The Siriaisa Algorithm for Minimum Independent Dominating Set

Frank Vega

Information Physics Institute, 840 W 67th St, Hialeah, FL 33012, USA

hexagon:2611.00043v2

Abstract

The Minimum Independent Dominating Set problem asks for a smallest maximal independent set of a graph. Unless P = NP it is not approximable within $n^{1-\varepsilon}$ for any $\varepsilon > 0$; it is NP-complete on cubic planar bipartite graphs and NP-hard to approximate within δB on bipartite graphs of maximum degree B. We present version 0.0.9 of Siriaisa, an ensemble of eighteen strategies that adapts the vertex cover algorithm Salvador and the independent set algorithm Esperanza to independent domination: its exact subproblems are forests, solved by a conditioned tree dynamic program, in place of the bipartite structures of Salvador; one base strategy splits every vertex of degree larger than $\lfloor \sqrt{n} \rfloor$ into unconnected copies; and the improvers include a weighted swap search. We prove that the output is an independent dominating set computed in $O(n + m)$ expected time, with ratio at most Δ, optimal on forests, and with ratio at most $\sqrt{n}$ on three classes of graphs. The degree-splitting strategy computes a dominating set of size at most $\lfloor \sqrt{n} \rfloor (i(G) + s(G))$ on every graph, for an explicit splitting excess $s(G)$, and the output satisfies the same bound when the decoded set is independent. The excess is at most $n(\sqrt{n} + 1)$, largest on complete graphs, zero for $n \geq D^2$ on every family of bounded degree D, among them the NP-hard and UG-hard bounded-degree families, and $O(\sqrt{n})$ on planar graphs; on complete multipartite, crown, wheel and double-star graphs the output is optimal. We conjecture the ratio $O(\sqrt{n})$ on all graphs; a proof would imply P = NP. In a reduction from satisfiability, every independent dominating set of the graph of a formula with N variables and t copies of every clause, t at least the largest clause width W, is a satisfying assignment, with N vertices, or has at least $t + N$ vertices; with $t = W$, every independent dominating set of ratio smaller than $1 + W/N$ is a satisfying assignment. This proves the $n^{1-\varepsilon}$ inapproximability directly from satisfiability, without a gap from the PCP theorem. Through a reduction of Chlebík and Chlebíková, with an explicit edge gadget that turns a graph H into a graph $h(H)$ with $i(h(H)) = \tau(H) + 2|E(H)|$, where $\tau(H)$ is the minimum vertex cover size, every output on $h(H)$ decodes into a vertex cover of H; we state the conditional consequences under the Unique Games Conjecture, for which a proof has been reported but not yet peer reviewed, on general graphs and, through the result of Austrin, Khot and Safra, on bounded-degree graphs. In reproducible experiments the output is optimal on 9 999 of 10 000 graphs with an exact integer-program optimum, on all 95 of 112 bipartite graphs whose optimum was proved, on all 18 instances of the set cover reduction, and on 2 of the 30 instances of the vertex cover reduction with a proved optimum; on ten Khot–Regev YES instances the decoded vertex cover has ratio between 1.25 and 1.8665. We also report the results on the 68 complements of the DIMACS clique benchmark.

Keywords: Minimum Independent Dominating Set, Minimum Maximal Independent Set, Approximation Algorithm, Forest Dynamic Programming, Local Search, Bipartite Graphs, Unique Games Conjecture, P versus NP
2020 MSC: 05C69, 68Q25, 68W25, 90C27

1. Introduction

An *independent dominating set* of an undirected graph $G = (V, E)$ is a set $D \subseteq V$ in which no two vertices are adjacent and every vertex of $V \setminus D$ has a neighbour in D; equivalently, D is a maximal independent set. The Minimum Independent Dominating Set problem (MIDS) asks for such a set of minimum size, denoted $i(G)$. We write $n = |V|$, $m = |E|$, $N(v)$ and $N[v] = N(v) \cup \{v\}$ for the open and closed neighbourhoods, $\deg(v) = |N(v)|$, Δ for the maximum

Email address: vega.frank@gmail.com (Frank Vega)

degree, $\tau(G)$ for the minimum vertex cover size and $\alpha(G)$ for the independence number. The ratio of a returned independent dominating set S is $\rho = |S|/i(G)$.

MIDS is among the hardest graph problems to approximate. Irving proved that, unless P = NP, no polynomial-time algorithm approximates it within any constant factor [1], and Halldórsson proved that it cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless P = NP [2]; Damian-Iordache and Pemmaraju proved that it is NP-complete on circle graphs and, unless P = NP, not approximable within n^{ε} for any $\varepsilon < 1$ on circle graphs with n vertices [3]. The problem remains NP-hard on bipartite graphs: Loverov and Orlovich proved that it is NP-complete already on cubic planar bipartite graphs [4]; Manlove proved it NP-hard on cubic planar graphs [5], and Bazgan, Brankovic, Casel and Fernau derived from Manlove's reduction that it is APX-hard on cubic graphs [6]. On graphs of maximum degree at most a constant B, every maximal independent set approximates MIDS within B, and the problem is APX-complete [7]; on cubic graphs, Alimonti and Calamoneri gave a polynomial-time algorithm with ratio 1.923, and the ratio 2 on graphs of maximum degree at most 3 [8], while Alimonti and Kann proved APX-completeness on cubic graphs of vertex cover, independent set and dominating set [9]. It is also hard to approximate on graphs of bounded degree, including bipartite ones: Chlebík and Chlebíková proved that there are absolute constants $\delta > 0$ and B_0 such that for every $B \geq B_0$ MIDS is NP-hard to approximate within δB on graphs of maximum degree at most B, and that the same holds on bipartite graphs; for small degrees they obtained the NP-hard factors $1 + \frac{1}{680}$ for $\Delta \leq 3$, and $1 + \frac{1}{293}$ and $1 + \frac{1}{151}$ for bipartite graphs with $\Delta \leq 4$ and $\Delta \leq 5$ [10]. These results hold without any conjecture.

Exact algorithms and heuristics. A minimum independent dominating set can be computed in $O(1.4423^n)$ time by enumerating the maximal independent sets; Gaspers and Liedloff gave a branch-and-reduce algorithm that runs in $O(1.3569^n)$ time, with a lower bound $\Omega(1.3247^n)$ on its worst-case running time [11], and Bourgeois, Della Croce, Escoffier and Paschos gave further exact algorithms [12]. Laforest and Phan proposed an exact algorithm and a greedy heuristic [13]. Haraguchi proposed a local search whose move, the k-swap, drops k vertices of a maximal independent set and adds any number of vertices: an improved solution in the neighbourhood is found, or shown not to exist, in $O(n\Delta)$ time for $k = 2$, and in $O(n\Delta^3)$ time for $k = 3$ when the solution is minimal with respect to the 2-swap; a metaheuristic that iterates this search was evaluated on DIMACS graphs [14]. Goddard and Henning survey independent domination in graphs [15]. For the Minimum Dominating Set problem, in which the set need not be independent, Lenzen, Pignolet and Wattenhofer gave a distributed algorithm that computes a constant-factor approximation on planar graphs in a constant number of rounds [16].

Under the Unique Games Conjecture (UGC) [17], Khot and Regev proved that Minimum Vertex Cover is hard to approximate within $2 - \varepsilon$ for every $\varepsilon > 0$ [18]. In October 2026 the OpenAI Math Team released a manuscript with Lean artifacts reporting a proof of UGC in the NP-hardness form that the theorem of Khot and Regev requires [19]; at the time of writing it has not been peer reviewed. If it holds, the hardness result of Khot and Regev holds unless P = NP. Through a reduction of Chlebík and Chlebíková (Section 7), every vertex cover instance H becomes an MIDS instance $h(H)$ whose optimum is $\tau(H)$ plus a known constant, so the instances of Khot and Regev give MIDS instances on which Siriaisa can be tested as a vertex cover algorithm.

This paper describes version 0.0.9 of Siriaisa [20], a Python implementation of an ensemble of eighteen strategies. The ensemble adapts the seventeen strategies of the vertex cover algorithm Salvador [21, 22] and the phases of the independent set algorithm Esperanza [23] to independent domination. Where Salvador builds a bipartite structure and solves vertex cover on it exactly by Kőnig's theorem, Siriaisa builds a *forest* and solves MIDS on it exactly by a linear-time tree dynamic program; MIDS is NP-hard on bipartite graphs [4], so the bipartite structures of Salvador carry no exact solver for MIDS. One base strategy splits every vertex of degree larger than $\lfloor \sqrt{n} \rfloor$ into copies, so that the auxiliary graph has maximum degree at most $\lfloor \sqrt{n} \rfloor$.

Contributions. The contributions of this paper are:

- The algorithm (Section 3): twelve base strategies, five improvers and a final refinement, an exact conditioned forest solver, induced-forest layers, a degree-splitting strategy at $\lfloor \sqrt{n} \rfloor$, and a weighted swap search, together with the proof that the output is an independent dominating set (Theorem 4.3).
- The running time $O(n + m)$, in expectation, for the default parameters (Theorem 5.2).
- Proved bounds (Section 6): the ratio is at most Δ on every graph (Corollary 6.2); the output is optimal on every forest (Theorem 6.4); the ratio is at most $\sqrt{n}$ on three classes (Theorem 6.5); and the degree-splitting strategy

yields a dominating set of size at most $\lfloor \sqrt{n} \rfloor (i(G) + s(G))$ on every graph, for an explicit splitting excess $s(G)$, with the same bound for the output when the decoded set is independent (Theorem 6.8); the excess is largest on complete graphs, of order $n^{3/2}$, it vanishes for $n \geq D^2$ on every family of bounded degree D, among them the NP-hard and UG-hard bounded-degree families, it is $O(\sqrt{n})$ on planar graphs, and on complete multipartite, crown, wheel and double-star graphs the output is optimal (Propositions 6.9, 6.10 and 6.12, Table 4).

- The conjecture that the ratio is $O(\sqrt{n})$ on every graph, with the theorem that a proof would imply P = NP (Conjecture 6.17, Theorem 6.19), and the graphs on which its bound is proved (Proposition 6.18).
- A reduction from satisfiability (Section 8): the graph of a formula with t copies of every clause, t at least the largest clause width, whose independent dominating sets are the satisfying assignments, with one vertex per variable, or have at least t more vertices (Theorem 8.2, Corollary 8.3); it proves that MIDS is not approximable within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless P = NP, directly from satisfiability and without a gap from the PCP theorem (Theorem 8.4).
- The vertex cover reduction of Chlebík and Chlebíková with an explicit gadget and the identity $i(h(H)) = \tau(H) + 2|E(H)|$ (Proposition 7.1), the decoding of every independent dominating set into a vertex cover (Lemma 7.2), and the conditional consequences for vertex cover approximation, on general graphs under the hardness of Khot and Regev and on bounded-degree graphs under that of Austrin, Khot and Safra (Theorem 7.3, Corollary 7.4).
- Reproducible experiments (Section 9): 10 000 structured and random graphs with an exact integer-program optimum, with four baseline heuristics on the same graphs (Section 9.3); 112 bipartite graphs; 55 instances of the two reductions of Chlebík and Chlebíková, ten of them built from Khot–Regev YES instances; and the 68 complements of the DIMACS clique benchmark graphs.

Section 2 records the code metadata, Section 3 describes the algorithm, Section 4 proves correctness, Section 5 the running time and Section 6 the bounds. Section 7 treats the reductions and the conditional consequences, Section 8 the reduction from satisfiability, Section 9 reports the experiments and Section 10 concludes. Table 1 lists the notation.

Table 1: Notation.

Symbol	Meaning
$G = (V, E)$, n, m	the input graph, its number of vertices and of edges
$N(v)$, $N[v]$, $\deg(v)$	open and closed neighbourhood and degree of v
Δ, $\delta(G)$	maximum and minimum degree
$i(G)$	minimum size of an independent dominating set
$\alpha(G)$, $\tau(G)$	independence number, minimum vertex cover size
S, $\rho = \lvert S \rvert / i(G)$	the output of Algorithm 1 and its ratio
$c_1, \ldots, c_{18}$	the candidates of the strategies (Table 3)
in, pre, force, P	flags and penalty of the forest solver (Section 3.2)
IN, DOM, FREE	states of a vertex in the forest solver (Section 3.2)
$u(c)$, $a(c)$, $\delta(v)$	terms of the recurrences of the forest solver (Section 3.2)
$k = \max(1, \lfloor \sqrt{n} \rfloor)$	degree threshold of the splitting strategy c_{13}
$B = \{v : \deg(v) > k\}$	the split vertices (Section 6)
$s(G)$	splitting excess, $\sum_{v \in B} (\lceil \deg(v)/k \rceil - 1)$
H (Section 6), S_0	copy graph of Algorithm 5 and decoded set
H, m_H, $h(H)$ (Section 7)	a vertex cover instance, its number of edges, its VC-IDS graph
φ, N, q, W, Λ	a formula, its numbers of variables and clauses, its largest clause width, its number of literal occurrences
$G_{\varphi,t}$	graph of φ with t copies of every clause (Definition 8.1)

2. Research Data

The algorithm is implemented in the Python package `siriaisa` [20]. The version described in this paper is v0.0.9. Table 2 lists the code metadata.

Nr.	Code metadata description	Metadata
C1	Current code version	v0.0.9
C2	Permanent link to code repository	https://github.com/frankvegadelgado/mids
C3	Package name	`siriaisa`
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Languages, tools, and services used	Python ≥ 3.12, NetworkX, NumPy, SciPy
C7	Main entry points	`iris`, `batch_iris`, `test_iris`

Table 2: Code metadata of the `siriaisa` package. SciPy is used only by the experiment scripts, for the exact integer programs.

3. The Siriaisa Algorithm

The input graph is converted once into an index graph adj on the vertices $0, \ldots, n-1$, and self-loops are removed. Every strategy returns a set that is turned into an independent dominating set by the repair-and-complete routine of Section 3.1; the strategies are computed in their numbering order, the base strategies c_1–c_{13} first and the improvers c_{14}–c_{18} last, and the output is c_{18}. Table 3 lists them, and Algorithm 1 gives the main loop.

Table 3: The eighteen strategies of Siriaisa 0.0.9. "Source" names the strategy of Salvador [21] or the phase of Esperanza [23] that it adapts.

Name	Strategy	Source
c_1	spanning-tree peeling: maximum spanning forest, tree DP, peel, repeat	—
c_2	endpoints of the NetworkX minimum maximal matching	Salvador c_2
c_3	greedy: the undominated vertex that dominates most undominated vertices	Salvador c_3
c_4	star-forest reduction solved by the tree DP	Salvador c_4
c_5	Min-to-Min	Salvador c_5
c_6	primal–dual on the domination linear program	Salvador c_6
c_7	BFS spanning forest solved by the tree DP	Salvador c_7
c_8	DFS spanning forest solved by the tree DP	Salvador c_8
c_9	induced-forest layers, each solved by the tree DP, in several orders	Salvador c_9
c_{10}	first forest layer re-solved exactly, the rest fixed from the best of c_1–c_9	Salvador c_{10}, c_{12}
c_{11}	feedback-core peeling and the conditioned forest DP	Salvador c_{11}
c_{12}	dynamic minimum-degree (Caro–Wei) greedy	Esperanza baseline
c_{13}	degree splitting at $\lfloor \sqrt{n} \rfloor$, decoding, repair	—
c_{14}	union of c_2–c_8, repaired by votes	Salvador c_{13}
c_{15}	descent and weighted swap search from the best of c_1–c_{14}	Salvador c_{14}, Esperanza
c_{16}	reduce-and-search: swap search and forest large-neighbourhood search	Salvador c_{15}
c_{17}	consensus forcing over c_1–c_{16}	Salvador c_{16}
c_{18}	refinement of every candidate, a longer search on the best	Salvador c_{17}, Esperanza

Algorithm 1 FindIndependentDominatingSet(G)

Require: Undirected graph $G = (V, E)$
Ensure: An independent dominating set of G

1: remove self-loops; build the index graph adj; $n \leftarrow |V|$, $m \leftarrow |E|$
2: $c_1 \leftarrow$ Complete(SpanningTreePeeling(adj)); $c_2 \leftarrow$ Complete(endpoints of `min_maximal_matching`); $c_3 \leftarrow$ Complete($\emptyset$)
3: $c_4, c_7, c_8 \leftarrow$ the star, BFS and DFS spanning forests solved by Algorithm 3 and completed
4: $c_5 \leftarrow$ MinToMin; $c_6 \leftarrow$ PrimalDual
5: $(c_9, \mathcal{L}) \leftarrow$ ForestLayers(adj) ▷ Algorithm 4; $\mathcal{L}$ the layers
6: $c_{10} \leftarrow$ ResolveLayer(best of $c_1..c_9, L_0$) if not larger
7: $c_{11} \leftarrow$ FeedbackForest(adj, best of $c_1..c_{10}$); $c_{12} \leftarrow$ CaroWei; $c_{13} \leftarrow$ DegreeSplit(adj) ▷ Algorithm 5
8: $LB \leftarrow \sum_{\text{components } C} \lceil n_C/(\Delta_C + 1) \rceil$ ▷ Lemma 6.3
9: $c_{14} \leftarrow$ Complete($c_2 \cup \cdots \cup c_8$, priority = votes)
10: $c_{15} \leftarrow$ ImproveIDS(best of $c_1..c_{14}$) ▷ Algorithm 6
11: $c_{16} \leftarrow$ three rounds of ImproveIDS followed by ForestLNS($\mathcal{L}$), from the best of $c_1..c_{15}$, each kept if not larger
12: $c_{17} \leftarrow$ ConsensusForce($c_1, \ldots, c_{16}$)
13: $R_j \leftarrow$ ImproveIDS(ForestLNS($c_j, \mathcal{L}$)) for every distinct c_j, $j \le 17$
14: $c_{18} \leftarrow$ Complete(ForestLNS(ImproveIDS($\min_j R_j$), $\mathcal{L}$))
15: verify every c_j; **return** c_{18}

Every search of lines 10, 11, 13 and 14 stops as soon as its best set has at most LB vertices.

3.1. Repair and completion

Algorithm 2 turns any seed into an independent dominating set; it is the counterpart of the pruning of Salvador and of the routine MaximizeSolution of Salvador and Esperanza. All sorts are stable counting sorts on integer keys.

