Incremental Dominating Set

Ilan Doron Arad
Massachusetts Institute of Technology, Cambridge, MA, USA

Jonathan Gal
Technion – Israel Institute of Technology, Haifa, Israel

Joseph (Seffi) Naor
Technion – Israel Institute of Technology, Haifa, Israel

Abstract

Dominating Set is a fundamental problem in graph theory: given a graph, find a minimum-weight subset of vertices such that every vertex is either selected or adjacent to a selected vertex. In online settings where vertices arrive sequentially, comparing algorithms against an offline optimum with full knowledge of the input leads to extremely strong lower bounds, where even a simple star graph shows that any online algorithm must have competitive ratio $\Omega(\Delta)$, with Δ the largest degree of any vertex in the graph, matching the trivial strategy of selecting all vertices.

We study the *incremental* dominating set problem, where the optimal algorithm is constrained to the same choices available to online algorithms. This introduces a benchmark that enables a meaningful comparison between algorithms.

We present the first results for vertex-weighted graphs and randomized algorithms in this model. For incremental dominating set, we give an $O(\Delta)$-competitive deterministic algorithm and an $O(\log^2 \Delta)$-competitive randomized algorithm. We extend these results to the Connected Dominating Set problem using a linear-programming formulation that captures connectivity through local constraints. When the neighborhood of each arriving vertex is known *in advance*, deterministic algorithms achieve similar polylogarithmic competitive ratios as their randomized counterparts. Finally, we establish matching lower bounds, showing that our results are optimal up to constant factors.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph theory

Keywords and phrases Algorithms, Online Algorithms, Dominating Set

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.5

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2606.27469

Funding *Joseph (Seffi) Naor*: Supported in part by ISF grant 3001/24 and United States – Israel BSF grant 2022418.

1 Introduction

Dominating Set is a fundamental and extensively studied problem in graph theory and algorithm design, e.g. [15, 18, 24]. Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, we say that vertex u dominates vertex v if either $u = v$ or $(u, v) \in E$. A subset of vertices $D \subseteq V$ is then called a dominating set if every vertex in V is dominated by at least one vertex in D. Variants include the Connected Dominating Set problem [10, 11, 27], which requires the dominating set to be connected, and the Total Dominating Set problem [2, 7], where every vertex must be dominated by a distinct neighbor. These, and related problems, have found applications in network routing [26], facility location [16, 22], efficient transmission [20], and more [23].

We consider online versions of these problems. In the online setting, domination requests arrive sequentially, requiring the algorithm to make irrevocable decisions, while maintaining feasibility at each step without knowledge of future requests [3, 8, 17].

Online domination has primarily been studied under two models: the known-graph and unknown-graph settings. In the known-graph model, the entire graph is known upfront, and requests for dominating specific vertices arrive sequentially. This model is a special case of the online set cover problem, for which the known competitive bounds are polylogarithmic in the number of sets and elements [1]. Several works have extended these bounds to variants of the known-graph model in the context of domination problems [21].

In the unknown-graph model, the vertices are revealed one by one along with the edges connecting them to previously seen vertices. Each revealed vertex must be immediately dominated. The version most commonly studied in this model is the late accept variant [3], in which a vertex can be added to the dominating set at any time after it arrives, but cannot be removed once it is added.

Most research in this setting analyzes online algorithms by comparing them to a powerful offline algorithm. Consider the star graph in Figure 1; it demonstrates that every online algorithm must have competitive ratio $\Theta(\Delta)$, where Δ is the maximum degree of a vertex, matching the trivial strategy that always adds all vertices to the dominating set. Thus, there has been very little research in this direction [3, 9].

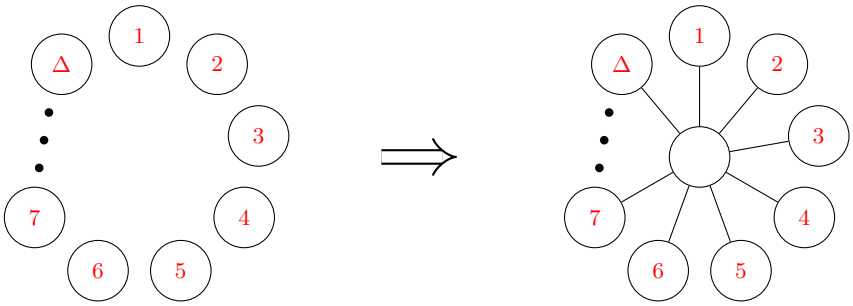

Figure 1 A star graph where the center vertex is revealed last. An online algorithm, unable to choose the center vertex before it is requested must select all leafs, while an offline algorithm needs only to select the center vertex, yielding a competitive ratio of $\Omega(\Delta)$. However, against the optimal **incremental** solution, which is also unable to choose the center vertex, the competitive ratio is 1.

To provide a more meaningful benchmark, we propose comparing online algorithms with their incremental counterparts. In the incremental dominating set (DS) problem, we are given a graph $G = (V, E)$ and an ordering $v_1, v_2, \ldots, v_n$ of the vertices. The goal is to choose a subset $S \subseteq V$ such that for every $t \in [n]$, the set $S \cap \{v_1, \ldots, v_t\}$ is a dominating set for the subgraph induced by $\{v_1, \ldots, v_t\}$. Since both the algorithm and the optimal incremental solution face the same arrival constraints, this benchmark is more informative. In the example in Figure 1, the competitive ratio is reduced to 1 from $\Omega(\Delta)$.

This problem naturally models constraints that online algorithms face in the unknown-graph setting. Boyar et al. [3] were the first to consider deterministic online algorithms for the incremental model on unweighted graphs in the unknown-graph setting. They showed that any deterministic online algorithm has a competitive ratio of $\Theta(\Delta)$. This result implies that, for unweighted graphs, no deterministic online algorithm can substantially outperform the naïve approach of simply selecting all vertices. However, the question of obtaining improved competitive bounds for other settings, such as weighted graphs, where achieving an $O(\Delta)$ competitive algorithm is non-trivial, or randomized algorithms, has remained open.

Table 1 Results for incremental DS and incremental CDS. Entries marked with - follow from results in other rows. All results apply to weighted graphs unless specified otherwise. Δ refers to the maximum degree of the graph induced by requested vertices, while Δ_G refers to the maximum degree of the revealed graph.

	Incremental DS	**Incremental CDS**	**Lower Bound**
Deterministic	$O(\Delta)$	$O(\Delta)$	$\Omega(\Delta)$ [3]
Randomized (Polynomial Time)	$O(\log^2 \Delta)$	$O(\log^2 \Delta)$	$\Omega(\log^2 \Delta)$
Randomized (General)			$\Omega(\log \Delta)$
Known Neighborhood Unweighted	$O(\log^2 \Delta_G)$	$O(\log^2 \Delta_G)$	$\Omega(\log^2 \Delta)$
Known Neighborhood (Polynomial Time)	$O(\log n \log \Delta)$	–	$\Omega(\log^2 \Delta)$
Known Neighborhood (Deterministic)			$\Omega(\log^2 \Delta / \log\log \Delta)$
Random Order Arrival	–	–	$\Omega(\log \Delta)$
Offline (Known Input)	$O(\log \Delta)$	$O(\log \Delta)$	$\Omega(\log \Delta)$

1.1 Our Contribution & Results

We address several open questions in online domination problems: obtaining competitive ratios for weighted graphs, randomized algorithms, and settings with additional structural information.

We start by formulating both incremental Dominating Set (DS) and incremental Connected Dominating Set (CDS) as covering problems. For incremental CDS we manage to encode both domination and connectivity requirements using a small number of linear constraints. This is the first use of such formulation in the online setting.

In the deterministic weighted setting, we show that incremental DS and incremental CDS admit an $O(\Delta)$-competitive algorithm, matching the unweighted lower bound [3]. For randomized algorithms, we achieve $O(\log^2 \Delta)$ competitive ratio and prove this is optimal up to a constant factor for polynomial-time algorithms. For computationally unbounded algorithms we prove an $\Omega(\log \Delta)$ lower bound.

