hexagon:2611.00043v1

The Siriaisa Algorithm for Minimum Independent Dominating Set

Frank Vega

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

Abstract

The Minimum Independent Dominating Set problem (MIDS) asks for a smallest maximal independent set of an undirected graph; unless P = NP, it cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$. We present version 0.0.8 of Siriaisa, an ensemble of eighteen strategies that adapts the vertex cover algorithm Salvador and the independent set algorithm Esperanza to MIDS, with every exact subproblem a forest solved by a conditioned tree dynamic program. We prove that the output is an independent dominating set computed in $O((n+m)\,\alpha_{\mathrm{A}}(n))$ expected time, that its ratio is at most Δ, that it is optimal on forests, and that its ratio is at most $\sqrt{n}$ on three classes of graphs. A degree-splitting strategy yields a dominating set of size at most $\lfloor \sqrt{n} \rfloor\,(i(G) + s(G))$ for an explicit splitting excess $s(G)$, which vanishes on bounded-degree families for large n and is largest on complete graphs, where the output is optimal. More generally, splitting at $\lfloor n^{1-\varepsilon} \rfloor$ gives the bound $n^{1-\varepsilon}\,(i(G) + s_{\varepsilon}(G))$ for every $\varepsilon \in (0, 1]$. We state the ratio $n^{3/4}$ as a conjecture whose proof would imply P = NP. Through a reduction of Chlebík and Chlebíková, every output decodes into a vertex cover, with conditional consequences for the Unique Games Conjecture. In experiments the output is optimal on 9 999 of 10 000 graphs with an exact integer-program optimum and on all 95 bipartite graphs whose optimum was proved. As an application, we analyse the SAT solver `satypy` 0.0.4: it answers every unsatisfiable formula correctly, every SATISFIABLE answer carries a satisfying assignment, a wrong answer requires a ratio larger than $2p$ for p variables, and its construction gives a self-contained proof of Halldórsson's inapproximability theorem.

Keywords: Minimum Independent Dominating Set, Minimum Maximal Independent Set, Approximation Algorithm, Forest Dynamic Programming, Local Search, Bipartite Graphs, Unique Games Conjecture, Boolean Satisfiability, 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 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

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

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.

Under the Unique Games Conjecture (UGC) [11], Khot and Regev proved that Minimum Vertex Cover is hard to approximate within $2 - \varepsilon$ for every $\varepsilon > 0$ [12]. 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 [13]; 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 becomes an MIDS instance whose optimum is τ 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 0.0.8 of Siriaisa [14], a Python implementation of an ensemble of eighteen strategies. The ensemble adapts the seventeen strategies of the vertex cover algorithm Salvador [15, 16] and the phases of the independent set algorithm Esperanza [17] 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 $O((n + m)\,\alpha_{\mathrm{A}}(n))$ for the default parameters, with α_{A} the inverse Ackermann function (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, 6.11 and 6.15, Table 3).
- The parametric bound $n^{1-\varepsilon}\,(i(G) + s_\varepsilon(G))$ for splitting at $\lfloor n^{1-\varepsilon} \rfloor$, $\varepsilon \in (0, 1]$, with its consequences, among them the ratio $n^{3/4}$ on every graph with $\Delta \leq \lfloor n^{3/4} \rfloor$ and $s_\varepsilon(K_n) \geq n$ for every fixed ε and large n (Theorem 6.13, Corollary 6.14).
- The conjecture that the ratio is at most $n^{3/4}$ on every graph, with the theorem that a proof would imply P = NP (Conjecture 6.20, Theorem 6.21).
- 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; 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.

- The application to the SAT solver `satypy` 0.0.4 (Section 8): the structure of the SAT-IDS graph, with $i = p$ on satisfiable and $i \geq \beta + p$ on unsatisfiable formulas (Lemma 8.2), the least number of clause copies β with $\beta + p > 2p^2$ (Lemma 8.4) and with $\beta + p > 2\sqrt{n}\,p$ (Lemma 8.3), the correctness of every answer except UNSATISFIABLE on a satisfiable formula, which requires a ratio larger than $2p$, the correctness of every answer when the ratio on the graph is at most $2p$ (Theorem 8.7), the failure of the bound of Theorem 6.8 to separate satisfiable from unsatisfiable formulas for every number of copies (Proposition 8.8), the exact branching on the clause named by the output (Proposition 8.13), the gap p against more than $p\,n^{1-1/t}$ with $\beta_t = p^t(3q)^{t-1} + 1$ copies (Theorem 8.9), which gives a self-contained proof of Halldórsson's theorem (Corollary 8.11) and the decision of every formula under Conjecture 6.20 (Corollary 8.12), the asymptotic $s(G) \sim 2pL$ of the splitting excess with β^* copies (Proposition 8.5), and the identity $\mu(\varphi) = \lfloor i(G)/\beta \rfloor$ with the minimum number $\mu(\varphi)$ of falsified clauses (Theorem 8.6); and, on every graph, the inequality $\lfloor \sqrt{n} \rfloor (i(G) + s(G)) \geq n - i(G)$ (Proposition 6.10).

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 application to Boolean satisfiability, Section 9 reports the experiments and Section 10 concludes.

2. Research Data

The algorithm is implemented in the Python package `siriaisa` [14]. The version described in this paper is v0.0.8. Table 1 lists the code metadata.

Nr.	Code metadata description	Metadata
C1	Current code version	v0.0.8
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 $\geq$ 3.12, NetworkX, NumPy, SciPy
C7	Main entry points	`iris`, `batch_iris`, `test_iris`

Table 1: 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 2 lists them, and Algorithm 1 gives the main loop.

Table 2: The eighteen strategies of Siriaisa 0.0.8. "Source" names the strategy of Salvador [15] or the phase of Esperanza [17] 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

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

where $\delta(v) = 0$ if $\mathrm{pre}(v)$ or some child attains $\mathrm{IN}(c) \le a(c)$, $\delta(v) = \min_c(\mathrm{IN}(c) - a(c))$ otherwise (one child is forced into X), and $\mathrm{DOM}(v) = \infty$ when no such child exists and $\neg\mathrm{pre}(v)$. A root r takes $\min(\mathrm{IN}(r), \mathrm{DOM}(r), \mathrm{FREE}(r) + u(r))$, and back-pointers recover X.

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

Require: An induced forest F, flags, penalty $P > |V(F)|$
Ensure: A minimiser X of Φ

for every tree of F **do**
  root it; compute a BFS order and the children lists
  for v in reverse BFS order **do** compute $\mathrm{IN}(v), \mathrm{DOM}(v), \mathrm{FREE}(v)$ and the forced child
  end for
  choose the state of the root; assign the states top-down by the recorded choices; add the IN vertices to X
end for
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 [18]; 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. The rounds stop at a work budget of $40(n + m)$, and the vertices left are completed by Algorithm 2.

c_2–c_8. c_2 completes the endpoints of the NetworkX [19] 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.

Algorithm 4 ForestLayers(adj)

Require: Index graph; layering budget $8(n + m)$; 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 budget allows **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 [20, 21].

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 $O((n + m)\,\alpha_{\mathrm{A}}(n))$ 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 union by size and path halving, $O(\alpha_{\mathrm{A}}(n))$ amortised per operation [18], within the work budgets $40(n + m)$ and $8(n + m)$, 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 $O((n + m)\,\alpha_{\mathrm{A}}(n))$. □

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 first round of c_1 fits in its budget, since its cost $4(n + 2m)$ plus the initial $n + m$ is at most $40(n + m)$; the maximum spanning forest is G itself, the forest optimum is independent in G and is kept entirely, and it dominates G, so the residual graph is empty and c_1 is the forest optimum of Lemma 4.2. Every candidate after c_1 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 $|c_{18}| \le |c_1| = i(G)$. □

The next theorem is the counterpart of the $O(\sqrt{n})$ theorem of Esperanza [17], 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*

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

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 $|S_0| \le |S_H|$, domination, and $c_{13} \le |S_0|$ when S_0 is independent; the output is no larger than c_{13}. □

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 {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 (Lower bound of the splitting bound)**.** *For every graph G on n vertices and $k = \max(1, \lfloor \sqrt{n} \rfloor)$, $k\,(i(G) + s(G)) \ge n - i(G)$. In particular the bound of Theorem 6.8 is at least $n - i(G)$ on every graph.*

Proof. Let D be a minimum independent dominating set. It dominates the other $n - i(G)$ vertices, so $\sum_{d \in D} \deg(d) \ge n - i(G)$. A vertex $d \in D \cap B$ contributes $\lceil \deg(d)/k \rceil - 1 \ge \deg(d)/k - 1$ to $s(G)$, and a vertex $d \in D \setminus B$ satisfies $\deg(d) \le k$, so $\deg(d)/k - 1 \le 0$; since every term of $s(G)$ is non-negative, $s(G) \ge \sum_{d \in D} (\deg(d)/k - 1) \ge (n - i(G))/k - i(G)$. Multiplying by k gives $k\,s(G) \ge n - i(G) - k\,i(G)$. □

Proposition 6.11 (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 6.11, 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 6.12 (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. □

Algorithm 5 is defined for every block size $k \geq 1$. For $\varepsilon \in (0, 1]$, let DegreeSplit$_\varepsilon$ denote Algorithm 5 with $k_\varepsilon = \max(1, \lfloor n^{1-\varepsilon} \rfloor)$ in place of $\max(1, \lfloor \sqrt{n} \rfloor)$, let $B_\varepsilon = \{v : \deg_G(v) > k_\varepsilon\}$, and define

$$s_\varepsilon(G) = \sum_{v \in B_\varepsilon} \Big(\Big\lceil \frac{\deg_G(v)}{k_\varepsilon} \Big\rceil - 1 \Big).$$

Strategy c_{13} is DegreeSplit$_{1/2}$, with $k_{1/2} = k$ and $s_{1/2}(G) = s(G)$.