Algorithm 2 Complete(adj, P, π)

Require: Index graph, seed $P \subseteq V$, optional integer priority π
Ensure: An independent dominating set S

$S \leftarrow P$; $\mathrm{conf}(v) \leftarrow |N(v) \cap S|$ for $v \in S$
for $v \in S$ with $\mathrm{conf}(v) > 0$, by decreasing conf, then increasing π, then increasing degree **do**
  if $v \in S$ and $\mathrm{conf}(v) > 0$ **then** remove v from S and decrement conf of its neighbours in S
  end if
end for
$U \leftarrow$ the vertices with no vertex of S in their closed neighbourhood; $g(v) \leftarrow |N[v] \cap U|$ for $v \in U$
while $U \neq \emptyset$ **do** ▷ bucket queue on g
  take $v \in U$ of largest $g(v)$; $S \leftarrow S \cup \{v\}$; remove $N[v] \cap U$ from U and update g on their neighbours
end while
return S

3.2. The conditioned forest solver

Let F be an induced forest of G and let the rest of G be fixed. Every vertex v of F carries three flags: $\mathrm{in}(v)$ (v may be selected; false when v has a selected neighbour outside F), $\mathrm{pre}(v)$ (v is already dominated from outside F) and $\mathrm{force}(v)$ (v is the only possible dominator of some vertex outside F). With a penalty $P > |F|$, Algorithm 3 minimises

$$\Phi(X) = |X| + P \cdot \left|\{v \in V(F) : v \notin X \cup N(X),\ \neg\mathrm{pre}(v)\}\right| + P \cdot \left|\{v : \mathrm{force}(v),\ v \notin X\}\right|$$

over the sets $X \subseteq V(F)$ that are independent and contain only vertices with $\mathrm{in}(v)$. Rooting every tree, the states of a vertex v are IN ($v \in X$), DOM ($v \notin X$, dominated by a child or from outside) and FREE ($v \notin X$, not dominated by its

children). With $u(c) = 0$ if $\mathrm{pre}(c)$ and $u(c) = P$ otherwise, and $a(c) = \min(\mathrm{DOM}(c), \mathrm{FREE}(c) + u(c))$, the recurrences over the children c of v are

$$\mathrm{IN}(v) = \begin{cases} 1 + \sum_c \min(\mathrm{DOM}(c), \mathrm{FREE}(c)) & \text{if } \mathrm{in}(v), \\ \infty & \text{otherwise,} \end{cases}$$
$$\mathrm{FREE}(v) = \textstyle\sum_c a(c) + P \cdot [\mathrm{force}(v)],$$
$$\mathrm{DOM}(v) = \textstyle\sum_c \min(\mathrm{IN}(c), a(c)) + \delta(v) + P \cdot [\mathrm{force}(v)],$$

where $[\cdot]$ is 1 if the condition holds and 0 otherwise. Each value is the smallest Φ on the subtree of v for the state of v, without the penalty of v itself when v is FREE. The term $u(c)$ is the penalty of a child that stays undominated: a FREE child c of a vertex $v \notin X$ has no neighbour in X, and it costs P unless $\mathrm{pre}(c)$. The term $a(c)$ is the cheapest value of a child $c \notin X$ whose parent is not in X. In $\mathrm{IN}(v)$ no child is in X and every child is dominated by v, so a FREE child costs no penalty. In $\mathrm{FREE}(v)$ no child is in X. In $\mathrm{DOM}(v)$ every child takes its cheapest state, of value $\min(\mathrm{IN}(c), a(c))$, and the term $\delta(v)$ is the additional cost that makes v dominated: $\delta(v) = 0$ if $\mathrm{pre}(v)$, or if the cheapest state of some child is IN, that is, $\mathrm{IN}(c) \le a(c)$; otherwise one child has to be moved from its cheapest state to IN, and $\delta(v) = \min_c(\mathrm{IN}(c) - a(c))$ is the smallest increase, attained by the *forced child*; if $\neg\mathrm{pre}(v)$ and v has no child with $\mathrm{IN}(c) < \infty$, in particular if v is a leaf, then $\mathrm{DOM}(v) = \infty$. A root r takes $\min(\mathrm{IN}(r), \mathrm{DOM}(r), \mathrm{FREE}(r) + u(r))$, and back-pointers recover X.

Example. Let F be the path ℓ–p–r–q rooted at r, with neutral flags and $P = 5$; the children of r are p and q, and ℓ is the child of p. For the leaves, $\mathrm{IN}(\ell) = \mathrm{IN}(q) = 1$, $\mathrm{FREE}(\ell) = \mathrm{FREE}(q) = 0$, $\mathrm{DOM}(\ell) = \mathrm{DOM}(q) = \infty$ and $a(\ell) = a(q) = \min(\infty, 0 + 5) = 5$. For p, $\mathrm{IN}(p) = 1 + \min(\infty, 0) = 1$, $\mathrm{FREE}(p) = a(\ell) = 5$, and $\mathrm{DOM}(p) = \min(\mathrm{IN}(\ell), a(\ell)) + \delta(p) = 1 + 0 = 1$, because $\mathrm{IN}(\ell) = 1 \le a(\ell) = 5$; hence $a(p) = \min(1, 5 + 5) = 1$. For the root, $\mathrm{IN}(r) = 1 + \min(\mathrm{DOM}(p), \mathrm{FREE}(p)) + \min(\mathrm{DOM}(q), \mathrm{FREE}(q)) = 1 + 1 + 0 = 2$, $\mathrm{FREE}(r) = a(p) + a(q) = 1 + 5 = 6$, and $\mathrm{DOM}(r) = \min(1, 1) + \min(1, 5) + \delta(r) = 2 + 0 = 2$. The root takes $\min(2, 2, 6 + 5) = 2$. With the state IN for r, the child p takes DOM and its child ℓ takes IN, and q takes FREE, which gives $X = \{r, \ell\}$; with the state DOM for r, both children take IN, which gives $X = \{p, q\}$. Both are minimum independent dominating sets of the path, and $i(F) = 2$.

Algorithm 3 ConditionedForestIDS(F, in, pre, force, P)

Require: An induced forest F, flags, penalty $P > |V(F)|$
Ensure: A minimiser X of Φ
1: **for** every tree of F **do**
2: root it; compute a BFS order and the children lists
3: **for** v in reverse BFS order **do** compute $\mathrm{IN}(v), \mathrm{DOM}(v), \mathrm{FREE}(v)$ and the forced child
4: **end for**
5: choose the state of the root; assign the states top-down by the recorded choices; add the IN vertices to X
6: **end for**
7: **return** X

With neutral flags (in ≡ true, pre ≡ force ≡ false) the algorithm returns a minimum independent dominating set of F (Lemma 4.2). ResolveLayer(S, L) fixes every vertex outside the induced forest L from a set S, sets the flags accordingly (a vertex outside L, not in S, without a neighbour in S outside L and with exactly one selectable neighbour in L forces that neighbour), solves L by Algorithm 3 and completes the result by Algorithm 2. ForestLNS applies ResolveLayer to every layer of $\mathcal{L}$, keeping a new set when it is not larger, for at most two passes and a work budget of $40(n + m)$.

3.3. Base strategies

c_1: spanning-tree peeling. Round by round, on the residual graph R (initially G), a maximum spanning forest of R with edge weights $\deg_R(u) + \deg_R(v)$ is computed by Kruskal's algorithm on the edges bucket-sorted by weight, with a union-find structure using union by size and path halving [24]; the forest is solved exactly by Algorithm 3; a maximal G-independent subset of the forest optimum, larger residual degree first, is kept; the kept vertices and their neighbours are removed. A round on a residual graph with n_R vertices and m_R edges is charged $4(n_R + m_R)$ work units, and the

rounds stop at a work budget of $40(n + m)$. Every pointer step of a find operation is charged to a second counter with the limit $320(n + m)$, tested after every edge; when it is exceeded, Kruskal's algorithm stops adding edges, the round is finished on the forest built so far, and it is the last round. The vertices left are completed by Algorithm 2.

c_2–c_8. c_2 completes the endpoints of the NetworkX [25] minimum maximal matching; every unmatched vertex has all its neighbours matched, so the endpoints and the isolated vertices dominate G. c_3 is Algorithm 2 from the empty seed. c_4 solves the spanning forest in which every vertex points to the vertex of largest pair (degree, index) in its closed neighbourhood; these pairs strictly increase along the pointers, so they form a forest. c_5 (Min-to-Min) repeatedly takes the undominated vertex of smallest residual degree and selects the free vertex of its closed neighbourhood that dominates most undominated vertices. c_6 processes the undominated vertices by increasing degree, raises the dual variable of v in the linear program $\min \sum_u x_u$, $\sum_{u \in N[v]} x_u \geq 1$, until a free vertex of $N[v]$ becomes tight, and selects the tight free vertex of largest degree. c_7 and c_8 solve a BFS and a DFS spanning forest, rooted by decreasing degree. The optimum of a spanning forest dominates G but need not be independent in G; Algorithm 2 repairs it.

c_9*: induced-forest layers.* Algorithm 4 splits every connected component into layers, each an induced forest, and solves the layers in several orders. The trees of the layer under construction are kept in a union-find structure with union by size and path halving. A pass over the remaining vertices is charged their number plus the sum of their degrees, within the budget $8(n + m)$, and every pointer step of a find operation is charged to a second counter with the limit $64(n + m)$, tested after every find; when either runs out, the layer under construction is closed with the vertices accepted so far, and the vertices not placed are left to the greedy completion.

Algorithm 4 ForestLayers(adj)

Require: Index graph; layering budget $8(n + m)$; find budget $64(n + m)$; at most 6 orders per component
Ensure: An independent dominating set and the layers $\mathcal{L}$

for every connected component C **do**
  $R \leftarrow V(C)$ sorted by increasing degree; $t \leftarrow 0$
  while $R \neq \emptyset$ and the budgets allow **do**
    $L_t \leftarrow \emptyset$
    for $v \in R$ **do** add v to L_t if its neighbours in L_t lie in pairwise different trees of L_t (union-find)
    end for
    $R \leftarrow R \setminus L_t$; $t \leftarrow t + 1$
  end while
  for every order σ of the layers (all of them if $t! \leq 6$, otherwise the natural and the reverse order and their rotations, at most 6) **do**
    $X \leftarrow \emptyset$
    for L in the order σ **do** $X \leftarrow X \cup$ ConditionedForestIDS($L \setminus (X \cup N(X))$) with neutral flags
    end for
    complete X greedily on the vertices still undominated; keep the smallest X
  end for
end for

c_{10} *and* c_{11}. c_{10} is ResolveLayer of the first layer of $\mathcal{L}$ with the rest fixed from the best of c_1–c_9, kept when it is not larger. c_{11} peels a feedback vertex set (the *core*) by the reductions of Salvador's c_{11} on a multigraph: a vertex of residual degree at most 1 lies on no cycle and is removed; a vertex v of residual degree 2 with distinct neighbours a, b is bypassed, replaced by an edge a–b; when both edges of a vertex of degree 2 go to the same b, the vertex b goes to the core; otherwise the vertex of largest residual degree goes to the core. The vertices outside the core induce a forest of G, which is solved by ResolveLayer for three choices on the core (the choice of the best of c_1–c_{10}, no core vertex, and the greedy solution of the core); the smallest set is kept.

c_{12}. c_{12} is the dynamic minimum-degree greedy of Esperanza, which repeatedly selects a vertex of smallest residual degree and deletes its closed neighbourhood [26, 27].

c_{13}*: degree splitting.* Algorithm 5 replaces every vertex of degree larger than $k = \lfloor \sqrt{n} \rfloor$ by unconnected copies, each joined to at most k of its neighbours, solves the copy graph and maps the solution back.

Algorithm 5 DegreeSplit(adj)

Require: Index graph on n vertices
Ensure: An independent dominating set of G
$k \leftarrow \max(1, \lfloor \sqrt{n} \rfloor)$; $H \leftarrow G$; owner$(v) \leftarrow v$
for $u \in V$ by decreasing degree (counting sort) **do**
  if $\deg_H(u) > k$ **then**
    let $w_1, \ldots, w_d$ be the neighbours of u in H; remove u from H
    for $j = 1, \ldots, \lceil d/k \rceil$ **do** add a node u_j with owner$(u_j) = u$, joined to $w_{(j-1)k+1}, \ldots, w_{\min(jk,d)}$
    end for
  end if
end for
for each inner solver $A \in \{\text{Complete}(\emptyset), \text{MinToMin}, \text{ForestLayers}\}$ **do**
  $S_H \leftarrow A(H)$; cnt$(v) \leftarrow |\{x \in S_H : \text{owner}(x) = v\}|$; $S_0 \leftarrow \{v : \text{cnt}(v) > 0\}$
  $S_A \leftarrow \text{Complete}(\text{adj}, S_0, \pi = \text{cnt})$ ▷ repair with pruning
end for
return the smallest S_A

3.4. Improvers and the weighted swap search

c_{14} completes the union of c_2–c_8, discarding first the vertices chosen by fewer of them. c_{17} (consensus forcing) runs three rounds: round k takes the vertices chosen by at least a fraction $q_k \in \{1, 0.9, 0.75\}$ of the sets in the pool, keeps an independent subset of them (most votes first), removes them with their neighbours, solves the residual graph by the best of Algorithm 2, Min-to-Min and Algorithm 4, and adds the result to the pool.

Algorithm 6 is the local search used by c_{15}, c_{16} and c_{18}. Its first phase is a descent: a *move* at a vertex $u \notin S$ inserts u, removes the $k = |N(u) \cap S|$ neighbours of u in S, and refills the vertices left undominated, which all lie in the neighbourhoods of the removed vertices, by the greedy of Algorithm 2 restricted to them; the move is kept when S shrinks, which requires $k \geq 2$. Its second phase is a weighted swap search on a set S of fixed size k that may leave vertices undominated and contain conflicting edges. With weights $w(v) \geq 1$ on the vertices and $w(e) \geq 1$ on the edges, it minimises

$$\text{cost}(S) = \sum_{v:\, N[v] \cap S = \emptyset} w(v) + \sum_{e \subseteq S} w(e),$$

so that every vertex, not only an undominated one, may enter S. When $\text{cost}(S) = 0$, S is an independent dominating set of size k; it is recorded and one vertex is removed to try $k - 1$.

Algorithm 6 ImproveIDS(adj, S_0, B, σ)

Require: An independent dominating set S_0; work budget $W = \max(B(n+m), F)$; seed σ; lower bound LB
Ensure: An independent dominating set S^* with $|S^*| \le |S_0|$

$S \leftarrow$ descent from S_0 (moves at vertices with $k \ge 2$, chained, within the budget); $S^* \leftarrow S$
for segment $i = 1, \ldots, 4$, each with a quarter of the remaining work and the weight cap $\theta_i \in (10, 100, 30, 300)$ rotated by σ **do**
    $S \leftarrow S^*$ minus the vertex whose removal increases the cost least; all weights $\leftarrow 1$
    while work remains and $|S^*| > LB$ **do**
        remove the vertex of S of largest score among 50 sampled ones (never the vertex added last)
        take a random undominated vertex v; add the vertex of $N[v]$ of largest score allowed by configuration checking
        add 1 to the weight of every undominated vertex and of every conflicting edge; halve all weights when their mean exceeds θ_i
        if cost$(S) = 0$ **then** $S^* \leftarrow S$; remove from S the vertex whose removal increases the cost least
        end if
    end while
end for
return S^* after a last descent

The score of a vertex is the decrease of the cost when it changes side; configuration checking forbids a removed vertex to re-enter before one of its neighbours has changed side. All scores, the domination counts and the sets of violations are updated incrementally in time proportional to the degrees of the two-hop neighbourhood of the changed vertex, and every operation is charged to a work counter. The default budgets are $B = 10$ for c_{15}, 5 per round for c_{16}, 3 per candidate and 30 for the final search of c_{18}, with the constant floors $F = \min(2 \cdot 10^6,\ 10(n+m)^2)$ and, for the final search, $F = \min(24 \cdot 10^6,\ 100(n+m)^2)$; the seed is 20261031.

4. Correctness of the Algorithm

Lemma 4.1 (Repair and completion)**.** *For every seed P, Algorithm 2 returns an independent dominating set of G, in $O(n + m + \max \pi)$ time.*

Proof. After the repair loop no vertex of S has a neighbour in S: a vertex is examined with its current conflict count and is removed whenever it is positive, and removals only decrease the counts of the others, so a vertex that keeps a positive count when examined is removed, and a vertex examined with count 0 keeps count 0 because no vertex is added during the loop. In the completion loop a vertex is added only when it lies in U, i.e. when no vertex of its closed neighbourhood is in S, so S stays independent, and the loop ends when $U = \emptyset$, i.e. when every vertex is dominated. The counting sorts cost $O(n + \Delta + \max \pi)$ and every vertex leaves U once, with one scan of its neighbourhood, so the bucket queue costs $O(n + m)$. □

Lemma 4.2 (Exact conditioned forest solver)**.** *Algorithm 3 returns a minimiser of Φ in $O(\sum_{v \in V(F)} \deg_G(v))$ time. With neutral flags and $P > |V(F)|$, the minimiser is a minimum independent dominating set of F.*

Proof. For a rooted subtree T_v, IN(v), DOM(v) and FREE(v) are the minimum of the part of Φ carried by T_v over independent sets $X \cap T_v$ in which v is selected, dominated by a child or from outside, or not dominated by its children, respectively; the penalty of v itself in the state FREE is paid by its parent ($u(c)$) unless the parent is selected, and the penalty of a forced unselected vertex is added in the states DOM and FREE. The recurrences enumerate exactly these cases: a selected vertex excludes selected children and dominates them; a dominated vertex needs one selected child unless it is pre-dominated, and the cheapest such child is forced when none is chosen freely; a vertex with an unselected parent pays $u(c)$ in the state FREE. Induction over the reverse BFS order gives the minimum at the root, and the top-down reconstruction follows the recorded minimising choices. Each vertex scans its neighbourhood a constant number of times. With neutral flags, every independent dominating set X of F has $\Phi(X) = |X| \le |V(F)| < P$, so a minimiser has no penalty: it is an independent dominating set of F, of minimum size. □

Theorem 4.3 (Correctness)**.** *For every finite undirected graph G, Algorithm 1 returns an independent dominating set of G, and every candidate $c_1, \ldots, c_{18}$ is an independent dominating set of G.*

Proof. Self-loops do not affect independence or domination of other vertices and are removed. c_1–c_4, c_7, c_8, c_{13} and c_{14} are outputs of Algorithm 2, hence independent dominating sets by Lemma 4.1. c_5, c_6 and c_{12} select only vertices with no selected vertex in their closed neighbourhood and stop when every vertex is dominated, so they are maximal independent sets. In Algorithm 4 the vertices of a layer that are not yet dominated have no neighbour in X, the layer is an induced forest, and the forest solution is independent in G (Lemma 4.2), so X stays independent; the final greedy completion adds only undominated vertices. ResolveLayer (c_{10}, c_{11}, ForestLNS) ends with Algorithm 2. c_{17} joins an independent set of forced vertices with an independent dominating set of the residual graph, which contains no neighbour of a forced vertex, and every vertex outside the residual graph is a forced vertex or one of its neighbours. In Algorithm 6 the descent inserts u after removing all its neighbours in S and refills only undominated vertices, so S stays a maximal independent set, and the swap search records a set only when it has no undominated vertex and no conflicting edge. Finally, the implementation verifies every candidate explicitly and raises an error otherwise. □

5. Runtime Analysis

Hash-table operations are counted as $O(1)$ expected time; they occur only in the relabelling of the input vertices, in the multigraph of c_{11} and in the copy graph of c_{13}.