We also study the **known neighborhood** model, where requesting a vertex reveals its full neighborhood. Surprisingly, knowing just the neighborhood of requested vertices in advance yields substantially stronger *deterministic* bounds: $O(\log^2 \Delta_G)$-competitive in the unweighted case for both incremental DS and incremental CDS where Δ_G is the maximum degree of the revealed graph (containing vertices which are not requested), and $O(\log n \log \Delta)$-competitive for weighted incremental DS. These competitive ratios hold at every step of the algorithm, and in particular the final solution remains competitive even if some revealed vertices were never requested. These bounds are asymptotically tight.

We also study offline and random-order models. In the offline setting, we establish tight $\Theta(\log \Delta)$ polynomial time approximation bounds unless NP $\subseteq$ BPP. In the random-order setting, we prove an $\Omega(\log \Delta)$ lower bound, showing that random arrival order does not substantially improve the competitive ratio.

A summary of our upper and lower bounds across different models appears in Table 1. We now discuss our results in detail.

1.1.1 Online Incremental

We begin with the online setting of incremental DS and CDS. Boyar et al. [3] have shown a $\Theta(\Delta)$ deterministic lower bound for incremental DS and incremental CDS, which matches the competitive bound of the trivial algorithm that selects all vertices in unweighted graphs. In contrast, we design a deterministic algorithm with a non-trivial competitive ratio for weighted graphs.

▶ **Theorem 1.** *There is a* $(\Delta + 1)$*-competitive deterministic algorithm for* incremental DS *and* incremental CDS*.*

In the randomized setting, the competitive ratio improves exponentially.

▶ **Theorem 2.** *There is an* $O(\log^2 \Delta)$*-competitive randomized algorithm for* incremental DS *and* incremental CDS*.*

We note that this result gives the first sublinear-competitive algorithm for weighted online CDS. In contrast, in the known-graph model (or in any model in which we only need to dominate a subset of the vertices) the problem reduces to non-metric TSP, for which obtaining an $f(n)$-approximation is hard for every polynomial-time computable function [11]. Thus, the incremental model allows substantially better guarantees.

We also study the **known neighborhood** model, where requesting a vertex v_i reveals its full neighborhood $N[v_i]$. Importantly, this extra information does not change the original setting, as the neighbors of v_i become visible to the algorithm, but cannot be added to the dominating set until they are requested. Surprisingly, this additional information enables deterministic algorithms to achieve polylogarithmic competitive ratios matching their randomized counterparts.

▶ **Theorem 3.** *When the neighborhood of requested vertices is known, and the graph is unweighted, then there is an* $O(\log^2 \Delta_G)$*-competitive deterministic algorithm for both* incremental DS *and* incremental CDS*.*

Here Δ_G denotes the maximum degree of the fully revealed graph, which may contain vertices that are never requested. The competitive ratio holds at every step of the algorithm, and remains valid even if the adversary stops requesting vertices before all revealed vertices have been requested. In the case where all revealed vertices are eventually requested we have $\Delta = \Delta_G$ and the algorithm is $O(\log^2 \Delta)$-competitive.

In the weighted case we still get polylogarithmic results for incremental DS.

▶ **Theorem 4.** *When the neighborhood of requested vertices is known, there is an* $O(\log n \log \Delta)$*-competitive deterministic algorithm for* incremental DS*.*

Here n denotes the number of revealed vertices and Δ the maximum degree of $G[V_n]$, the subgraph induced by requested vertices. As in the unweighted case, the competitive ratio holds at every step of the algorithm.

We establish matching lower bounds for our randomized algorithms.

▶ **Theorem 5.** *There is no polynomial-time online algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log^2 \Delta)$ *unless* NP $\subseteq$ BPP*, even when the neighborhood of requested vertices is known.*

For deterministic algorithms, we obtain a stronger lower bound.

▶ **Theorem 6.** *There is no deterministic online algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log^2 \Delta / \log\log \Delta)$*, even when the neighborhood of requested vertices is known.*

Finally, in the random-order setting where vertices arrive according to a uniformly random permutation, we establish a logarithmic lower bound.

▶ **Theorem 7.** *There is no online random-order algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log \Delta)$*.*

The random-order model is a restriction of the fully adversarial model, the above lower bound immediately implies the following.

▶ **Corollary 8.** *There is no online algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log \Delta)$*. This lower bound also holds when the neighborhood of requested vertices is known.*

1.1.2 The Offline Setting

We study the offline variants of incremental DS and CDS, where the entire graph and the vertex request order are known in advance. We show that these problems remain NP-complete and hard to approximate, and establish tight approximation bounds up to constant factors.

▶ **Theorem 9.** Incremental DS *and* incremental CDS *are* NP-complete*. Furthermore, unless* NP $\subseteq$ BPP *there is no polynomial-time* $o(\log \Delta)$ *approximation algorithm for either problem.*

Complementing this lower bound, we give a matching polynomial-time $O(\log \Delta)$-approximation algorithm for both problems.

▶ **Theorem 10.** *There is a polynomial-time* $O(\log \Delta)$*-approximation algorithm for* incremental DS *and* incremental CDS*.*

1.2 Our Techniques

We start in Section 3 by formulating both incremental DS and incremental CDS as covering problems. The formulation of incremental DS is standard, but for incremental CDS we introduce a more involved formulation that maintains connectivity throughout. For each newly requested vertex, we create different constraints depending on whether it connects to zero, one, or more than one existing connected component. This formulation captures both domination and connectivity requirements, while ensuring that both the maximum constraint size and maximum frequency are linear in Δ, enabling the application of standard techniques for online covering problems. This formulation is the foundation for most of our results.

As an immediate result from the linear formulation we prove the $O(\log^2 \Delta)$-competitive ratio for online randomized algorithms, and an $O(\log \Delta)$ approximation ratio for the offline setting, where both the graph and request order are known in advance. Another result of this formulation is a greedy deterministic $O(\Delta)$-competitive algorithm for the weighted variant of both settings.

In Section 4, we use the extra information provided by knowing the neighborhood of requested vertices to exponentially improve the competitive ratio for deterministic algorithms. When the graph is unweighted, we maintain both a fractional and an integral solution, coupled through a potential function that is exponential in the fractional coverage of elements not yet covered integrally. The exponential growth ensures that at most $O(|\mathrm{OPT}|)$ elements can be covered fractionally without being covered integrally. We combine this bound with an algorithm that guarantees that whenever a new uncovered vertex is requested, we add a

logarithmic number of vertices to the dominating set. Together, these techniques provide the desired competitive ratio. We note that our potential function approach is inspired by Alon et al. [1], yet our solution gives improved competitive bounds and handles a wider range of settings.

We note that Δ_G can increase over time, to handle this we use a guess-and-square technique that achieves the desired $O(\log^2 \Delta_G)$ competitive ratio, improving upon the $O(\log^3 \Delta_G)$ bound obtained by the standard guess-and-double approach (see Section 4.1.2 for more details). This technique is used several times throughout the paper for algorithms where the maximum degree of the graph is not known in advance.

To get a similar competitive ratio for incremental CDS, we partition constraints into two sets: the first one contains the constraints handled by the incremental DS algorithm and the second one contains at most $O(|\mathrm{OPT}|)$ other constraints. The small size of the second set ensures that satisfying these constraints preserves the competitive ratio.

For the weighted case, we augment the potential function with a term that bounds the difference between the cost of the integral and fractional solutions while simultaneously bounding the increase of the potential throughout the runtime, yielding a logarithmic competitive ratio for incremental DS.

To obtain matching lower bounds on our competitive ratios, we provide reductions from dominating set and online set cover. A key insight in both reductions is the use of auxiliary vertices that are revealed first and must be selected. Then, we reveal vertices which are covered by these auxiliary vertices, revealing information about the instance to the incremental algorithm (either the sets or the graph depending on the reduction), thus enabling us to construct the reductions. These reductions preserve both the instance size and the optimal solution size, allowing us to transfer known lower bounds to our setting.

For the random-order lower bound, we replicate vertices so that a random permutation induces the adversarial set cover sequence with high probability, transferring random-order set cover lower bounds to the random order setting.

