An Approximate Solution to the Minimum Vertex Cover Problem: The Salvador Algorithm

Frank Vega

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

hexagon:2610.00337v2

Abstract

Minimum Vertex Cover has a classical linear-time ratio of 2, and under the Unique Games Conjecture no polynomial-time algorithm achieves a fixed ratio below 2 unless P = NP. We present version 0.1.6 of Salvador, a linear-time ensemble of seventeen strategies, numbered in the order in which they are computed: twelve base strategies (component amplification before a maximal matching, the maximal matching itself, degree and weight heuristics, reductions to auxiliary problems that are solved exactly, a decomposition into bipartite layers, a weighted subdivision reduction and a bounded-treewidth dynamic program), followed by five improvers (a union pass, local search, reduce-and-search, consensus forcing and a swap-maximize refinement). We prove that every strategy runs in $O((n+m)\,\alpha(n))$ time, and that the returned cover has at most $2\tau - r$ vertices, where r is the number of components with an edge, so the ratio is at most 1.999999999 whenever $\tau \leq 10^9 r$. We prove exactness or a bounded ratio on several classes of graphs, among them every bipartite graph on which the Hopcroft–Karp phases of the layer strategy reach a maximum matching, and we give a per-instance certificate: a cover with at most $2L - |M|$ vertices, for a lower bound $L \leq \tau$ and the maximal matching M, is within ratio 3/2 on that instance. In a reproducible suite the returned cover satisfies $|C| \leq 2\tau - |M|$ on all 291 tested graphs with a known optimum and is certified on the six without one, although single strategies violate it; on the 68 DIMACS clique complements it is never worse than Hvala and optimal on 39 instances against 18. Whether $|C| \leq 2\tau - |M|$ holds on every graph is stated as a conjecture: a proof would give a linear-time 3/2-approximation and would refute the Unique Games Conjecture unless P = NP.

Keywords: Vertex Cover, Approximation Algorithm, Maximal Matching, Split Graph, Bipartite Layers, Ensemble Heuristics, Linear Time, Unique Games Conjecture, P versus NP
2020 MSC: 05C69, 05C70, 68Q25, 90C27

1. Introduction

The **Minimum Vertex Cover** (VC) problem is one of the twenty-one NP-complete problems identified by Karp [1]. Given an undirected graph $G = (V, E)$, a set $C \subseteq V$ is a *vertex cover* if every edge has at least one endpoint in C; the goal is a cover of minimum size $\tau(G)$. We write $n = |V|$, $m = |E|$, $\nu(G)$ for the matching number, and $r(G)$ for the number of connected components of G that have an edge.

The approximation landscape is well charted. Both endpoints of a maximal matching form a 2-approximation computable in linear time [2], and LP-based refinements reach $2 - \Theta(1/\sqrt{\log n})$ [3, 4], which is $2 - o(1)$ but not a constant below 2. On the hardness side, Dinur and Safra ruled out ratios below $10\sqrt{5} - 21 \approx 1.3606$ unless P = NP [5], and the proof of the 2-to-2 Games Theorem raised this unconditional threshold to $\sqrt{2} - \varepsilon$ [6]. Under the Unique Games Conjecture (UGC) [7], Khot and Regev showed that no fixed ratio $2 - \varepsilon$ is achievable in polynomial time unless P = NP [8], and Austrin, Khot and Safra refined this to graphs of bounded degree [9], matching Halperin's algorithm [10]. The barrier persists on triangle-free graphs. Every vertex cover contains at least two vertices of each triangle, so taking all three vertices of a triangle and deleting them, until no triangle is left, and then running a

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

$(2-\varepsilon)$-approximation for triangle-free graphs on the rest would approximate every graph within $\max\{3/2,\ 2-\varepsilon\}$. Hence approximating Minimum Vertex Cover within $2-\varepsilon$ on triangle-free graphs is UGC-hard, as observed by Bonnet and Rzążewski [11], who also note that for graphs of girth at least 5 neither a $(2-\varepsilon)$-approximation nor a UGC-hardness proof is known. 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 [12]; at the time of writing it has not been peer reviewed. If it holds, every fixed ratio below 2 is NP-hard, and in particular so is the ratio $3/2$, which lies above the unconditional threshold $\sqrt{2}$ but below 2.

Salvador is an ensemble algorithm, implemented in the Python package `Salvador` [13], that computes seventeen strategies $c_1, \ldots, c_{17}$ in linear time and returns the smallest refined cover. The strategies are numbered in the order in which they are computed. The twelve *base strategies* $c_1, \ldots, c_{12}$ build covers from the graph: component amplification before a maximal matching (c_1), the maximal matching itself (c_2), degree and weight heuristics (c_3 to c_6), three reductions to auxiliary problems that are solved exactly (c_7, c_8 and c_{10}), a decomposition into bipartite layers (c_9), a weighted subdivision reduction with exact reductions and folding (c_{11}), and a bounded-treewidth dynamic program (c_{12}). The five *improvers* $c_{13}, \ldots, c_{17}$ work on the covers already found: a union pass, a $(1,2)$-swap local search, a search that alternates with exact reductions, consensus forcing and a final swap-maximize refinement of every candidate. Three of the base strategies and the pruning pass adapt the published Hvala ensemble [14, 15], so Salvador contains Hvala and inherits its worst-case ratio 2.

The guarantees of the ensemble come from different strategies on different graphs. The maximal matching c_2 gives a cover of at most $2\tau - r$ vertices on every graph (Theorem 5.3), because every component with an edge saves one vertex. The component amplification c_1 uses this observation in the other direction: it creates components before matching. Every vertex u has two copies $(u, 0)$ and $(u, 1)$; an edge of the maximal matching M whose endpoints both have index 0 is split off as an isolated edge, and the other edges of its endpoints move to the copies of index 1. Splitting stops when the split graph has r' components, the largest value not above $|M|$, so that $|M| - 1 \le r' \le |M| \le \tau$. The bipartite layers c_9 handle odd cycles differently: a non-bipartite component is split into at most $\lfloor \log_2 m \rfloor + 1$ bipartite layers, each solved by a matching, a few Hopcroft–Karp phases and Kőnig's theorem, and a vertex chosen in one layer is discarded from the later ones. Minimum-degree rules make c_5 and c_8 exact on every regular bipartite graph, a Caro–Wei argument bounds c_8 on every graph, and the exact reductions and dynamic programs of c_{10}, c_{11} and c_{12} certify their own optimality whenever their cores are empty.

Contributions. The main contributions of this paper are:

- The ensemble of seventeen strategies (Section 3, Algorithm 1), with the component amplification c_1 (Algorithms 3 and 4, Figure 1) and the bipartite layers c_9 (Algorithm 5).
- Linear running time: with the default parameters every strategy, and therefore the whole ensemble, runs in $O((n+m)\,\alpha(n))$ time, with α the inverse Ackermann function (Theorem 6.1).
- A bound for every graph: the returned cover satisfies $|C| \le 2\tau(G) - r(G)$ (Theorem 5.3); hence the ratio is at most 1.999999999 on every graph with $\tau \le 10^9 r$, in particular on every graph with at most $10^9 + 1$ vertices (Corollary 5.4). The margin r/τ tends to 0 as τ grows, so this is not a $(2-\varepsilon)$-approximation.
- Per-class guarantees (Section 5.2): c_5 is exact on forests, cycles, complete and complete bipartite graphs (Theorem 5.7); c_5 and c_8 are exact on every regular or degree-separated bipartite graph (Theorem 5.11); c_8 meets the Caro–Wei bound, hence ratio $2 - 2/(d+1)$ on d-regular graphs (Lemma 5.8, Theorem 5.9); c_3 achieves $11/6$ on subcubic graphs (Theorem 5.14); c_4 satisfies a Cauchy–Schwarz bound (Theorem 5.15); c_9 uses at most $\lfloor \log_2 m \rfloor + 1$ layers and is a minimum cover of every bipartite component on which its phases reach a maximum matching (Proposition 5.21); and a combined certificate with an exact a-posteriori form on the residue $\mathcal{R}$ of nearly Kőnig–Egerváry graphs (Theorems 5.22 and 5.24).
- The split graph of c_1: its component count is computed for every number of splits in one union-find pass, and $r(G) \le r' \le |M| \le \tau$ with $r' \ge |M| - 1$ (Lemmas 5.25 and 5.26).
- A per-instance $3/2$ certificate (Proposition 5.28): any cover with $|C| \le 2L - |M|$ or $|C| \le \frac{3}{2}L$, for a computable lower bound $L \le \tau$, is within ratio $3/2$ of the optimum on that instance; and the conjecture that the returned cover

always satisfies $|C| \leq 2\tau - |M|$ (Conjecture 5.30), with the conditional theorem that a proof would refute UGC unless P = NP, and would imply P = NP if the theorem of [12] holds (Theorem 5.32).