Lemma 5.1 (Copy graph)**.** *In Algorithm 5 let $B = \{v : \deg_G(v) > k\}$. The copy graph H has maximum degree at most k, exactly m edges, every edge of H joins two nodes whose owners are adjacent in G, every edge of G corresponds to exactly one edge of H, two copies of the same vertex are never adjacent, and $|V(H)| = n - |B| + \sum_{v \in B} \lceil \deg_G(v)/k \rceil \le n + 2m/k$. It is built in $O(n + m)$ expected time.*

Proof. Splitting u replaces every edge uw of H by one edge $u_j w$ and changes no other degree; hence when u is examined $\deg_H(u) = \deg_G(u)$, the copies of u get at most k neighbours each, and every vertex that is not split keeps its degree, at most k. Edges are moved one to one, so H has m edges, each between nodes with adjacent owners, and the copies of u are joined only to neighbours of u. Since $\lceil d/k \rceil \le d/k + 1$, $|V(H)| \le n + \sum_{v \in B} \deg_G(v)/k \le n + 2m/k$. Every edge is moved at most twice, once for each endpoint. □

Theorem 5.2 (Running time)**.** *For the default parameters, Algorithm 1 runs in $O(n + m)$ expected time.*

Proof. The conversion, the components, the lower bound LB and the verifications are linear. No step uses a comparison sort: every sort is a counting sort on integer keys at most $n + m$. c_2 uses the linear NetworkX matching; c_3–c_8 and c_{12} are single passes with bucket queues, $O(n + m)$, using Lemmas 4.1 and 4.2. c_1 and c_9 run Kruskal's algorithm and the layering with a union-find structure with union by size and path halving [24]. Apart from the pointer steps of the find operations, the work of a round of c_1 and of a pass of the layering is within a constant factor of the units charged to the budgets $40(n + m)$ and $8(n + m)$. The pointer steps are charged to the second counters, with the limits $320(n + m)$ and $64(n + m)$; a tree of the structure with a vertex at depth h has at least 2^h vertices, so one find takes at most $\log_2 n$ pointer steps, and the counters are tested after at most two finds, so they are exceeded by at most $2 \log_2 n$. Hence c_1 and the layering take $O(n + m)$ time in the worst case, and c_9 solves at most 6 orders per component, each $O(n + m)$. c_{10} and c_{11} are $O(n + m)$: every rule of the core peeling removes one vertex with $O(\deg)$ expected dictionary operations, and the bypass does not change degrees. c_{13} is linear by Lemma 5.1, since the inner solvers and the three repairs are linear in $|V(H)| + m = O(n + m)$. The improvers are bounded by work counters: every scan, update, insertion, removal and undo of Algorithm 6 and of ForestLNS is charged to a counter that stops the search at $\max(B(n + m), F)$ units, with fixed B and constant F, and a single move exceeds the limit by at most $O(n + m)$. c_{17} runs three rounds of linear work. There are at most eighteen distinct candidates, so the total is $O(n + m)$. □

6. Proved Bounds

Lemma 6.1 (Maximal-set bound)**.** *Every maximal independent set M of a graph G of maximum degree $\Delta \ge 1$ satisfies $n/(\Delta + 1) \le i(G) \le |M| \le \Delta\, i(G)$.*

Proof. A minimum independent dominating set D satisfies $V = \bigcup_{d\in D} N[d]$, so $n \le (\Delta+1)|D|$. M is an independent dominating set, so $i(G) \le |M|$. For $d \in D$, $|M \cap N[d]| = 1$ if $d \in M$, and $M \cap N[d] \subseteq N(d)$ has at most Δ vertices otherwise; since $M \subseteq \bigcup_{d\in D} N[d]$, $|M| \le \Delta|D|$. □

Corollary 6.2 (Bounded degree)**.** *For every graph G, the output S of Algorithm 1 satisfies $|S| \le \max(1,\Delta)\, i(G)$; in particular the ratio is at most Δ on every graph with an edge.*

Proof. Isolated vertices belong to S and to every independent dominating set, and on every other component S is a maximal independent set (Theorem 4.3), so Lemma 6.1 applies componentwise and the bounds add up. □

Lemma 6.3 (Certified lower bound)**.** *For every graph G with components C, $i(G) \ge LB = \sum_C \lceil n_C/(\Delta_C+1)\rceil$. If the output has at most LB vertices, it is optimal.*

Proof. An independent dominating set of G restricts to one of every component, and Lemma 6.1 gives $i(C) \ge n_C/(\Delta_C+1)$, an integer bound after rounding up. □

Theorem 6.4 (Forests)**.** *If G is a forest, the output of Algorithm 1 is a minimum independent dominating set.*

Proof. On a forest, the core of c_{11} is empty. Indeed, the residual multigraph stays a forest: removing a vertex keeps it one, and bypassing a vertex v of degree 2 with neighbours $a \ne b$ contracts an edge and creates no parallel edge, because a and b are not adjacent (otherwise a, v, b would lie on a cycle). So no vertex of degree 2 has both edges to the same vertex, and as long as vertices remain one of them has degree at most 1; the rules that send a vertex to the core never apply. Hence c_{11} is ResolveLayer on the forest G with no vertex fixed, that is, Algorithm 3 with neutral flags and $P = n+1$, which returns a minimum independent dominating set of G by Lemma 4.2, followed by Algorithm 2, which removes a vertex only when it has a neighbour in the set and adds a vertex only when it is undominated, and so leaves an independent dominating set unchanged. Every candidate after c_{11} is either a strategy output or the result of a search that never returns a larger set, and c_{18} is no larger than any of them, so $|c_{18}| \le |c_{11}| = i(G)$. □

The next theorem is the counterpart of the $O(\sqrt{n})$ theorem of Esperanza [23], which gives the ratio $2\sqrt{n}$ for maximum independent set on graphs with $\Delta \le \sqrt{n}$; for MIDS the bound is Δ itself.

Theorem 6.5 (Ratio $\sqrt{n}$ on three classes)**.** *Let S be the output of Algorithm 1 on a graph G with n vertices. If at least one of the following conditions holds, then $|S|/i(G) \le \sqrt{n}$:*

(a) $\Delta \le \sqrt{n}$*, and then* $|S|/i(G) \le \max(1,\Delta)$*;*

(b) $i(G) \ge \sqrt{n}$*;*

(c) $\alpha(G) \le \sqrt{n}\, i(G)$*.*

Proof. (a) is Corollary 6.2. (b) $|S| \le n$, so $|S|/i(G) \le n/\sqrt{n}$. (c) S is independent, so $|S| \le \alpha(G) \le \sqrt{n}\, i(G)$. □

The degree-splitting strategy c_{13} extends condition (a) to graphs with vertices of degree larger than $\sqrt{n}$. For $k = \max(1, \lfloor\sqrt{n}\rfloor)$ and $B = \{v : \deg_G(v) > k\}$ define the *splitting excess*

$$s(G) = \sum_{v\in B}\Big(\Big\lceil \frac{\deg_G(v)}{k}\Big\rceil - 1\Big) \le \sum_{v\in B} \frac{\deg_G(v)}{k} \le \frac{2m}{k}.$$

Lemma 6.6 (Lifting)**.** *For the copy graph H of Algorithm 5, $i(H) \le i(G) + s(G)$.*

Proof. Let D be a minimum independent dominating set of G, and let L be the set of nodes of H whose owner lies in D; $|L| = \sum_{u\in D}\lceil \deg_G(u)/k\rceil$ over split u, plus one per unsplit u. L is independent in H, since an edge of H joins nodes with adjacent owners (Lemma 5.1) and D is independent. Let z be a node of H with owner $w \notin D$. The vertex w has a neighbour $u \in D$, and the edge uw of G corresponds to exactly one edge of H, between a node owned by u, which lies in L, and a node owned by w. If w is not split, that node is z, so z is dominated by L; if w is split, at least one of its $\lceil \deg_G(w)/k\rceil$ copies is dominated. Completing L greedily to a maximal independent set of H adds only undominated nodes, at most $\lceil \deg_G(w)/k\rceil - 1$ for every split $w \notin D$. Hence $i(H) \le |D| + \sum_{u\in D\cap B}(\lceil \deg_G(u)/k\rceil - 1) + \sum_{w\in B\setminus D}(\lceil \deg_G(w)/k\rceil - 1) = i(G) + s(G)$. □

Lemma 6.7 (Decoding)**.** *For every independent dominating set S_H of H, the set $S_0 = \{\text{owner}(x) : x \in S_H\}$ is a dominating set of G with $|S_0| \le |S_H|$. If S_0 is independent in G, then* Complete$(\text{adj}, S_0, \pi) = S_0$.

Proof. $|S_0| \le |S_H|$ because the owner map is a function. Let $w \in V$. If w is not split, the node w of H is in S_H, so $w \in S_0$, or it has a neighbour $x \in S_H$, whose owner is a neighbour of w in G. If w is split, its first copy has at least one neighbour, and the same argument applies to that copy. If S_0 is independent, the repair loop of Algorithm 2 removes nothing, and $U = \emptyset$ because S_0 dominates G, so nothing is added. □

Theorem 6.8 (Degree splitting)**.** *Let $k = \max(1, \lfloor \sqrt{n} \rfloor)$. For each inner solver of Algorithm 5, the decoded set S_0 is a dominating set of G with*

$$|S_0| \;\le\; k\,(i(G) + s(G)).$$

If S_0 is independent in G for one of the inner solvers, then the output S of Algorithm 1 satisfies $|S| \le k\,(i(G) + s(G))$, so $|S|/i(G) \le \lfloor \sqrt{n} \rfloor\,(1 + s(G)/i(G))$; in particular $|S|/i(G) \le 2\sqrt{n}$ when $s(G) \le i(G)$.

Proof. Every inner solver returns a maximal independent set S_H of H (Lemma 4.1 and Theorem 4.3). H has maximum degree at most k by Lemma 5.1, so Lemma 6.1, applied componentwise in H with $\max(1, \Delta(H)) \le k$, gives $|S_H| \le k\, i(H)$, and Lemma 6.6 gives $i(H) \le i(G) + s(G)$. Lemma 6.7 gives $|S_0| \le |S_H|$ and domination. Strategy c_{13} applies Algorithm 2 to S_0. If S_0 is independent, Algorithm 2 returns S_0 itself (Lemma 6.7), so the candidate c_{13} has at most $|S_0|$ vertices, and the output is no larger than c_{13}. If S_0 is not independent, Algorithm 2 removes vertices of S_0 and can add other vertices, and the bound is proved for S_0 only. □

On a graph with $\Delta \le k$ no vertex is split, $s(G) = 0$, $H = G$, and Theorem 6.8 reduces to Theorem 6.5(a). On a star $K_{1,n-1}$, $s(G) = \lceil (n-1)/k \rceil - 1$, the copy graph is a disjoint union of $\lceil (n-1)/k \rceil$ stars, and every maximal independent set of H that contains the copies of the centre decodes to the optimum $\{\text{centre}\}$. The bound of Theorem 6.8 on S_0 holds on every graph, whereas the bound on the output requires the independence of S_0.

Worst cases of the splitting excess. The excess $s(G)$ is largest on dense graphs; the next two propositions show that the graphs on which it is largest are graphs on which the output is optimal.

Proposition 6.9 (Splitting excess)**.** *Let G be a graph on $n \ge 3$ vertices with m edges, $k = \lfloor \sqrt{n} \rfloor$ and $B = \{v : \deg(v) > k\}$.*

(i) $s(G) = 0$ *if and only if $\Delta \le k$; then no vertex is split, $H = G$, and the scan of Algorithm 5 ends at its first vertex.*

(ii) $s(G) \le \sum_{v \in B} \frac{\deg(v)-1}{k} \le \min\left\{ \frac{2m-|B|}{k},\ \frac{|B|\,(n-2)}{k} \right\} \le |B|\,(\sqrt{n}+1) \le n\,(\sqrt{n}+1)$.

(iii) The complete graph maximises the excess: $s(G) \le s(K_n) = n(\lceil (n-1)/k \rceil - 1)$, *and* $s(K_n) \ge n^{3/2} - n - \sqrt{n}$.

(iv) For the complete bipartite graph $K_{a,b}$ with $a \le b$ and $a + b = n$, $s(K_{a,b}) = a\,(\lceil b/k \rceil - 1) + b\,(\lceil a/k \rceil - 1)$; a term vanishes when its degree is at most k. In particular $s(K_{n/2,n/2}) = n\,(\lceil n/(2k) \rceil - 1)$ for even n, and $s(K_{1,n-1}) = \lceil (n-1)/k \rceil - 1$.

Proof. (i) $\lceil d/k \rceil - 1 = 0$ exactly when $d \le k$. The vertices are scanned by decreasing degree and splitting changes no other degree (Lemma 5.1), so when the first vertex has degree $\Delta \le k$ no vertex is split. (ii) For every integer $d \ge 1$, $\lceil d/k \rceil \le (d-1)/k + 1$. Summing over B gives the first inequality, and $\sum_{v \in B} \deg(v) \le 2m$ and $\deg(v) \le n - 1$ give the second. Since $k > \sqrt{n} - 1 > 0$ and $n - 2 < n - 1 = (\sqrt{n} - 1)(\sqrt{n} + 1)$, $(n-2)/k < \sqrt{n} + 1$. (iii) $\lceil d/k \rceil$ is non-decreasing in d, every degree is at most $n - 1$ and $|B| \le n$; in K_n every vertex has degree $n - 1 > \sqrt{n} \ge k$ for $n \ge 3$, so every vertex lies in B and attains the maximum. Moreover $\lceil (n-1)/k \rceil \ge (n-1)/\sqrt{n} = \sqrt{n} - 1/\sqrt{n}$, which gives the lower bound. (iv) The a vertices of one side have degree b and the b vertices of the other side have degree a. □

Proposition 6.10 (Rigid dense families)**.** *If every connected component of G is a complete multipartite graph (in particular a complete graph, a complete bipartite graph or a star), a crown graph with $t \ge 3$ vertices on each side (the complete bipartite graph $K_{t,t}$ minus a perfect matching), a wheel or a double star, the output of Algorithm 1 is a minimum independent dominating set.*

Proof. Gains are local to the components, and whenever the greedy c_3 selects a vertex of a component C, that vertex has the largest gain $|N[v] \cap U|$ among the undominated vertices of C; we show that c_3 is optimal on every such component, and the output is no larger than c_3 (proof of Theorem 6.4). *Complete multipartite* C with parts $P_1, \dots, P_r$: a vertex of P_j has gain $|C| - |P_j| + 1$, so the first vertex selected lies in a smallest part P; it dominates every other part, the other vertices of P are undominated, have no undominated neighbour and are all selected, so $c_3 \cap C = P$. An independent dominating set that contains a vertex of a part contains no vertex of the other parts, hence all of that part to dominate it, so $i(C) = \min_j |P_j|$. *Crown* with sides $\{a_1, \dots, a_t\}$, $\{b_1, \dots, b_t\}$ and $a_i b_j \in E$ iff $i \neq j$: all gains are t, so c_3 first selects, say, a_1, which dominates every b_j with $j \neq 1$; then b_1 has gain t and every a_i, $i \geq 2$, has gain $2 < t$, so b_1 is selected and dominates the rest; $c_3 \cap C = \{a_1, b_1\}$, and no single vertex dominates C, so $i(C) = 2$. *Wheel* (a hub joined to a cycle): the hub has the largest gain $|C|$ and dominates C, so $c_3 \cap C$ is the hub and $i(C) = 1$. *Double star* (adjacent hubs x, y with $a \geq b \geq 1$ leaves): x has the largest gain $a + 2$; selecting it dominates its leaves and y, and the b leaves of y are then selected, so $|c_3 \cap C| = 1 + b$. An independent dominating set contains x (and then all leaves of y), or y (and all leaves of x), or neither (and all $a + b$ leaves), so $i(C) = 1 + \min(a, b) = 1 + b$. □

By Propositions 6.9 and 6.10, the bound $\lfloor \sqrt{n} \rfloor\, (i(G) + s(G))$ of Theorem 6.8 is weakest on complete graphs, where $s(G)$ is of order $n^{3/2}$ and $i(G) = 1$, and on balanced complete bipartite graphs, where $s(G)$ is of order $n^{3/2}/2$; on both, $i(G)$ is attained by the greedy strategy c_3 and the output is optimal. The bound is useful when few vertices have degree larger than $\sqrt{n}$.

Corollary 6.11 (Few vertices of large degree)**.** *If $|B| \leq i(G)/(\sqrt{n} + 1)$, in particular if $|B|\,(\sqrt{n} + 1)(\Delta + 1) \leq n$, then $s(G) \leq i(G)$, and the decoded set S_0 of Theorem 6.8 has at most $2\lfloor \sqrt{n} \rfloor\, i(G)$ vertices; if S_0 is independent, the output has ratio at most $2\sqrt{n}$.*

Proof. By Proposition 6.9(ii), $s(G) \leq |B|(\sqrt{n} + 1) \leq i(G)$, and $i(G) \geq n/(\Delta + 1)$ by Lemma 6.1. Theorem 6.8 gives the rest. □

The splitting excess on hard families. Proposition 6.12 and Table 4 give $s(G)$ on the families on which MIDS is NP-hard, NP-hard to approximate, or on which, through the VC-IDS reduction of Section 7, it encodes a UG-hard vertex cover problem, and on further structured families.