2 Preliminaries

We introduce the notation used throughout the paper, formally defining the problems we study, and presenting their covering formulations.

Let $G = (V, E, w)$ denote an undirected graph with vertex set V, edge set E, and weight function $w : V \to \mathbb{R}^+$. Let $w(S) = \sum_{s \in S} w(s)$ denote the total weight of all elements in $S \subseteq V$. The set of all connected components of G is denoted by $C(G)$.

Without loss of generality, we assume that $V = \{v_1, \ldots, v_n\}$, and that vertices are requested in this order, i.e., vertex v_1 is requested first, then v_2, and so on. Denote $V_i = \{v_1, \ldots, v_i\}$ when the order is understood from context.

For a vertex v, we denote by $N(v)$ the set of neighbors of v and $N[v] = N(v) \cup \{v\}$. We define the natural extensions to sets $N[U] = \bigcup_{u \in U} N[u]$.

For any $H \subseteq V$, let $G[H]$ be the induced subgraph of G on H.

We now define the competitive ratio used to benchmark our algorithms throughout the paper.

▶ **Definition 11** (Competitive Ratio). *The* competitive ratio *of an online algorithm $\mathcal{A}$ is*

$$\sup \frac{w(\mathcal{A})}{w(\mathrm{OPT})},$$

where OPT *denotes the cost of the optimal algorithm, and the supremum is taken over all graphs, weight functions, and request sequences.*

In this paper, OPT always refers to an optimal **incremental** algorithm.

2.1 Problem Definitions

We begin by defining the dominating set and connected dominating set problems.

▶ **Definition 12** (Dominating Set). *For a graph G, a* dominating set *is a subset $S \subseteq V$ such that $\bigcup_{v \in S} N[v] = V$. The objective is to minimize $w(S)$.*

In addition to covering all vertices, we may impose the requirement that the dominating set be connected within each connected component of the graph.

▶ **Definition 13** (Connected Dominating Set). *A* connected dominating set (CDS) *is a* dominating set *of G such that for every connected component C of G, the set $S \cap C$ is connected.*

To model the decisions an online algorithm must make incrementally, we define the incremental dominating set problem and its connected variant. Recall that V_i denotes the set containing the first i vertices requested by the online process.

▶ **Definition 14** (Incremental Dominating Set). *For a graph G, an* incremental dominating set *is a subset $S \subseteq V$ such that for all $i \in [n]$, $S \cap V_i$ is a* dominating set *of $G[V_i]$.*

▶ **Definition 15** (Incremental Connected Dominating Set). *For a graph G, an* incremental connected dominating set *is a subset $S \subseteq V$ such that for all $i \in [n]$, $S \cap V_i$ is a* connected dominating set *of $G[V_i]$.*

In the online setting, the vertices of a graph are requested one by one. In this paper, we consider two information models: at step i, the algorithm either knows the induced subgraph $G[V_i]$, which corresponds to the standard vertex arrival model, or the neighborhood subgraph $G[N[V_i]]$. Throughout the paper, the competitive ratio is measured against OPT, the optimal incremental dominating set.

▶ **Definition 16** (Known Neighborhood Model). *In the* known neighborhood model*, at step i when vertex v_i is requested, the algorithm observes the induced subgraph $G[N[V_i]]$ rather than just $G[V_i]$. Vertices in $N[V_i] \setminus V_i$ are visible but cannot be selected for the dominating set until requested.*

We work in the late-accept model [3, 4], where vertices can be added to the solution at any time but cannot be removed once chosen. We now define the online variants of the incremental Dominating Set and incremental Connected Dominating Set problems.

▶ **Definition 17** (Online Incremental Dominating Set). *In the* online incremental dominating set *problem, the algorithm must maintain a dominating set of $G[V_i]$ at each step i.*

▶ **Definition 18** (Online Incremental Connected Dominating Set). *In the* online incremental connected dominating set *problem, the algorithm must maintain a connected dominating set of $G[V_i]$ at each step i.*

Since any superset of a CDS is also a CDS, the late accept setting does not violate the incrementality requirements of previous steps. Adding vertices to the solution at later steps preserves the CDS property for all previous graphs.

2.2 Covering Problems

In this paper, we use linear programs with covering constraints

▶ **Definition 19** (Covering Problem). *A* covering problem *is a linear program of the form:*

$$\begin{aligned} \min \quad & \sum_{i\in[n]} w(i)\cdot x_i \\ & s.t. \\ a^j \cdot \vec{x} &\geq 1, && \forall j \in [m] \\ x_i &\geq 0 && \forall i \in [n] \end{aligned} \tag{1}$$

where $a^j \in \{0,1\}^n$ *for all* $j \in [m]$.

3 Covering LPs for Incremental DS and Incremental CDS

We now formulate linear programming (LP) relaxations for incremental DS and incremental CDS, expressing both as covering problems with each constraint involving at most $O(\Delta)$ variables and each variable participating in $O(\Delta)$ constraints. Formulating both problems as covering LPs has two main advantages. First, it allows us to leverage techniques already known in the literature for online covering problems. Second, it provides a unified framework for designing algorithms for both problems simultaneously, an approach we exploit in later sections.

We formalize these observations in the following theorem,

▶ **Theorem 20.** *Both* incremental DS *and* incremental CDS *have* LP *relaxations that are covering problems. The size of each constraint is bounded, and so is the number of variables in each constraint. More precisely:*

1. *For* incremental DS*, each constraint involves at most* $\Delta + 1$ *variables.*
2. *For* incremental CDS*, each constraint involves at most* Δ *variables.*
3. *In both relaxations, each variable appears in at most* $\Delta + 1$ *covering constraints, and a single non-negativity constraint.*

3.1 Incremental DS as a Covering Problem

We begin by formulating the LP relaxation for incremental DS. For each vertex v_i, we introduce a variable $x_i \in [0,1]$.

Proof of Theorem 20(1). When v_i is requested, either v_i or one of its (previously requested) neighbors needs to be in the dominating set, so

$$\sum_{j:v_j\in N[v_i]\cap V_i} x_j \geq 1.$$

Due to the monotone nature of the solution, it suffices to ensure that each newly requested vertex is covered when it appears. This gives us the following relaxation

$$\begin{aligned} \min \quad & \sum_{i\in[n]} x_i \cdot w(v_i) \\ & \text{s.t.} \\ \sum_{j:v_j\in N[v_i]\cap V_i} & x_j \geq 1, && \forall i \in [n] \\ x_i &\geq 0, && \forall i \in [n] \end{aligned} \tag{2}$$

This relaxation is a covering problem. Each constraint involves at most $\Delta + 1$ variables since it includes v_i and its neighbors in V_i. Moreover, each variable x_j appears only in the non-negativity constraint and constraints for vertices v_i such that $v_j \in N[v_i]$. Since $|N[v_i]| \leq \Delta + 1$, each variable x_j appears in at most $\Delta + 1$ covering constraints. ◀

3.2 Incremental CDS as a Covering Problem

Here we prove Theorem 20(2) by identifying a small set of integral constraints that ensures the chosen subset is both a dominating set and is connected. We provide the intuition for the LP formulation here, and defer the details to the full version of the paper.

To express connectivity using local constraints, we classify each new vertex by how it interacts with existing components: it may start a new one, attach to one, or join several together. The cases require different types of constraints to maintain both domination and connectivity. To the best of our knowledge, this is the first time that connectivity is captured using only local constraints.

Proof of Theorem 20(2). We start by classifying the vertices into three types:

1. Opener – A vertex that, upon arrival, is not connected to any other vertex. Formally, $N[v_i] \cap V_i = \{v_i\}$. For example, the first vertex v_1 is always an opener.
2. Attacher – A vertex that, upon arrival, is connected to only one component. Formally, there exists $C \in C(G[V_{i-1}])$ such that for all $C' \neq C$, $N[v_i] \cap V_i \cap C' = \emptyset$.
3. Joiner – A vertex that connects at least two components upon arrival. Formally, there exist $C_1, C_2 \in C(G[V_{i-1}])$ such that $N[v_i] \cap V_i \cap C_1 \neq \emptyset$ and $N[v_i] \cap V_i \cap C_2 \neq \emptyset$.