Theorem 6.13 (Parametric degree splitting)**.** *Let $\varepsilon \in (0, 1]$. For each inner solver of* DegreeSplit$_\varepsilon$*, the decoded set S_0 is a dominating set of G with*

$$|S_0| \;\leq\; k_\varepsilon\,(i(G) + s_\varepsilon(G)) \;\leq\; n^{1-\varepsilon}\,(i(G) + s_\varepsilon(G)).$$

If S_0 is independent in G for one of the inner solvers, then the set S_ε returned by DegreeSplit$_\varepsilon$ *is an independent dominating set of G with $|S_\varepsilon| \leq k_\varepsilon\,(i(G) + s_\varepsilon(G))$. For $\varepsilon = \frac{1}{2}$ this is Theorem 6.8.*

Proof. The statements and proofs of Lemmas 5.1, 6.6 and 6.7 use only that k is a positive integer, so they hold with k_ε, B_ε and $s_\varepsilon(G)$ in place of k, B and $s(G)$: the copy graph H_ε has maximum degree at most k_ε, $i(H_\varepsilon) \leq i(G) + s_\varepsilon(G)$, and decoding gives a dominating set S_0 with $|S_0| \leq |S_H|$. Every inner solver returns a maximal independent set S_H of H_ε, and Lemma 6.1, applied componentwise with $\max(1, \Delta(H_\varepsilon)) \leq k_\varepsilon$, gives $|S_H| \leq k_\varepsilon\, i(H_\varepsilon)$. Finally $k_\varepsilon \leq n^{1-\varepsilon}$, since $n^{1-\varepsilon} \geq 1$ for $n \geq 1$. If S_0 is independent, Algorithm 2 returns S_0 (Lemma 6.7), and S_ε is the smallest of the repaired sets, each an independent dominating set by Lemma 4.1. □

Corollary 6.14 (Consequences of Theorem 6.13)**.** *Let G be a graph on $n \geq 1$ vertices with m edges, and let $\varepsilon \in (0, 1]$.*

(i) Monotonicity. *If $0 < \varepsilon' \leq \varepsilon$, then $k_{\varepsilon'} \geq k_\varepsilon$, $B_{\varepsilon'} \subseteq B_\varepsilon$ and $s_{\varepsilon'}(G) \leq s_\varepsilon(G)$.*

(ii) Weighted excess. $\sum_{v \in B_\varepsilon} (\deg(v) - k_\varepsilon) \leq k_\varepsilon\, s_\varepsilon(G) \leq \sum_{v \in B_\varepsilon} (\deg(v) - 1) \leq 2m - |B_\varepsilon|$*; hence the bound of Theorem 6.13 satisfies $k_\varepsilon\,(i(G) + s_\varepsilon(G)) \leq k_\varepsilon\, i(G) + 2m - |B_\varepsilon|$.*

(iii) Lower bound. $k_\varepsilon\,(i(G) + s_\varepsilon(G)) \geq n - i(G)$.

(iv) Vanishing excess. $s_\varepsilon(G) = 0$ *if and only if* $\Delta \le k_\varepsilon$, *and then* $k_\varepsilon\,(i(G) + s_\varepsilon(G)) = k_\varepsilon\, i(G) \ge \max(1,\Delta)\, i(G)$. *If* $\varepsilon < 1$ *and* $\Delta \le D$ *for an integer* $D \ge 1$, *then* $s_\varepsilon(G) = 0$ *for every* $n \ge D^{1/(1-\varepsilon)}$; *in particular* $s_{1/4}(G) = 0$ *for every* $n \ge D^{4/3}$, *and on every cubic graph with* $n \ge 6$.

(v) Ratio $n^{1-\varepsilon}$. *If* $s_\varepsilon(G) = 0$, *the output* S *of Algorithm 1 satisfies* $|S|/i(G) \le n^{1-\varepsilon}$; *in particular Conjecture 6.20 holds on every graph with* $\Delta \le \lfloor n^{3/4} \rfloor$.

(vi) Bounded excess. *If* $s_\varepsilon(G) \le i(G)$, *then* $|S_0| \le 2k_\varepsilon\, i(G) \le 2n^{1-\varepsilon}\, i(G)$ *for every inner solver of* DegreeSplit$_\varepsilon$, *and* $|S_\varepsilon| \le 2n^{1-\varepsilon}\, i(G)$ *when* S_0 *is independent.*

(vii) Complete graphs. *For* $n \ge 2$, $s_\varepsilon(K_n) = n\,(\lceil (n-1)/k_\varepsilon \rceil - 1)$, *which is* 0 *if* $k_\varepsilon \ge n-1$ *and at least* n *otherwise; for every fixed* $\varepsilon \in (0,1]$ *there is* n_0 *such that* $s_\varepsilon(K_n) \ge n$ *for every* $n \ge n_0$. *On* K_n *the output of Algorithm 1 is optimal.*

Proof. (i) $n^{1-\varepsilon'} \ge n^{1-\varepsilon}$ for $n \ge 1$, so $k_{\varepsilon'} \ge k_\varepsilon$; a vertex of degree larger than $k_{\varepsilon'}$ has degree larger than k_ε, and $\lceil d/k \rceil$ is non-increasing in k, so every term of $s_{\varepsilon'}(G)$ is at most the corresponding term of $s_\varepsilon(G)$, and the terms are non-negative. (ii) For integers $d, k \ge 1$, $d/k \le \lceil d/k \rceil \le (d-1)/k + 1$, so $d - k \le k(\lceil d/k \rceil - 1) \le d - 1$; summing over B_ε and using $\sum_{v \in B_\varepsilon} \deg(v) \le 2m$ gives the inequalities. (iii) The proof of Proposition 6.10 uses only that k is a positive integer, and applies with k_ε, B_ε and $s_\varepsilon(G)$. (iv) $\lceil d/k_\varepsilon \rceil - 1 = 0$ exactly when $d \le k_\varepsilon$, so $s_\varepsilon(G) = 0$ if and only if $B_\varepsilon = \emptyset$, that is $\Delta \le k_\varepsilon$; then $k_\varepsilon \ge \max(1,\Delta)$. For an integer $D \ge 1$, $\lfloor n^{1-\varepsilon} \rfloor \ge D$ if and only if $n^{1-\varepsilon} \ge D$, that is $n \ge D^{1/(1-\varepsilon)}$, and then $k_\varepsilon \ge D \ge \Delta$. For $\varepsilon = \frac{1}{4}$ the exponent is $\frac{4}{3}$; a cubic graph has even $n \ge 4$, and $\lfloor 6^{3/4} \rfloor = 3$, so $k_{1/4} \ge 3 = \Delta$ for $n \ge 6$. (v) By (iv), $\max(1,\Delta) \le k_\varepsilon \le n^{1-\varepsilon}$, and Corollary 6.2 gives $|S| \le \max(1,\Delta)\, i(G)$. For $\varepsilon = \frac{1}{4}$, $\Delta \le \lfloor n^{3/4} \rfloor$ gives $\max(1,\Delta) \le n^{3/4}$ for $n \ge 1$. (vi) Theorem 6.13 with $i(G) + s_\varepsilon(G) \le 2i(G)$. (vii) Every vertex of K_n has degree $n-1$. If $k_\varepsilon \ge n-1$, then $B_\varepsilon = \emptyset$; otherwise every vertex lies in B_ε and $\lceil (n-1)/k_\varepsilon \rceil \ge 2$, so every vertex contributes at least 1. For fixed $\varepsilon > 0$, $n^{1-\varepsilon}/(n-1) \to 0$, so $k_\varepsilon \le n^{1-\varepsilon} < n-1$ for every n large enough. Optimality is Proposition 6.11. □

The splitting excess on hard families. Proposition 6.15 and Table 3 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 6.15 (Splitting excess on graph families)**.** *Let* $k = \lfloor \sqrt{n} \rfloor$.

(a) Bounded degree. *If* $\Delta \le D$ *for a constant* D, *then* $s(G) = 0$ *for every* $n \ge D^2$. *This holds for cubic graphs with* $n \ge 10$ *(Proposition 6.16); for graphs and bipartite graphs of maximum degree at most* B *with* $n \ge B^2$; *for the SC-IDS graphs of maximum degree* B *with* $n \ge B^2$; *for* $h(H)$ *with* $\Delta(H) \le D$ *and* $|V(H)| + 6|E(H)| \ge \max(D,3)^2$; *for graphs of maximum degree* d *with* $n \ge d^2$, *the setting of [22]; for grids with* $n \ge 16$; *and for hypercubes* Q_q *with* $q \ge 4$.