- A reproducible experimental suite (Section 7): small graphs with known optima, large adversarial graphs with certified bounds, a strategy report and an ablation, bipartite, Khot–Regev YES-type, regular and triangle-free graphs, a study of the bags of c_{10}, a test of the conjectured bound and of the layers of c_9, a comparison with Hvala on the 68 DIMACS clique complements, and the running time.

Section 3 describes the algorithm. Section 4 proves correctness, Section 5 the approximation results, and Section 6 the running time. Section 7 reports the experiments, and Section 8 concludes.

2. Research Data

The algorithm is implemented in the Python package `Salvador`, freely available on the Python Package Index [13]. The version described in this paper is v0.1.6. Code metadata is summarised in Table 1.

Nr.	Code metadata description	Metadata
C1	Current code version	v0.1.6
C2	Permanent link to code repository	https://github.com/frankvegadelgado/salvador
C3	Permanent link to reproducible capsule	https://pypi.org/project/salvador/
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Languages, tools, and services used	Python $\geq$ 3.12, NetworkX $\geq$ 3.4.2, NumPy $\geq$ 2.2.1, SciPy $\geq$ 1.15.0
C7	Compilation requirements and dependencies	Python, NetworkX, NumPy, SciPy

Table 1: Code metadata for the `Salvador` package (v0.1.6).

3. Description of the Algorithm

3.1. Main ensemble

Salvador's entry point, FindVertexCover (Algorithm 1), removes self-loops and isolated vertices and then computes the strategies in their numbering order. The base strategies $c_1, \ldots, c_9$ are computed from the graph alone; c_{10}, c_{11} and c_{12} also receive the best cover found before them, which they use on the parts of the graph that their exact methods do not reach; the improvers $c_{13}, \ldots, c_{16}$ combine or improve the covers already found; and c_{17} refines every candidate $c_1, \ldots, c_{16}$ and returns the smallest refined cover.

Every strategy that receives an earlier cover is guarded: c_{10}, c_{12}, c_{14} and c_{15} are replaced by the cover they start from whenever that cover is smaller, c_{16} returns the smallest cover of its pool, and the refinement never enlarges a cover. Every candidate is pruned by Algorithm 2, which removes in one pass every vertex whose neighbours all lie in the current set.

3.2. Base strategies

Throughout, G has no self-loops and no isolated vertices, M is the maximal matching returned by the NetworkX routine `min_maximal_matching`, which builds a maximal matching greedily in one scan of the edges, and $\mathrm{pos}(u)$ is the position of u in the vertex order of G, used to break ties. The same NetworkX routine is the only maximal-matching subroutine of every strategy.

Algorithm 1 FindVertexCover(G) — Salvador 0.1.6

Require: Undirected graph $G = (V, E)$
Ensure: A vertex cover $C \subseteq V$

remove all self-loops and isolated vertices from G; **if** $E = \emptyset$ **return** $\emptyset$
$adj \leftarrow \{v \mapsto N_G(v)\}$ ▷ base strategies
$c_1 \leftarrow$ ComponentSplitCover(G) ▷ Algorithm 4
$c_2 \leftarrow$ MaximalMatchingCover(G); $c_3 \leftarrow$ BucketDegreeGreedy(adj); $c_4 \leftarrow$ Hallelujah(G)
$c_5 \leftarrow$ MinToMinCover(adj); $c_6 \leftarrow$ LinearWeightedCover(G); $c_7 \leftarrow$ SolveVC(G)
$c_8 \leftarrow$ BipartitePlanarCover(adj); $c_i \leftarrow$ Prune(adj, c_i) for $i = 2, \ldots, 8$
$c_9 \leftarrow$ Prune(adj, BipartiteLayersCover(adj)) ▷ Algorithm 5
$c_{10} \leftarrow$ EDSTreewidthCover(adj, arg min$_{i \le 9} |c_i|$) ▷ bipartite components
$c_{11} \leftarrow$ SubdivisionCover(adj, arg min$_{i \le 10} |c_i|$)
$c_{12} \leftarrow$ TreewidthDPCover(adj, arg min$_{i \le 11} |c_i|$)
$c_{13} \leftarrow$ Prune($adj, c_2 \cup \cdots \cup c_8$) ▷ improvers
$c_{14} \leftarrow$ LocalSearchCover(adj, arg min$_{i \le 13} |c_i|$)
$c_{15} \leftarrow$ ReduceSearchCover(adj, arg min$_{i \le 14} |c_i|$)
$c_{16} \leftarrow$ ConsensusForceCover($adj, \{c_1, \ldots, c_{15}\}$)
$v_{\max} \leftarrow$ a vertex of maximum degree
$c_{17} \leftarrow$ arg min$_{1 \le i \le 16} |$RefineCover($G, c_i, v_{\max}$)$|$
return c_{17}

Algorithm 2 Prune(adj, C)

Require: Adjacency map adj, a vertex cover C
Ensure: A vertex cover $C' \subseteq C$

$C' \leftarrow C$
for v in a fixed snapshot of C **do**
    if $N_G(v) \subseteq C'$ **then** $C' \leftarrow C' \setminus \{v\}$
    end if
end for
return C'

Component amplification: c_1.. The nodes of the *split graph* H are *copies* (u, i) of vertices u of G with an *index* $i \in \{0, 1\}$. Initially H has the copy $(u, 0)$ of every vertex and the edge $((u, 0), (v, 0))$ for every edge $\{u, v\}$ of G; the *projection* $\pi(u, i) = u$ maps the edges of H onto the edges of G. To *split* an edge $((u, 0), (v, 0))$ whose endpoints both have index 0, the edge is kept, every other edge of $(u, 0)$ moves to $(u, 1)$, and every other edge of $(v, 0)$ moves to $(v, 1)$; the split edge becomes an isolated K_2 of H (Figure 1(a)). Only edges with index 0 at both endpoints are split, and every vertex is an endpoint of at most one split edge, so no index larger than 1 is ever needed and no bridge has to be found. The edges split are those of M, in the order of the scan: they are pairwise disjoint, so when an edge of M comes up both its endpoints still have index 0. Splitting the first k edges F_k of M leaves the split graph with

$$\text{count}(k) \;=\; k + c(G - F_k)$$

components that have an edge, where $c(\cdot)$ counts the components with an edge: the k split edges are isolated, and every other edge of G keeps its endpoints connected through the copies of index 1. Algorithm 3 computes all the values $\text{count}(0), \ldots, \text{count}(|M|)$ in one backward pass of a union-find structure, starting from $G - F_{|M|}$ and adding the edges of M back one by one, and stops at the first k at which $\text{count}(k)$ reaches the largest value that does not exceed $|M|$. Lemma 5.26 shows that this value r' satisfies $r(G) \le r' \le |M|$ and $r' \ge |M| - 1$.

Algorithm 3 SplitGraph(G, M)

Require: Graph G without isolated vertices, the maximal matching $M = (e_1, \ldots, e_{|M|})$ of G
Ensure: Split graph H with copies of index 0 and 1, and r' components with an edge

$UF \leftarrow$ union-find over V with the edges $E \setminus M$; $c \leftarrow$ its trees that contain an edge
$\text{count}(|M|) \leftarrow |M| + c$
for $k = |M| - 1$ **down to** 0 **do**
    join the endpoints of e_{k+1} in UF and update c; $\text{count}(k) \leftarrow k + c$
end for
$r' \leftarrow \max\{\text{count}(k) : \text{count}(k) \le |M|\}$; $k^* \leftarrow \min\{k : \text{count}(k) = r'\}$
$S \leftarrow$ the endpoints of $e_1, \ldots, e_{k^*}$
$H \leftarrow$ the edges $((u, 0), (v, 0))$ for $uv \in \{e_1, \ldots, e_{k^*}\}$, and $((u, [u \in S]), (v, [v \in S]))$ for every other $uv \in E$
return H, r'

Let M_H be the maximal matching of H returned by `min_maximal_matching`; it contains every isolated K_2, because such an edge has no neighbouring edge. Every matched edge is written $((a, i), (b, j))$ with $(i, \text{pos}(a)) \le (j, \text{pos}(b))$, and the edges are processed in increasing order of (i, j) and then of $\text{pos}(a)$, by a counting sort. A vertex is selected through at most one of its copies: an edge with an endpoint already selected adds nothing. An edge with $i = j$ selects its endpoint of larger degree in G, ties by position; with only two indices no residual graph is needed. An edge with $i < j$ selects an endpoint by a *rule*: `low` (index 0), `high` (index 1), or `deg_low` and `deg_high`, the endpoint of larger degree in G with ties broken toward index 0 or index 1. The decoded set is in general not a cover of G, since an edge whose copy is not matched is covered only if an endpoint was selected through another copy; it is completed by a bucket-queue greedy cover of the uncovered edges and pruned (Algorithm 4).