We now add the appropriate constraints for each type to enforce both domination and connectivity.

Case 1: Opener. An Opener creates a new component and must be chosen to dominate itself. The only way to satisfy the dominating set requirement is to include it in the dominating set, thus $x_i \geq 1$.

Case 2: Attacher. An Attacher connects to one existing component C. The domination and connectivity constraints are satisfied as follows:

- If v_i is added to the CDS, at least one of its neighbors in C must also be in the CDS to maintain connectivity.
- If v_i is **not** added to the CDS, at least one of its neighbors must be added to satisfy the domination requirement.

Both scenarios are enforced by the constraint:

$$\sum_{v_j \in N(v_i) \cap V_i} x_j \geq 1.$$

This constraint is sufficient because the neighbors of v_i are already adjacent to the CDS, so adding any neighbor maintains both domination and connectivity constraints.

Case 3: Joiner. The joiner connects multiple components $C_1, \ldots, C_k$. After adding the joiner the dominating set of any pair of components must have a path between them, but such a path must go through the joiner. As a result the joiner must be added to the dominating set. To maintain connectivity:

- v_i must be included in the CDS, $x_i \geq 1$.
- For each component C_h, at least one neighbor of v_i within C_h must be included:

$$\sum_{v_j \in N(v_i) \cap V_i \cap C_h} x_j \geq 1.$$

This requirement is sufficient since any other vertex is already adjacent to the CDS, so adding a neighbor keeps the joiner connected to the dominating set in C_h.

The maximum size of a constraint is Δ for attachers. Moreover, each constraint corresponds to a vertex v_i and involves only its neighbors. Any variable x_j appears in one non-negativity constraint, at most one constraint for each neighbor v_i, and at most one constraint for v_j itself. Since each vertex has at most Δ neighbors, each variable appears in at most $\Delta + 2$ constraints overall. ◀

3.3 Algorithmic Implications

We start by showing the strength of Theorem 20 by combining it with several known theorems about covering problems to obtain immediate results.

The first of such results is for the offline problem, where from Srinivasan [25] we have the following theorem.

▶ **Theorem 21** (Srinivasan [25]). *For any covering problem with sparsity d (i.e., each variable appears in at most d constraints), there is an $O(\log d)$-approximation algorithm.*

Combining Theorems 20 and 21 yields the approximation ratio for the offline setting of both incremental DS and incremental CDS.

▶ **Theorem 10.** *There is a polynomial-time $O(\log \Delta)$-approximation algorithm for* incremental DS *and* incremental CDS*.*

Proof. By Theorem 20, both incremental DS and incremental CDS can be formulated as covering integer programs where each variable appears in at most $\Delta + 1$ constraints, meaning that the column sparsity is $d = \Delta + 1$. Applying Theorem 21 yields an $O(\log(\Delta + 1)) = O(\log \Delta)$-approximation algorithm for both problems. ◀

For the online setting, we apply the following theorem from Gupta and Nagarajan [13].

▶ **Theorem 22** (Gupta and Nagarajan [13]). *There exists an $O(\log k \cdot \log d)$-competitive randomized online algorithm for covering problems, where k is the maximum number of variables in any constraint and d is the sparsity (maximum number of constraints any variable appears in).*

Again, combining Theorems 20 and 22 immediately yields the randomized competitive ratio.

▶ **Theorem 2.** *There is an $O(\log^2 \Delta)$-competitive randomized algorithm for* incremental DS *and* incremental CDS*.*

Proof. By Theorem 20, the LP relaxations of both problems are online covering problems with row sparsity $k \leq \Delta + 1$ and column sparsity $d \leq \Delta + 1$. Applying Theorem 22 gives an $O(\log^2 \Delta)$-competitive randomized online algorithm. ◀

Greedy Deterministic Online Algorithm

We present a deterministic greedy algorithm for the weighted variants of incremental DS and incremental CDS. Upon the arrival of each covering constraint, the algorithm selects the minimum-weight vertex satisfying the constraint, adding it to the dominating set if it has not been selected previously. The analysis is based on the covering formulation of Theorem 20. Since each variable participates in at most $\Delta + 1$ covering constraints, the greedy algorithm is $(\Delta + 1)$-competitive. The complete proof is straightforward and is deferred to the full version of the paper.

▶ **Theorem 1.** *There is a $(\Delta + 1)$-competitive deterministic algorithm for* incremental DS *and* incremental CDS*.*

4 Know Thy Neighbor

In this section we present improved deterministic algorithms for the case where the neighborhood of each requested vertex v_i ($N[V_i]$) is revealed when v_i needs to be dominated. The setting remains incremental and vertices which have not been requested (yet were revealed earlier) cannot be used to cover requested vertices. In other words, when a vertex is requested, we learn about its neighbors, but only the requested vertex itself needs to be dominated.

Knowing the neighborhood of each requested vertex complies with the arrival model studied in previous sections, except that additional information is provided at runtime. This extra information allows us to design deterministic algorithms with polylogarithmic competitive ratios, a substantial improvement over the $\Omega(\Delta)$ lower bound from the setting where only requested vertices are known (without full neighborhood information).

Since revealed vertices need not all be requested, we distinguish between Δ_G, the maximum degree of the full revealed graph, and Δ, the maximum degree of $G[V_n]$, the subgraph induced by requested vertices.

In Section 4.1, we present an $O(\log^2 \Delta_G)$-competitive deterministic algorithm for the unweighted case, for both incremental DS and incremental CDS. Then, in Section 4.2, we show that in the weighted case there exists an $O(\log n \log \Delta)$-competitive deterministic algorithm for incremental DS. We leave improving the upper bound for weighted incremental CDS as an open problem.

This model differs from that of [14], where a vertex must be covered only when its last neighbor is revealed, and late acceptance is disallowed.

4.1 Unweighted Case

We first present a deterministic algorithm for incremental DS. Our approach is based on maintaining both a fractional solution and an integral dominating set D, coupled through a potential function. The potential function penalizes vertices that are covered fractionally but not integrally, ensuring that vertices with high fractional coverage are added to the integral solution.

The algorithm maintains a parameter m, initialized to an upper bound on the maximum degree of the revealed graph Δ_G, and updated dynamically to remain an upper bound as new vertices arrive (see Section 4.1.2). We also maintain a fractional solution x (possibly infeasible), initialized with $x_i = \frac{1}{4m}$ for each vertex v_i.

To track how well each vertex is covered fractionally, we define the potential cover c_v for each known vertex v. This measure has two components: the fractional coverage from revealed neighbors, and a correction term that accounts for potential neighbors that have not yet been revealed. Specifically, when a new neighbor of v is revealed, it contributes $\frac{1}{4m}$ to the fractional coverage, while simultaneously decreasing the correction term by the same amount, ensuring c_v remains unchanged. Formally, for each revealed vertex v we define:

$$c_v = \sum_{u_j \in N[v] \cap V_i} x_j + \frac{m - |N[v] \cap V_i|}{4m}.$$

Let D be the dominating set maintained by the algorithm at the current step, and denote the set of uncovered elements as $U_i = N[V_i] \setminus N[D]$. Define the potential at step i as $\Phi_i = \sum_{v \in U_i} m^{3c_v}$. The exponential term ensures that vertices with high fractional coverage cannot remain uncovered integrally.

At each step, we first initialize any newly revealed neighbors of v_i with fractional value $\frac{1}{4m}$. If v_i is uncovered, we snapshot the current potential as Φ_s, then increase the weights of vertices in $N[v_i] \cap V_i$ until v_i is covered fractionally. We then add the smallest subset of $N[v_i] \cap V_i$ to D such that the potential is restored to at most Φ_s; by Lemma 23 this subset has size $O(\log m)$. Finally, if v_i remains uncovered after this process, we add v_i itself to D. A pseudocode of the algorithm is described in Algorithm 1.