(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* $s(G) \le n(\lceil ds_b/k \rceil - 1)$, *and* $s(G) = 0$ *when* $N \ge d^2 s_b$.

(c) Sparse graphs. $s(G) \le 2m/k$; *for planar graphs* $s(G) \le (6n-12)/k$, *for bipartite planar graphs* $s(G) \le (4n-8)/k$ *(sharper bounds in Proposition 6.18), and for graphs with* $m \le cn$, $s(G) \le 2c\,n/k \le 2c(\sqrt{n} + 2)$ *for* $n \ge 4$.

(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* $s(G) \ge n(\lceil \delta(G)/k \rceil - 1)$. *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 3: 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 [12, 22]	$\max(D, 3)$	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 [12]	$\leq ds_b$	0	—	ratio $\leq ds_b$
Khot–Regev YES graphs, any N	vertex cover: UG-hard [12]	$\leq ds_b$	$\leq n(\lceil ds_b/k \rceil - 1)$	—	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 6.15(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 6.11). On all 55 instances of Section 9.4, $\Delta \leq \lfloor \sqrt{n} \rfloor$ (Table 6: $n \geq 106$ 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 6.16 (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 6.17 (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 6.18 (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 6.15(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 6.11. □

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 6.19 (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 6.11 the optimality. □

Conjecture 6.20 (Ratio $n^{1-\varepsilon}$ with $\varepsilon = \frac{1}{4}$)**.** *For every graph G on n vertices, the output S of Algorithm 1 satisfies $|S|/i(G) \le n^{3/4}$.*

Since $\sqrt{n} \le n^{3/4}$ for every $n \ge 1$ and $2\sqrt{n} \le n^{3/4}$ for every $n \ge 16$, Theorems 6.5 and 6.8, Corollary 6.12 and Proposition 6.11 show that Conjecture 6.20 holds on every graph with $\Delta \le \sqrt{n}$, with $i(G) \ge \sqrt{n}$, or with $\alpha(G) \le \sqrt{n}\, i(G)$, on every graph with $n \ge 16$, $s(G) \le i(G)$ and an independent decoded set, and on every graph whose components are complete multipartite, crown, wheel or double-star graphs.

Theorem 6.21 (Conditional consequence of Conjecture 6.20)**.** *If Conjecture 6.20 is proved, then* P = NP.

Proof. Algorithm 1 runs in polynomial time (Theorem 5.2), so Conjecture 6.20 would give a polynomial-time approximation of MIDS within $n^{1-\varepsilon}$ with $\varepsilon = \frac{1}{4}$, which is impossible unless P = NP [2] (see also Corollary 8.11). □

Corollary 6.22 (Circle graphs)**.** *If Conjecture 6.20 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). 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 graph $h(H)$ 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.

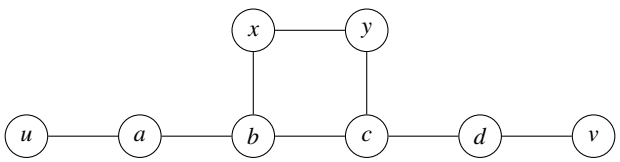


Figure 1: The edge gadget of the VC-IDS reduction: the edge uv of H is replaced by the path u–a–b–c–d–v and the square b–x–y–c.

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 [12], and in particular if the theorem reported in [13] 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 [12] 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 $2 - (2 + o_d(1)) \log\log d / \log d$ [22]. 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 [22] 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 [13] had not been peer reviewed at the time of writing; we cite it as reported and have not checked the formalization. Section 9.4 reports the measured values of $|D| - i(h(H))$ and $|C_D|/\tau(H)$ on VC-IDS instances.

8. Application to Boolean Satisfiability

The SAT solver `satypy` 0.0.4 [23] decides a formula in conjunctive normal form in three stages: unit propagation and pure literal elimination, which answers UNSATISFIABLE on a conflict and otherwise commits literals and leaves a residual formula; an exact implication-graph solver when every residual clause has at most two literals; and otherwise a reduction of the residual formula to MIDS, solved by Siriaisa 0.0.8. Every SATISFIABLE answer is returned with an assignment, formed by the committed literals and the decoded literals, that is checked against every clause of the input formula. This section describes the third stage. Let φ be the residual formula, with p variables $x_1, \dots, x_p$ and q clauses $C_1, \dots, C_q$, every clause a nonempty set of literals with no complementary pair.

Definition 8.1 (SAT-IDS graph)**.** For an integer $\beta \geq 1$, the graph $G_\beta(\varphi)$ has a *literal vertex* for each of the $2p$ literals $x_r, \neg x_r$ and β *clause vertices* $c_{j,1}, \dots, c_{j,\beta}$ for each clause C_j; its edges are $x_r \neg x_r$ for every variable and $c_{j,t}\ell$ for every clause C_j, every $t \leq \beta$ and every literal $\ell \in C_j$. It has $n = 2p + q\beta$ vertices and $p + \beta \sum_j |C_j|$ edges.

Lemma 8.2 (Structure of the SAT-IDS graph)**.** *Let $\beta \geq p$ and let D be an independent dominating set of $G = G_\beta(\varphi)$.*

(i) If D contains no clause vertex, then D contains exactly one of x_r and $\neg x_r$ for every r, $|D| = p$, and the literals of D form a satisfying assignment of φ.

(ii) If D contains a clause vertex $c_{j,t}$, then D contains $c_{j,1}, \dots, c_{j,\beta}$.

(iii) If D contains a clause vertex, then $|D| \geq \beta + p$.

(iv) $i(G) = p$ if φ is satisfiable, and $i(G) \geq \beta + p$ otherwise; no independent dominating set of G has a size strictly between p and $\beta + p$.

(v) If an assignment of φ falsifies exactly one clause, then $i(G) \leq \beta + p$; in particular $i(G) = \beta + p$ when φ is unsatisfiable and such an assignment exists.

Proof. (i) The neighbours of x_r and $\neg x_r$ are each other and clause vertices, none of which lies in D, so D contains one of them, and not both since D is independent. Hence D consists of p literal vertices, one per variable, which form an assignment; every clause vertex $c_{j,t} \notin D$ has a neighbour in D, a literal of C_j set true. (ii) The neighbours of $c_{j,t}$ are the literals of C_j, so none of them lies in D; every $c_{j,s}$ has the same neighbours, so it is dominated only by itself and lies in D. (iii) If D contains clause vertices of two clauses, then $|D| \geq 2\beta \geq \beta + p$ by (ii). Otherwise D contains the β clause vertices of exactly one clause C_j. For every variable x_r, C_j contains at most one of x_r and $\neg x_r$; let y be one that C_j does not contain. The neighbours of y are its complement and clause vertices of clauses other than C_j, which are not in D; so if neither x_r nor $\neg x_r$ lies in D, then y is not dominated. Hence D contains a literal vertex of every variable and $|D| \geq \beta + p$. (iv) A satisfying assignment, as a set of p literal vertices, is independent and dominates the complementary literals and every clause vertex, and by (i) and (iii) every independent dominating set has p or at least $\beta + p$ vertices; if φ is unsatisfiable, (i) excludes the first case. (v) Let the assignment σ falsify only C_j, and let D be the p literals of σ together with $c_{j,1}, \dots, c_{j,\beta}$. The neighbours of $c_{j,t}$ are the literals of C_j, all false, so D is independent; the false literals are dominated by their complements, and the clause vertices of the other clauses by a true literal. So $|D| = \beta + p$, and (iv) gives the equality. □

Lemma 8.3 (Number of copies)**.** *For integers $p, q \geq 1$, let $\Lambda = 4q^2p^4 + 8p^3 - 4qp^3$ and $\beta^* = 2qp^2 + \lfloor \sqrt{\Lambda} \rfloor + 1 - p$. Then β^* is the least positive integer β with $\beta + p > 2p\sqrt{2p + q\beta}$. Moreover $p \leq \beta^* \leq 4qp^2 + 1$.*

Proof. Write $C = \beta + p$, so that $2p + q\beta = qC + 2p - qp > 0$. Both sides of the inequality are positive, so it is equivalent to $f(C) = C^2 - 4qp^2C - (8p^3 - 4qp^3) > 0$. The discriminant of f is 4Λ with $\Lambda = 4p^3(q^2p + 2 - q) > 0$, since $q^2p \geq q$, so f has the real roots $r_{1,2} = 2qp^2 \mp \sqrt{\Lambda}$. Since $f(p) = p^2 - 8p^3 < 0$, $r_1 < p < r_2$, and every $C \geq p + 1$ satisfies $f(C) > 0$ if and only if $C > r_2$. As $2qp^2$ is an integer, the least integer larger than r_2 is $2qp^2 + \lfloor \sqrt{\Lambda} \rfloor + 1$, which gives β^*. Since $\Lambda \geq 0$, $\beta^* \geq 2qp^2 + 1 - p \geq p$. If $q \geq 2$, then $\Lambda \leq 4q^2p^4$ and $\beta^* \leq 4qp^2 + 1 - p$; if $q = 1$, then $\Lambda = 4p^4 + 4p^3 < (2p^2 + p)^2$ and $\beta^* \leq 4p^2 + 1$. □

Lemma 8.4 (Copies for the ratio $2p$)**.** *For an integer $p \geq 1$, let $\hat{\beta} = 2p^2 - p + 1$. Then $\hat{\beta}$ is the least integer β with $\beta + p > 2p^2$, $\hat{\beta} \geq p + 1$, and $G_{\hat{\beta}}(\varphi)$ has $n = 2p + q\hat{\beta} \leq 2p + 2qp^2$ vertices and $p + \hat{\beta} \sum_j |C_j|$ edges.*

Proof. For an integer β, $\beta + p > 2p^2$ if and only if $\beta \geq 2p^2 - p + 1$. Moreover $\hat{\beta} - (p+1) = 2p(p-1) \geq 0$, and the counts follow from Definition 8.1. □

For a literal ℓ of φ, let o_ℓ be the number of clauses that contain ℓ, let $L = \sum_j |C_j| = \sum_\ell o_\ell$ be the number of literal occurrences of φ, and let λ be the number of literals with $o_\ell \geq 1$, so that $\lambda \leq 2p$.