Maximal matching, greedy and Hallelujah: c_2, c_3 and c_4.. Candidate c_2, MaximalMatchingCover, takes both endpoints of M; this is the classical 2-approximation [2]. BucketDegreeGreedy (c_3) repeatedly takes a vertex of maximum current degree from a bucket queue; it is the greedy set cover algorithm on the stars of G. Hallelujah (c_4) is the degree-1 weighted reduction of [16]: every vertex u of degree d is replaced by d copies of weight $1/d$, one per incident edge, so that the auxiliary instance is a perfect matching; in Salvador the auxiliary edges are resolved selection-aware, scanning the vertices by ascending degree, adding nothing when an endpoint is already selected, selecting the lighter endpoint otherwise, and the neighbour of the scanned vertex on a tie.

Min-to-Min and primal-dual: c_5 and c_6.. MinToMinCover (c_5) repeatedly pops a vertex of minimum positive degree from a bucket queue and puts an arbitrary current neighbour of it into the cover; when that degree is 1 this is the exact leaf rule of every exact vertex cover solver. It processes exactly one minimum-degree vertex per step and selects the neighbour in $O(1)$, without rescanning a bucket, so the number of pushes and pops is $O(n + m)$. LinearWeightedCover

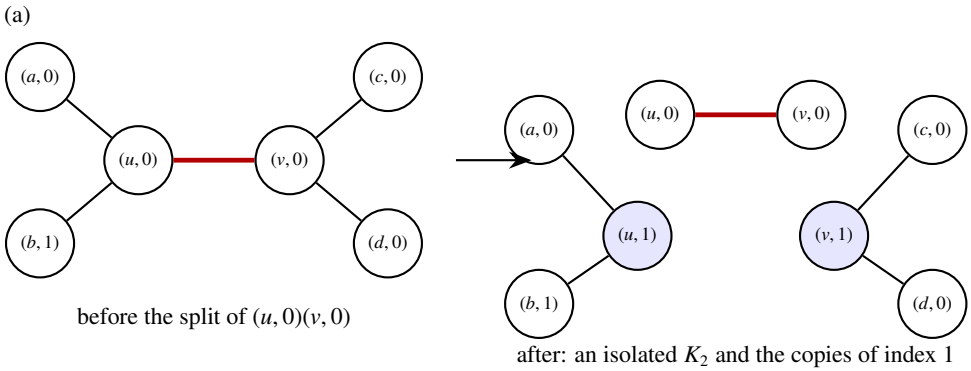


Figure 1: Strategy c_1. (a) Splitting $(u,0)(v,0)$ keeps it as an isolated K_2 and moves the other edges of u and v to their copies of index 1 (shaded); only edges whose endpoints both have index 0 are split. (b) The path $a\,b\,c\,d\,e$, with $\tau = 2$. The maximal matching $M = (ab, cd)$ fixes the target $|M| = 2$, and one split, of ab, reaches it. The matched edges $(a,0)(b,0)$ and $(d,0)(e,0)$ have equal indices and select their endpoint of larger degree in G, b and d; the matched edge $((b,1),(c,0))$ is then covered by b. The decoded set $\{b,d\}$ is a minimum cover.

Algorithm 4 ComponentSplitCover(G) — strategy c_1

Require: Graph G without self-loops or isolated vertices
Ensure: A minimal vertex cover c_1

$M \leftarrow$ `min_maximal_matching`(G); $(H, r') \leftarrow$ SplitGraph(G, M) ▷ Algorithm 3
$M_H \leftarrow$ `min_maximal_matching`(H), edges $((a,i),(b,j))$ ordered by $(i, j, \mathrm{pos}(a))$
for *rule* $\in$ {`deg_high`, `high`, `deg_low`} **do**
  $S \leftarrow \emptyset$
  for $((a,i),(b,j)) \in M_H$ **with** $a \notin S$ **and** $b \notin S$ **do**
    if $i = j$ **then** add the endpoint of larger degree in G to S **else** add $rule(a,i,b,j)$ to S
  end for
  $S \leftarrow S \cup$ BucketGreedy$(\{uv \in E : u \notin S,\ v \notin S\})$; $D_{rule} \leftarrow$ Prune(adj, S)
end for
return $c_1 \leftarrow$ the smallest D_{rule}

(c_6) is the primal-dual, local-ratio 2-approximation of Bar-Yehuda and Even [17]: it scans every edge once, raises a dual variable at each endpoint up to the remaining slack, and adds an endpoint to the cover when its budget saturates.

Weighted MIDS gadget on a forest core: c_7.. SolveVC (c_7) builds by union-find a spanning-forest core G_p of G and records the cycle-closing edges. Every vertex v of the core becomes a *choice node* $(v, 0)$ of weight 1 joined to a pendant node $(v, 1)$ of weight 0, and every core edge uv becomes a *hub* h_{uv} of weight $P = N + 1$, with N the number of core vertices, joined to $(u, 0)$ and $(v, 0)$ (Figure 2). A minimum-weight independent dominating set of this gadget never contains a hub, since a hub costs more than all choice nodes together, so it dominates every hub through one of its choice nodes, and the decoded set $\{v : (v, 0) \text{ selected}\}$ is a minimum cover of the core. The gadget is a subdivision of a forest with pendant leaves, hence a forest; it is solved by a Baker-style layered pass [18] that at the default parameter is its linear-time greedy baseline, which takes the weight-0, then the weight-1, then the heavy nodes into a maximal independent set. Uncovered core edges and the cycle-closing edges are finally repaired in one scan, each by its endpoint of larger degree.

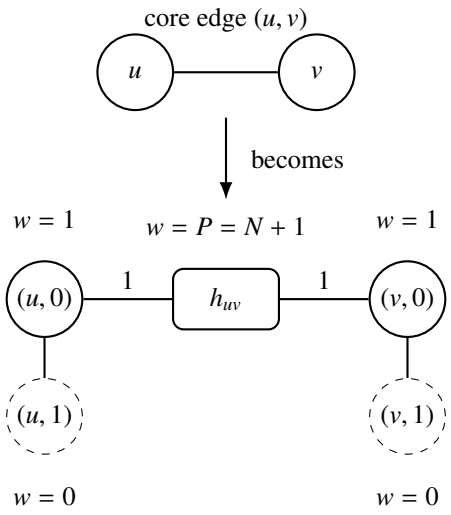


Figure 2: Local weighted-MIDS gadget for a core edge (u, v). Selecting $(u, 0)$ or $(v, 0)$ corresponds to paying for a vertex-cover endpoint, while selecting the heavy hub h_{uv} incurs penalty $P = N + 1$.

Oriented-incidence cycles: c_8.. BipartitePlanarCover (c_8) processes the vertices in a fixed order. When u is processed, each edge $\{u, v_j\}$ to a not-yet-processed neighbour, $j = 1, \ldots, d$, creates two *oriented-incidence* nodes x_{uv_j} and x_{v_ju}, of weights $1/\deg u$ and $1/\deg v_j$, joined by a *forcing edge*, and consecutive incidences are joined by *local edges* that close an even cycle C_{2d} when $d \geq 2$ (Figure 3). Every edge of G is handled once, from its first processed endpoint, so the auxiliary graph is a vertex-disjoint union of even cycles and single edges with $2m$ nodes: it is bipartite and planar, and a minimum-weight vertex cover of it is computed exactly by a dynamic program along each cycle, with no flow computation. A cover of the auxiliary graph contains an endpoint of every forcing edge and is decoded to $\{u : x_{uv} \text{ selected}\}$, a cover of G. The *residual variant* always processes a vertex of minimum residual degree d and weighs the gadget by residual degrees; its neighbour side then weighs $\sum_j 1/d_{\text{res}}(v_j) \leq 1$, so the exact cycle solve selects the neighbours, and the decoded cover is the complement of a greedy minimum-degree maximal independent set. Candidate c_8 keeps the better of the input-order and the residual variant.