Algorithm 1 Unweighted Known Neighborhood Deterministic.

```
   // Notation as defined in the preceding text.
1  Initialize D ← ∅ (initial dominating set).
2  for vertex v_i ∈ V do
3      For each newly revealed vertex v_j, set x_j ← 1/(4m), and increment Φ_i by
       m^{3c_{v_j}} ≤ m^{3/4}. if v_i ∈ U_i then
4          Set Φ_s ← Φ_i.
5          while c_{v_i} < 1 do
6              For each v_j ∈ N[v_i] ∩ V_i, multiply x_j by two.
7          Add the smallest subset of vertices to D for which Φ_i ≤ Φ_s. // By Lemma 23,
           the size of such a subset is O(log m).
8      if v_i is not covered then
9          Add v_i to D.
```

To begin the analysis, we show that after each weight increase the algorithm can avoid an increase in the potential function by adding only a small subset of vertices to the dominating set in Line 7.

▶ **Lemma 23.** *If v_i is uncovered at step i, then there exists a subset of $N[v_i] \cap V_i$ of size at most $6 \log m$ such that adding the vertices in the subset to D ensures that the potential does not increase, i.e.,*

$$\Phi_i \leq \Phi_s,$$

where Φ_s is the potential prior to the weight increases in Line 6.

The proof is deferred to the full version of the paper. We next move from considering how much is added per update to bounding how often such updates happen.

▶ **Lemma 24.** *The total number of times the* **if** *statement in Line 3 in Algorithm 1 is triggered is at most $O(|\mathrm{OPT}| \log m)$.*

Proof. Every coordinate x_j is initialized to $1/(4m)$ and the algorithm only multiplies coordinates by 2 (Line 6). Moreover, Line 5 ensures that the algorithm never increases the value of any coordinate beyond 2. Thus, for every index j the number of times x_j can be doubled is at most $\log(8m) = O(\log m)$.

Since v_i must be dominated by a vertex in OPT, each time the **if** statement is triggered at least one coordinate corresponding to a vertex of OPT is doubled.

Combining these results, the total number of times that the **if** statement can be triggered is upper bounded by $O(|\mathrm{OPT}| \cdot \log m)$. ◀

Having bounded the number of update steps, we now show that the increase in the potential function itself remains bounded throughout the execution. Since each vertex added in the final while loop decreases the potential by a large amount, bounding the total potential increase immediately implies a bound on the number of vertices added in that phase.

▶ **Lemma 25.** *At step i, $\Phi_i \leq i \cdot \Delta_G \cdot m$.*

Proof. The potential can change in three places: when a vertex is requested, when weights are increased in Line 6, and when new vertices are revealed.

First, consider the moment when a vertex $v_i \in N[u]$ is requested. Its initial contribution of $1/(4m)$ is added to the first summand of c_u, but at the same time $|N[u] \cap V_i|$ increases by one, which decreases the correction term $(m - |N[u] \cap V_i|)/4m$ by exactly $1/(4m)$. The two changes cancel, and c_u does not change.

Second, by Lemma 23, after each weight increase in Line 6 the vertices added to D in Line 7 ensure that the potential does not increase, so weight increases cause no net increase in the potential.

Therefore the only source of increase is when new vertices are revealed. For each new vertex u, the increase in the potential is at most $m^{3c_u} \leq m$, since $c_u \leq \frac{1}{4}$ at reveal. After i steps, the number of revealed vertices is bound by $i \cdot \Delta_G$, the total increase is bounded by $i \cdot \Delta_G \cdot m$. ◀

We now use the bounded increase in Φ to bound the number of vertices added in the final loop of the algorithm. Intuitively, since each additional vertex in that phase reduces the potential by a large amount, a global bound on the total potential increase immediately limits how many such insertions can occur.

▶ **Lemma 26.** *Denote the number of total requested vertices throughout the algorithm as r. The number of vertices added by Lines 8-9 is bounded by $\frac{r \cdot \Delta_G}{m^2}$, and if $m \geq \Delta_G$ then the bound is $2 \cdot |\mathrm{OPT}|$.*

Proof. When we insert v_i to D in Line 9, we remove at least $m^{3c_u} \geq m^3$ from the potential. By lemma 25 the number of added vertices is at most

$$\frac{r \cdot \Delta_G \cdot m}{m^3} = \frac{r \cdot \Delta_G}{m^2}$$

If $m \geq \Delta_G$ then $|\mathrm{OPT}| \geq r/(\Delta_G + 1) \geq r/(2\Delta_G)$, giving us the upper bound. ◀

We are now ready to combine the previous bounds to obtain the overall competitive ratio.

▶ **Lemma 27.** *If $m \geq \Delta_G$, the competitive ratio of algorithm 1 is $O(\log^2 m)$.*

Proof. By Lemma 24 we enter the **if** in Line 3 at most $O(|\mathrm{OPT}| \cdot \log m)$ times, and by Lemma 23 each time we add $4 \log m$ vertices to D. By Lemma 26 we then add at most $2|\mathrm{OPT}|$ vertices in Lines 8-9. Hence

$$|D| \;\leq\; O(|\mathrm{OPT}| \cdot \log m) \cdot 4 \log m + 2|\mathrm{OPT}| \;=\; O(|\mathrm{OPT}| \cdot \log^2 m),$$

proving the claimed competitive ratio. ◀

We note that Algorithm 1 extends naturally to the case where constraints are of the form $N(v_i) \cap D$ (as with attacher constraints), where the requested vertex itself is excluded. If $N(v_i) = \emptyset$ then v_i is an isolated vertex and no constraint is enforced. Otherwise, all we need to do is replace all instances of $N[v_i]$ with $N(v_i)$. The potential function argument is unchanged, and the same $O(\log^2 \Delta_G)$-competitive bound holds.

4.1.1 Connected Dominating Set

In this part we extend the result of Section 4.1 to incremental CDS. The algorithm is similar to Algorithm 1, but with a special case for joiners (recall that a joiner is a vertex that joins two connected components, see Section 3.2 for the formal definition). When a joiner is requested, for each constraint created by it, we immediately add an arbitrary vertex from the constraint to the dominating set. We prove that the number of vertices added this way is bounded by $O(|\mathrm{OPT}|)$.

▶ **Lemma 28.** *The number of constraints added by joiners is bounded by* $2 \cdot |\mathrm{OPT}|$.

Proof. We actually prove a stronger statement, namely that the total number of constraints created by openers and joiners is at most $2 \cdot |\mathrm{OPT}|$. Let $\mathcal{O}$ be the number of openers and let $\mathcal{J}$ be the number of joiners. By the transformation in Section 3.2, every opener and joiner must belong to the dominating set, so $|\mathrm{OPT}| \geq \mathcal{O} + \mathcal{J}$.

We use a charging argument. When an opener is requested, it becomes the representative of its component. When a joiner is requested, each connected component it merges creates one constraint, which we charge to that component's representative. The joiner then becomes the representative of the merged component.

Each vertex can be charged at most twice: once when requested (as an opener or joiner), and at most once later as a representative. Therefore, the total number of constraints is at most $2(\mathcal{O} + \mathcal{J}) \leq 2 \cdot |\mathrm{OPT}|$. ◀

4.1.2 Guess and Increase

To extend our algorithm to the case where the maximum degree Δ_G is not known in advance, we use a guess-and-update technique inspired by the standard "guess-and-double" framework. A straightforward adaptation, starting with an underestimate of Δ_G and doubling it whenever the maximum degree exceeds the current guess, leads to a competitive ratio of

$$\sum_{j=1}^{\log \Delta_G} O(\log^2(\Delta_G/2^j)) \cdot \mathrm{OPT} = O(\log^3 \Delta_G) \cdot \mathrm{OPT},$$

which is asymptotically worse than the optimal bound. To handle this discrepancy, instead of doubling the current estimate of Δ_G, we square it upon discovering that the maximum degree violates the current guess.

We are now ready to prove the main theorem of the section.

▶ **Theorem 3.** *When the neighborhood of requested vertices is known, and the graph is unweighted, then there is an* $O(\log^2 \Delta_G)$*-competitive deterministic algorithm for both* incremental DS *and* incremental CDS*.*

Proof. Denote by OPT_i the cost of the optimal solution for the first i vertices, and let Δ_i be the maximum degree of the revealed graph at this step. Since each new vertex introduces additional constraints, it follows that $\mathrm{OPT}_i \leq \mathrm{OPT}$ for all i.