Proposition 6.12 (Splitting excess on graph families)**.** *Let $k = \lfloor \sqrt{n} \rfloor$.*

(a) Bounded degree. *If $\Delta \leq D$ for a constant D, then $s(G) = 0$ for every $n \geq D^2$. This holds for cubic graphs with $n \geq 10$ (Proposition 6.13); for graphs and bipartite graphs of maximum degree at most B with $n \geq B^2$; for the SC-IDS graphs of maximum degree B with $n \geq B^2$; for the VC-IDS graph $h(H)$ of Section 7 with $\Delta(H) \leq D$ and $|V(H)| + 6|E(H)| \geq \max(D, 3)^2$; for graphs of maximum degree d with $n \geq d^2$, the setting of [28]; for grids with $n \geq 16$; and for hypercubes Q_q with $q \geq 4$.*

(b) Khot–Regev graphs. *The YES graph with N blocks of s_b subsets on a d-regular constraint graph has $n = Ns_b$ vertices and maximum degree at most ds_b, so $s(G) \leq n(\lceil ds_b/k \rceil - 1)$, and $s(G) = 0$ when $N \geq d^2 s_b$.*

(c) Sparse graphs. *$s(G) \leq 2m/k$; for planar graphs $s(G) \leq (6n-12)/k$, for bipartite planar graphs $s(G) \leq (4n-8)/k$ (sharper bounds in Proposition 6.15), and for graphs with $m \leq cn$, $s(G) \leq 2c\,n/k \leq 2c(\sqrt{n} + 2)$ for $n \geq 4$.*

(d) Sparse random graphs. *For a constant $c > 0$, the random graph $G(n, c/n)$ satisfies $s(G) = 0$ with probability tending to 1 as $n \to \infty$.*

(e) Few hubs. *$s(K_{1,n-1}) = s(W_n) = \lceil (n-1)/k \rceil - 1$ for the star and the wheel, and the double star with a and b leaves has $s = (\lceil (a+1)/k \rceil - 1) + (\lceil (b+1)/k \rceil - 1)$.*

(f) Dense graphs. *If the minimum degree satisfies $\delta(G) > k$, then $s(G) \geq n(\lceil \delta(G)/k \rceil - 1)$. The complete multipartite graph with parts of sizes $n_1, \dots, n_r$ has $s = \sum_j n_j(\lceil (n - n_j)/k \rceil - 1)$, and the crown graph on $n = 2t$ vertices has $s = n(\lceil (t-1)/k \rceil - 1)$.*

Proof. (a) $\lfloor \sqrt{n} \rfloor \geq D$ if and only if $n \geq D^2$, and then $\Delta \leq k$, so $s(G) = 0$ by Proposition 6.9(i). The graph $h(H)$ has $|V(H)| + 6|E(H)|$ vertices and maximum degree $\max(\Delta(H), 3)$; Q_q has 2^q vertices and degree $q \leq 2^{q/2}$ for $q \geq 4$; grids have degree at most 4. (b) A vertex (v, x) is adjacent only to vertices of the d neighbouring blocks, each with s_b vertices; $ds_b \leq \sqrt{N s_b}$ if and only if $N \geq d^2 s_b$. (c) Proposition 6.9(ii) with $m \leq 3n - 6$ for planar and $m \leq 2n - 4$ for bipartite planar graphs; $n/\lfloor \sqrt{n} \rfloor < n/(\sqrt{n} - 1) = \sqrt{n} + 1 + 1/(\sqrt{n} - 1) \leq \sqrt{n} + 2$ for $n \geq 4$. (d) For $t = k + 1 > \sqrt{n}$, a fixed vertex has degree at least t with probability at most $\binom{n-1}{t}(c/n)^t \leq (ec/t)^t \leq (ec/\sqrt{n})^{\sqrt{n}}$ once $t > ec$, so the probability that some vertex has degree larger than k is at most $n(ec/\sqrt{n})^{\sqrt{n}} \to 0$, and then $s(G) = 0$. (e) The hubs have degrees $n - 1$, $a + 1$ and $b + 1$, every other vertex has degree at most $3 \leq k$ for $n \geq 9$, and the formula of s applies. (f) Every vertex lies in B and $\lceil \deg(v)/k \rceil \geq \lceil \delta(G)/k \rceil$; a vertex of the part j has degree $n - n_j$, and every vertex of the crown has degree $t - 1$. □

Table 4: The splitting excess $s(G)$ ($k = \lfloor \sqrt{n} \rfloor$) and the output on graph families. "Hardness" cites the hardness of MIDS on the family, or of the vertex cover problem it encodes; "Output" is the guarantee proved for Algorithm 1.

Family	Hardness	Δ	$s(G)$	$i(G)$	Output
cubic, $n \in \{4, 6, 8\}$	—	3	n	$\geq n/4$	ratio ≤ 2
cubic, $n \geq 10$	APX-hard [6]	3	0	$\geq n/4$	ratio ≤ 2
cubic planar, $n \geq 10$	NP-hard [5]	3	0	$\geq n/4$	ratio ≤ 2
cubic planar bipartite, $n \geq 10$	NP-complete [4]	3	0	$\geq n/4$	ratio ≤ 2
maximum degree ≤ 3	NP-hard within $1 + \frac{1}{680}$ [10]	≤ 3	≤ 8; 0 for $n \geq 9$	$\geq n/4$	ratio ≤ 3
maximum degree $\leq B$, bipartite or not, $n \geq B^2$	NP-hard within δB [10]	$\leq B$	0	$\geq n/(B + 1)$	ratio $\leq B$
SC-IDS of degree B, $n \geq B^2$	reduction of [10]	B	0	$\mathrm{sc} + \lvert S \rvert$	ratio $\leq B$
VC-IDS $h(H)$, $\Delta(H) \leq D$, $\lvert V(h(H)) \rvert \geq \max(D, 3)^2$	vertex cover: UG-hard [18, 28]	$\max(D, 3)$	0	$\tau(H) + 2\lvert E(H) \rvert$	ratio $\leq \max(D, 3)$
Khot–Regev YES graphs, $N \geq d^2 s_b$	vertex cover: UG-hard [18]	$\leq ds_b$	0	—	ratio $\leq ds_b$
Khot–Regev YES graphs, any N	vertex cover: UG-hard [18]	$\leq ds_b$	$\leq n(\lceil ds_b/k \rceil - 1)$	—	ratio $\leq ds_b$
planar, $n \geq 400$	NP-hard [5]	unbounded	$< 2\sqrt{n} + 19$; $\geq 2\sqrt{n} - 3$ for some graphs	—	—
bipartite planar, $n \geq 400$	NP-hard [4]	unbounded	$< 2\sqrt{n} + 12$; $\geq 2\sqrt{n} - 3$ for some graphs	—	—
circle graphs	not within n^{ε}, $\varepsilon < 1$ [3]	unbounded	up to $s(K_n)$ (maximum)	—	—
$m \leq cn$	—	unbounded	$\leq 2c(\sqrt{n} + 2)$	—	—
$G(n, c/n)$	—	$\leq k$ w.h.p.	0 w.h.p.	—	ratio $\leq \Delta$
grids, $n \geq 16$; hypercubes Q_q, $q \geq 4$	—	≤ 4; q	0	—	ratio $\leq \Delta$
star $K_{1,n-1}$, wheel W_n	—	$n - 1$	$\lceil (n - 1)/k \rceil - 1$	1	optimal
double star ($a \geq b$ leaves)	—	$a + 1$	$\sum_{h \in \{a,b\}} (\lceil (h + 1)/k \rceil - 1)$	$1 + b$	optimal
complete graph K_n	—	$n - 1$	$n(\lceil (n - 1)/k \rceil - 1)$ (maximum)	1	optimal
complete bipartite $K_{a,b}$, $a \leq b$	—	b	$a(\lceil b/k \rceil - 1) + b(\lceil a/k \rceil - 1)$	$\min(a, b)$	optimal
complete multipartite	—	$n - \min_j n_j$	$\sum_j n_j(\lceil (n - n_j)/k \rceil - 1)$	$\min_j n_j$	optimal
crown graph, $n = 2t$	—	$t - 1$	$n(\lceil (t - 1)/k \rceil - 1)$	2	optimal
minimum degree $\delta > k$	—	$\geq \delta$	$\geq n(\lceil \delta/k \rceil - 1)$	—	—

By Proposition 6.12(a), on every family of bounded degree D on which MIDS is NP-hard or NP-hard to approximate, and on the VC-IDS graphs that encode UG-hard vertex cover instances of bounded degree, the splitting excess vanishes for $n \geq D^2$: strategy c_{13} then solves G itself, and Theorem 6.8 gives the ratio $\lfloor \sqrt{n} \rfloor$, weaker than the ratio Δ of Corollary 6.2. The excess is positive only on graphs with vertices of degree larger than $\sqrt{n}$; it is of order at most $\sqrt{n}$ on graphs with a constant number of such vertices or with $O(n)$ edges, and of order $n^{3/2}$ on dense graphs, among them the complete, complete multipartite and crown graphs, on which the output is optimal (Proposition 6.10). On all 55 instances of Section 9.5, $\Delta \leq \lfloor \sqrt{n} \rfloor$ (Table 8: $n \geq 106$ and $\Delta \leq 6$ for SC-IDS, $n \geq 500$ and $\Delta \leq 5$ for the regular VC-IDS graphs, $n \geq 48\,172$ and $\Delta = 64$ for the VC-IDS graphs of Khot–Regev instances), so $s = 0$ on all of them.

Cubic graphs. MIDS is APX-hard on cubic graphs [6] and NP-hard on cubic planar [5] and cubic planar bipartite graphs [4]. On these graphs the splitting excess is bounded by an absolute constant, and it vanishes from ten vertices on.

Proposition 6.13 (Splitting excess on cubic and bounded-degree graphs)**.** *Let* $k = \max(1, \lfloor \sqrt{n} \rfloor)$.

(i) If G is cubic, then $s(G) = n$ for $n \in \{4, 6, 8\}$ and $s(G) = 0$ for every $n \geq 10$; hence $s(G) \leq 8$ for every cubic graph.

(ii) If $\Delta(G) \leq 3$, then $s(G) \leq 8$, and $s(G) = 0$ for every $n \geq 9$.

(iii) If $\Delta(G) \leq D$ for an integer $D \geq 2$, then $s(G) < D^2(D - 1)$, and $s(G) = 0$ for every $n \geq D^2$.

Proof. (i) A cubic graph has an even number $n \geq 4$ of vertices, since $3n = 2m$. For $n \in \{4, 6, 8\}$, $k = 2$, every vertex has degree $3 > k$ and contributes $\lceil 3/2 \rceil - 1 = 1$, so $s(G) = n \leq 8$. For even $n \geq 10$, $k \geq 3 = \Delta$, so $s(G) = 0$ by

Proposition 6.9(i). (ii) For $n \le 3$, $k = 1$, the degrees are at most 2, and every vertex contributes at most $\lceil 2/1 \rceil - 1 = 1$, so $s(G) \le 3$. For $4 \le n \le 8$, $k = 2$, a vertex of degree 3 contributes 1 and a vertex of degree at most 2 contributes 0, so $s(G) \le n \le 8$. For $n \ge 9$, $k \ge 3 \ge \Delta$ and $s(G) = 0$. (iii) For $n \ge D^2$, $k \ge D \ge \Delta$ and $s(G) = 0$. For $n < D^2$, every vertex contributes $\lceil \deg(v)/k \rceil - 1 \le D - 1$, since $k \ge 1$, so $s(G) \le n(D-1) < D^2(D-1)$. □

On a cubic graph with $n \ge 10$, Algorithm 5 therefore splits no vertex, $H = G$, and c_{13} applies its inner solvers to G itself. The ratio of the output on cubic graphs is bounded by the regularity of the graph.

Theorem 6.14 (Regular graphs)**.** *If G is d-regular with $d \ge 1$, every independent dominating set S of G, in particular the output of Algorithm 1, satisfies $|S| \le n/2$ and $|S|/i(G) \le (d+1)/2$. On cubic graphs, $n/4 \le i(G) \le |S| \le n/2$ and the ratio is at most 2.*

Proof. Let I be an independent set of G. Every vertex of I has its d neighbours in $V \setminus I$, and every vertex of $V \setminus I$ has at most d neighbours in I, so counting the edges between I and $V \setminus I$ gives $d|I| \le d(n - |I|)$, that is $|I| \le n/2$. The output is an independent dominating set (Theorem 4.3), and Lemma 6.1 gives $i(G) \ge n/(d+1)$, so $|S|/i(G) \le (n/2)/(n/(d+1)) = (d+1)/2$. For $d = 3$, $n/4 \le i(G)$ and $(d+1)/2 = 2$. □

Planar graphs. MIDS is NP-hard on cubic planar graphs [5] and NP-complete on cubic planar bipartite graphs [4], hence on planar and on bipartite planar graphs. On planar graphs the splitting excess is not bounded by a constant, but it is at most $2\sqrt{n} + O(1)$, and the leading constant 2 is attained.

Proposition 6.15 (Splitting excess on planar graphs)**.** *Let G be a planar graph on n vertices, $k = \lfloor \sqrt{n} \rfloor$, $B = \{v : \deg(v) > k\}$ and $b = |B|$.*

(i) If $n \ge 36$, then $b < 2n/(k-5)$ and $s(G) < 2n/k + 10n/(k(k-5))$; for $n \ge 400$, $s(G) < 2\sqrt{n} + 19$.

(ii) If G is bipartite and $n \ge 16$, then $b < 2n/(k-3)$ and $s(G) < 2n/k + 6n/(k(k-3))$; for $n \ge 400$, $s(G) < 2\sqrt{n} + 12$.

(iii) For every $n \ge 16$, the planar bipartite graph $K_{2,n-2}$ satisfies $s(K_{2,n-2}) = 2(\lceil (n-2)/k \rceil - 1) \ge 2\sqrt{n} - 3$, and the output of Algorithm 1 on it is optimal.

Proof. Let $R = V \setminus B$, let e_B be the number of edges with both ends in B, e_{BR} the number of edges between B and R, and $D = \sum_{v \in B} \deg(v) = 2e_B + e_{BR}$. The subgraph of the edges between B and R is bipartite and planar, so $e_{BR} < 2n$ (a bipartite planar graph on $p \ge 3$ vertices has at most $2p - 4$ edges, and on $p \le 2$ vertices at most 1). (i) $G[B]$ is planar, so $e_B \le 3b$ and $D < 6b + 2n$; every vertex of B has degree at least $k+1$, so $b(k+1) \le D < 6b + 2n$, that is $b(k-5) < 2n$, and $k \ge 6$ for $n \ge 36$. By Proposition 6.9(ii), $s(G) \le (D-b)/k < (5b+2n)/k < 2n/k + 10n/(k(k-5))$. For $n \ge 400$, $k \ge 20$, so $k - 5 \ge 3k/4$, and $n < (k+1)^2$ gives $n/k^2 < (21/20)^2 < 1.11$; hence $10n/(k(k-5)) \le \frac{40}{3} n/k^2 < 14.8$, and $n/k < \sqrt{n} + 2$ (proof of Proposition 6.12(c)) gives $s(G) < 2\sqrt{n} + 4 + 14.8 < 2\sqrt{n} + 19$. (ii) $G[B]$ is bipartite planar, so $e_B \le 2b$, $D < 4b + 2n$, $b(k-3) < 2n$ with $k \ge 4$ for $n \ge 16$, and $s(G) < (3b+2n)/k < 2n/k + 6n/(k(k-3))$. For $n \ge 400$, $k - 3 \ge 17k/20$, so $6n/(k(k-3)) \le \frac{120}{17} n/k^2 < 7.9$ and $s(G) < 2\sqrt{n} + 4 + 7.9 < 2\sqrt{n} + 12$. (iii) $K_{2,n-2}$ is planar and bipartite. Its two vertices of degree $n - 2 > \sqrt{n} \ge k$ form B, and the other vertices have degree $2 \le k$, so $s = 2(\lceil (n-2)/k \rceil - 1) \ge 2(n-2)/\sqrt{n} - 2 = 2\sqrt{n} - 2 - 4/\sqrt{n} \ge 2\sqrt{n} - 3$ for $n \ge 16$. The output is optimal by Proposition 6.10. □

Circle graphs. A circle graph is the intersection graph of a set of chords of a circle; MIDS is NP-complete on circle graphs and, unless P = NP, not approximable within n^{ε} for any $\varepsilon < 1$ [3]. A chord model of n chords has $2n$ distinct endpoints, which we number $1, \dots, 2n$ around the circle from a point that is not an endpoint, and we write the chord of v as (a_v, b_v) with $a_v < b_v$; two chords cross if and only if exactly one endpoint of one of them lies strictly between the endpoints of the other.

Proposition 6.16 (Splitting excess on circle graphs)**.** *(i) For a circle graph given by a chord model, $\deg(v) = b_v - a_v - 1 - 2\,|\{u : a_v < a_u < b_u < b_v\}|$ for every chord v, so all degrees, B and $s(G)$ can be computed in $O(n \log n)$ time from the model, without listing the edges.*

(ii) For every $n \ge 3$, the complete graph K_n and every complete bipartite graph $K_{a,b}$ with $a + b = n$ are circle graphs; hence the largest splitting excess among circle graphs on n vertices equals the largest one among all graphs, $s(K_n) = n(\lceil (n-1)/k \rceil - 1) \ge n^{3/2} - n - \sqrt{n}$, and on these graphs the output of Algorithm 1 is optimal.

Proof. (i) The other $2n-2$ endpoints are distinct, and $b_v - a_v - 1$ of them lie strictly between a_v and b_v. A chord with both endpoints there does not cross v and contributes two of them, a chord with exactly one endpoint there crosses v and contributes one, and the other chords contribute none and do not cross v; this gives the formula. The numbers $|\{u : a_v < a_u < b_u < b_v\}|$ are counted for all v by scanning the chords by decreasing a_v and querying and updating a Fenwick tree over the right endpoints, in $O(n \log n)$ time; B and $s(G)$ then take $O(n)$ time. (ii) The n diameters of a regular $2n$-gon cross pairwise, so K_n is a circle graph. For $K_{a,b}$, take a horizontal and b vertical chords of the unit circle, all at distance less than $\frac{1}{2}$ from the centre: chords of the same direction are disjoint, and every horizontal chord crosses every vertical one. Proposition 6.9(iii) gives the value of $s(K_n)$ and its maximality, and Proposition 6.10 the optimality. □

Conjecture 6.17 (Ratio $O(\sqrt{n})$)**.** *For every graph G on n vertices, there is a constant c, independent of G, such that the output S of Algorithm 1 satisfies $|S|/i(G) \le c\sqrt{n}$.*

The bound of Conjecture 6.17 holds, with an explicit constant, on the graphs of the next proposition.