Bipartite layers: c_9.. BipartiteLayersCover (c_9) treats every connected component on its own (Algorithm 5). On a bipartite graph it uses a near-optimal linear-time bipartite solver, BipartiteCover: the maximal matching of `min_maximal_matching`, enlarged by at most $p = 8$ phases of the Hopcroft–Karp algorithm [19], each of which finds a maximal set of vertex-disjoint shortest augmenting paths in $O(n + m)$, and Kőnig's construction [20]: with Z the vertices reachable from the unmatched vertices of one side T by alternating paths, the set $(T \setminus Z) \cup (Z \setminus T)$ is returned, the smaller of the two choices of T. A non-bipartite component is split into *bipartite layers*. A breadth-first 2-colouring

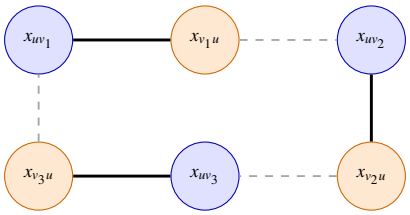


Figure 3: Oriented-incidence gadget of a vertex u with $d = 3$ unprocessed neighbours: an even cycle C_6. Solid forcing edges $\{x_{uv_j}, x_{v_ju}\}$ encode the original edges $\{u, v_j\}$; dashed local edges close the cycle. Blue nodes (weight $1/\deg u$) decode to u, orange nodes (weight $1/\deg v_j$) to v_j.

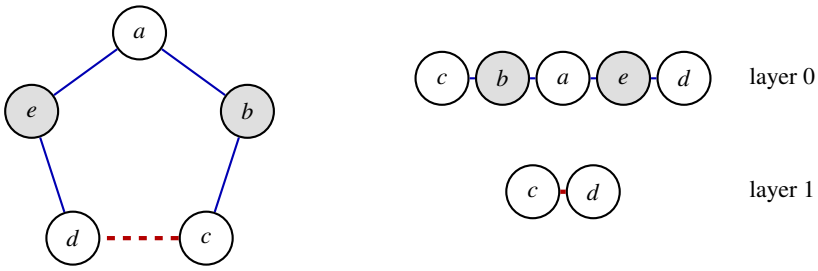
gives each vertex the colour that disagrees with most of its coloured neighbours, ties against its parent, so that at least half of the edges join different colours; these edges form layer 0, and their endpoints receive the copy index 0. The remaining edges, without the vertices they leave isolated, are coloured in the same way and form layer 1, with copy index 1, and so on until no edge is left (Figure 4). Every edge lies in exactly one layer. The layers are then solved in an order: in each layer the copies of the vertices selected in an earlier layer of the same order are discarded, together with their edges, which are covered, and the rest of the layer is solved by BipartiteCover. Different orders give different covers; all orders are tried when there are at most $q = 6$ of them, and otherwise q orders (the natural order, the reverse order and their rotations), and the smallest cover is kept.

Algorithm 5 BipartiteLayersCover(adj) — strategy c_9

Require: Adjacency map of a graph without self-loops; limits $p = 8$ (phases) and $q = 6$ (orders)
Ensure: A vertex cover C

$C \leftarrow \emptyset$
for each connected component K **do**
  $R \leftarrow E(K)$; $t \leftarrow 0$
  while $R \neq \emptyset$ **do**
    $\chi \leftarrow$ breadth-first majority 2-colouring of $(V(R), R)$
    $B_t \leftarrow \{uv \in R : \chi(u) \neq \chi(v)\}$ ▷ layer t, copies (u, t); $|B_t| \geq |R|/2$
    $R \leftarrow R \setminus B_t$; $t \leftarrow t + 1$
  end while
  $O \leftarrow$ all orders of $B_0, \ldots, B_{t-1}$ if $t! \leq q$, else q rotations of the natural and the reverse order
  for each order $(B_{\sigma(1)}, \ldots, B_{\sigma(t)}) \in O$ **do**
    $S \leftarrow \emptyset$
    for $s = 1, \ldots, t$ **do** $S \leftarrow S \cup$ BipartiteCover($\{uv \in B_{\sigma(s)} : u \notin S,\ v \notin S\}, p$)
    end for
  end for
  $C \leftarrow C \cup$ the smallest S
end for
return C

Labelled, weighted edge dominating set: c_{10}.. Candidate c_{10} reduces every bipartite component to an edge dominating set problem on a gadget D. Every edge $\{u, v\}$ of G becomes an *edge-node* (u, v), and a vertex u with neighbours $n_0, \ldots, n_{d-1}$ receives *incidence nodes* $(u, 0), \ldots, (u, d-2)$ that thread its edge-nodes into a path $(u, n_0) - (u, 0) - (u, n_1) - \cdots - (u, n_{d-1})$, whose edges are the *path edges* of u; a chosen path edge of u decodes to u (Figure 5). Every vertex carries a cover label shared along its path, choosing some but not all path edges of a vertex costs a prohibitive penalty, the path edges of every vertex weigh $K^\star$ in total, where $K^\star$ exceeds the number of edges, and every edge-node whose two endpoints are labelled 1 costs one unit of overlap. A penalty-free solution is then the set of path edges of a set C of vertices labelled 1; it is an edge dominating set exactly when C is a vertex cover, and it costs $K^\star|C| + e(C)$, with $e(C)$ the number of edges inside C. Its minimum is therefore a minimum vertex cover with the fewest doubly covered edges. The problem is solved by bucket elimination along a greedy minimum-degree order of D, eliminating a node only

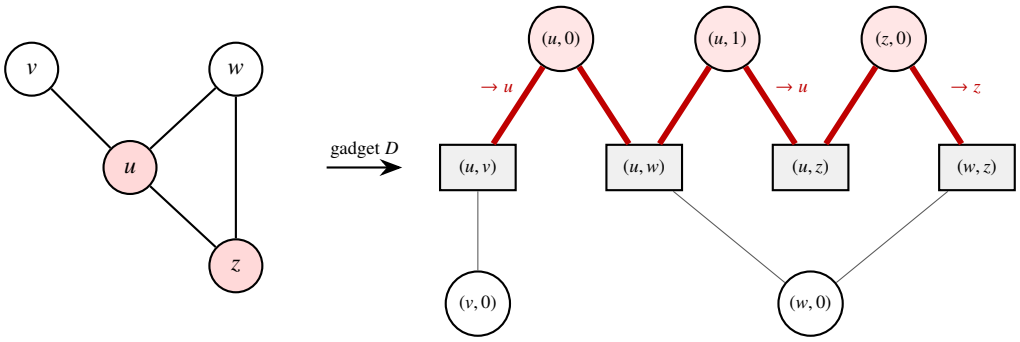


(a) majority colouring of C_5 from a: grey = 1, white = 0

(b) order (0, 1): layer 0 gives $\{b, e\}$, layer 1 adds c or d: $3 = \tau(C_5)$

Figure 4: Strategy c_9 on the odd cycle C_5. (a) The breadth-first majority colouring from a puts four of the five edges between the colours (solid): they form the bipartite layer 0, a path. The edge cd joins two vertices of colour 0 and forms layer 1 (dashed), with the copies $(c, 1)$ and $(d, 1)$. (b) Solving layer 0 exactly gives $\{b, e\}$; in layer 1 neither c nor d is selected yet, and one of them is added. The cover has $3 = \tau(C_5)$ vertices.

while its fill degree is at most 5, so that no bag has more than six nodes, and within a table budget of 1024 times the gadget size; the nodes that are not eliminated form a *core*, fixed as in the best of $c_1, \ldots, c_9$. With an empty core c_{10} is a minimum cover of the component. Minimum edge dominating set is NP-hard even on bipartite graphs of maximum degree 3 [21], so the core is what keeps c_{10} linear.



G, $\tau(G) = 2$

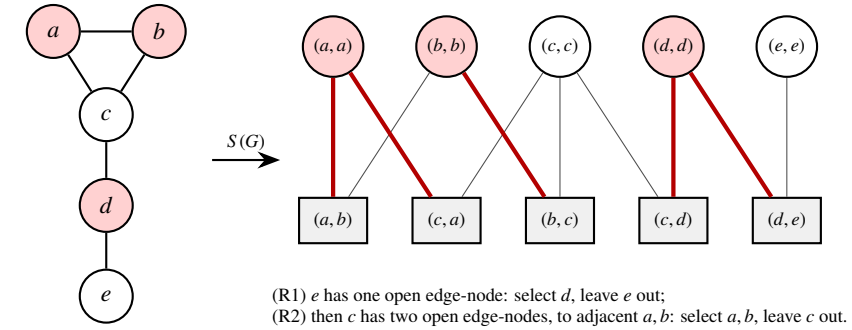
Figure 5: The edge-dominating-set gadget of c_{10}. Left: a graph G in which u has neighbours v, w, z (adjacency order) and w, z are adjacent. Right: its gadget D. Every edge of G is an edge-node (grey box); every vertex receives incidence nodes (circles) that thread its edge-nodes into a path, e.g. $(u, v) - (u, 0) - (u, w) - (u, 1) - (u, z)$ for u, while v, of degree 1, has the single node $(v, 0)$. The incidence index k in (u, k) is a position on the path of u, never a neighbour. The thick red edges are all the path edges of the vertices labelled 1, u and z (shaded): they form an edge dominating set of D, since every gadget edge shares an endpoint with one of them, and they decode to the cover $\{u, z\}$. Its weight is $2K^\star + 1$: one unit $K^\star$ per vertex, and an overlap of 1 because the edge uz has both endpoints in the cover. Every minimum cover of G ($\{u, z\}$ or $\{u, w\}$) has this weight, and no other labelling is lighter.