We maintain an adaptive upper bound Δ' on the current maximum degree, initialized to a constant. As vertices arrive, whenever $\Delta_i > \Delta'$, we update $\Delta' \leftarrow (\Delta')^2$, restart the algorithm with the new bound, and include in the global solution all vertices selected during the new run while retaining vertices selected during previous solutions.

By Lemma 27, each phase with bound Δ' has cost $O(\log^2 \Delta') \cdot \mathrm{OPT}$. Since Δ' grows quadratically, after k such increases the total cost is bounded by

$$O\left(\sum_{j=0}^{k} \log^2\left(\Delta_G^{1/2^{k-j}}\right)\right) \cdot \mathrm{OPT} \leq O\left(\sum_{j=0}^{\infty} \log^2\left(\Delta_G^{2^{-j}}\right)\right) \cdot \mathrm{OPT} = O(\log^2 \Delta_G) \cdot \mathrm{OPT}.$$

Therefore, the adaptive algorithm preserves the $O(\log^2 \Delta_G)$ competitive ratio even when Δ_G is unknown in advance.

To extend the result to incremental CDS, we use the CDS formulation from Section 3.2. We use Algorithm 1 to handle the attacher constraints (as defined in Section 3.2, these are the constraints corresponding to vertices that attach to an existing component) and achieve an $O(\log^2 \Delta_G)$-competitive dominating set for these constraints. Since the optimal solution for the attacher constraints is at most $|\mathrm{OPT}|$, the number of vertices selected by Algorithm 1 is bounded by $O(\log^2 \Delta_G) \cdot |\mathrm{OPT}|$. By Lemma 28, there are at most $2|\mathrm{OPT}|$ constraints created by openers and joiners, and adding one vertex per such constraint contributes at most $2|\mathrm{OPT}|$ additional vertices. Combining both contributions results in a valid incremental CDS of size $O(\log^2 \Delta_G) \cdot |\mathrm{OPT}|$, finishing the proof. ◀

4.2 Weighted Case

In this part, we show an $O(\log m \log \Delta)$ deterministic algorithm for incremental DS that uses an upper bound on the value of the optimal solution α, and a parameter m which represents an upper bound on the number of **requested** vertices n. In this section Δ represents the degree of $G[V_n]$, the subgraph induced by requested vertices, disregarding vertices which are revealed but not requested. We apply doubling to handle the case where α is not known, and we later show how to increase m dynamically, using a technique similar to doubling, to obtain the desired competitive ratio of $O(\log n \log \Delta)$. If the number of requested vertices n is known in advance we can set $m = n$ instead.

Our algorithm maintains both a fractional and an integral solution. We use the fractional solution to guide which vertices to add to the integral dominating set D, while a potential function ensures the two solutions remain balanced. To obtain the fractional solution, we apply the following result from Buchbinder et al. [5]:

▶ **Theorem 29.** *There is a deterministic $O(\log d)$-competitive fractional algorithm for covering problems with constraint size up to d.*

Since each covering constraint involves at most $\Delta + 1$ variables, Theorem 29 guarantees an online monotone fractional solution which is $O(\log \Delta)$ competitive.

Denote the dominating set as D. Let the set of vertices adjacent to requested vertices that are not yet dominated at step i be $U_i = N[V_i] \setminus N[D]$. Denote $c_{v_i} = \sum_{u_j \in N[v_i] \cap V_i} x_j$ as the fractional coverage of v_i, and define the potential at step i as

$$\Phi_i = \sum_{v \in U_i} m^{2c_v} + \exp\left(\frac{1}{2\alpha}\left(\sum_{v \in D} w(v) - 3 \log m \sum_{v_j \in V_i} w(v_j) x_j\right)\right).$$

The first sum ensures that every vertex is covered when it is requested, and the second term ensures that the integral and fractional solutions do not differ by too much.

At each step a vertex v_i is requested to be dominated. We update the potential function for the newly revealed neighbors of v_i, and then add the covering constraint $\sum_{j \in N[v_i] \cap V_i} x_j \geq 1$ to the LP. We increase the weights of vertices in $N[v_i] \cap V_i$ in the fractional solution until this constraint is satisfied. Then, we add a subset of vertices to D that ensures that the

potential does not increase. Vertices v_j with $w(v_j) > \alpha$ are kept at $x_j = 0$ in the fractional solution, ensuring that heavy vertices are never selected. The algorithm is described formally in Algorithm 2.

Algorithm 2 Known Neighborhood Deterministic.

```
   // Notation as defined in the preceding text.
1  Initialize D ← ∅ (initial dominating set).
2  Initialize x ← 0⃗, Φ_0 ← e^0 = 1.
3  for vertex v_i ∈ V do
4      For each newly revealed vertex u, increment Φ_i by m^{2c_u} = 1.
5      Add covering constraint for v_i and compute the online fractional solution x̃ from
        Theorem 29.
6      for j = 1, ..., i do
7          Set x_j ← x̃_j. // Update fractional solution
8          If Φ_i increased by the change in last line, add v_j to D
```

The analysis follows the same potential-function framework as in the unweighted case, where vertices are added to the integral solution in a way that ensures the potential function decreases. Combining this approach with the $O(\log \Delta)$-competitive fractional algorithm yields an $O(\log m \log \Delta)$ competitive ratio. The detailed analysis is deferred to the full version of the paper, and establishes the following result.

▶ **Theorem 4.** *When the neighborhood of requested vertices is known, there is an* $O(\log n \log \Delta)$*-competitive deterministic algorithm for* incremental DS.

5 Lower Bounds

5.1 Lower Bounds via Online Set Cover

We present a reduction from the online set cover problem to the online incremental Dominating Set problem. The reduction preserves the instance size up to polynomial factors and maintains the optimal solution size up to a constant additive term. This allows us to transfer known hardness results for online set cover and prove Theorems 5 and 6.

Given an instance of online set cover with universe $U = \{e_1, e_2, \ldots, e_n\}$ and collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ with $S_i \subseteq U$, construct the incremental dominating set instance as follows. The vertex set is

$$V = \{r, t, t'\} \cup \mathcal{S} \cup U,$$

where r is a root adjacent to all set nodes, and t is a terminal adjacent to all element nodes, with a neighbor t'. The edge set is

$$E = (\{r\} \times \mathcal{S}) \cup \{(S_i, e_j) \mid e_j \in S_i\} \cup (U \times \{t\}) \cup \{(r, t), (t, t')\}.$$

That is, every set S_i is adjacent to r, a set S_i and element e_j are adjacent iff $e_j \in S_i$, every element of U is adjacent to t, and t is adjacent to both r and t'.

In the online adversarial setting, vertex r is requested first, followed by the vertices in $\mathcal{S}$. When an element is requested in the online set cover instance, the corresponding vertex is requested in the incremental DS instance. Once the set cover instance is finished, the terminal t is requested, followed by t', and then vertices in U which were not yet requested. A visualization of the reduction appears in Figure 2. We prove the following lemma,

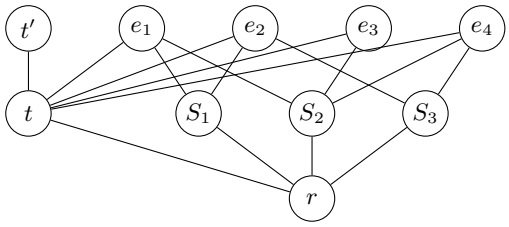

Figure 2 The resulting reduction from a set cover instance with 4 elements $E = \{e_1, e_2, e_3, e_4\}$ and 3 sets $S_1 = \{e_1, e_2\}$, $S_2 = \{e_1, e_3, e_4\}$ and $S_3 = \{e_2, e_4\}$. The vertices r and S_1, S_2, S_3 are requested first, and then when an element e_i is requested in the online set cover instance its corresponding vertex is requested in the reduction. After the online set cover instance is finished, t and t' are requested, followed by the rest of the vertices in E.

▶ **Lemma 30.** *There is an optimal solution for the reduced incremental instance that contains only elements belonging to* $\{r, t\} \cup \mathcal{S}$.