Proposition 8.5 (Splitting excess of the SAT-IDS graph)**.** *Let $\beta = \beta^*$, $G = G_\beta(\varphi)$ with n vertices and $k = \lfloor \sqrt{n} \rfloor$.*

(i) $k \geq qp$; the vertices of degree larger than k are exactly the literal vertices ℓ with $o_\ell \geq 1$, the literal vertex ℓ has degree $1 + \beta o_\ell$, and $s(G) = \sum_{\ell:\, o_\ell \geq 1} (\lceil (1 + \beta o_\ell)/k \rceil - 1)$.

(ii) $2pL - pL/k - \lambda < s(G) < 2pL + ((p+1)L + \lambda)/k$; hence

$$1 - \frac{1}{2k} - \frac{1}{L} \;<\; \frac{s(G)}{2pL} \;<\; 1 + \frac{1}{k}\Big(1 + \frac{1}{L}\Big).$$

(iii) If every variable of φ occurs in φ both positively and negatively, then $\lambda = 2p$, $L \geq 2p$ and $s(G) > 4p^2 - 2p^2/k - 2p$.

Proof. (i) By Lemma 8.3, $n > q\beta \geq q(2qp^2 + 1 - p) = q^2p^2 + q(qp^2 + 1 - p) > q^2p^2$, so $k \geq qp \geq p$. A clause vertex $c_{j,t}$ has degree $|C_j| \leq p \leq k$. A literal vertex ℓ is adjacent to its complement and to the β clause vertices of each of the o_ℓ clauses that contain it, so it has degree $1 + \beta o_\ell$; for $o_\ell = 0$ this is $1 \leq k$. Since $\beta + p > 2p\sqrt{n} \geq 2pk$ (Lemma 8.3), $\beta > 2pk - p \geq k$, so for $o_\ell \geq 1$ the degree $1 + \beta o_\ell$ exceeds k. The formula for $s(G)$ is the definition of the splitting excess. (ii) Since $\beta^* \geq 2qp^2 + 1 - p \geq 2$, the integer $\beta - 1 \geq 1$ is not a solution of the inequality of Lemma 8.3, so $\beta - 1 + p \leq 2p\sqrt{2p + q(\beta - 1)} < 2p\sqrt{n} < 2p(k+1)$, that is $\beta < 2pk + p + 1$; together with $\beta > 2pk - p$ this gives $2p - p/k < \beta/k < 2p + (p+1)/k$. For every real x, $x - 1 \leq \lceil x \rceil - 1 < x$, so by (i)

$$\frac{\beta L + \lambda}{k} - \lambda \;\leq\; s(G) \;<\; \frac{\beta L + \lambda}{k}.$$

The lower bound is larger than $(2p - p/k)L - \lambda$, and the upper bound is smaller than $(2p + (p+1)/k)L + \lambda/k$. Dividing by $2pL$ and using $\lambda \leq 2p$ and $(p+1)/(2p) \leq 1$ gives the ratio bounds. (iii) Every variable gives two literals with $o_\ell \geq 1$, so $\lambda = 2p$ and $L = \sum_\ell o_\ell \geq 2p$; since $2p - p/k > 0$, (ii) gives $s(G) > (2p - p/k) \cdot 2p - 2p$. □

By Proposition 8.5(i), $k \geq qp$, so both error terms in (ii) tend to 0 as qp and L grow, and $s(G)$ is asymptotic to $2pL$.

Let $\mu(\varphi)$ be the least number of clauses of φ falsified by an assignment, so that $\mu(\varphi) = 0$ if and only if φ is satisfiable. For a set F of clause indices, let $U(F) = \bigcup_{j \in F} C_j$ and let $c(F)$ be the number of variables x_r with $x_r \in U(F)$ and $\neg x_r \in U(F)$. A set T of literals *completes* F if, for every variable x_r, T contains the complement of ℓ when ℓ is the only one of $x_r, \neg x_r$ in $U(F)$, exactly one of $x_r, \neg x_r$ when neither lies in $U(F)$, and neither when both lie in $U(F)$, and if T contains a literal of every clause C_j with $j \notin F$. F is *admissible* if some T completes it.

Theorem 8.6 (Independent domination and maximum satisfiability)**.** *Let $\beta \geq 1$ and $G = G_\beta(\varphi)$.*

(i) The independent dominating sets of G are exactly the sets $T \cup \{c_{j,t} : j \in F,\ t \leq \beta\}$ with F admissible and T completing F, and such a set has $\beta|F| + p - c(F)$ vertices; hence $i(G) = \min_{F \text{ admissible}} (\beta|F| + p - c(F))$.

(ii) $\beta\,\mu(\varphi) \leq i(G) \leq \beta\,\mu(\varphi) + p$; if $\beta > p$, then $\mu(\varphi) = \lfloor i(G)/\beta \rfloor$.

(iii) If $\beta > p$, every independent dominating set D of G contains the clause vertices of exactly $\lfloor |D|/\beta \rfloor$ clauses, and its literal vertices, completed by arbitrary values on the variables with no literal in D, form an assignment that falsifies at most $\lfloor |D|/\beta \rfloor$ clauses.

(iv) If $\beta > p$ and $|D| \leq \rho\, i(G)$ for some $\rho \geq 1$, that assignment falsifies at most $\rho\,(\mu(\varphi) + p/\beta)$ clauses.

For $\beta = \hat{\beta}$ and $\beta = \beta^*$, (ii)–(iv) apply, since $\hat{\beta} \geq p + 1$ and $\beta^* \geq 2qp^2 + 1 - p > p$ (Lemmas 8.4 and 8.3).

Proof. (i) Let D be an independent dominating set, F the set of indices of the clauses with a clause vertex in D, and T the set of literal vertices of D. By Lemma 8.2(ii), D contains $c_{j,1}, \ldots, c_{j,\beta}$ for every $j \in F$ and no other clause vertex. Since D is independent, T contains no literal of $U(F)$, which is adjacent to a clause vertex of D, and at most one of $x_r, \neg x_r$. A literal vertex $\ell \notin T$ is dominated by its complement or by a clause vertex of D, that is, its complement lies in T or $\ell \in U(F)$. If both $x_r, \neg x_r$ lie in $U(F)$, neither lies in T. If only ℓ lies in $U(F)$, then $\ell \notin T$ and the complement of ℓ, which is not in $U(F)$, is dominated only if it lies in T. If neither lies in $U(F)$, one of them lies in T. A clause vertex $c_{j,t}$ with $j \notin F$ is dominated by a literal of C_j in T. Hence T completes F. Conversely, if T completes F, the set $D = T \cup \{c_{j,t} : j \in F,\ t \le \beta\}$ is independent, because clause vertices are pairwise non-adjacent, T holds at most one literal of every variable and no literal of $U(F)$; it dominates every literal vertex not in T, by its complement or by a clause vertex of F, and every clause vertex $c_{j,t}$ with $j \notin F$, by a literal of C_j in T. T contains one literal of each of the $p - c(F)$ variables that do not have both literals in $U(F)$, so $|D| = \beta|F| + p - c(F)$. (ii) Let σ be an assignment that falsifies exactly the $\mu(\varphi)$ clauses of a set F_σ. Every literal of $U(F_\sigma)$ is false under σ, so no variable has both literals in $U(F_\sigma)$ and $c(F_\sigma) = 0$, and the set of true literals of σ completes F_σ; by (i), $i(G) \le \beta\mu(\varphi) + p$. Conversely, let T complete an admissible F. T holds at most one literal of every variable, so it extends to an assignment by arbitrary values on the variables with no literal in T; this assignment satisfies every clause C_j with $j \notin F$, so $\mu(\varphi) \le |F|$ and $\beta|F| + p - c(F) \ge \beta|F| \ge \beta\mu(\varphi)$, since $c(F) \le p$. If $\beta > p$, then $\beta\mu(\varphi) \le i(G) < \beta(\mu(\varphi) + 1)$. (iii) By (i), $|D| = \beta|F| + p - c(F)$ with $0 \le p - c(F) \le p < \beta$, so $|F| = \lfloor |D|/\beta \rfloor$, and the assignment satisfies every clause outside F, as in (ii). (iv) By (iii) and (ii), the number of falsified clauses is at most $|D|/\beta \le \rho\, i(G)/\beta \le \rho\,(\mu(\varphi) + p/\beta)$. □

Lemma 8.2(iv) and (v) are the cases $\mu(\varphi) = 0$ and $\mu(\varphi) = 1$ of Theorem 8.6.