Weighted subdivision reduction: c_{11}.. The *subdivision graph* $S(G)$ has a *vertex-node* (u, u) for every vertex and an *edge-node* (u, w) for every edge, joined to (u, u) and (w, w); it is bipartite, with $n + m$ nodes and $2m$ edges. A set X of vertices *dominates* the edge-node (u, w) when $u \in X$ or $w \in X$; with vertex-nodes of weight 1 and an undominated edge-node costing $n + 1$, a minimum-weight selection dominates every edge-node and is a minimum vertex cover (Lemma 3.1). A vertex cover of $S(G)$ itself would not encode the problem: it may take the edge-node instead of either endpoint, so on a forest its minimum is the set of edge-nodes, which decodes to the empty set; c_{11} keeps the domination constraint, which carries the hardness, and uses the bipartite structure for bookkeeping. SubdivisionCover (c_{11}) keeps a residual graph of undecided vertices and undominated edges and applies four exact rules: (R0) a vertex of residual degree 0 is left out; (R1) the neighbour of a vertex of residual degree 1 is selected; (R2) the two neighbours of a vertex of residual degree 2 are selected when they are adjacent; and (R3) a vertex u of residual degree 2 with non-adjacent neighbours a, b is folded with them into one vertex z adjacent to $(N(a) \cup N(b)) \setminus \{u\}$, as in the folding rule of Chen,

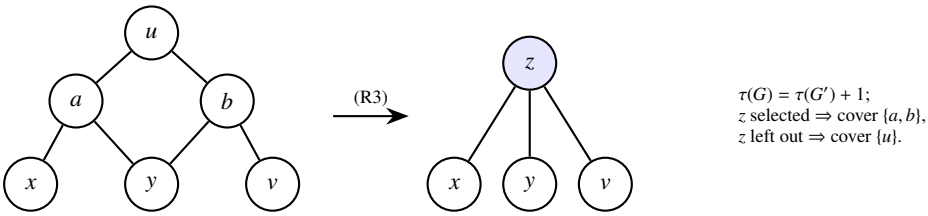
Kanj and Jia [22], and unfolded at the end into $\{a, b\}$ if z was selected and into u otherwise. Folding is charged to a work budget of $4(n + m)$. When no rule applies a vertex of largest residual degree is *peeled* into the cover, either among all vertices (`max`) or among the vertices of the best of $c_1, \ldots, c_{10}$ (`ref`), following the reducing-peeling scheme of Chang, Li and Zhang [23]. The ensemble keeps the smaller of the two variants, both with folding. Figure 6 shows the reduction on a small graph.

Lemma 3.1 (Domination encodes vertex cover)**.** *A set $X \subseteq V$ dominates every edge-node of $S(G)$ if and only if X is a vertex cover of G. Every minimiser of $\Phi(X) = |X| + (n + 1) \cdot |\{(u, w) : u \notin X,\ w \notin X\}|$ is a minimum vertex cover, and $\min_X \Phi(X) = \tau(G)$.*

Proof. The edge-node (u, w) is dominated exactly when X covers $\{u, w\}$, so a cover pays no penalty and $\Phi(X) = |X|$. A set that leaves an edge-node undominated has $\Phi(X) \geq n + 1 > n \geq \Phi(V)$, so a minimiser is a cover of minimum size. □



(a) G, $\tau(G) = 3$ (b) $S(G)$: the selection $\{a, b, d\}$ dominates every edge-node



(c) folding a vertex u of degree 2 with non-adjacent neighbours a, b

Figure 6: The weighted subdivision reduction of c_{11}. (a) A graph G with a triangle abc and a pendant path $c\,d\,e$. (b) Its subdivision $S(G)$: vertex-nodes (circles) keep the vertices, every edge becomes an edge-node (grey box) joined to its two endpoints, so $S(G)$ is bipartite. The reductions select d by (R1) and then a, b by (R2); the selected vertex-nodes (shaded) dominate every edge-node through the thick edges, and no peel was needed, so $\{a, b, d\}$ is a minimum vertex cover (Theorem 5.18). (c) The fold (R3): u, a, b become one vertex z adjacent to $N(a) \cup N(b) \setminus \{u\}$, and the optimum drops by exactly one.

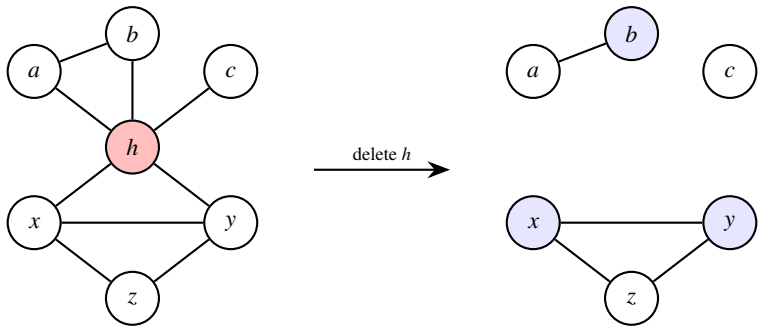
Bounded-treewidth dynamic program: c_{12}.. TreewidthDPCover (c_{12}) solves vertex cover exactly by bucket elimination along a greedy minimum-degree order, eliminating a vertex only while its degree in the fill graph is at most 10 and the total table size stays within $256(n + m)$; the remaining vertices form a core, fixed as in the best of $c_1, \ldots, c_{11}$.

3.3. Improvers

Union and local search: c_{13} and c_{14}.. Candidate c_{13} prunes the union of $c_2, \ldots, c_8$, the pruning pass of Hvala applied to all the independent heuristics at once. LocalSearchCover (c_{14}) improves the complementary independent set $S = V \setminus C$ of the best of $c_1, \ldots, c_{13}$. For $v \notin S$ let $\text{tight}(v) = |N(v) \cap S|$. A $(1, 2)$-swap of Andrade, Resende and Werneck [24] removes a vertex $x \in S$ and inserts two non-adjacent neighbours of x with tightness 1, enlarging S by one; an iterated

local search step forces a vertex into S, evicts its neighbours, reinserts freed vertices and is undone if S shrank. The cover found is then improved by the weighted swap search of Cai, Su, Luo and Sattar [25]: it keeps a set of fixed size k that may leave edges uncovered, gives every edge a weight, and at each step removes a vertex of the set and inserts an endpoint of a random uncovered edge, guided by the weights of the uncovered edges and by configuration checking; the weight of every uncovered edge grows by one at each step, and all weights are halved when their mean exceeds a threshold. Whenever no edge is uncovered the cover is recorded and k is decreased by one. The search is run in four segments with the thresholds 10, 100, 30 and 300, each restarted with fresh weights from the best cover found, and a last descent of $(1, 2)$-swaps follows. Every adjacency scan is charged to a work counter; the iterated local search uses the first half and the weighted swap search the second half of a limit of $\max\{b(n + m), 20{,}000\}$ work with $b = 100$.

Reduce-and-search and consensus: c_{15} and c_{16}.. The swap search of c_{14} stops on plateaus, and the reductions of c_{11} stop when every residual degree is at least 3. ReduceSearchCover (c_{15}) alternates the two for three rounds from the best of $c_1, \ldots, c_{14}$: a swap search with a different fixed seed in each round, which also accepts a cover of equal size and so moves along the plateau, and a pass of c_{11} guided by the searched cover, whose exact rules correct it wherever a vertex of residual degree at most 2 appears; a cover is accepted whenever it is not larger. ConsensusForceCover (c_{16}) turns the agreement of the earlier covers into a reduction. A vertex can be forced into every cover by a self-loop on it; c_{16} obtains the same effect by deleting the forced vertices and adding them to the cover. In at most three rounds it forces the vertices contained in all, then 90%, then 75% of the covers of its pool, which starts with $c_1, \ldots, c_{15}$ and receives the cover of every round, solves the residual graph again with the best of the pruned maximal-matching cover, c_5 and c_{11} (`max` with folding), and prunes the union on G (Figure 7). It stops early when its best cover equals a certified matching lower bound L, the larger of $|M|$ and a Karp–Sipser matching [26], or when a round would force the same set as the previous one.