Proof. Let $\mathcal{I}$ be a solution to the reduced incremental instance. Vertex $r \in \mathcal{I}$, as r is the first requested vertex. Let $\mathcal{S}' = \mathcal{S} \cap \mathcal{I}$ and $U' = U \cap \mathcal{I}$. For each $u \in U'$ we choose some set $s_u \in \mathcal{S}$ that contains it. Define $\mathcal{S}_{U'} = \{s_u | u \in U'\}$, and let $\mathcal{I}' = \{r, t\} \cup \mathcal{S}' \cup \mathcal{S}_{U'}$.

The solution $\mathcal{I}'$ dominates all requested vertices: each $u \in U'$ is adjacent to $s_u \in \mathcal{I}'$, every set vertex is dominated by r, and r dominates itself. Since t' has only one neighbor t, any solution must contain t or t', so replacing whichever appears in $\mathcal{I}$ with t does not increase the solution size. Finally, $|\mathcal{S}_{U'}| \leq |U'|$ gives $|\mathcal{I}'| \leq |\mathcal{I}|$. ◀

We now proceed to proving that the reduction satisfies several properties.

▶ **Lemma 31.** *Our reduction satisfies the following properties:*

1. *The number of vertices in the resulting incremental instance is* $n + m + 3$.
2. *The optimal solution size in the incremental instance differs from the optimal set cover size by 2.*
3. *The reduction is online, meaning that each request in the online set cover instance is translated into a corresponding vertex in the online incremental DS instance.*
4. *There is an optimal solution that is connected.*

Proof. Property 1 is true by the way the reduction is defined.

The set cover solution selects a subset $\mathcal{S}' \subseteq \mathcal{S}$ to cover all the requested elements. The set $\{r, t\} \cup \mathcal{S}'$ is a valid solution to the reduction, since all vertices corresponding to sets in $\mathcal{S}$ are dominated by r, each requested element in U must be covered by an element in $\mathcal{S}'$, and all other elements in U are dominated by t. Thus a set-cover solution of size k yields an incremental solution of size $k+2$. By Lemma 30, let $\mathcal{I} \subseteq \{r, t\} \cup \mathcal{S}$ be the optimal solution to the reduced incremental instance, then $\mathcal{I} \backslash \{r, t\}$ is a feasible cover for the set cover instance of size $|\mathcal{I}| - 2$. Combining both directions proves the claim.

By Lemma 30, an optimal solution lies in $\{r, t\} \cup \mathcal{S}$. Since r and $\mathcal{S}$ are revealed before any element is requested, each subsequent element request corresponds exactly to the arrival of its matching vertex, and once t is requested it dominates all remaining elements of U without further additions, proving Property 3.

By Lemma 30, there exists an optimal solution containing only r, t, and elements from $\mathcal{S}$, where each vertex in $\mathcal{S}$ is adjacent to r. Since r is requested first and all other vertices in the optimal solution are connected to r, the partial solution remains connected at every step, thus proving Property 4. ◀

By combining these properties with known hardness results for online set cover [1, 19] the following results are established. The analysis is straightforward and is provided to the full version of the paper.

▶ **Theorem 5.** *There is no polynomial-time online algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log^2 \Delta)$ *unless* NP $\subseteq$ BPP*, even when the neighborhood of requested vertices is known.*

▶ **Theorem 6.** *There is no deterministic online algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log^2 \Delta / \log\log \Delta)$*, even when the neighborhood of requested vertices is known.*

5.2 Random Order Lower Bound

We present here a lower bound of $\Omega(\log n)$ on random order incremental DS. This proves Theorem 7. As a result, we can conclude that no algorithm has an $o(\log n)$ competitive ratio in the adversarial case, proving Theorem 8. We start with the result of Gupta et al. [12].

▶ **Theorem 32** (Theorem 5.1 in [12])**.** *There is an instance of set cover with n elements and $\Theta(n)$ sets, where any online algorithm for random order set cover is $\Omega(\log n)$-competitive.*

Let (S, E) be the set cover instance used in Theorem 32, where S is the collection of subsets and E is the ground set of elements. We construct an incremental Dominating Set instance as follows:

The vertex set includes all elements in E. For each subset $s_i \in S$, we create a clique $S_i = s_i^1, \ldots, s_i^{n^3}$. Each vertex in S_i is connected to all elements $e \in s_i$. Additionally, the clique $R = r^1, \ldots, r^{n^5}$ is added, where each $r^j \in R$ is connected to all vertices in every S_i.

Let η be the event that a vertex in R is requested first, and also for each S_i a vertex $s \in S_i$ is requested before all the vertices in E. We show that event η occurs with probability at least $1/2$.

▶ **Lemma 33.** *For sufficiently large n, $\mathbb{P}[\eta] \geq \frac{1}{2}$.*

The main idea is that $|R| = n^5$ is large relative to the $\Theta(n^4)$ remaining vertices, and each S_i's clique of size n^3 is large relative to the $\Theta(n)$ vertices of E that depend on it, so both events hold with high probability for sufficiently large n. The detailed calculation is provided in the full version of the paper. We now prove the lower bound.

▶ **Theorem 7.** *There is no online random-order algorithm for* incremental DS *or* incremental CDS *with competitive ratio* $o(\log \Delta)$.

Proof. Assume the event η occurs. Then some vertex in R is selected first, and for every $s_i \in S$, a vertex $s_i^j \in S_i$, which is already dominated by r, is requested before any $e \in E$.

When vertices in E are requested, each $e \in E$ is adjacent only to vertices s_i^j such that $e \in s_i$. To maintain the domination invariant, the algorithm must select such an s_i^j upon e's arrival. Selecting e itself would dominate only e, while selecting a neighboring s_i^j may also dominate future elements in E, so we may assume the algorithm avoids selecting e directly.

This simulates the random order set cover problem of Theorem 32, where the algorithm must cover elements of E using sets s_i. Given η, any algorithm incurs competitive ratio $\Omega(\log n)$. By Lemma 33 η occurs with constant probability, so the expected competitive ratio is $\Omega(\log n)$. The reduction produces a graph of size polynomial in n with maximum degree $\Delta = \Theta(n^5)$, so this establishes a lower bound of $\Omega(\log \Delta)$ for random-order incremental Dominating Set. ◀

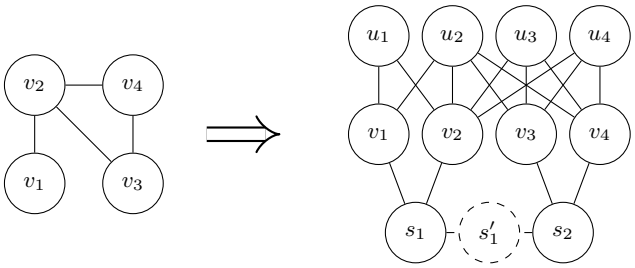

Figure 3 An example of the reduction presented in Section 5.3. The original graph is on the left, and the reduced instance on the right. Vertices in the reduced instance are requested from the bottom row to the top. The vertex s'_1 (dashed) is added for incremental CDS.

5.3 Hardness of Approximation

We construct a reduction from the offline Dominating Set (DS) problem to the incremental Dominating Set and incremental Connected Dominating Set (CDS) problems. The reduction preserves the number of vertices and maintains the maximum degree up to a constant factor. This construction is used to prove Theorem 9.

Let $G = (V, E)$ be an input graph with $V = \{v_1, \ldots, v_n\}$, maximum degree Δ, and minimum dominating set of size OPT. We define a new graph $G' = (V', E')$ by adding $S = \{s_1, \ldots, s_{n/\Delta}\}$ (wlog we assume that Δ divides n) and $U = \{u_1, \ldots, u_n\}$ to V. The full vertex set is $V' = V \cup S \cup U$, consisting of $n(2 + 1/\Delta)$ vertices.

We define the following edge sets in the new graph:

$$E_s = \{(s_i, v_{\Delta \cdot i+j}) \mid i \in [n/\Delta],\ j \in [\Delta]\},$$
$$E_u = \{(v_i, u_j) \mid (v_i, v_j) \in E\} \cup \{(v_i, u_i) \mid i \in [n]\}.$$

The final edge set is $E' = E_s \cup E_u$.