Proposition 6.18 (Graphs on which the bound of Conjecture 6.17 holds)**.** *Let S be the output of Algorithm 1 on a graph G with $n \ge 1$ vertices, minimum degree δ and maximum degree Δ.*

(a) Excluded induced star. *If G has no induced subgraph $K_{1,r}$ for an integer $r \ge 2$, then $|S|/i(G) \le r-1$. This holds with $r = \Delta+1$ when $\Delta \ge 1$, and with $r = 3$ on claw-free graphs, among them the line graphs. Hence $|S|/i(G) \le D$ for every n on every family of maximum degree at most D, and $|S|/i(G) \le 2$ on claw-free graphs.*

(b) Independence number. *$|S|/i(G) \le \alpha(G)(\Delta+1)/n$ and $|S|/i(G) \le \alpha(G) \le n-\delta$. Hence $|S|/i(G) \le \sqrt{n}$ if $\alpha(G)(\Delta+1) \le n^{3/2}$, if $\alpha(G) \le \sqrt{n}\, i(G)$, if the vertices of G are covered by at most $\sqrt{n}$ cliques, or if $\delta \ge n - \sqrt{n}$.*

(c) Distant vertices. *If G has q vertices such that no two of them are adjacent or have a common neighbour, then $i(G) \ge q$ and $|S|/i(G) \le n/q$. Hence $|S|/i(G) \le \sqrt{n}$ if $i(G) \ge \sqrt{n}$, if G has at least $\sqrt{n}$ connected components, or if a connected component of G has diameter at least $3\sqrt{n}-1$.*

(d) Regular graphs. *If G is d-regular with $d \ge 1$, then $|S|/i(G) \le \min((d+1)/2,\ n-d)$. Hence $|S|/i(G) \le \sqrt{n}$ if $d \le 2\sqrt{n}-1$ or $d \ge n-\sqrt{n}$.*

(e) Splitting excess. *If the decoded set S_0 of Theorem 6.8 is independent for one of the inner solvers and $s(G) \le t\, i(G)$ for a real number $t \ge 0$, then $|S|/i(G) \le (1+t)\sqrt{n}$; for $t = 1$, in particular under the hypothesis of Corollary 6.11, $|S|/i(G) \le 2\sqrt{n}$.*

(f) Exact families. *If G is a forest, or if every connected component of G is a complete multipartite graph, a crown graph with at least three vertices on each side, a wheel or a double star, then $|S| = i(G)$.*

The bounds of (a), (b), (c) and (d) hold for every independent dominating set of G in the place of S.

Proof. S is an independent dominating set of G (Theorem 4.3); let D be a minimum one. (a) Every vertex of $S \setminus D$ has a neighbour in D, and this neighbour is not in S, because S is independent. A vertex $u \in D \setminus S$ has at most $r-1$ neighbours in S: its neighbours in S are pairwise non-adjacent, and r of them would induce a $K_{1,r}$ with u. Hence $|S \setminus D| \le (r-1)|D \setminus S|$ and $|S| \le |S \cap D| + (r-1)|D \setminus S| \le (r-1)|D|$. A vertex of degree at most Δ is not the centre of a $K_{1,\Delta+1}$. In a line graph, three edges that meet an edge uv and are distinct from it contain two edges through u or two edges through v, which meet; so a line graph has no induced $K_{1,3}$. If $\Delta \le D$ and $\Delta \ge 1$ the bound is $\Delta \le D$, and if $\Delta = 0$ then $S = V = D$. (b) $|S| \le \alpha(G)$ because S is independent, and $n \le (\Delta+1)\, i(G)$ because the closed neighbourhoods of the vertices of D cover V; $i(G) \ge 1$ gives $|S|/i(G) \le \alpha(G)$. An independent set that contains a vertex v is disjoint from $N(v)$, so it has at most $n - \deg(v) \le n - \delta$ vertices. An independent set has at most one vertex in each clique, so $\alpha(G)$ is at most the number of cliques of a cover. (c) The closed neighbourhoods of the q vertices are pairwise disjoint and each contains a vertex of D, so $|D| \ge q$, and $|S| \le n$. If $i(G) \ge \sqrt{n}$, then $|S|/i(G) \le n/\sqrt{n}$. Vertices of different components are neither adjacent nor have a common neighbour, so one vertex of each component gives q equal to the number of components. On a shortest path $v_0, \dots, v_p$ between two vertices at distance p, the distance between v_a and v_b is $|a-b|$, so the $\lfloor p/3 \rfloor + 1 \ge (p+1)/3$ vertices $v_0, v_3, v_6, \dots$ are pairwise at distance at least 3; for $p \ge 3\sqrt{n}-1$ this gives $q \ge \sqrt{n}$. (d) Theorem 6.14 gives $(d+1)/2$, and (b) gives $n - \delta = n - d$. (e) By

Theorem 6.8, $|S|/i(G) \le \lfloor \sqrt{n} \rfloor (1 + s(G)/i(G)) \le (1 + t)\sqrt{n}$; Corollary 6.11 gives $s(G) \le i(G)$. (f) Theorem 6.4 and Proposition 6.10. The proofs of (a), (b) and (c) use only that S is an independent dominating set, and so does the proof of Theorem 6.14. □

Under each of the hypotheses of Proposition 6.18 that end with the bound 1, 2, $\sqrt{n}$ or $2\sqrt{n}$, the output satisfies $|S|/i(G) \le 2\sqrt{n}$, and on a family of maximum degree at most D it satisfies $|S|/i(G) \le D$, so the bound of Conjecture 6.17 holds on such a family with $c = \max(2, D)$. By Proposition 6.18(a) this covers the families of bounded degree of Table 4: the cubic, cubic planar and cubic planar bipartite graphs ($D = 3$, and the ratio is at most 2 by Theorem 6.14), the graphs and bipartite graphs of maximum degree at most B, the SC-IDS graphs of maximum degree B, the VC-IDS graphs $h(H)$ with $\Delta(H) \le D$, the grids and, with $\Delta = q \le \sqrt{n}$ for $q \ge 4$, the hypercubes Q_q. It covers the Khot–Regev YES graphs with $N \ge d^2 s_b/4$ blocks, on which $\Delta \le d s_b \le 2\sqrt{n}$, and the random graph $G(n, c/n)$ with probability tending to 1 (Proposition 6.12(d)). By Proposition 6.18(f) the bound holds with ratio 1 on the forests and on the stars, wheels, double stars, complete graphs, complete bipartite graphs, complete multipartite graphs and crown graphs of Table 4.

Theorem 6.19 (Conditional consequence of Conjecture 6.17)**.** *If Conjecture 6.17 is proved, then* P = NP.

Proof. Algorithm 1 runs in polynomial time (Theorem 5.2). For $n \ge c^4$, $c\sqrt{n} \le n^{3/4}$, and the graphs with $n < c^4$ vertices are solved exactly in constant time, so Conjecture 6.17 would give a polynomial-time approximation of MIDS within $n^{1-\varepsilon}$ with $\varepsilon = \frac{1}{4}$, which is impossible unless P = NP [2]. □

Corollary 6.20 (Circle graphs)**.** *If Conjecture 6.17 is proved for circle graphs only, then* P = NP.

Proof. Algorithm 1 takes the graph and needs no chord model, and it runs in polynomial time (Theorem 5.2). For $n \ge c^4$, $c\sqrt{n} \le n^{3/4}$, and the graphs with $n < c^4$ vertices are solved exactly in constant time, so the conjecture on circle graphs would give a polynomial-time approximation of MIDS on circle graphs within n^{ε} with $\varepsilon = \frac{3}{4} < 1$, which is impossible unless P = NP [3]. □

7. Hard Instances and the Unique Games Conjecture

Chlebík and Chlebíková prove their bounded-degree results with two reductions [10], which we use as instance generators.

SC-IDS. For a set system $\mathcal{S}$ over a universe $\mathcal{U}$, take the bipartite incidence graph of $\mathcal{S}$ and $\mathcal{U}$ and attach to every set node S a path S–S'–S''. The graph is bipartite, its maximum degree is the largest of the set sizes plus one and the element frequencies, and $i = \mathrm{sc}(\mathcal{S}) + |\mathcal{S}|$, where sc is the minimum set cover size [10].

VC-IDS. For a graph H with m_H edges, replace every edge uv by the gadget of Figure 1: six new vertices a, b, c, d, x, y and the eight edges $ua, ab, bc, cd, dv, bx, xy, yc$. The resulting graph is denoted $h(H)$; it has $|V(H)| + 6m_H$ vertices, $8m_H$ edges and maximum degree $\max(\Delta(H), 3)$. Chlebík and Chlebíková use an edge gadget with these parameters and the identity of Proposition 7.1 [10]; we prove the identity for the gadget of Figure 1.

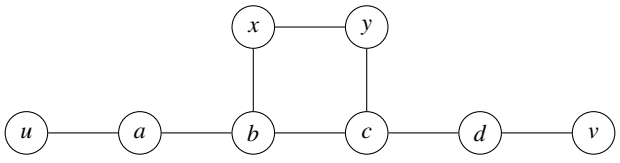


Figure 1: The edge gadget of the VC-IDS reduction: the edge uv of H is replaced by the path u–a–b–c–d–v and the square b–x–y–c.

Proposition 7.1 (VC-IDS identity)**.** *For every graph H without isolated vertices and with m_H edges, $i(h(H)) = \tau(H) + 2m_H$.*

Proof. Let D be an independent dominating set of $h(H)$ and fix the gadget of an edge uv. The vertex x is dominated only from $\{b, x, y\}$ and y only from $\{x, y, c\}$; since b and c are adjacent, D cannot dominate both without containing x or y, and since x and y are adjacent, D contains exactly one of them. Suppose $x \in D$ (the case $y \in D$ is symmetric, with u, a, b and v, d, c exchanged). Then $b, y \notin D$, so c is dominated only from $\{c, d\}$ and D contains c or d; and a is dominated only from $\{u, a\}$, so D contains a when $u \notin D$. Hence D meets the gadget in at least 2 vertices, and in at least 3 when neither u nor v is in D. Let $X = D \cap V(H)$ and let q be the number of edges of H with no endpoint in X; then $|D| \geq |X| + 2m_H + q$. Adding one endpoint of each of these q edges to X gives a vertex cover of H, so $|D| \geq \tau(H) + 2m_H$. Conversely, for a minimum vertex cover C of H, take C and, in every gadget, $\{a, y\}$ if only $v \in C$, $\{d, x\}$ if only $u \in C$, and $\{x, c\}$ if $u, v \in C$. The set is independent and dominates every gadget vertex; a vertex $u \notin C$ has an edge, every edge at u has its other endpoint in C, and the corresponding gadget contains the neighbour a or d of u, so u is dominated. The set has $\tau(H) + 2m_H$ vertices. □

Lemma 7.2 (Decoding a vertex cover)**.** *For every independent dominating set D of $h(H)$, the set C_D formed by $D \cap V(H)$ and one endpoint of every edge of H with no endpoint in D is a vertex cover of H with $|C_D| \leq |D| - 2m_H$, computable in $O(|V(H)| + m_H)$ time.*

Proof. This is the inequality $|D| \geq |X| + 2m_H + q$ of the proof of Proposition 7.1. □

Theorem 7.3 (Conditional consequence for vertex cover)**.** *Assume that for every fixed $\varepsilon_0 > 0$ it is NP-hard to approximate Minimum Vertex Cover within $2 - \varepsilon_0$. If for some fixed $\varepsilon_0 > 0$ the output D of Algorithm 1 satisfies $|D| \leq i(h(H)) + (1 - \varepsilon_0)\,\tau(H)$ for every graph H without isolated vertices, then* P = NP*. The assumption holds if UGC holds [18], and in particular if the theorem reported in [19] holds; without any assumption, such a bound would refute UGC unless* P = NP.

Proof. Isolated vertices belong to no minimum vertex cover and can be removed from H. $h(H)$ has polynomial size, Algorithm 1 runs in polynomial time, and by Lemma 7.2 and Proposition 7.1 the decoded cover satisfies $|C_D| \leq |D| - 2m_H \leq \tau(H) + (1 - \varepsilon_0)\tau(H) = (2 - \varepsilon_0)\tau(H)$. Under the assumption this solves an NP-hard problem in polynomial time. The theorem of Khot and Regev [18] derives the assumption from UGC. □

Bounded degree. Austrin, Khot and Safra proved that, assuming UGC, Minimum Vertex Cover on graphs of maximum degree d is hard to approximate within $2 - (2 + o_d(1)) \log\log d / \log d$ [28]. Since $h(H)$ has maximum degree $\max(\Delta(H), 3)$, the VC-IDS reduction transfers this bound to MIDS instances of bounded degree.

Corollary 7.4 (Bounded-degree instances)**.** *Assume UGC and* P ≠ NP*. For every $\delta > 0$ there is d_0 such that for every $d \geq d_0$ and every polynomial-time algorithm $\mathcal{A}$ that returns independent dominating sets, in particular Algorithm 1, there is a graph H of maximum degree at most d without isolated vertices with*

$$|\mathcal{A}(h(H))| \;>\; i(h(H)) + \Big(1 - (2 + \delta)\frac{\log\log d}{\log d}\Big)\tau(H),$$

and $h(H)$ has maximum degree at most $\max(d, 3)$.

Proof. Write the factor of [28] as $2 - (2 + \varepsilon_d) \log\log d / \log d$ with $\varepsilon_d \to 0$, and choose d_0 with $\varepsilon_d < \delta$ for $d \geq d_0$. If the inequality failed for every such H, Lemma 7.2 and Proposition 7.1 would give a polynomial-time vertex cover algorithm with ratio at most $2 - (2 + \delta) \log\log d / \log d$, a smaller factor, on all graphs of maximum degree at most d (isolated vertices are removed first), which is UG-hard; under P ≠ NP this contradicts UGC. □

Theorem 7.3 and Corollary 7.4 are conditional implications, not claims about Algorithm 1. The theorem of [19] had not been peer reviewed at the time of writing; we cite it as reported and have not checked the formalization. Section 9.5 reports the measured values of $|D| - i(h(H))$ and $|C_D|/\tau(H)$ on VC-IDS instances.

8. A Reduction from Satisfiability

A *literal* is a variable x or its negation $\neg x$, a *clause* is a non-empty set of literals that does not contain a variable together with its negation, and a formula in conjunctive normal form is a list of clauses $C_1, \ldots, C_q$; it is *satisfiable* if some assignment of the variables makes true at least one literal of every clause. Deciding whether a formula is satisfiable is NP-complete [29]. A clause that contains a variable and its negation is satisfied by every assignment and can be deleted, and a formula with an empty clause is unsatisfiable, so these restrictions lose no generality. Let φ be a formula, $x_1, \ldots, x_N$ the variables that occur in it, $N \geq 1$, $\Lambda = \sum_{i=1}^{q} |C_i|$ its number of literal occurrences and $W = \max_i |C_i|$ its largest clause width; a clause holds at most one literal of each variable, so $W \leq N$.

Definition 8.1 (Graph of a formula with t copies)**.** For an integer $t \geq 1$, the graph $G_{\varphi,t}$ has one *literal vertex* $[\ell]$ for each of the $2N$ literals ℓ and t *clause vertices* $c_i^1, \ldots, c_i^t$ for each clause C_i. Its edges are $\{[x_j], [\neg x_j]\}$ for $j = 1, \ldots, N$ and $\{c_i^r, [\ell]\}$ for every i, every r and every literal $\ell \in C_i$. It has $2N + tq$ vertices and $N + t\Lambda$ edges.

For an assignment τ, let D_τ be the set of the vertices $[\ell]$ of the literals ℓ that τ makes true.

Theorem 8.2 (Independent dominating sets of $G_{\varphi,t}$)**.** *Let $t \geq W$. If τ satisfies φ, then D_τ is an independent dominating set of $G_{\varphi,t}$ with N vertices. Every independent dominating set D of $G_{\varphi,t}$ satisfies exactly one of the following: (i) D contains no clause vertex, and then $D = D_\tau$ for an assignment τ that satisfies φ, so $|D| = N$; (ii) D contains all the t copies of $k \geq 1$ clauses and no other clause vertex, and then $|D| = kt + N - \lambda$, where λ is the number of variables x_j with $[x_j] \notin D$ and $[\neg x_j] \notin D$, $2\lambda \leq kW$, $\lambda = 0$ if $k = 1$, and $|D| \geq t + N$. Consequently $i(G_{\varphi,t}) = N$ if φ is satisfiable, and $i(G_{\varphi,t}) \geq t + N$ otherwise.*

Proof. Let τ satisfy φ. D_τ holds exactly one of $[x_j]$ and $[\neg x_j]$ for every j and no clause vertex, and the only edges between literal vertices are the edges $\{[x_j], [\neg x_j]\}$, so D_τ is independent. A literal vertex $[\ell] \notin D_\tau$ is adjacent to $[\neg\ell] \in D_\tau$, and a clause vertex c_i^r is adjacent to $[\ell] \in D_\tau$ for a literal $\ell \in C_i$ that τ makes true. Hence D_τ is an independent dominating set, and $|D_\tau| = N$.

Let D be an independent dominating set. (i) Suppose that D contains no clause vertex. D holds at most one of the adjacent vertices $[x_j]$, $[\neg x_j]$; the neighbours of $[x_j]$ are $[\neg x_j]$ and clause vertices, so $[x_j] \in D$ or $[\neg x_j] \in D$. Hence D holds exactly one literal vertex of every variable, $D = D_\tau$ for the assignment τ that makes the literals of D true, and $|D| = N$. A clause vertex c_i^1 is not in D, so it has a neighbour $[\ell] \in D$ with $\ell \in C_i$, and τ makes ℓ true: τ satisfies φ.