(a) h lies in every cover of the pool: votes 13/13, forced in round 1

(b) residual graph $G - h$: an edge, a triangle and the isolated c; the residual solver returns $\{b, x, y\}$, which is optimal

(c) c_{16} = Prune($\{h\} \cup \{b, x, y\}$) = $\{h, b, x, y\}$, and $\tau(G) = 4$.

Figure 7: One round of consensus forcing (c_{16}). (a) The vertex h is in every cover of the pool, so it is forced: it is deleted and added to the cover, as a self-loop on h would require. (b) The residual graph splits into small components, on which the linear-time solvers are exact. (c) The union is pruned on G. Forcing helps when the strategies agree on the vertices of an optimal cover and disagree elsewhere: the residual graph then carries only the undecided part of the instance.

Swap-maximize refinement: c_{17}.. RefineCover (c_{17}) refines every candidate. For a cover C with complementary independent set I, and for every $u \in C \cup \{v_{\max}\}$, with $v_{\max}$ a vertex of maximum degree, it moves u into the independent set, evicts the neighbours of u from it, regrows a maximal independent set by scanning the cover vertices adjacent to an evicted vertex in ascending order of degree, and keeps the complement when it is strictly smaller than the best cover found so far. A move may evict many vertices at once, which the $(1, 2)$-swaps cannot do, and it succeeds when the low-degree vertices it frees outnumber the evicted ones. The moves stop after $100(n + m)$ work per candidate, sorting included.

4. Correctness of the Algorithm

Theorem 4.1. *Algorithm 1 always returns a vertex cover of the input graph G.*

Proof. Isolated vertices and self-loops need no cover, so it suffices that every candidate $c_1, \dots, c_{16}$ is a cover of the cleaned graph and that the refinement preserves this, since c_{17} is one of the refined candidates. Pruning preserves validity: a vertex is removed only when all its neighbours are in the set, so its edges stay covered. Candidate c_1 is completed by a greedy cover of the edges that its decoded set leaves uncovered, and then pruned. Candidate c_2 consists of both endpoints of a maximal matching, and an edge with no matched endpoint would extend the matching. Candidates c_3 and c_5 add vertices until no active vertex has positive degree, and every deletion removes only covered edges. In c_4 every original edge is an auxiliary edge, and when its first scanned endpoint is processed it is either already covered or one of its endpoints is selected. Candidate c_6 adds an endpoint of every edge whose slack is exhausted, which the local-ratio scan guarantees for every edge [17]. Candidate c_7 ends with a repair scan that adds an endpoint of every uncovered edge. In c_8 every edge of G is a forcing edge of the auxiliary graph, whose cover is exact, so an endpoint is decoded. Candidate c_9 is a cover by Proposition 5.21(i). Candidate c_{10} keeps only penalty-free dynamic-programming states, which decode to covers, and takes the earlier best cover on non-bipartite components. Candidate c_{11} stops only when the residual graph has no edge, an edge leaves it only when an endpoint is selected, and unfolding a fold vertex z adds $\{a, b\}$ if z was selected and u otherwise, which covers every edge at u, a and b. In c_{12} the dynamic program covers every edge between eliminated vertices and every edge to the core, and the core is fixed as in a cover. A union of covers is a cover, which handles c_{13}. Candidates c_{14}, the search step of c_{15} and every refinement are complements of independent sets, hence covers, and a reduction pass of c_{15} is accepted only after it is checked to be a cover. In c_{16} a round returns $S \cup C'$ with C' a cover of $G - S$, so every edge with an endpoint in S is covered by S and every other edge by C'. □

5. Approximation Ratio

5.1. Ratio 2 and the bound $2\tau - r$

Theorem 5.1 (Every pruned candidate is minimal)**.** *Every output of Algorithm 2 is a minimal vertex cover. In particular $c_1, \dots, c_{13}$ are minimal covers.*

Proof. A vertex v that survives its test has a neighbour outside the current set, and later removals only enlarge the outside, so after the pass every remaining vertex has a neighbour outside the cover; that is, no vertex can be removed. □

Theorem 5.2 (Unconditional ratio 2)**.** *For every graph G, Algorithm 1 returns a cover C with $|C| \le 2\tau(G)$.*

Proof. Before pruning, c_2 consists of both endpoints of a maximal matching M; any cover contains an endpoint of each of the $|M|$ disjoint edges of M, so $\tau(G) \ge |M|$ and $|c_2| \le 2|M| \le 2\tau(G)$ [2]. The returned cover is the smallest refined candidate, and a refinement never enlarges a cover, so $|C| \le |c_2|$. □

Theorem 5.3 ($2\tau - r$ on every graph)**.** *Let $r(G)$ be the number of connected components of G that have an edge. The pruned candidate c_2, for every maximal matching, and therefore the cover C returned by Algorithm 1, satisfy*

$$|C| \;\le\; |c_2| \;\le\; 2\,\tau(G) - r(G), \qquad \textit{so} \qquad \frac{|C|}{\tau(G)} \;\le\; 2 - \frac{r(G)}{\tau(G)}.$$

Proof. Whether pruning removes a vertex depends only on its neighbours, and the restriction of a maximal matching to a component is a maximal matching of it, so it suffices to prove $|c_2| \le 2\tau(G) - 1$ for a connected graph G with an edge and to sum over the components. Let M be the maximal matching and $V(M)$ its matched vertices. Always $|c_2| \le |V(M)| = 2|M| \le 2\nu(G) \le 2\tau(G)$; suppose for a contradiction that $|c_2| = 2\tau(G)$. Then equality holds throughout: pruning removed nothing, so $c_2 = V(M)$ is a minimal cover by Theorem 5.1, and $|M| = \nu(G) = \tau(G)$, so M is a maximum matching. Take an edge $uv \in M$. By minimality u has a neighbour $a \notin V(M)$ and v has a neighbour $b \notin V(M)$. If $a \ne b$, then $a\,u\,v\,b$ is an M-augmenting path, which contradicts the maximality of $|M|$. Hence u and v have a single common neighbour $a_{uv} \notin V(M)$, and u, v, a_{uv} form a triangle. Let $A = \{a_{uv} : uv \in M\}$ and, for $a \in A$, let M_a be the edges uv of M with $a_{uv} = a$. A cover K of G contains at least $|M_a| + 1$ vertices of $\{a\} \cup V(M_a)$: if $a \in K$,

then a and an endpoint of each edge of M_a; if $a \notin K$, both endpoints of every edge of M_a, since they are adjacent to a. These sets are disjoint for different a and the sets M_a partition M, so $\tau(G) \geq |M| + |A| \geq |M| + 1 > \tau(G)$, a contradiction. Hence $|c_2| \leq 2\tau(G) - 1$, and $|C| \leq |c_2|$ as in Theorem 5.2. □

Corollary 5.4 (Ratio 1.999999999 below 10^9)**.** *If $\tau(G) \leq 10^9\, r(G)$, then the cover C returned by Algorithm 1 satisfies $|C| \leq 1.999999999\, \tau(G)$. In particular this holds for every graph with at most $10^9 + 1$ vertices.*

Proof. $2 - r/\tau \leq 2 - 10^{-9}$ is equivalent to $\tau \leq 10^9 r$. A component with $n_i \geq 2$ vertices has $\tau_i \leq n_i - 1$, so if $n \leq 10^9 + 1$ then $\tau \leq 10^9 \leq 10^9 r$. □

Theorem 5.3 is elementary, and the same bound holds for any algorithm that returns a pruned maximal-matching cover. It is not a $(2 - \varepsilon)$-approximation: along any sequence of connected graphs with $\tau \to \infty$ the margin $1/\tau$ tends to 0, as for the classical $2 - \Theta(1/\sqrt{\log n})$ guarantees [3], so it is consistent with every hardness result. It does settle a question that experiments cannot: every graph in any experiment we know of has far fewer than 10^9 vertices, so on such graphs a ratio of at most 1.999999999 is a theorem, not an empirical finding.