With $\beta = \hat{\beta}$, `satypy` builds $G = G_\beta(\varphi)$ and runs Algorithm 1 on it, with output S. Without branching (Proposition 8.13), it answers SATISFIABLE, with the literals of S, if $|S| < \beta$ and the assignment passes the check; UNSATISFIABLE if $|S| > 2p^2$, that is, if $|S|/p > 2p$; and UNKNOWN otherwise. The number of copies β^* of Lemma 8.3, with the threshold $|S| > 2\sqrt{n}\,p$, is the choice for the ratio $2\sqrt{n}$.

Theorem 8.7 (Decisions of the MIDS stage)**.** *Let $\beta = \hat{\beta}$ and $G = G_\beta(\varphi)$ with n vertices, and let S be the output of Algorithm 1 on G.*

(i) If $|S| < \beta$, then the literal vertices of S form a satisfying assignment of φ, and the answer is SATISFIABLE.

(ii) If $|S| \ge \beta$, then $|S| \ge \beta + p > 2p^2$, and the answer is UNSATISFIABLE; the stage never answers UNKNOWN.

(iii) If φ is unsatisfiable, the answer is UNSATISFIABLE.

(iv) If φ is satisfiable, the answer is UNSATISFIABLE if and only if S contains a clause vertex, and then $|S|/i(G) \ge (\beta + p)/p > 2p$.

(v) If $|S|/i(G) \le 2p$, the answer is correct.

Proof. S is an independent dominating set of G (Theorem 4.3), and $\beta \ge p + 1$ (Lemma 8.4). (i) $|S| < \beta < \beta + p$, so S contains no clause vertex by Lemma 8.2(iii), and Lemma 8.2(i) gives the assignment. It consists of one literal of every variable of φ, and the committed literals of the first stage satisfy the clauses that the first stage removed from the input formula and contain no variable of φ, so their union passes the check. (ii) $|S| \ge \beta > p$, so $|S| \ge \beta + p$ by Lemma 8.2(iv), and $\beta + p > 2p^2$ by Lemma 8.4. (iii) $|S| \ge i(G) \ge \beta + p$ by Lemma 8.2(iv), and (ii) applies. (iv) $i(G) = p$ by Lemma 8.2(iv); by (i) and (ii) the answer is UNSATISFIABLE if and only if $|S| \ge \beta$, that is, by Lemma 8.2(i) and (iii), if and only if S contains a clause vertex, and then $|S| \ge \beta + p$. (v) If φ is satisfiable, $|S| \le 2p\, i(G) = 2p^2 < \beta + p$, so S contains no clause vertex by Lemma 8.2(iii), and the answer is SATISFIABLE by (iv) and (i); if φ is unsatisfiable, (iii) applies. □

Independently of any conjecture, Theorem 8.7 shows that `satypy` answers UNSATISFIABLE on every unsatisfiable formula, that every SATISFIABLE answer carries a satisfying assignment, and that a wrong answer can only be UNSATISFIABLE on a satisfiable formula on which Algorithm 1 has ratio larger than $2p = 2i(G)$.

The proved bound of the degree-splitting strategy. Theorem 6.8 cannot replace the ratio assumption of Theorem 8.7(v), for any number of copies.

Proposition 8.8 (No separation by the splitting bound)**.** *For every $\beta \geq 1$ and every satisfiable φ, the graph $G = G_\beta(\varphi)$ satisfies $\lfloor \sqrt{n} \rfloor (i(G) + s(G)) \geq p + q\beta \geq \beta + p$. Hence the bound of Theorem 6.8 on a satisfiable formula is never smaller than the size $\beta + p$ that Lemma 8.2(iii) guarantees for every independent dominating set with a clause vertex, and it does not imply that the output contains no clause vertex; in particular this holds for $\beta = \hat{\beta}$, for $\beta = \beta^*$ and for $\beta = \beta_t$ of Theorem 8.9.*

Proof. $i(G) = p$ by Lemma 8.2(iv), and $n = 2p + q\beta$, so Proposition 6.10 gives $\lfloor \sqrt{n} \rfloor (p + s(G)) \geq n - p = p + q\beta \geq p + \beta$. □

SAT-IDS graphs for the ratio $n^{1-1/t}$. The number of copies can be chosen for any ratio of the form $n^{1-\varepsilon}$ with $\varepsilon = 1/t$.

Theorem 8.9 (Gap of the SAT-IDS graph for the ratio $n^{1-1/t}$)**.** *For an integer $t \geq 2$, let $\beta_t = p^t(3q)^{t-1} + 1$ and $G = G_{\beta_t}(\varphi)$, with $n = 2p + q\beta_t = 3^{t-1}p^tq^t + q + 2p$ vertices.*

(i) $\beta_t + p > p\,n^{1-1/t}$; hence $i(G) = p$ if φ is satisfiable, and $i(G) \geq \beta_t + p > p\,n^{1-1/t}$ otherwise.

(ii) For every independent dominating set D of G with $|D| \leq n^{1-1/t}\, i(G)$: φ is satisfiable if and only if $|D| < \beta_t$, and then the literal vertices of D form a satisfying assignment.

(iii) G has $p + \beta_t \sum_j |C_j|$ edges and is built in time polynomial in p and q for every fixed t.

Proof. (i) $\beta_t > p$, so $2p + q\beta_t < 3q\beta_t$. Since $\beta_t > p^t(3q)^{t-1}$, $\beta_t^{1/t} > p\,(3q)^{(t-1)/t}$, and $\beta_t = \beta_t^{1/t}\beta_t^{(t-1)/t} > p\,(3q\beta_t)^{1-1/t} > p\,n^{1-1/t}$, so $\beta_t + p > p\,n^{1-1/t}$. Lemma 8.2(iv) applies because $\beta_t \geq p$. (ii) If φ is satisfiable, $|D| \leq n^{1-1/t}p < \beta_t + p$, so D contains no clause vertex by Lemma 8.2(iii), and Lemma 8.2(i) gives $|D| = p < \beta_t$ and the assignment. If φ is unsatisfiable, $|D| \geq i(G) \geq \beta_t + p$. (iii) Definition 8.1; n and the number of edges are polynomials in p and q of degree $2t$ and $2t + 1$. □

Corollary 8.10 (Two values of the ratio)**.** *Let φ be satisfiable, $\beta \geq p$, $G = G_\beta(\varphi)$ with n vertices, and D an independent dominating set of G. Then either $|D|/i(G) = 1$, and the literal vertices of D form a satisfying assignment, or D contains a clause vertex and $|D|/i(G) \geq (\beta + p)/p$. For $\beta = \hat{\beta}$, $(\beta + p)/p > 2p$; for $\beta = \beta^*$, $(\beta + p)/p > 2\sqrt{n}$; for $\beta = \beta_t$ with $t \geq 2$, $(\beta + p)/p > n^{1-1/t}$, in particular $(\beta_4 + p)/p > n^{3/4}$. Hence, for $\beta = \hat{\beta}$ and every $\rho \leq 2p$, for $\beta = \beta^*$ and every $\rho \leq 2\sqrt{n}$, and for $\beta = \beta_t$ and every $\rho \leq n^{1-1/t}$, $|D|/i(G) \leq \rho$ holds if and only if D is a minimum independent dominating set of G.*

Proof. $i(G) = p$ by Lemma 8.2(iv). If D contains no clause vertex, $|D| = p$ and its literal vertices form a satisfying assignment by Lemma 8.2(i); otherwise $|D| \geq \beta + p$ by Lemma 8.2(iii). The inequalities for $\hat{\beta}$, β^* and β_t are Lemmas 8.4 and 8.3 and Theorem 8.9(i), divided by p; in both cases the second alternative gives $|D|/i(G) > \rho$, and the first gives $|D| = i(G)$. □

Corollary 8.11 (Inapproximability through the SAT-IDS graph)**.** *For every $\varepsilon > 0$, unless* P = NP*, no polynomial-time algorithm returns, on every graph with n vertices, an independent dominating set of size at most $n^{1-\varepsilon}\, i(G)$.*

Proof. Let $t \geq \max(2, 1/\varepsilon)$, so that $n^{1-\varepsilon} \leq n^{1-1/t}$. Given a formula in conjunctive normal form, unit propagation and pure literal elimination either decide it or leave a residual formula φ, every clause a nonempty set of literals with no complementary pair; the graph $G_{\beta_t}(\varphi)$ is built in polynomial time by Theorem 8.9(iii), and by Theorem 8.9(ii) the size of the returned set decides φ. Satisfiability of formulas in conjunctive normal form is NP-complete. □

Corollary 8.11 is the theorem of Halldórsson [2]; Theorem 8.9 gives a self-contained proof of it through the SAT-IDS graph.