(ii) Suppose that D contains a clause vertex, and let H be the set of the indices i such that D contains a copy of C_i, with $k = |H| \geq 1$. Let $i \in H$ and $c_i^r \in D$. By independence no vertex $[\ell]$ with $\ell \in C_i$ is in D. Every copy c_i^s has exactly these neighbours, so it is not dominated by another vertex and $c_i^s \in D$. Hence D contains the t copies of each clause C_i with $i \in H$ and no other clause vertex, that is, kt clause vertices. D holds at most one of $[x_j]$ and $[\neg x_j]$ for every j, and none of them exactly for the λ variables counted in the statement, so $|D| = kt + N - \lambda$. Let x_j be such a variable. The vertex $[x_j]$ is dominated, its neighbour $[\neg x_j]$ is not in D, and its other neighbours are copies of the clauses that contain x_j, so $x_j \in C_i$ for some $i \in H$; in the same way $\neg x_j \in C_{i'}$ for some $i' \in H$, and $i' \neq i$ because no clause contains both literals of a variable. So every such variable has one literal occurrence in each of two different clauses of H, and different variables give different occurrences. The clauses of H have at most kW literal occurrences, so $2\lambda \leq kW$, and $\lambda = 0$ if $k = 1$. If $k = 1$, then $|D| = t + N$. If $k \geq 2$, then, by $t \geq W$,

$$|D| \geq kt + N - \frac{kW}{2} = t + N + (k-1)t - \frac{kW}{2} \geq t + N + \Big(\frac{k}{2} - 1\Big)W \geq t + N.$$

The two cases exclude each other. If φ is satisfiable, the sets of case (i) exist and have $N < t + N$ vertices, so $i(G_{\varphi,t}) = N$; if not, case (i) does not occur and every independent dominating set has at least $t + N$ vertices. □

Corollary 8.3 (Ratio on the graph of a satisfiable formula)**.** *Let $t \geq W$ and let φ be satisfiable. Every independent dominating set D of $G_{\varphi,t}$ satisfies $|D|/i(G_{\varphi,t}) = 1$, and then $D = D_\tau$ for a satisfying assignment τ, or $|D|/i(G_{\varphi,t}) \geq 1 + t/N$. In particular, for $t = W$, every independent dominating set of ratio smaller than $1 + W/N$ is a satisfying assignment.*

Proof. By Theorem 8.2, $i(G_{\varphi,t}) = N$, and $|D| = N$ in case (i) and $|D| \geq t + N$ in case (ii). □

Theorem 8.4 (Inapproximability within $n^{1-\varepsilon}$)**.** *Let $0 < \varepsilon < 1$. If a polynomial-time algorithm returns, on every graph G with n vertices, an independent dominating set with at most $n^{1-\varepsilon}\, i(G)$ vertices, then* P = NP.

Proof. Let φ be a formula with $N \geq 1$ variables and $q \geq 1$ clauses, and let

$$t = \lfloor (N\,(2N+q)^{1-\varepsilon})^{1/\varepsilon} \rfloor + 1,$$

so that $t^{\varepsilon} > N(2N+q)^{1-\varepsilon}$ and $t > N^{1/\varepsilon} \geq N \geq W$. The graph $G = G_{\varphi,t}$ has $n = 2N + tq \leq t(2N+q)$ vertices, and for a fixed ε its size is polynomial in $N + q + \Lambda$, since $t \leq N^{1/\varepsilon}(2N+q)^{(1-\varepsilon)/\varepsilon} + 1$. Let S be the set that the algorithm returns on G. If φ is satisfiable, $i(G) = N$ by Theorem 8.2, and

$$|S| \leq n^{1-\varepsilon} N \leq t^{1-\varepsilon}(2N+q)^{1-\varepsilon} N < t^{1-\varepsilon}\, t^{\varepsilon} = t < t + N.$$

If φ is unsatisfiable, $|S| \geq i(G) \geq t + N$ by Theorem 8.2. Hence φ is satisfiable if and only if $|S| < t + N$, and in that case $S = D_\tau$ for a satisfying assignment τ, by Theorem 8.2. This decides satisfiability in polynomial time. □

Theorem 8.4 is the bound of Halldórsson [2]. The reduction $\varphi \mapsto G_{\varphi,t}$ proves it for every $\varepsilon > 0$ directly from satisfiability: the graph of a satisfiable formula has an independent dominating set with N vertices, every independent dominating set that is not a satisfying assignment has at least $t + N$ vertices, and no gap is obtained from the PCP theorem.

9. Experiments

All experiments use Siriaisa 0.0.9 and the scripts of the folder `car/` of the repository: `run_car.py` (Section 9.2), `car_bipartite.py` (Section 9.4), `car_ugc.py` (Section 9.5), and `car_split.py`, `car_cubic.py`, `car_circle.py` and `car_bounded.py` (Section 9.6); Section 9.7 uses the command `batch_iris` of the package. The scripts were written with Claude (Claude Opus 5.5) and run by the author; they import the repository's own `siriaisa/`, so they run whether or not the package is installed. The environment is Windows 11, Python 3.12.3, NetworkX 3.6.1 [25], NumPy 2.4.6 and SciPy 1.17.1 [30]; wall-clock times are machine dependent.

9.1. Reference values

The reference values are computed by the scripts only and are never given to Siriaisa; every set returned by Siriaisa or by an integer program is verified to be an independent dominating set (respectively a vertex cover or a set cover) before its size is used.

- *Integer programs.* The optimum $i(G)$ is the value of

$$\min \sum_{v \in V} x_v \quad \text{s.t.} \quad \sum_{u \in N[v]} x_u \geq 1 \;\; \forall v \in V, \qquad x_u + x_v \leq 1 \;\; \forall \{u,v\} \in E, \qquad x \in \{0,1\}^V,$$

  solved by `scipy.optimize.milp`, which calls the HiGHS branch-and-cut solver [31], with a sparse constraint matrix (`scipy.sparse`) and a time limit. When HiGHS reports an optimal status, the rounded objective is $i(G)$. When the time limit is reached, the dual bound `mip_dual_bound` of HiGHS gives the certified lower bound $\lceil L \rceil \leq i(G)$, and the incumbent, if any, an upper bound; the interval $[\lceil L \rceil, \min(|S|, \text{incumbent})]$ is reported and the instance is *open* unless $|S| = \lceil L \rceil$. The vertex cover program ($\min \sum x_v$, $x_u + x_v \geq 1$) and the set cover program ($\min \sum_S x_S$, $\sum_{S \ni e} x_S \geq 1$) are solved in the same way.

- *Linear relaxation of vertex cover.* For the VC-IDS instances, the value of the linear relaxation of vertex cover is half the size of a maximum matching of the bipartite double cover $H \times K_2$ [32]; it is computed with the Hopcroft–Karp algorithm [33] of NetworkX (`bipartite.hopcroft_karp_matching`) and rounded up, a certified lower bound on $\tau(H)$ that needs no time limit.

- *Closed forms and identities.* $i = \lceil n/3 \rceil$ for paths and cycles, $i(K_{a,b}) = \min(a,b)$ and $i = 2$ for crown graphs, checked against the integer program whenever it proves optimality; on a complete multipartite graph, in which every maximal independent set is a whole part, i is the size of the smallest part, and `car_split.py` uses this value for its complete, complete bipartite and complete multipartite graphs without an integer program; $i = \text{sc} + |\mathcal{S}|$ for SC-IDS [10] and $i(h(H)) = \tau(H) + 2m_H$ for VC-IDS (Proposition 7.1), which turn bounds on sc and τ into bounds on i.

- *Constructive upper bounds.* For the Khot–Regev instances, the planted cover of the reduction, made independent greedily on violated constraints, bounds $\tau(H)$ from above, and the construction of Proposition 7.1 turns it into an explicit independent dominating set of $h(H)$, which is verified; the cover C_D decoded from the output of Siriaisa (Lemma 7.2) is a second upper bound.
- *Generators.* All random graphs are built with NetworkX generators (`gnp_random_graph`, `bipartite.random_graph`, `random_regular_graph`, `bipartite.configuration_model`, `random_labeled_tree` and others), each with its own seed derived from the master seed and the instance name.
- *Two runs of the references.* Every integer program, and every reference built from it, was computed in two runs of the scripts on the same instances (the same seeds and generators); the reference of an instance combines them, with the larger of the two certified lower bounds and the smaller of the two upper bounds. The sizes returned by Siriaisa are the same in both runs, and the times reported are those of the second run.

The certified lower bound LB of Lemma 6.3 is used inside Siriaisa to stop its searches, not as a reference value.

9.2. Structured and random families

The script `run_car.py` generates 10 000 instances with at most 40 vertices from fifteen families (master seed 12345) and computes the exact optimum by the integer program with a time limit of 30 s. Every instance is tagged with the bound of Corollary 6.2 for its family: 1 for the rigid families, and the maximum degree for the bounded-degree and random families. Table 5 reports the results. The output is optimal on 9 999 instances; the remaining one is a Barabási–Albert graph with $n = 20$, $m = 36$ and $\Delta = 7$, on which $|S| = 6$ and $i(G) = 5$. No instance exceeds its bound. Siriaisa took 73.8 ms per instance on average, and the integer program 20.0 ms.

Table 5: The suite of 10 000 graphs: number of instances, optimal outputs, mean and maximum ratio $|S|/i(G)$, the largest bound of Corollary 6.2 in the family, the mean maximum degree and the number of instances above the bound.

Family	Kind	Inst.	Opt.	Mean	Max	Bound	Mean Δ	Viol.
clique	rigid	667	667	1.0000	1.0000	1	6.47	0
complete bipartite	rigid	667	667	1.0000	1.0000	1	8.49	0
crown	rigid	666	666	1.0000	1.0000	1	3.09	0
double star	rigid	666	666	1.0000	1.0000	1	9.32	0
star	rigid	667	667	1.0000	1.0000	1	14.98	0
balanced tree	bounded	667	667	1.0000	1.0000	4	3.49	0
cycle	bounded	667	667	1.0000	1.0000	2	2.00	0
grid	bounded	667	667	1.0000	1.0000	4	3.60	0
ladder	bounded	667	667	1.0000	1.0000	3	2.95	0
lollipop	bounded	667	667	1.0000	1.0000	8	5.52	0
path	bounded	667	667	1.0000	1.0000	2	1.98	0
random regular	bounded	667	667	1.0000	1.0000	5	3.53	0
Barabási–Albert	random	666	665	1.0003	1.2000	21	8.91	0
Erdős–Rényi	random	666	666	1.0000	1.0000	20	8.07	0
random tree	random	666	666	1.0000	1.0000	7	3.77	0
All		**10000**	**9999**	**1.0000**	**1.2000**			**0**

9.3. Baseline heuristics

The script `car_baselines.py` runs four baseline heuristics on the 10 000 instances of Section 9.2, with the optima of that section as references: a random maximal independent set of NetworkX (`maximal_independent_set`); the minimum-degree greedy, which selects a free vertex with the fewest free neighbours and deletes its closed neighbourhood; the maximum-degree greedy, which selects a free vertex with the most free neighbours; and the maximum-degree greedy followed by a local search with the 2-swap move of [14], in which two vertices of the set are dropped and one vertex is added, applied until no such move exists. Every set is verified to be an independent dominating set. Table 6 reports, for every family and every method, the number of optimal outputs and the mean and

maximum ratio. Over the 10 000 instances, Siriaisa is optimal on 9 999, the random maximal independent set on 2 301, the minimum-degree greedy on 1 253, the maximum-degree greedy on 7 262, and the maximum-degree greedy with the 2-swap search on 7 920; the largest ratios are 1.2, 30, 30, 2 and 2. The mean time per instance in this run is 29.1 ms for Siriaisa and 0.032 ms, 0.034 ms, 0.022 ms and 0.035 ms for the four baselines.

Table 6: Siriaisa and four baseline heuristics on the suite of 10 000 graphs: number of optimal outputs (Opt.) and mean and maximum ratio to the optimum (file `car_baselines_summary.csv`).

		Siriaisa			Random maximal			Min-degree greedy			Max-degree greedy			Max-degree + 2-swap		
Family	Inst.	Opt.	Mean	Max	Opt.	Mean	Max	Opt.	Mean	Max	Opt.	Mean	Max	Opt.	Mean	Max
balanced_tree	667	667	1.000	1.000	12	2.150	3.333	0	2.658	3.333	329	1.203	1.400	329	1.203	1.400
barabasi_albert	666	665	1.000	1.200	44	1.684	6.000	3	1.921	6.000	452	1.089	2.000	474	1.079	2.000
clique	667	667	1.000	1.000	667	1.000	1.000	667	1.000	1.000	667	1.000	1.000	667	1.000	1.000
complete_bipartite	667	667	1.000	1.000	250	2.352	12.000	57	2.646	12.000	667	1.000	1.000	667	1.000	1.000
crown	666	666	1.000	1.000	251	1.811	3.000	117	2.045	3.000	666	1.000	1.000	666	1.000	1.000
cycle	667	667	1.000	1.000	124	1.210	1.500	70	1.356	1.500	667	1.000	1.000	667	1.000	1.000
double_star	666	666	1.000	1.000	100	2.305	6.500	5	2.599	6.500	666	1.000	1.000	666	1.000	1.000
erdos_renyi	666	666	1.000	1.000	71	1.472	2.667	27	1.704	3.000	332	1.146	2.000	417	1.095	2.000
grid	667	667	1.000	1.000	123	1.361	2.000	24	1.629	2.000	111	1.309	1.667	248	1.197	1.667
ladder	667	667	1.000	1.000	72	1.439	1.857	33	1.673	1.889	75	1.268	1.500	260	1.117	1.333
lollipop	667	667	1.000	1.000	165	1.325	2.000	47	1.472	2.000	667	1.000	1.000	667	1.000	1.000
path	667	667	1.000	1.000	90	1.277	2.000	31	1.459	2.000	667	1.000	1.000	667	1.000	1.000
random_tree	666	666	1.000	1.000	94	1.373	3.000	40	1.548	3.000	385	1.075	2.000	468	1.046	1.667
regular	667	667	1.000	1.000	147	1.274	2.500	99	1.417	2.500	244	1.161	1.750	390	1.081	1.667
star	667	667	1.000	1.000	91	14.063	30.000	33	14.979	30.000	667	1.000	1.000	667	1.000	1.000
All	10000	9999	1.000	1.200	2301	2.407	30.000	1253	2.674	30.000	7262	1.083	2.000	7920	1.055	2.000

9.4. Bipartite graphs

MIDS is NP-complete on bipartite graphs [4], so on these graphs there is no Kőnig-type formula for the optimum. The script `car_bipartite.py` (option `-quick`) generates 112 bipartite graphs with 64 to 256 vertices from sixteen families: random bipartite graphs $G(n_1, n_2, p)$, balanced and with sides $1 : 2$, of average degree 3, 6 and 20; random d-regular bipartite graphs ($d = 3, 4, 8$); bipartite graphs with power-law degrees; double covers $G \times K_2$ of random 3- and 5-regular graphs; a perfect matching plus random edges; random trees, paths and even cycles; 2D and 3D grids, 2D grids with 30% of the edges deleted; complete bipartite and crown graphs; hypercubes Q_6, Q_8 and their random halves. Labels are shuffled and every instance has its own generator, derived from the seed 20261031 and its name. The reference is the integer program with a time limit of 60 s and the closed forms of Section 9.1.

Table 7 reports the results. The optimum was proved on 95 instances, and the output is optimal on all of them. On the 17 open instances, all with $n \geq 200$, the ratio column gives the certified upper bound $|S|/\lceil L \rceil$; on 8 of them $|S|$ equals the best solution found by the integer programs, on 7 it is smaller, and on two ($G(n/3, 2n/3, p)$ with $n = 200$ and average degree 20) it exceeds it by one vertex (15 against 14, and 16 against 15). The strategies c_9 and c_{11} were exact (one forest layer, empty core) on the 8 trees and paths, as Theorem 6.4 requires. Siriaisa took at most 12.1 s per instance and 391.0 s in total; the integer programs took 1 453.9 s and the whole run 1 847.2 s.

Table 7: Bipartite graphs: instances, instances with the optimum proved ("Ref."), optimal, suboptimal and open outputs, maximum and mean ratio (certified upper bounds on open instances), and the time of Siriaisa in seconds.

Family	Inst.	Ref.	Opt.	Subopt.	Open	Max ratio	Mean ratio	Time
2D grid	2	2	2	0	0	1.0000	1.0000	4.8
2D grid, 70% edges kept	6	6	6	0	0	1.0000	1.0000	30.0
3D grid	2	1	1	0	1	1.0750	1.0375	5.6
complete bipartite	2	2	2	0	0	1.0000	1.0000	3.4
crown graph	2	2	2	0	0	1.0000	1.0000	0.2
double cover $G \times K_2$ of random regular	12	8	8	0	4	1.1765	1.0370	46.9
even cycle	2	2	2	0	0	1.0000	1.0000	0.0
hidden perfect matching + random edges	12	12	12	0	0	1.0000	1.0000	42.3
hypercube	2	2	2	0	0	1.0000	1.0000	4.9
hypercube, 50% edges kept	2	2	2	0	0	1.0000	1.0000	10.9
path	2	2	2	0	0	1.0000	1.0000	0.0
power-law bipartite	6	6	6	0	0	1.0000	1.0000	25.1
random bipartite (1:2)	18	15	15	0	3	1.4545	1.0657	56.9
random bipartite (balanced)	18	15	15	0	3	1.4167	1.0602	60.7
random d-regular bipartite	18	12	12	0	6	1.2917	1.0623	78.9
random tree	6	6	6	0	0	1.0000	1.0000	20.3
All	**112**	**95**	**95**	**0**	**17**	**1.4545**	**1.0349**	**391.0**

9.5. Instances of the hardness reductions

The script `car_ugc.py` (option `-quick`, integer programs limited to 60 s) builds 55 instances of the two reductions of Section 7. The SC-IDS instances are bipartite, of maximum degree $B \in \{4, 5, 6\}$, built from random set systems with 30 or 60 sets of size $B - 1$ in which every element lies in at most B sets; the reference is the set cover integer program and $i = \mathrm{sc} + |\mathcal{S}|$. The VC-IDS instances are $h(H)$, none of whose graphs H has an isolated vertex, for random d-regular graphs H with $d \in \{3, 4, 5\}$ and 50, 100 or 200 vertices, of maximum degree d, and for ten Khot–Regev YES instances H with 640 vertices (seed 20261009, the instances and names of Part D of the Salvador experiments [21]: a Unique Games instance on a random 4-regular graph with 40 vertices and a planted labelling, 16 subsets per block drawn from the p-biased measure, the base family $k = 4$, $p = 0.45$, $\eta = 0$ and the variants $k = 3$, $p = 0.35$, $\eta = 0.1$ and linear constraints, two instances each [18]); these $h(H)$ have 48 172 to 75 166 vertices and maximum degree 64. The reference is $\tau(H)$: L_τ is the larger of the linear relaxation and the integer-program dual bound, and U_τ the smaller of the integer-program value, the planted cover and the decoded cover of Lemma 7.2; by Proposition 7.1, $L_\tau + 2m_H \le i(h(H)) \le U_\tau + 2m_H$, with equality when $L_\tau = U_\tau$ (Section 9.1).