Intuitively, the set U is a duplicate of V as the edges are mirrored from V to U, and each vertex $v_i \in V$ is also connected to its counterpart $u_i \in U$. The vertices are requested in the following order: first the vertices in S, then those in V, and finally those in U.

The degree of vertex s_i is Δ, of v_i is bounded by $2\Delta + 1$ and of u_i is at most $\Delta + 1$. Thus, the maximum degree is $O(\Delta)$. In Figure 3 an example of the reduction is given.

We now prove that the size of any solution is almost preserved between the two models

▶ **Lemma 34.** *If D is a dominating set of G, then $D' = S \cup D$ is an incremental dominating set of G' with $|D'| \leq 3|D|$.*

Proof. If D is a dominating set of G, then the set $D' = S \cup D$ is an incremental dominating set in G'. Each vertex s_i is added to the solution immediately upon arrival. By construction, every vertex $v_i \in V$ is adjacent to at least one such s_j, ensuring that V is dominated once it arrives. For each vertex $u_i \in U$, if $v_i \in D$, then u_i is directly adjacent to $v_i \in D'$. Otherwise, since D is a dominating set in G, there exists a neighbor $v_j \in D$ of v_i, and in this case u_i is adjacent to $v_j \in D'$ via the edge $(v_j, u_i) \in E_u$.

The size of D' is

$$|D| + \frac{n}{\Delta} \leq |D| + 2\frac{n}{\Delta + 1} \leq 3|D|.$$

The first inequality holds because $\Delta > 0$, and the second follows from the standard lower bound on the size of any dominating set in a graph with maximum degree Δ. ◀

For the converse direction:

▶ **Lemma 35.** *If I is an incremental dominating set of G', then the set $D = \{v_i \mid v_i \in I \text{ or } u_i \in I\}$ is a dominating set of G and satisfies $|D| = O(|I|)$.*

Proof. Fix any $v_i \in V$.

- If $v_i \in D$, then v_i is trivially dominated.
- If $v_i \notin D$, but v_i is adjacent in G to some $v_j \in I$, then v_i is dominated via $v_j \in D$.
- Else, $u_i \in V'$ must be dominated in G', and since $u_i \notin I$, there must exist a vertex $v_j \in I$ such that $(v_j, u_i) \in E_u$. By construction of E_u, this implies $(v_i, v_j) \in E$, and $v_j \in D$.

In all cases, v_i is adjacent to some $v_j \in D$, so D is a dominating set of G.

To bound the size of D, note that $S \subseteq I$, and each $v_i \in D$ corresponds to at most two vertices in I. Therefore,

$$2|D| \geq |I| - |S| = |I| - \frac{n}{\Delta} \geq |I| - 2\frac{n}{\Delta + 1}.$$

Using $|D| \geq \frac{n}{\Delta+1}$, we get $4|D| \geq 2|D| + 2\frac{n}{\Delta+1} \geq |I|$. ◀

To extend the reduction to incremental CDS, we introduce an additional set of vertices $S' = \{s'_1, \ldots, s'_{n/\Delta-1}\}$ and add the edges (s_i, s'_i) and (s_i, s'_{i+1}) for all $i \in [n/\Delta - 1]$. These vertices are requested after S and must be selected to maintain connectivity. As a result, any valid incremental CDS solution must include the entire set S' to maintain connectivity. Moreover, the reduction preserves the asymptotic relationship between the size of the original dominating set and the size of the corresponding incremental CDS solution.

The reduction preserves the maximum degree up to a constant factor, and by Lemmas 34 and 35 the sizes of optimal solutions in G and G' are linearly related, a polynomial-time $o(\log \Delta)$-approximation algorithm for incremental DS would yield the same for classical Dominating Set, contradicting known hardness results [6]. This establishes the following.

▶ **Theorem 9.** Incremental DS *and* incremental CDS *are* NP-complete. *Furthermore, unless* NP $\subseteq$ BPP *there is no polynomial-time $o(\log \Delta)$ approximation algorithm for either problem.*

References

1 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009. doi:10.1137/060661946.

2 Maryam Atapour and Nasrin Soltankhah. On total dominating sets in graphs. *International Journal of Contemporary Mathematical Sciences*, 4(6):253–257, 2009.

3 Joan Boyar, Stephan J Eidenbenz, Lene M Favrholdt, Michal Kotrbčík, and Kim S Larsen. Online dominating set. *Algorithmica*, 81(5):1938–1964, 2019. doi:10.1007/S00453-018-0519-1.

4 Joan Boyar, Lene M Favrholdt, Michal Kotrbčík, and Kim S Larsen. Relaxing the irrevocability requirement for online graph algorithms. *Algorithmica*, 84(7):1916–1951, 2022. doi:10.1007/S00453-022-00944-W.

5 Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal–dual approach. *Theoretical Computer Science*, 3(2-3):93–263, 2007.

6 Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008. doi:10.1016/J.IC.2008.07.003.

7 Ernest J Cockayne, RM Dawes, and Stephen T Hedetniemi. Total domination in graphs. *Networks*, 10(3):211–219, 1980. doi:10.1002/NET.3230100304.

8 Bevan Das and Vaduvur Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Proceedings of ICC'97-International Conference on Communications*, volume 1, pages 376–380. IEEE, 1997. doi:10.1109/ICC.1997.605303.

9 Minati De, Sambhav Khurana, and Satyam Singh. Online dominating set and independent set. *CoRR*, 2021. arXiv:2111.07812.

10 Ding-Zhu Du and Peng-Jun Wan. *Connected dominating set: theory and applications*, volume 77. Springer Science & Business Media, 2012.

11 Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1998. doi:10.1007/PL00009201.

12 Anupam Gupta, Gregory Kehne, and Roie Levin. Random order online set cover is as easy as offline. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1253–1264. IEEE, 2022.

13 Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. *Mathematics of Operations Research*, 39(4):998–1011, 2014. doi:10.1287/MOOR.2014.0652.

14 Hovhannes A. Harutyunyan, Denis Pankratov, and Jesse Racicot. Online Domination: The Value of Getting to Know All Your Neighbors. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.57.

15 Stephen T Hedetniemi and Renu C Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. In *Annals of discrete mathematics*, volume 48, pages 257–277. Elsevier, 1991.

16 John N Hooker, Robert S Garfinkel, and CK Chen. Finite dominating sets for network location problems. *Operations Research*, 39(1):100–118, 1991. doi:10.1287/OPRE.39.1.100.

17 Koji M. Kobayashi. Improved Bounds for Online Dominating Sets of Trees. In *28th International Symposium on Algorithms and Computation (ISAAC 2017)*, volume 92 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ISAAC.2017.52.

18 D. (Dénes) König. *Theory of Finite and Infinite Graphs*. Birkhäuser, Boston, 1990. English translation of *Theorie der endlichen und unendlichen Graphen* (1950).

19 Simon Korman. On the use of randomization in the online set cover problem. *Weizmann Institute of Science*, 2:34, 2004.

20 Chung Laung Liu et al. *Introduction to combinatorial mathematics*, volume 181. McGraw-Hill New York, 1968.

21 Christine Markarian and Abdul-Nasser Kassar. Online deterministic algorithms for connected dominating set & set cover leasing problems. In *ICORES*, pages 121–128, 2020. doi:10.5220/0008866701210128.

22 Avraham Mehrez and Alan Stulman. The facility location problem when the underlying distribution is either dominated or dominating. *Zeitschrift für Operations Research*, 28:B157–B161, 1984. doi:10.1007/BF01920506.

23 EC Milner. The theory of graphs and its applications, 1964.

24 Oystein Ore. *Theory of graphs*. Providence, R.I. : American Mathematical Society, 1962.

25 Aravind Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999. doi:10.1137/S0097539796314240.

26 Jie Wu and Hailan Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, 18:13–36, 2001. doi:10.1023/A:1016783217662.

27 Jiguo Yu, Nannan Wang, Guanghui Wang, and Dongxiao Yu. Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey. *Computer Communications*, 36(2):121–134, 2013. doi:10.1016/J.COMCOM.2012.10.005.