Corollary 8.12 (Conditional consequence for SAT)**.** *If Conjecture 6.20 is proved, then the MIDS stage of* `satypy` *with $\beta = \beta_4 = 27p^4q^3 + 1$ in place of $\hat{\beta}$, answering SATISFIABLE if and only if $|S| < \beta_4$, decides every formula in conjunctive normal form correctly in polynomial time.*

Proof. Unit propagation, pure literal elimination and the implication-graph solver are exact and run in polynomial time. $G_{\beta_4}(\varphi)$ has $n = 27p^4q^4 + q + 2p$ vertices and is built in polynomial time (Theorem 8.9(iii)), and Algorithm 1 runs on it in polynomial time (Theorem 5.2). Conjecture 6.20 gives $|S| \le n^{3/4}\, i(G)$, and Theorem 8.9(ii) with $t = 4$ gives the correctness. □

Corollary 8.12 is a second route to the conclusion P = NP of Theorem 6.21.

Branching on the clause named by the output. When $|S| \ge \beta$, the output itself names a clause on which the formula can be split exactly. `satypy` decodes the literal vertices of S, completes them with the value false on the variables with no literal in S, and, if the resulting assignment satisfies φ, answers SATISFIABLE with it; otherwise it chooses a shortest falsified clause C and decides every formula $\varphi \wedge \ell$, $\ell \in C$, by the same procedure with the depth decreased by one, up to a fixed depth d ($d = 2$ by default). It answers SATISFIABLE if a branch is satisfiable, with the checked assignment of that branch, and UNSATISFIABLE if every branch is unsatisfiable; at depth 0 it uses the rule of Theorem 8.7.

Proposition 8.13 (Branching)**.** *Let $\beta > p$, $G = G_\beta(\varphi)$, S an independent dominating set of G with $|S| \ge \beta$, and w the largest clause width.*

(i) S contains the clause vertices of exactly $\lfloor |S|/\beta \rfloor \ge 1$ clauses, and the decoded assignment falsifies at most $\lfloor |S|/\beta \rfloor$ clauses; if it falsifies none, φ is satisfiable.

(ii) For every clause C of φ, φ is satisfiable if and only if $\varphi \wedge \ell$ is satisfiable for some literal $\ell \in C$.

(iii) Branching to depth d runs Algorithm 1 at most $\sum_{h=0}^{d} w^h$ times, each time on the SAT-IDS graph of a formula with at most p variables and q clauses, so it runs in polynomial time for every fixed d.

(iv) Every UNSATISFIABLE answer on an unsatisfiable formula and every SATISFIABLE answer is correct. If a satisfiable φ is answered UNSATISFIABLE, there are satisfiable formulas $\varphi = \varphi_0, \varphi_1, \ldots, \varphi_d$, each φ_{h+1} the residual formula of $\varphi_h \wedge \ell_h$ for a literal ℓ_h, such that the output of Algorithm 1 on the SAT-IDS graph of every φ_h contains a clause vertex.

Proof. (i) Theorem 8.6(iii); $\lfloor |S|/\beta \rfloor \ge 1$ since $|S| \ge \beta$. (ii) Every satisfying assignment of φ satisfies C, hence some $\ell \in C$, and satisfies $\varphi \wedge \ell$; the converse is immediate. (iii) Every node of the branching tree runs Algorithm 1 once and has at most w children, and the tree has depth at most d. Fixing a literal and simplifying removes at least one variable and no clause is added, so every formula of the tree has at most p variables and q clauses, and its SAT-IDS graph has polynomial size by Lemma 8.4. (iv) SATISFIABLE answers carry checked assignments. An unsatisfiable formula is answered UNSATISFIABLE at every node: by (ii) every branch is unsatisfiable, and at depth 0 by Theorem 8.7(iii). Let φ_0 be satisfiable and answered UNSATISFIABLE. The exact stages answer it correctly, so it reaches the MIDS stage and the output on its graph has $|S| \ge \beta$, hence contains a clause vertex (Lemma 8.2(i) and (iii)). If $d \ge 1$, by (ii) some branch $\varphi_0 \wedge \ell_0$ is satisfiable, and since every branch is answered UNSATISFIABLE, so is this one; its residual formula φ_1 is satisfiable and answered UNSATISFIABLE with depth $d - 1$, and induction on d gives the chain. □

9. Experiments

All experiments use Siriaisa 0.0.8 and the scripts of the folder `car/` of the repository: `run_car.py` (Section 9.2), `car_bipartite.py` (Section 9.3), `car_ugc.py` (Section 9.4), and `car_split.py`, `car_cubic.py`, `car_circle.py` and `car_bounded.py` (Section 9.5); Section 9.6 uses the command `batch_iris` of the package. Section 9.7 uses the scripts `small_formulas.py` and `strategy_search.py` of the folder `car/` of the `satypy` repository [23], which imports that repository's own `satypy/`. 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 [19], NumPy 2.4.6 and SciPy 1.17.1 [24]; 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 [25], 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 $H \times K_2$ [26]; it is computed with the Hopcroft–Karp algorithm [27] 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; $i = \text{sc} + |\mathcal{S}|$ 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.

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 4 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 36.6 ms per instance on average, and the integer program 9.5 ms.

Table 4: 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. 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 5 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 15 of them $|S|$ equals the best solution found by the integer program, 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 10.8 s per instance and 362.2 s in total; the integer programs took 1 439.5 s and the whole run 1 803.9 s.

Table 5: 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	23.4
3D grid	2	1	1	0	1	1.0750	1.0375	5.7
complete bipartite	2	2	2	0	0	1.0000	1.0000	2.9
crown graph	2	2	2	0	0	1.0000	1.0000	0.1
double cover $G \times K_2$ of random regular	12	8	8	0	4	1.1765	1.0370	52.6
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	49.1
hypercube	2	2	2	0	0	1.0000	1.0000	3.3
hypercube, 50% edges kept	2	2	2	0	0	1.0000	1.0000	6.8
path	2	2	2	0	0	1.0000	1.0000	0.0
power-law bipartite	6	6	6	0	0	1.0000	1.0000	25.6
random bipartite (1:2)	18	15	15	0	3	1.4545	1.0657	50.3
random bipartite (balanced)	18	15	15	0	3	1.4167	1.0602	53.0
random d-regular bipartite	18	12	12	0	6	1.2917	1.0623	63.7
random tree	6	6	6	0	0	1.0000	1.0000	20.8
All	**112**	**95**	**95**	**0**	**17**	**1.4545**	**1.0349**	**362.2**

9.4. 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 [15]: 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 [12]); 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 6 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 [22], so they measure the algorithm on bounded-degree VC-IDS graphs without testing Corollary 7.4. The whole run took 1 061.5 s; Siriaisa took at most 25.6 s per instance.

Table 6: 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**

9.5. 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 7 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 $K_{200,200}$, whose integer program reached its time limit and for which $i = 200$ by Proposition 6.11; 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 1 286.8 s.

Table 7: 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	19.1
planar (Delaunay)	4	100–400	12	4	0	0.000	4	4	3	1	0	1.0161	18.6
random 4-regular	4	100–400	4	4	0	0.000	4	2	2	0	2	1.1481	20.6
random 5-regular	4	100–400	5	4	0	0.000	4	2	2	0	2	1.2239	18.5
random 6-regular	4	100–400	6	4	0	0.000	4	2	2	0	2	1.2586	17.8
random 4-regular bipartite*	4	100–400	4	4	0	0.000	4	2	2	0	2	1.1481	19.8
random 5-regular bipartite*	4	100–400	5	4	0	0.000	4	2	2	0	2	1.2059	17.8
SC-IDS, $B = 4$	4	94–375	4	4	0	0.000	4	4	3	1	0	1.0080	21.9
SC-IDS, $B = 6$	4	93–384	6	4	0	0.000	4	4	4	0	0	1.0000	19.3
VC-IDS, 3-regular H	4	240–1 000	3	4	0	0.000	4	2	2	0	2	1.4400	34.6
$G(n, 1/2)$	4	100–400	230	0	3 783	0.473	0	2	2	0	2	2.5000	10.0
complete K_n	4	100–400	399	0	7 600	0.950	0	4	4	0	0	1.0000	2.1
$K_{n/2,n/2}$	4	100–400	200	0	3 600	0.450	4	4	4	0	0	1.0000	9.5
$K_{n/4,3n/4}$	4	100–400	300	0	2 600	0.325	2	4	4	0	0	1.0000	9.4
complete 4-partite, equal parts	4	100–400	300	0	5 600	0.700	0	4	4	0	0	1.0000	10.7
complete 5-partite, random parts	4	100–400	392	0	5 889	0.736	0	4	4	0	0	1.0000	7.5
crown	4	100–400	199	0	3 600	0.450	0	4	4	0	0	1.0000	1.1
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	11.8
split graph	4	100–400	232	0	209	0.055	0	4	4	0	0	1.0000	4.5
$G(n, 3/n)$	4	100–400	11	4	0	0.000	4	4	3	1	0	1.0075	29.4
Barabási–Albert, $m = 3$	4	100–400	67	0	19	0.013	0	4	2	2	0	1.0532	17.8
power-law bipartite	4	100–400	55	0	6	0.004	1	4	4	0	0	1.0000	23.6
2D grid	4	100–400	4	4	0	0.000	4	2	2	0	2	1.0814	24.9
hypercube	2	128–512	9	2	0	0.000	2	1	1	0	1	1.2308	5.1
Khot–Regev YES	2	640	64	0	323	0.020	0	0	0	0	2	15.9000	7.2
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	19.2
VC-IDS, Khot–Regev H	2	49 096–51 550	64	2	0	0.000	2	0	0	0	2	22.1825	64.0
All	**110**			**54**	**7 600**	**0.950**	**69**	**87**	**79**	**8**	**23**	**22.1825**	**465.9**

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 6.16(i)–(ii), that the copy graph equals G when $s(G) = 0$, and Theorem 6.17: $|S| \le n/2$ on regular graphs and $|S| \le 2L$ on cubic graphs. Table 8 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 582.7 s.