Table 8 reports the results. On the 18 SC-IDS instances the output is optimal. On the VC-IDS instances $\tau(H)$ was proved on 30 of 37; on these the output is optimal on 2 (one 3-regular and one 4-regular graph with 50 vertices) and exceeds the optimum on 28; the certified ratio $|D|/L$ is at most 1.0175 on the instances of Khot and Regev and at most 1.0375 on the regular graphs, and the largest excess $|D| - L$ is 292 vertices. The decoded cover satisfies $|C_D|/L_\tau \le 1.35$ on the regular graphs and lies between 1.25 and 1.8665 on the instances of Khot and Regev (mean 1.4826), where the planted cover has 347 to 416 vertices and the decoded cover 480 to 629. The random regular graphs H are not the hard instances of [28], so they measure the algorithm on bounded-degree VC-IDS graphs without testing Corollary 7.4. The whole run took 1 654.0 s; Siriaisa took at most 72.8 s per instance.

Table 8: Instances of the reductions of Chlebík and Chlebíková. "Ref." counts instances with i proved; "Opt.", "Subopt." and "Open" classify $|D|$ against it; "max $|D|/L$" is the largest certified ratio, "max exc." the largest $|D| - L$; the last two columns give the number of optimal decoded covers and the largest $|C_D|/L_\tau$.

Family	Inst.	n	Δ	Ref.	Opt.	Subopt.	Open	max $\lvert D\rvert/L$	max exc.	C_D opt. / max $\lvert C_D\rvert/L_\tau$
SC-IDS, $B = 4$	6	113–225	4	6	6	0	0	1.0000	0	—
SC-IDS, $B = 5$	6	108–228	5	6	6	0	0	1.0000	0	—
SC-IDS, $B = 6$	6	106–224	6	6	6	0	0	1.0000	0	—
VC-IDS, 3-regular H	9	500–2 000	3	9	1	8	0	1.0309	22	1 / 1.1982
VC-IDS, 4-regular H	9	650–2 600	4	8	1	7	1	1.0360	33	1 / 1.2821
VC-IDS, 5-regular H	9	800–3 200	5	6	0	6	3	1.0375	42	0 / 1.3500
VC-IDS, Khot–Regev H	10	48 172–75 166	64	7	0	7	3	1.0175	292	0 / 1.8665
All	**55**			**48**	**20**	**28**	**7**	**1.0375**	**292**	**2 / 1.8665**

9.6. The splitting excess on hard families

Four scripts measure the splitting excess $s(G)$ and the output on the families of Section 6; all of them were run with the option `-quick`, so that $n \le 400$ on the random families, and the integer program has a time limit of 30 s for $n \le 400$ (`car_split.py`, `car_cubic.py`, `car_circle.py`) or of 60 s for $n \le 1\,100$ (`car_bounded.py`). Each script asserts on every instance the statements of Section 6 that apply to it, and no assertion failed.

Hard and structured families. The script `car_split.py` (seed 20261101) builds 110 instances of 29 families and computes, on each, k, B, $s(G)$, the copy graph H of Algorithm 5, the decoded set S_0 of each inner solver and whether it is independent in G; it asserts the bounds of Proposition 6.9(ii)–(iii), that $|V(H)| = n - |B| + \sum_{v\in B}\lceil \deg(v)/k\rceil$ and $\Delta(H) \le k$, that every S_0 dominates G (Lemma 6.7), and that $|S_0| \le k\,(U + s(G))$ (Theorem 6.8). Table 9 reports the results. $s(G) = 0$ on all 54 instances of maximum degree at most k; the largest excess, $7\,600 = s(K_{400})$, is attained by the complete graph, and $s(G)/n^{3/2} \le 0.950$ on all instances. The decoded set S_0 of some inner solver is independent on 69 instances, 15 of them with $s(G) > 0$, and the largest value of $|S_0|/(k\,(U + s(G)))$ is 0.1264. The optimum is proved on 87 instances, among them the 20 complete, complete bipartite and complete multipartite graphs, for which i is the size of the smallest part (Section 9.1); the output is optimal on 79 of them and exceeds the optimum by 1 to 5 vertices on the other 8 (a Delaunay graph, an SC-IDS graph, $G(n, 3/n)$, two Barabási–Albert graphs and three Khot–Regev graphs of constraint degree 3). On the 23 open instances the lower bound is the integer-program dual bound for $n \le 400$ and $\lceil n/(\Delta + 1)\rceil$ otherwise; the ratios 15.9 and 22.2 of the Khot–Regev and VC-IDS Khot–Regev graphs are ratios to $\lceil n/(\Delta + 1)\rceil$. The run took 1 542.9 s.

Table 9: Splitting excess on hard and structured families ($k = \lfloor \sqrt{n} \rfloor$): largest maximum degree, instances with $s(G) = 0$, largest $s(G)$ and $s(G)/n^{3/2}$, and instances on which the decoded set S_0 of some inner solver is independent. "Ref." counts instances with $i(G)$ proved (integer program, closed form, planted value or reduction identity); "Opt.", "Subopt." and "Open" classify the output against it; "Max ratio" is the largest $|S|/L$ for the certified lower bound L (an upper bound on the ratio of open instances); "Time" is the total time of SIRIAISA in seconds. *Union of d random perfect matchings with repeated edges merged (maximum degree d).

Family	Inst.	n	Δ	$s = 0$	max s	max $s/n^{3/2}$	S_0 ind.	Ref.	Opt.	Subopt.	Open	Max ratio	Time
random 3-regular bipartite*	4	100–400	3	4	0	0.000	4	2	2	0	2	1.0784	38.0
planar (Delaunay)	4	100–400	12	4	0	0.000	4	4	3	1	0	1.0161	33.3
random 4-regular	4	100–400	4	4	0	0.000	4	2	2	0	2	1.1481	35.5
random 5-regular	4	100–400	5	4	0	0.000	4	2	2	0	2	1.2239	34.5
random 6-regular	4	100–400	6	4	0	0.000	4	2	2	0	2	1.2586	36.3
random 4-regular bipartite*	4	100–400	4	4	0	0.000	4	2	2	0	2	1.1481	32.3
random 5-regular bipartite*	4	100–400	5	4	0	0.000	4	2	2	0	2	1.2059	30.0
SC-IDS, $B = 4$	4	94–375	4	4	0	0.000	4	4	3	1	0	1.0080	33.0
SC-IDS, $B = 6$	4	93–384	6	4	0	0.000	4	4	4	0	0	1.0000	25.0
VC-IDS, 3-regular H	4	240–1 000	3	4	0	0.000	4	2	2	0	2	1.4400	59.2
$G(n, 1/2)$	4	100–400	230	0	3 783	0.473	0	2	2	0	2	2.5000	19.5
complete K_n	4	100–400	399	0	7 600	0.950	0	4	4	0	0	1.0000	4.9
$K_{n/2,n/2}$	4	100–400	200	0	3 600	0.450	4	4	4	0	0	1.0000	19.0
$K_{n/4,3n/4}$	4	100–400	300	0	2 600	0.325	2	4	4	0	0	1.0000	13.8
complete 4-partite, equal parts	4	100–400	300	0	5 600	0.700	0	4	4	0	0	1.0000	22.0
complete 5-partite, random parts	4	100–400	392	0	5 889	0.736	0	4	4	0	0	1.0000	17.8
crown	4	100–400	199	0	3 600	0.450	0	4	4	0	0	1.0000	2.8
star	4	100–400	399	0	19	0.009	4	4	4	0	0	1.0000	0.1
wheel	4	100–400	399	0	19	0.009	4	4	4	0	0	1.0000	0.2
double star	4	100–400	200	0	18	0.008	0	4	4	0	0	1.0000	25.3
split graph	4	100–400	232	0	209	0.055	0	4	4	0	0	1.0000	7.7
$G(n, 3/n)$	4	100–400	11	4	0	0.000	4	4	3	1	0	1.0075	47.6
Barabási–Albert, $m = 3$	4	100–400	67	0	19	0.013	0	4	2	2	0	1.0532	25.8
power-law bipartite	4	100–400	55	0	6	0.004	1	4	4	0	0	1.0000	43.9
2D grid	4	100–400	4	4	0	0.000	4	2	2	0	2	1.0814	31.2
hypercube	2	128–512	9	2	0	0.000	2	1	1	0	1	1.2308	4.8
Khot–Regev YES	2	640	64	0	323	0.020	0	0	0	0	2	15.9000	10.2
Khot–Regev YES, $d = 3$, $s_b \in \{4, 8\}$	4	160–320	24	2	39	0.007	2	4	1	3	0	1.0361	23.7
VC-IDS, Khot–Regev H	2	49 096–51 550	64	2	0	0.000	2	0	0	0	2	22.1825	87.0
All	**110**			**54**	**7 600**	**0.950**	**69**	**87**	**79**	**8**	**23**	**22.1825**	**764.3**

Cubic graphs. The script `car_cubic.py` (seed 20261102) builds 61 instances: every cubic graph on 4, 6 and 8 vertices (one, two and six graphs, collected as isomorphism classes of random 3-regular graphs), six random graphs of maximum degree 3 on 3 to 8 vertices, twelve named cubic graphs, and random cubic, cubic bipartite (three edge-disjoint random perfect matchings), double covers of random cubic graphs, cubic planar (duals of Delaunay triangulations with a triangular outer face), cubic planar bipartite (grown from the cube by inserting two rungs across an even face, which preserves planarity, bipartiteness and degree 3), random cubic graphs with 20% of the edges deleted, prisms, Möbius ladders and generalized Petersen graphs with 100 to 402 vertices. It asserts Proposition 6.13(i)–(ii), that the copy graph equals G when $s(G) = 0$, and Theorem 6.14: $|S| \le n/2$ on regular graphs and $|S| \le 2L$ on cubic graphs. Table 10 reports the results. On the cubic graphs, $s(G) = n$ for $n \in \{4, 6, 8\}$ and $s(G) = 0$ for $n \ge 10$, and on the graphs of maximum degree 3 with $n \le 8$, $s(G) \le 6$. The optimum is proved on 49 instances and the output is optimal on all of them; on the 12 open instances, all with 400 or 402 vertices, $|S|/L \le 1.0792$ except on $GP(200, 2)$, where $|S| = 120$ and $L = 103$. $|S|/n$ is at most 0.5, attained by $K_{3,3}$, and at most 0.3333 for $n \ge 10$; strategy c_{13} alone is optimal on 15 of the 30 cubic graphs with $n \ge 10$ and a proved optimum. The run took 564.1 s.

Table 10: Cubic and subcubic graphs: the values of $s(G)$, instances on which the copy graph is G (no vertex split), the classification of the output, and the largest $|S|/n$. "Ref." counts instances with $i(G)$ proved (integer program, closed form, planted value or reduction identity); "Opt.", "Subopt." and "Open" classify the output against it; "Max ratio" is the largest $|S|/L$ for the certified lower bound L (an upper bound on the ratio of open instances); "Time" is the total time of Sriaisa in seconds. †Petersen, Heawood, Möbius–Kantor, Pappus, Desargues, dodecahedron, truncated tetrahedron, truncated cube, Frucht, Tutte, McGee and Foster graphs.

Family	Inst.	n	$s(G)$	$H = G$	Ref.	Opt.	Subopt.	Open	Max ratio	max $\lvert S\rvert/n$	Time
cubic, $n = 4$	1	4	4	0	1	1	0	0	1.0000	0.2500	0.0
cubic, $n = 6$	2	6	6	0	2	2	0	0	1.0000	0.5000	0.0
cubic, $n = 8$	6	8	8	0	6	6	0	0	1.0000	0.3750	0.0
$\Delta \le 3$, $n \le 8$	6	3–8	1,4,5,6	0	6	6	0	0	1.0000	0.4000	0.0
named cubic graphs†	12	10–90	0	12	12	12	0	0	1.0000	0.3333	2.1
random cubic	4	100–400	0	4	2	2	0	2	1.0693	0.2700	19.6
random cubic bipartite	4	100–400	0	4	2	2	0	2	1.0693	0.2700	25.1
double cover of random cubic	4	100–400	0	4	2	2	0	2	1.0693	0.2800	24.5
cubic planar (dual of Delaunay)	4	100–400	0	4	2	2	0	2	1.0792	0.2725	25.0
cubic planar bipartite	4	100–400	0	4	2	2	0	2	1.0588	0.2800	20.8
random cubic, 20% edges deleted	4	100–400	0	4	4	4	0	0	1.0000	0.3250	25.1
prism $C_t \times K_2$	4	100–402	0	4	3	3	0	1	1.0099	0.2600	13.2
Möbius ladder	2	100–400	0	2	2	2	0	0	1.0000	0.2525	10.3
generalized Petersen $GP(t, 2)$	2	100–400	0	2	1	1	0	1	1.1650	0.3000	11.7
generalized Petersen $GP(t, 3)$	2	100–400	0	2	2	2	0	0	1.0000	0.2600	5.2
All	**61**			**46**	**49**	**49**	**0**	**12**	**1.1650**	**0.5000**	**182.7**

Circle graphs. The script `car_circle.py` (seed 20261103) builds 40 circle graphs from chord models with 100 and 400 chords: chords with uniform random endpoints, short random chords of expected degree about 3, 10 and $\sqrt{n}$, permutation graphs, two or four long chords each crossed by nested short chords, and the models of Proposition 6.16(ii) of K_n, $K_{n/2,n/2}$ and $K_{n/4,3n/4}$, a star, a path and a cycle. It computes the degrees and $s(G)$ from the model by Proposition 6.16(i), checks them against the circle graph built from the crossings, and checks that the models of the complete, complete bipartite, star, path and cycle families give these graphs. Table 11 reports the results. The degrees from the model agree with the graph on all instances; computing $s(G)$ from the model took at most 1.4 ms per instance and building the graph at most 49 ms. The excess reaches $s(K_{400}) = 7\,600$ on the diameters, $s(G)/s(K_n) \le 0.563$ on the other families, and $s(G) = 0$ only on paths, cycles and three of the four sparsest random models. The optimum is proved on 38 instances, with the closed forms $i(K_{a,b}) = \min(a, b)$ and $i = \lceil n/3 \rceil$ for paths and cycles; the output is optimal on 34 and exceeds the optimum by 1 to 3 vertices on four graphs of short chords with 400 chords; the two open instances are uniform random models with 400 chords, with $|S|/L \le 1.1905$. The run took 283.8 s.

Table 11: Circle graphs built from chord models: largest number of edges and maximum degree, instances with $s(G) = 0$, largest $s(G)$ and $s(G)/s(K_n)$, instances with an independent decoded set S_0. "Ref." counts instances with $i(G)$ proved (integer program, closed form, planted value or reduction identity); "Opt.", "Subopt." and "Open" classify the output against it; "Max ratio" is the largest $|S|/L$ for the certified lower bound L (an upper bound on the ratio of open instances); "Time" is the total time of Sriaisa in seconds.

Family	Inst.	n	max m	Δ	$s = 0$	max s	max $s/s(K_n)$	S_0 ind.	Ref.	Opt.	Subopt.	Open	Max ratio	Time
uniform random chords	4	100–400	27 268	219	0	2 513	0.331	0	2	2	0	2	1.1905	6.5
short chords, degree ≈ 3	4	100–400	904	14	3	2	0.002	3	4	4	0	0	1.0000	19.4
short chords, degree ≈ 10	4	100–400	2 500	29	0	107	0.119	0	4	2	2	0	1.0405	12.5
short chords, degree $\approx \sqrt{n}$	4	100–400	5 488	57	0	354	0.092	0	4	2	2	0	1.0600	8.6
permutation graph	4	100–400	41 274	393	0	3 925	0.563	0	4	4	0	0	1.0000	11.0
2 hub chords	4	100–400	532	102	0	15	0.007	1	4	4	0	0	1.0000	16.5
4 hub chords	4	100–400	599	54	0	14	0.006	1	4	4	0	0	1.0000	16.9
diameters (K_n)	2	100–400	79 800	399	0	7 600	1.000	0	2	2	0	0	1.0000	0.8
$K_{n/2,n/2}$	2	100–400	40 000	200	0	3 600	0.474	2	2	2	0	0	1.0000	4.4
$K_{n/4,3n/4}$	2	100–400	30 000	300	0	2 600	0.361	1	2	2	0	0	1.0000	5.0
star $K_{1,n-1}$	2	100–400	399	399	0	19	0.010	2	2	2	0	0	1.0000	0.0
path	2	100–400	399	2	2	0	0.000	2	2	2	0	0	1.0000	0.0
cycle	2	100–400	400	2	2	0	0.000	2	2	2	0	0	1.0000	0.1
All	**40**				**7**	**7 600**	**1.000**	**14**	**38**	**34**	**4**	**2**	**1.1905**	**101.7**

Bounded degree with $n \ge B^2$. On graphs of maximum degree at most B with $n \ge B^2$, $s(G) = 0$ (Proposition 6.12(a)), and MIDS is NP-hard to approximate within δB for every $B \ge B_0$ [10]. The script `car_bounded.py` (seed 20261104) builds 44 such graphs for $B \in \{4, 8, 16, 32\}$ with $n \ge \max(B^2, 256)$: random B-regular graphs, double covers of random B-regular graphs, B-regular graphs with a planted perfect code, SC-IDS graphs of degree B and VC-IDS graphs $h(H)$

of random B-regular graphs H, and Khot–Regev YES graphs with constraint degree 3 and $s_b \in \{4, 8\}$ samples per block ($\Delta \le 3s_b$, with $9s_b + 4$ blocks). A planted perfect code is a set D of t vertices, each with B private neighbours, the tB other vertices being joined by a random $(B-1)$-regular graph; the graph is B-regular, D is an independent dominating set, and $|D| = n/(B+1) = \lceil n/(\Delta+1) \rceil$, so $i(G) = |D|$ by Lemma 6.3. For SC-IDS and VC-IDS the identities $i = \mathrm{sc} + |S|$ and $i = \tau(H) + 2m_H$, with sc and $\tau(H)$ bounded by integer programs, narrow the interval. The script asserts $s(G) = 0$, $|S| \le n/2$ on the regular graphs and that the planted code has size $\lceil n/(\Delta+1) \rceil$. Table 12 reports the results. $s(G) = 0$ on all 44 instances. The optimum is proved on 26 instances and the output is optimal on 18 of them, among them all 8 planted perfect codes, on graphs with up to 1 056 vertices and 16 896 edges; it exceeds the optimum by 1 vertex on two SC-IDS graphs and three VC-IDS graphs of degree 16 and 32, and by 2, 1 and 4 vertices on three Khot–Regev graphs. On the 16 open random regular graphs the integer program did not improve on $\lceil n/(\Delta+1) \rceil$, and $|S|/L$ grows from 1.1346 for $B = 4$ to 2.2500 for $B = 32$; on the two open SC-IDS graphs of degree 32 the output has 357 and 355 vertices against the integer-program bound 348 and $L = 338$. The largest time of Siriaisa on an instance is 12.7 s, and the run took 1 579.5 s.