5.2. Per-class guarantees of the strategies

Because Salvador returns the smallest refined candidate, a guarantee proved for one candidate on a class of graphs holds for the returned cover on that class. Throughout, $R_0 = 1.999999999 = 2 - \varepsilon_0$ with $\varepsilon_0 = 10^{-9}$, $\alpha(G)$ is the independence number, $\bar{d}$ the average degree, Δ the maximum degree and $W(G) = \sum_{uv \in E} 1/\max\{\deg u, \deg v\}$.

Lemma 5.5 (Safe vertices)**.** *For every graph G and vertex w, $\tau(G - w) \geq \tau(G) - 1$, with equality if and only if w belongs to some minimum vertex cover. If v has degree 1 and w is its neighbour, then w belongs to some minimum vertex cover.*

Proof. A cover of $G - w$ plus w covers G; a minimum cover of G containing w, minus w, covers $G - w$; and conversely a minimum cover of $G - w$ plus w is a minimum cover of G when $\tau(G - w) = \tau(G) - 1$. A minimum cover contains v or w, and replacing v by w keeps it a cover, since vw is the only edge at v. □

Lemma 5.6 (Exactness criterion for Min-to-Min)**.** *Every vertex added by c_5 is a neighbour of a vertex of minimum positive degree in the current active graph, and is then removed from it. Consequently, if every added vertex belongs to some minimum vertex cover of the current active graph, then $|c_5| = \tau(G)$.*

Proof. Every vertex of positive current degree k has an entry in the bucket B_k, since an entry is pushed whenever a degree changes, and stale entries are skipped. Removing a vertex lowers the degree of each neighbour by one, so all positive degrees stay at least the current minimum minus one, and the bucket pointer, decreased by at most one per step and advanced to the first non-empty bucket, always indexes the minimum positive degree. Let $w_1, \ldots, w_k$ be the added vertices, $G_0 = G$ and $G_i = G_{i-1} - w_i$; the last graph has no edge. If each w_i lies in some minimum cover of G_{i-1}, Lemma 5.5 gives $\tau(G_i) = \tau(G_{i-1}) - 1$, so $|c_5| = k = \tau(G)$. □

Theorem 5.7 (Classes solved exactly by c_5)**.** *If every component of G is a tree, a path, a cycle, a complete graph or a complete bipartite graph, then $|c_5| = \tau(G)$, and* Salvador *returns a minimum vertex cover.*

Proof. It suffices that each added vertex is safe in the component K of the popped vertex v, which has minimum degree in K, and that deleting it leaves components of the listed types. In a tree or a path v is a leaf, so its neighbour is safe by Lemma 5.5, and deleting a vertex leaves trees or paths. A cycle C_k, a complete graph K_n and a balanced $K_{a,a}$ are vertex-transitive, so every vertex lies in some minimum cover, and $C_k - w = P_{k-1}$, $K_n - w = K_{n-1}$, $K_{a,a} - w = K_{a-1,a}$. In $K_{a,b}$ with $a < b$ the vertices of minimum degree are on the side of size b, so the added vertex lies on the side of size a, a minimum cover by Kőnig's theorem [20], and $K_{a,b} - w = K_{a-1,b}$. Lemma 5.6 concludes. □

Lemma 5.8 (Caro–Wei bound for the residual candidate)**.** *For every graph G without isolated vertices, the residual variant of c_8 returns, before pruning, a cover C_b with $|C_b| \leq n - \sum_{v \in V} \frac{1}{\deg(v)+1}$. Consequently $|C| \leq n - \lceil \sum_v 1/(\deg v + 1) \rceil$.*

Proof. Let u be processed with minimum residual degree $k \geq 1$ and residual neighbours $v_1, \ldots, v_k$. Its gadget is the even cycle $a_1 b_1 \cdots a_k b_k$, with $a_j = x_{uv_j}$ of weight $1/k$ and $b_j = x_{v_j u}$ of weight $w_j = 1/d_{\mathrm{res}}(v_j) \leq 1/k$. The forcing edges $a_j b_j$ are a perfect matching of the cycle, so every cover of it weighs at least $\sum_j \min\{1/k, w_j\} = \sum_j w_j$, the weight of the neighbour side; with ties broken toward the neighbours the exact solve returns the neighbour side. So every step puts $v_1, \ldots, v_k$ into C_b, u into the independent set $S = V \setminus C_b$, and deletes $u, v_1, \ldots, v_k$. Let $\Phi = \sum_x 1/(d_{\mathrm{res}}(x) + 1)$ over the vertices not yet deleted. The $k + 1$ deleted vertices have residual degree at least k, so they contribute at most 1, and the terms of the surviving vertices can only grow; hence each step lowers Φ by at most 1. When no residual edge remains the r surviving vertices contribute 1 each and belong to S, so after t steps $|S| = t + r \geq \Phi_0 = \sum_v 1/(\deg v + 1)$, the greedy form of the Caro–Wei bound [27, 28]. Pruning and the ensemble only shrink the cover, and cover sizes are integers. □

Theorem 5.9 (Regular graphs)**.** *If G is d-regular with $d \geq 1$, then $|C| \leq \frac{2d}{d+1}\,\tau(G) = (2 - \frac{2}{d+1})\tau(G)$. In particular cubic graphs get ratio at most $3/2$, and every d-regular graph with $d \leq 1{,}999{,}999{,}999$ gets ratio at most 1.999999999.*

Proof. By Lemma 5.8, $|C| \leq n - n/(d + 1) = nd/(d + 1)$, and $\tau(G) \geq m/d = n/2$ since each vertex covers d of the $m = nd/2$ edges. The last statement is $2 - 2/(d + 1) \leq 2 - 10^{-9}$. □

Lemma 5.8 is a per-instance certificate on every graph, computable from the degree sequence and a lower bound on τ; it is weak on very irregular graphs, where other candidates take over, and it gives the uniform bound of Theorem 5.9 on regular graphs. Part E of Section 7 finds that it is the best a-priori certificate on all of its instances.

Corollary 5.10 (Degree-heterogeneous graphs)**.** *For every graph without isolated vertices,*

$$\frac{|C|}{\tau(G)} \;\leq\; \frac{2\Delta}{\bar{d} + 1}.$$

In particular the ratio is at most R_0 whenever $\Delta \leq R_0(\bar{d} + 1)/2$.

Proof. The function $x \mapsto 1/(x + 1)$ is convex, so $\sum_v 1/(\deg v + 1) \geq n/(\bar{d} + 1)$ by Jensen's inequality, and Lemma 5.8 gives $|C| \leq n\bar{d}/(\bar{d} + 1)$. Every vertex covers at most Δ edges, so $\tau \geq m/\Delta = n\bar{d}/(2\Delta)$. □

On regular bipartite graphs c_5 and c_8 do much better than Theorem 5.9: they are exact, whatever the degree, because the minimum-degree rule never leaves the side on which it starts.

Theorem 5.11 (Degree-ordered bipartite graphs)**.** *Let G be a graph without isolated vertices each of whose components K is bipartite with a bipartition (L_K, R_K) such that either (a) K is d_K-regular for some $d_K \geq 1$, or (b) K is* degree-separated*: $\deg r > \deg l$ for every $r \in R_K$ and $l \in L_K$. Then $|c_5| = |c_8| = \tau(G)$, and* Salvador *returns a minimum vertex cover.*

Proof. The optimum. In case (a) both sides have $|V(K)|/2$ vertices and $\tau(K) \geq |E(K)|/d_K = |V(K)|/2$, so each side is a minimum cover. In case (b) let $a = \max_{L_K} \deg$ and $b = \min_{R_K} \deg > a$; for $S \subseteq R_K$ the edges between S and $N(S)$ give $b|S| \leq a|N(S)|$, so $|N(S)| > |S|$ for $S \neq \emptyset$, Hall's theorem [29] gives a matching saturating R_K, and R_K is a minimum cover. So it suffices that each candidate contains at most one side of each component, the side R_K in case (b).

Candidate c_5. In case (a) call L_K the side of the first vertex of K that c_5 pops. We show that every vertex popped in K lies in L_K, so that every added vertex, a neighbour of it, lies in R_K. While this holds, no vertex of L_K is removed, so every remaining vertex of R_K keeps its original degree, while vertices of L_K can only lose degree. A popped vertex has minimum positive degree δ (Lemma 5.6). Suppose a remaining $r \in R_K$ were popped. In case (b) any neighbour l of r has positive degree at most $\deg l < \deg r = \delta$, a contradiction. In case (a) $\delta = d_K$, so every vertex of L_K has degree 0 or d_K; a vertex of L_K adjacent to r has degree d_K and hence all its neighbours remain, and every remaining vertex of R_K keeps all its neighbours, so alternating between the sides from r reaches all of K through remaining vertices and no vertex of R_K has been removed. But the first vertex popped in K lies in L_K and its added neighbour, in R_K, was removed. Hence $c_5 \cap V(K) \subseteq R_K$ and $|c_5| \leq \tau(G)$.