Table 8: 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 SIRIAISA 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	1.4
random cubic	4	100–400	0	4	2	2	0	2	1.0693	0.2700	24.1
random cubic bipartite	4	100–400	0	4	2	2	0	2	1.0693	0.2700	25.5
double cover of random cubic	4	100–400	0	4	2	2	0	2	1.0693	0.2800	24.1
cubic planar (dual of Delaunay)	4	100–400	0	4	2	2	0	2	1.0792	0.2725	22.3
cubic planar bipartite	4	100–400	0	4	2	2	0	2	1.0693	0.2800	25.0
random cubic, 20% edges deleted	4	100–400	0	4	4	4	0	0	1.0000	0.3250	33.5
prism $C_t \times K_2$	4	100–402	0	4	3	3	0	1	1.0099	0.2600	15.1
Möbius ladder	2	100–400	0	2	2	2	0	0	1.0000	0.2525	12.4
generalized Petersen $GP(t, 2)$	2	100–400	0	2	1	1	0	1	1.1650	0.3000	13.0
generalized Petersen $GP(t, 3)$	2	100–400	0	2	2	2	0	0	1.0000	0.2600	5.9
All	**61**			**46**	**49**	**49**	**0**	**12**	**1.1650**	**0.5000**	**202.4**

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 6.19(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 6.19(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 9 reports the results. The degrees from the model agree with the graph on all instances; computing $s(G)$ from the model took at most 2.0 ms per instance and building the graph at most 53 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 $|S|/L \le 1.2143$. The run took 241.0 s.

Table 9: 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 SIRIAISA 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	1.2143	5.1
short chords, degree ≈ 3	4	100–400	904	14	3	2	0.002	3	4	4	0	0	1.0000	14.7
short chords, degree ≈ 10	4	100–400	2 500	29	0	107	0.119	0	4	2	2	0	1.0405	8.8
short chords, degree $\approx \sqrt{n}$	4	100–400	5 488	57	0	354	0.092	0	4	2	2	0	1.0600	6.1
permutation graph	4	100–400	41 274	393	0	3 925	0.563	0	4	4	0	0	1.0000	7.4
2 hub chords	4	100–400	532	102	0	15	0.007	1	4	4	0	0	1.0000	11.9
4 hub chords	4	100–400	599	54	0	14	0.006	1	4	4	0	0	1.0000	13.0
diameters (K_n)	2	100–400	79 800	399	0	7 600	1.000	0	2	2	0	0	1.0000	0.7
$K_{n/2,n/2}$	2	100–400	40 000	200	0	3 600	0.474	2	2	2	0	0	1.0000	3.4
$K_{n/4,3n/4}$	2	100–400	30 000	300	0	2 600	0.361	1	2	2	0	0	1.0000	3.2
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**	**1.2143**	**74.4**

Bounded degree with $n \ge B^2$. On graphs of maximum degree at most B with $n \ge B^2$, $s(G) = 0$ (Proposition 6.15(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 $i = \mathrm{sc} + |\mathcal{S}|$ 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 10 reports the results. $s(G) = 0$ on all 44 instances. The optimum is proved on 25 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 1 and 2 vertices on two 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 $L = 338$, and on the open Khot–Regev graph it has 173 vertices against 170 and $L = 158$. The largest time of Sriaisa on an instance is 13.2 s, and the run took 1 551.2 s.

Table 10: 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 Sriaisa 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	11.2
random B-regular bipartite (double cover)	4	2	256	512	2	0	0	0	2	1.1154	1.1058	12.1
B-regular, planted perfect code	4	2	260	520	2	2	2	0	0	1.0000	1.0000	0.9
SC-IDS	4	2	323	430	2	2	2	0	0	1.0000	1.0000	13.7
VC-IDS, B-regular H	4	2	260	320	2	2	2	0	0	1.0000	1.0000	15.0
random B-regular	8	2	256	1 024	2	0	0	0	2	1.3793	1.3621	5.1
random B-regular bipartite (double cover)	8	2	256	1 024	2	0	0	0	2	1.3448	1.3448	5.3
B-regular, planted perfect code	8	2	261	1 044	2	2	2	0	0	1.0000	1.0000	0.5
SC-IDS	8	2	331–334	769	2	2	2	0	0	1.0000	1.0000	10.7
VC-IDS, B-regular H	8	2	250	320	2	2	2	0	0	1.0000	1.0000	15.5
random B-regular	16	2	256	2 048	2	0	0	0	2	1.5625	1.5312	3.3
random B-regular bipartite (double cover)	16	2	256	2 048	2	0	0	0	2	1.5625	1.5625	4.4
B-regular, planted perfect code	16	2	272	2 176	2	2	2	0	0	1.0000	1.0000	0.6
SC-IDS	16	2	318–321	1 353	2	2	0	2	0	1.0115	1.0114	9.7
VC-IDS, B-regular H	16	2	882	1 152	2	2	1	1	0	1.0033	1.0016	20.4
random B-regular	32	2	1 024	16 384	2	0	0	0	2	2.2188	2.2188	6.0
random B-regular bipartite (double cover)	32	2	1 024	16 384	2	0	0	0	2	2.2500	2.2500	4.3
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	6.8
VC-IDS, B-regular H	32	2	3 298	4 352	2	2	0	2	0	1.0009	1.0009	13.6
Khot–Regev YES, $d = 3$, $s_b = 4$	12	2	160	382	2	2	1	1	0	1.0364	1.0182	12.3
Khot–Regev YES, $d = 3$, $s_b = 8$	24	2	608	2 938	2	1	0	1	1	1.0949	1.0505	12.4
All		**44**			**44**	**25**	**18**	**7**	**19**	**2.2500**	**1.2112**	**185.3**

9.6. Complements of the DIMACS clique benchmark

The complements of the 68 graphs of the DIMACS clique benchmark [28], in the form distributed with the NPBench collection [29], were solved with `batch_iris` on the folder `experiment/` of the repository, with the size option and logging; Table 11 reports the size of the returned independent dominating set and the time of Sriaisa recorded in the log. The optimum is not known for these instances, and no integer program was run. The time ranged from 0.016 s (`MANN_a9`) to 15.0 s (`p_hat700-1`), with mean 2.87 s and total 195.1 s.

Table 11: Complements of the DIMACS clique benchmark graphs: size $|S|$ of the returned set and time of Siriaisa in seconds.

Instance	$\lvert S\rvert$	Time	Instance	$\lvert S\rvert$	Time
brock200_1	8	1.13	brock200_2	4	1.40
brock200_3	5	1.16	brock200_4	6	1.16
brock400_1	10	1.66	brock400_2	11	1.54
brock400_3	11	1.53	brock400_4	11	1.58
brock800_1	8	6.97	brock800_2	9	6.72
brock800_3	9	6.99	brock800_4	9	6.68
c-fat200-1	10	1.21	c-fat200-2	22	1.21
c-fat200-5	56	1.43	c-fat500-1	12	6.10
c-fat500-10	124	4.06	c-fat500-2	24	6.25
c-fat500-5	62	4.94	C1000.9	31	3.43
C125.9	14	2.02	C250.9	18	1.34
C500.9	25	1.58	gen200_p0.9_44	16	1.45
gen200_p0.9_55	16	1.54	gen400_p0.9_55	21	1.41
gen400_p0.9_65	22	1.41	gen400_p0.9_75	24	1.36
hamming10-2	128	2.90	hamming10-4	13	5.25
hamming6-2	12	0.91	hamming6-4	2	0.04
hamming8-2	32	2.72	hamming8-4	4	1.83
johnson16-2-4	8	1.34	johnson32-2-4	16	1.61
johnson8-2-4	4	0.36	johnson8-4-4	7	1.96
keller4	5	1.40	keller5	10	4.31
MANN_a27	27	0.13	MANN_a45	45	0.19
MANN_a81	81	0.34	MANN_a9	9	0.02
p_hat1000-3	15	7.88	p_hat300-1	3	2.18
p_hat300-2	5	1.73	p_hat300-3	10	1.20
p_hat500-1	3	5.24	p_hat500-2	7	7.33
p_hat500-3	11	4.20	p_hat700-1	3	15.02
p_hat700-2	7	10.86	p_hat700-3	14	6.19
san200_0.7_1	7	1.99	san200_0.7_2	6	1.84
san200_0.9_1	16	2.24	san200_0.9_2	16	2.28
san200_0.9_3	15	2.39	san400_0.5_1	4	3.81
san400_0.7_1	8	1.93	san400_0.7_2	8	1.88
san400_0.7_3	8	1.91	san400_0.9_1	22	1.36
sanr200_0.7	7	1.09	sanr200_0.9	16	1.45
sanr400_0.5	5	2.62	sanr400_0.7	9	1.89

9.7. The SAT-IDS graphs of satypy

The scripts `small_formulas.py` and `strategy_search.py` decide formulas with `satypy` 0.0.4, with $\beta = \hat{\beta}$ copies (Lemma 8.4) and branching to depth 2 (Proposition 8.13), and with exhaustive search over all assignments, and compare the answers.