Table 12: Graphs of maximum degree B with $n \ge B^2$, on which $s(G) = 0$: instances, number of vertices and largest number of edges, instances with $s(G) = 0$, the classification of the output, and the largest and mean $|S|/L$. "Ref." counts instances with $i(G)$ proved (integer program, closed form, planted value or reduction identity); "Opt.", "Subopt." and "Open" classify the output against it; "Max ratio" is the largest $|S|/L$ for the certified lower bound L (an upper bound on the ratio of open instances); "Time" is the total time of Siriaisa in seconds.

Family	B	Inst.	n	max m	$s = 0$	Ref.	Opt.	Subopt.	Open	Max ratio	Mean ratio	Time
random B-regular	4	2	256	512	2	0	0	0	2	1.1346	1.1346	10.9
random B-regular bipartite (double cover)	4	2	256	512	2	0	0	0	2	1.1154	1.1058	10.9
B-regular, planted perfect code	4	2	260	520	2	2	2	0	0	1.0000	1.0000	1.1
SC-IDS	4	2	323	430	2	2	2	0	0	1.0000	1.0000	18.4
VC-IDS, B-regular H	4	2	260	320	2	2	2	0	0	1.0000	1.0000	20.0
random B-regular	8	2	256	1 024	2	0	0	0	2	1.3793	1.3621	8.5
random B-regular bipartite (double cover)	8	2	256	1 024	2	0	0	0	2	1.3448	1.3448	6.7
B-regular, planted perfect code	8	2	261	1 044	2	2	2	0	0	1.0000	1.0000	0.6
SC-IDS	8	2	331–334	769	2	2	2	0	0	1.0000	1.0000	13.5
VC-IDS, B-regular H	8	2	250	320	2	2	2	0	0	1.0000	1.0000	24.8
random B-regular	16	2	256	2 048	2	0	0	0	2	1.5625	1.5312	4.0
random B-regular bipartite (double cover)	16	2	256	2 048	2	0	0	0	2	1.5625	1.5625	5.0
B-regular, planted perfect code	16	2	272	2 176	2	2	2	0	0	1.0000	1.0000	0.7
SC-IDS	16	2	318–321	1 353	2	2	0	2	0	1.0115	1.0114	11.4
VC-IDS, B-regular H	16	2	882	1 152	2	2	1	1	0	1.0033	1.0016	19.8
random B-regular	32	2	1 024	16 384	2	0	0	0	2	2.2188	2.2188	5.8
random B-regular bipartite (double cover)	32	2	1 024	16 384	2	0	0	0	2	2.2500	2.2500	4.2
B-regular, planted perfect code	32	2	1 056	16 896	2	2	2	0	0	1.0000	1.0000	1.5
SC-IDS	32	2	1 313	10 787	2	0	0	0	2	1.0562	1.0533	7.1
VC-IDS, B-regular H	32	2	3 298	4 352	2	2	0	2	0	1.0009	1.0009	13.1
Khot–Regev YES, $d = 3$, $s_b = 4$	12	2	160	382	2	2	1	1	0	1.0364	1.0182	10.6
Khot–Regev YES, $d = 3$, $s_b = 8$	24	2	608	2 938	2	2	0	2	0	1.0237	1.0149	11.2
All		**44**			**44**	**26**	**18**	**8**	**18**	**2.2500**	**1.2095**	**209.7**

9.7. Complements of the DIMACS clique benchmark

The complements of the 68 graphs of the DIMACS clique benchmark [34], in the form distributed with the NPBench collection [35], were solved with `batch_iris` on the folder `experiment/` of the repository, with the size option and logging; Table 13 reports the size of the returned independent dominating set and the time of Siriaisa recorded in the log. The optimum is not known for these instances, and no integer program was run. The time ranged from 0.028 s (`MANN_a9`) to 11.9 s (`c-fat500-2`), with mean 3.47 s and total 236.0 s.

Table 13: Complements of the DIMACS clique benchmark graphs: size $|S|$ of the returned set and time of Sriaisa in seconds.

Instance	$\lvert S\rvert$	Time	Instance	$\lvert S\rvert$	Time
brock200_1	8	1.01	brock200_2	4	1.28
brock200_3	5	1.07	brock200_4	6	1.09
brock400_1	10	1.66	brock400_2	11	1.52
brock400_3	11	1.54	brock400_4	11	1.61
brock800_1	8	7.39	brock800_2	9	6.74
brock800_3	9	7.11	brock800_4	9	10.11
c-fat200-1	10	2.27	c-fat200-2	22	2.35
c-fat200-5	56	2.58	c-fat500-1	12	11.69
c-fat500-10	124	7.12	c-fat500-2	24	11.85
c-fat500-5	62	8.88	C1000.9	31	6.95
C125.9	14	3.50	C250.9	18	2.35
C500.9	25	2.87	gen200_p0.9_44	16	2.55
gen200_p0.9_55	16	2.82	gen400_p0.9_55	21	2.50
gen400_p0.9_65	22	2.53	gen400_p0.9_75	24	2.41
hamming10-2	128	5.12	hamming10-4	13	9.62
hamming6-2	12	1.74	hamming6-4	2	0.10
hamming8-2	32	4.48	hamming8-4	4	4.11
johnson16-2-4	8	3.09	johnson32-2-4	16	3.61
johnson8-2-4	4	1.13	johnson8-4-4	7	4.07
keller4	5	2.83	keller5	10	9.46
MANN_a27	27	0.14	MANN_a45	45	0.20
MANN_a81	81	0.35	MANN_a9	9	0.03
p_hat1000-3	15	7.82	p_hat300-1	3	2.09
p_hat300-2	5	1.59	p_hat300-3	10	1.13
p_hat500-1	3	4.98	p_hat500-2	7	3.96
p_hat500-3	11	2.12	p_hat700-1	3	9.54
p_hat700-2	7	7.03	p_hat700-3	14	3.97
san200_0.7_1	7	1.05	san200_0.7_2	6	1.23
san200_0.9_1	16	1.56	san200_0.9_2	16	1.48
san200_0.9_3	15	1.47	san400_0.5_1	4	2.64
san400_0.7_1	8	1.95	san400_0.7_2	8	2.03
san400_0.7_3	8	2.01	san400_0.9_1	22	1.43
sanr200_0.7	7	1.06	sanr200_0.9	16	1.45
sanr400_0.5	5	2.78	sanr400_0.7	9	2.25

10. Conclusion

Sriaisa 0.0.9 returns an independent dominating set (Theorem 4.3) in $O(n + m)$ expected time for the default parameters (Theorem 5.2). It combines twelve base strategies, five improvers and a refinement; the exact solver of the ensemble is a conditioned dynamic program on induced forests, which replaces the bipartite exact solvers of Salvador because MIDS is NP-complete on bipartite graphs. The output has ratio at most Δ (Corollary 6.2), is optimal on forests (Theorem 6.4), and has ratio at most $\sqrt{n}$ on the classes of Theorem 6.5; the degree-splitting strategy yields a dominating set of size at most $\lfloor \sqrt{n} \rfloor (i(G) + s(G))$ on every graph, and the output satisfies the same bound when the decoded set is independent (Theorem 6.8); the excess $s(G)$ is largest on complete graphs, where the output is optimal, and it vanishes on the NP-hard and UG-hard families of bounded degree once $n \geq \Delta^2$ (Propositions 6.9, 6.10 and 6.12). On cubic graphs, on which MIDS is APX-hard, $s(G) \leq 8$, with $s(G) = 0$ for $n \geq 10$, and the output has ratio at most 2 (Proposition 6.13 and Theorem 6.14); on planar graphs with $n \geq 400$, $s(G) < 2\sqrt{n} + 19$, and $K_{2,n-2}$ attains $s \geq 2\sqrt{n} - 3$ (Proposition 6.15). On circle graphs, on which MIDS is not approximable within n^{ε} for any $\varepsilon < 1$ unless P = NP, $s(G)$ can be computed from a chord model in $O(n \log n)$ time and reaches its largest value $s(K_n)$ (Proposition 6.16), and Conjecture 6.17 restricted to circle graphs would imply P = NP (Corollary 6.20). The graph of a formula with t copies of every clause, t at least the largest clause width W, has only independent dominating sets with N vertices, the satisfying assignments, or with at least $t + N$ vertices (Theorem 8.2), so that with $t = W$ every independent dominating set of ratio smaller than $1 + W/N$ is a satisfying assignment (Corollary 8.3); it proves the inapproximability of MIDS within $n^{1-\varepsilon}$ for every $\varepsilon > 0$ directly from satisfiability, without a gap from the PCP theorem (Theorem 8.4). On 10 000 structured and random graphs the output is optimal on 9 999; on 112 bipartite graphs it is optimal on all 95 instances with a proved optimum, and 17 are open; on the instances of the reductions of Chlebík and Chlebíková it is optimal on the 18 SC-IDS instances and on 2 of the 30 VC-IDS instances with a proved optimum, and the vertex cover decoded on the Khot–Regev instances has ratio between 1.25 and 1.8665. In the experiments of Section 9.6, the output is optimal on all 49 cubic graphs with a proved optimum, on all 8 planted perfect codes of degree up to 32, on 34 of the 38

circle graphs with a proved optimum, and on 79 of the 87 instances of the hard and structured families with a proved optimum.

We do not claim that Siriaisa has ratio $O(\sqrt{n})$ on every graph; this is Conjecture 6.17, and a proof would imply P = NP (Theorem 6.19). Nor do we claim any bound of the form of Theorem 7.3 or Corollary 7.4, which would refute UGC unless P = NP, and would imply P = NP if the theorem reported in [19] holds. The implementation is freely available as the `siriaisa` package on PyPI [20].

Acknowledgements

The author would like to thank Iris, Marilin, Sonia, Yoselin, and Arelis for their support.

Statements and Declarations

Declaration of interests

The author declares no competing interests.

Funding sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of generative AI in scientific writing

The author designed the strategies of Siriaisa, and used Claude Opus 5.5 to implement them by vibe coding, to write the experiment scripts of the folder `car/`, and to improve the readability of this manuscript; the author ran all the experiments. After using this tool, the author reviewed and edited the code and the content as needed and takes full responsibility for the content of the publication.

Data availability

Code and data: https://github.com/frankvegadelgado/mids (release `v0.0.9`); the experiments of Section 9 are reproduced by running the scripts of `car/` and the command `batch_iris` on the folder `experiment/`.

References

[1] R. W. Irving, On approximating the minimum independent dominating set, Information Processing Letters 37 (4) (1991) 197–200. doi:10.1016/0020-0190(91)90188-N.

[2] M. M. Halldórsson, Approximating the minimum maximal independence number, Information Processing Letters 46 (4) (1993) 169–172. doi:10.1016/0020-0190(93)90022-2.

[3] M. Damian-Iordache, S. V. Pemmaraju, Hardness of approximating independent domination in circle graphs, in: Algorithms and Computation (ISAAC 1999), Vol. 1741 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1999, pp. 56–69. doi:10.1007/3-540-46632-0_7.

[4] Y. A. Loverov, Y. L. Orlovich, NP-completeness of the independent dominating set problem in the class of cubic planar bipartite graphs, Journal of Applied and Industrial Mathematics 14 (2) (2020) 352–366. doi:10.1134/S1990478920020131.

[5] D. F. Manlove, On the algorithmic complexity of twelve covering and independence parameters of graphs, Discrete Applied Mathematics 91 (1–3) (1999) 155–175. doi:10.1016/S0166-218X(98)00147-4.

[6] C. Bazgan, L. Brankovic, K. Casel, H. Fernau, Domination chain: Characterisation, classical complexity, parameterised complexity and approximability, Discrete Applied Mathematics 280 (2020) 23–42. doi:10.1016/j.dam.2019.10.005.

[7] P. Crescenzi, V. Kann, A compendium of NP optimization problems: Minimum Independent Dominating Set, Online compendium (2000). URL https://www.csc.kth.se/~viggo/wwwcompendium/node14.html

[8] P. Alimonti, T. Calamoneri, Improved approximations of independent dominating set in bounded degree graphs, in: Graph-Theoretic Concepts in Computer Science (WG 1996), Vol. 1197 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1997, pp. 2–16. doi:10.1007/3-540-62559-3_2.

[9] P. Alimonti, V. Kann, Hardness of approximating problems on cubic graphs, in: Algorithms and Complexity (CIAC 1997), Vol. 1203 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1997, pp. 288–298. doi:10.1007/3-540-62592-5_80.

[10] M. Chlebík, J. Chlebíková, Approximation hardness of dominating set problems in bounded degree graphs, Information and Computation 206 (11) (2008) 1264–1275. doi:10.1016/j.ic.2008.07.003.

[11] S. Gaspers, M. Liedloff, A branch-and-reduce algorithm for finding a minimum independent dominating set, Discrete Mathematics and Theoretical Computer Science 14 (1) (2012) 29–42.
[12] N. Bourgeois, F. Della Croce, B. Escoffier, V. T. Paschos, Fast algorithms for min independent dominating set, Discrete Applied Mathematics 161 (4) (2013) 558–572.
[13] C. Laforest, R. Phan, Solving the minimum independent domination set problem in graphs by exact algorithm and greedy heuristic, RAIRO – Operations Research 47 (3) (2013) 199–221.
[14] K. Haraguchi, An efficient local search for the minimum independent dominating set problem, in: 17th International Symposium on Experimental Algorithms (SEA 2018), Vol. 103 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, pp. 13:1–13:13. doi:10.4230/LIPIcs.SEA.2018.13.
[15] W. Goddard, M. A. Henning, Independent domination in graphs: A survey and recent results, Discrete Mathematics 313 (2013) 839–854.
[16] C. Lenzen, Y.-A. Pignolet, R. Wattenhofer, Distributed minimum dominating set approximations in restricted families of graphs, Distributed Computing 26 (2) (2013) 119–137.
[17] S. Khot, On the Power of Unique 2-Prover 1-Round Games, in: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, Montreal, Québec, Canada, 2002, pp. 767–775. doi:10.1145/509907.510017.
[18] S. Khot, O. Regev, Vertex Cover Might Be Hard to Approximate to Within $2-\epsilon$, Journal of Computer and System Sciences 74 (3) (2008) 335–349. doi:10.1016/j.jcss.2007.06.019.
[19] OpenAI Math Team, The Unique Games Conjecture and Optimal Approximation Thresholds (Result Family 102), GitHub Repository, manuscripts and Lean artifacts available at https://github.com/openai/math/blob/main/lean/docs/102.md (October 2026). URL https://github.com/openai/math
[20] F. Vega, Siriaisa: Approximate Independent Dominating Set Solver, https://pypi.org/project/siriaisa, python package version 0.0.9 (2026).
[21] F. Vega, An Approximate Solution to the Minimum Vertex Cover Problem: The Salvador Algorithm, manuscript; preprint at https://www.preprints.org/manuscript/202605.2000 (2026).
[22] F. Vega, Salvador: Approximate Vertex Cover Solver, https://pypi.org/project/salvador, python package version 0.1.4 (2026).
[23] F. Vega, Esperanza: Approximate Independent Set Solver, https://pypi.org/project/esperanza, python package version 0.1.6 (2026).
[24] R. E. Tarjan, Efficiency of a Good But Not Linear Set Union Algorithm, Journal of the ACM 22 (2) (1975) 215–225. doi:10.1145/321879.321884.
[25] A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using NetworkX, in: Proceedings of the 7th Python in Science Conference (SciPy 2008), Pasadena, CA, USA, 2008, pp. 11–15.
[26] Y. Caro, New results on the independence number, Tech. rep., Tel-Aviv University (1979).
[27] V. K. Wei, A lower bound on the stability number of a simple graph, Tech. Rep. Technical Memorandum 81-11217-9, Bell Laboratories (1981).
[28] P. Austrin, S. Khot, M. Safra, Inapproximability of Vertex Cover and Independent Set in Bounded Degree Graphs, Theory of Computing 7 (3) (2011) 27–43. doi:10.4086/toc.2011.v007a003.
[29] S. A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, ACM, 1971, pp. 151–158. doi:10.1145/800157.805047.
[30] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., Scipy 1.0: fundamental algorithms for scientific computing in python, Nature Methods 17 (3) (2020) 261–272. doi:10.1038/s41592-019-0686-2.
[31] Q. Huangfu, J. A. J. Hall, Parallelizing the dual revised simplex method, Mathematical Programming Computation 10 (1) (2018) 119–142. doi:10.1007/s12532-017-0130-5.
[32] G. L. Nemhauser, L. E. Trotter, Vertex packings: Structural properties and algorithms, Mathematical Programming 8 (1) (1975) 232–248. doi:10.1007/BF01580444.
[33] J. E. Hopcroft, R. M. Karp, An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs, SIAM Journal on Computing 2 (4) (1973) 225–231. doi:10.1137/0202019.
[34] D. S. Johnson, M. A. Trick (Eds.), Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Vol. 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, USA, 1996. doi:10.1090/dimacs/026.
[35] DynaRoARS, NPBench: A collection of benchmarks (in DIMACS format) for various NP-complete problems, GitHub Repository. URL https://github.com/dynaroars/npbench