Small formulas. The script `small_formulas.py` used the eight DIMACS files of the folders `benchmarks/simple` and `benchmarks/test_formulas` of the `satypy` repository with at most ten variables and 180 random 3-CNF formulas, five for every number of variables from 3 to 5 and every number of clauses from 1 to 12 (seed 2026). All 188 answers agree with exhaustive search: 70 formulas are decided by unit propagation and pure literal elimination, 115 satisfiable formulas on the SAT-IDS graph of their residual formula, with $|S| = p$, and 3 unsatisfiable formulas by branching. The SAT-IDS graphs have 38 to 1 084 vertices and 99 to 3 222 edges, and each formula takes at most 5.0 seconds. Table 12 lists the graphs by residual size (p, q). On the 3 unsatisfiable formulas $|S| = \beta + p$, so by Theorem 8.6(iii) the output contains the clause vertices of exactly one clause, and $\mu(\varphi) = 1$.

Table 12: The SAT-IDS graphs of the 118 formulas of `small_formulas.py` decided on the MIDS graph, grouped by the residual size (p, q) and the answer of exhaustive search. "Count" is the number of formulas, $\beta = \hat{\beta}$, $|S|$ the size of the output of Algorithm 1 on the graph of the residual formula, the same for all formulas of a row, and $2p^2$ the threshold of the rule of Theorem 8.7.

p	q	Answer	Count	β	n	$\lvert E\rvert$	$\lvert S\rvert$	$2p^2$
3	2	SAT	5	16	38	99	3	18
3	3	SAT	5	16	54	147	3	18
3	4	SAT	8	16	70	195	3	18
3	5	SAT	6	16	86	243	3	18
3	6	SAT	5	16	102	291	3	18
3	7	SAT	6	16	118	339	3	18
3	8	SAT	5	16	134	387	3	18
3	8	UNSAT	1	16	134	387	19	18
3	9	SAT	5	16	150	435	3	18
3	10	SAT	4	16	166	483	3	18
3	10	UNSAT	1	16	166	483	19	18
3	11	SAT	5	16	182	531	3	18
3	12	SAT	4	16	198	579	3	18
3	12	UNSAT	1	16	198	579	19	18
4	4	SAT	4	29	124	352	4	32
4	5	SAT	2	29	153	439	4	32
4	6	SAT	5	29	182	526	4	32
4	7	SAT	4	29	211	613	4	32
4	8	SAT	3	29	240	700	4	32
4	9	SAT	5	29	269	787	4	32
4	10	SAT	5	29	298	874	4	32
4	11	SAT	5	29	327	961	4	32
4	12	SAT	5	29	356	1 048	4	32
5	7	SAT	2	46	332	971	5	50
5	8	SAT	1	46	378	1 109	5	50
5	9	SAT	5	46	424	1 247	5	50
5	10	SAT	2	46	470	1 385	5	50
5	11	SAT	4	46	516	1 523	5	50
5	12	SAT	4	46	562	1 661	5	50
6	16	SAT	1	67	1 084	3 222	6	72

Formulas built for the MIDS stage. The script `strategy_search.py` generates 3-CNF formulas with 3 to 6 variables and at most 16 clauses (seed 2026) of three kinds and keeps those whose residual formula reaches the SAT-IDS graph: satisfiable formulas with a single satisfying assignment, planted and completed by clauses it satisfies; satisfiable formulas satisfied by neither constant assignment; and unsatisfiable formulas. It obtained 20, 20 and 19 such formulas. For every formula it records the answer of `satypy` and, on the SAT-IDS graph of the residual formula, the output of every strategy $c_1, \ldots, c_{18}$ of Algorithm 1. All 59 answers agree with exhaustive search. On all 40 satisfiable formulas the output has $|S| = p$ and the answer is decided on the graph without branching; the 19 unsatisfiable formulas are decided by branching, each with $|S| = \beta + p$ and $\mu(\varphi) = 1$. On all 59 formulas $\lfloor |S|/\beta \rfloor \geq \mu(\varphi)$, and the decoded assignment falsifies at most $\lfloor |S|/\beta \rfloor$ clauses, as Theorem 8.6(ii) and (iii) state. The graphs have 166 to 1 084 vertices and 483 to 3 222 edges, and each formula takes at most 30.7 seconds. Table 13 gives, for every strategy, the number of satisfiable formulas on which it returns p vertices.

Table 13: Number of the 40 satisfiable formulas of `strategy_search.py` on whose SAT-IDS graph each strategy of Algorithm 1 returns a minimum independent dominating set (p vertices).

Strategy	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
Optimal	14	13	26	9	32	17	0	0	14
Strategy	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}
Optimal	36	36	0	11	0	39	39	39	40

By Lemma 8.4, the SAT-IDS graph with $\hat{\beta} = 2p^2 - p + 1$ copies grows as $2qp^2$. Table 14 gives its size for residual 3-CNF formulas with the numbers of variables and clauses of the uniform random benchmark `uf20-91` and of three instances of the `aim` family.

Table 14: Size of the SAT-IDS graph of a residual 3-CNF formula with p variables and q clauses and $\hat{\beta}$ copies (Lemma 8.4); $|E| = p + 3q\hat{\beta}$.

Size of	p	q	$\hat{\beta}$	n	$\lvert E\rvert$
`uf20-91`	20	91	781	71 111	213 233
`aim-50-1_6`	50	80	4 951	396 180	1 188 290
`aim-100-1_6`	100	160	19 901	3 184 360	9 552 580
`aim-200-2_0`	200	400	79 801	31 920 800	95 761 400

10. Conclusion

Siriaisa 0.0.8 returns an independent dominating set (Theorem 4.3) in $O((n + m)\,\alpha_{\mathrm{A}}(n))$ 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, 6.11 and 6.15). 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 6.16 and Theorem 6.17); 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 6.18). 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 6.19), and Conjecture 6.20 restricted to circle graphs would imply P = NP (Corollary 6.22). 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 9.5, 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.

On the SAT-IDS graph of the solver `satypy` 0.0.4, every independent dominating set has p or at least $\beta + p > 2p^2$ vertices, so `satypy` answers every unsatisfiable formula correctly, every SATISFIABLE answer carries a satisfying assignment, and every answer is correct when the ratio on the graph is at most $2p$ (Theorem 8.7); the bound of Theorem 6.8 never separates satisfiable from unsatisfiable formulas (Proposition 8.8), branching on the clause named by the output is exact and polynomial for every fixed depth (Proposition 8.13), and with $\beta_t = p^t(3q)^{t-1} + 1$ copies every $n^{1-1/t}$-approximation decides satisfiability (Theorem 8.9), so that Conjecture 6.20 would decide every formula in polynomial time (Corollary 8.12); its independent domination number determines the minimum number of falsified clauses (Theorem 8.6), with β^* copies its splitting excess is asymptotic to $2pL$ (Proposition 8.5), and on 247 small formulas every answer agrees with exhaustive search, with an optimal output on all 155 satisfiable instances of the graph (Section 9.7).

We do not claim that Siriaisa has ratio $n^{3/4}$ on every graph; this is Conjecture 6.20, and a proof would imply P = NP (Theorem 6.21). 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 [13] holds. The implementation is freely available as the `siriaisa` package on PyPI [14].

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 `v0.0.8`); 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. 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.

[12] 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.

[13] 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

[14] F. Vega, Siriaisa: Approximate Independent Dominating Set Solver, https://pypi.org/project/siriaisa, python package version 0.0.8 (2026).

[15] 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).

[16] F. Vega, Salvador: Approximate Vertex Cover Solver, https://pypi.org/project/salvador, python package version 0.1.4 (2026).

[17] F. Vega, Esperanza: Approximate Independent Set Solver, https://pypi.org/project/esperanza, python package version 0.1.6 (2026).

[18] 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.

[19] 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.
[20] Y. Caro, New results on the independence number, Tech. rep., Tel-Aviv University (1979).
[21] V. K. Wei, A lower bound on the stability number of a simple graph, Tech. Rep. Technical Memorandum 81-11217-9, Bell Laboratories (1981).
[22] 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.
[23] F. Vega, Satypy: A SAT Solver, https://pypi.org/project/satypy, python package version 0.0.4 (2026).
[24] 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.
[25] 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.
[26] G. L. Nemhauser, L. E. Trotter, Vertex packings: Structural properties and algorithms, Mathematical Programming 8 (1) (1975) 232–248. doi:10.1007/BF01580444.
[27] 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.
[28] 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.
[29] DynaRoARS, NPBench: A collection of benchmarks (in DIMACS format) for various NP-complete problems, GitHub Repository. URL https://github.com/dynaroars/npbench