Candidate c_8. By the proof of Lemma 5.8, the residual variant processes a vertex u of minimum positive residual degree, puts all its residual neighbours into the cover and deletes u with them. With L_K defined by the first processed vertex in case (a), the argument for c_5, with residual degrees, shows that every processed vertex of K lies in L_K; so every vertex put into the cover lies in R_K, and every vertex of R_K is eventually put there. Pruning keeps a minimum cover. □

Case (a) contains every regular bipartite graph: even cycles, hypercubes, crown graphs, balanced complete bipartite graphs, bipartite double covers $H \times K_2$ of regular graphs, and random regular bipartite graphs, which are expanders of linear treewidth. Case (b) contains every unbalanced complete bipartite graph. Neither case needs a bound on the degree or on the treewidth, and neither depends on how ties are broken.

Theorem 5.12 (Matching certificate)**.** *For every graph with an edge, $|C| \le 2\nu(G)$, so $|C|/\tau(G) \le 2\nu(G)/\tau(G)$, and the ratio is at most R_0 whenever $\tau(G) \ge 2\nu(G)/R_0$.*

Proof. $|C| \le |c_2| \le 2|M| \le 2\nu(G)$. □

Since $\tau \le 2\nu$ always holds, Theorem 5.12 leaves out exactly the *nearly Kőnig–Egerváry* graphs, with $\nu \le \tau < 2\nu/R_0$. On every other graph the matching candidate alone beats R_0. Every candidate is a minimal cover (Theorem 5.1), which gives a guarantee that does not depend on the strategy.

Corollary 5.13 (Low independence ratio)**.** *If $\alpha(G) \le (1 - 1/R_0)\, n$, then every minimal cover, and in particular every candidate, has ratio at most R_0. More precisely: (a) if V can be partitioned into cliques of size at least $k \ge 3$, every minimal cover has ratio at most $k/(k-1) \le 3/2$; (b) if G is d-regular with smallest adjacency eigenvalue $\lambda_{\min} \ge -(1 - \varepsilon_0)d$, every minimal cover has ratio at most $1 + |\lambda_{\min}|/d \le R_0$, for every degree d.*

Proof. A minimal cover has at most n vertices and $\tau = n - \alpha \ge n/R_0$. For (a), α is at most the number of cliques, at most n/k. For (b), Hoffman's ratio bound [30] gives $\alpha \le n|\lambda_{\min}|/(d + |\lambda_{\min}|)$, so $n/\tau \le 1 + |\lambda_{\min}|/d$. □

Theorem 5.14 (Max-degree greedy: c_3)**.** *For every graph of maximum degree Δ, $|C| \le |c_3| \le H(\Delta)\,\tau(G)$, where $H(k) = \sum_{j=1}^{k} 1/j$. In particular every graph of maximum degree at most 3 gets ratio at most $H(3) = 11/6$.*

Proof. Vertex cover is the set cover instance whose elements are the edges and whose sets are the stars of the vertices, of size at most Δ. The bucket queue processes degrees from the highest down and degrees only decrease, so every selected vertex has maximum current degree, and c_3 is the greedy set cover algorithm, whose ratio is at most $H(\Delta)$ by Johnson's theorem [31]. Pruning only shrinks c_3. □

Theorem 5.14 covers irregular and bipartite subcubic graphs, a class on which vertex cover is APX-hard [32], which excludes an approximation scheme but not a fixed ratio such as $11/6$.

Theorem 5.15 (Hallelujah bound: c_4)**.** *Let S be the set produced by the weighted pass of the selection-aware Hallelujah resolution. For every graph with an edge,*

$$|c_4|^2 \;\le\; |S|^2 \;\le\; W(G) \sum_{v \in S} \deg v \;\le\; 2m\, W(G) \;\le\; 2m\, \tau(G).$$

Consequently the ratio is at most R_0 on every graph with $2mW(G) \le R_0^2 \tau(G)^2$.

Proof. Candidate c_4 is the smaller of S and an unweighted pass, so $|c_4| \le |S|$. Each vertex enters S once, while an edge $\{u, w\}$ with no selected endpoint is inspected, and it is then an endpoint of minimum weight $1/\max\{\deg u, \deg w\}$ of that edge; the edge is covered afterwards, so distinct vertices of S are charged to distinct edges and $\sum_{v \in S} 1/\deg v \le W(G)$. This is the minimum weight of a cover of the auxiliary perfect matching, which is at most $\tau(G)$, because the $\deg v$ copies of each vertex of an optimal cover weigh 1 in total. By the Cauchy–Schwarz inequality, $|S|^2 \le (\sum_{v \in S} 1/\deg v)(\sum_{v \in S} \deg v) \le W(G) \cdot 2m$. □

Proposition 5.16 (Selection-aware Hallelujah on regular components)**.** *If every component G_j of G is d_j-regular with n_j vertices, then before pruning c_4 is the complement of a maximal independent set, and $|c_4| \le \sum_j n_j d_j/(d_j + 1) \le (2 - 2/(\max_j d_j + 1))\,\tau(G)$.*

Proof. Adjacent vertices have equal degree, so every weight comparison ties and the tie rule selects the still-unselected neighbours of each scanned unselected vertex u. Let I be the vertices unselected when scanned: no vertex of I is ever selected, since a later neighbour of it is already selected and is skipped, so I is independent, and every selected vertex has a neighbour in I, so I is maximal. In G_j each vertex of I dominates at most $d_j + 1$ vertices, so $|I \cap V(G_j)| \ge n_j/(d_j + 1)$; and $\tau(G_j) \ge n_j/2$. □

Theorem 5.15 is strong when degrees are heterogeneous, and Proposition 5.16 when they are equal; the published Hallelujah rule [16], which resolves ties by labels, reaches $2 - 1/k$ on the even cycle C_{2k}, while the selection-aware rule returns k vertices there.

Proposition 5.17 (Edge-dominating-set gadget: c_{10})**.** *For every edge dominating set F of the gadget D, the set of vertices that F decodes to is a vertex cover; for every vertex cover C, the path edges of the vertices of C form an edge dominating set that decodes to C. On a bipartite component K the minimum cost of a labelled edge dominating set is $K^\star\tau(K) + \min\{e(C) : C \text{ a minimum cover of } K\}$, so c_{10} is a minimum cover of K, with the fewest doubly covered edges, whenever its core is empty, and no bag of its dynamic program has more than six nodes.*

Proof. Let $\{u, v\} \in E$ and let (u, k) be an incidence node joined to the edge-node (u, v). The gadget edge between them is dominated by a chosen edge at (u, k) or at (u, v), and every gadget edge there lies on the path of u or of v and decodes to u or v. Conversely, a gadget edge $\{(u, k), (u, x)\}$ is chosen if $u \in C$; otherwise $x \in C$ and a path edge of x at (u, x) dominates it. A solution that pays a penalty costs more than the penalty-free solution of all vertices, so every optimum is penalty-free, chooses the path edges of the set C labelled 1, is feasible exactly when C is a cover, and costs $K^\star|C| + e(C)$ with $e(C) \le |E(K)| < K^\star$. With an empty core the bucket elimination over the penalty-free states is exact, and nodes are eliminated only while their fill degree is at most 5. □

Theorem 5.18 (Subdivision reduction: c_{11})**.** *Let p be the number of peels made by one variant of c_{11} and S the cover it returns. Then each rule (R0)–(R3) is exact, $|S| \le \tau(G) + p$, and S is a minimum vertex cover when no peel is needed.*

Proof. (R0) changes nothing. For (R1), the neighbour of a vertex of residual degree 1 is safe by Lemma 5.5. For (R2), every cover contains two vertices of the triangle u, a, b, and since u has no other residual edge some minimum cover contains a and b. For (R3), the folding rule of Chen, Kanj and Jia [22] satisfies $\tau(G_t) = \tau(G') + 1$, and unfolding adds exactly one vertex. So the quantity $\Psi_t = |S_t| + F_t + \tau(G_t)$, with S_t the selected vertices, F_t the folds and G_t the residual graph, is unchanged by every rule, while a peel raises it by $1 + \tau(G_t - x) - \tau(G_t) \in \{0, 1\}$ by Lemma 5.5. At the start $\Psi_0 = \tau(G)$, and at the end the residual graph has no edge and unfolding adds one vertex per fold, so $|S| = \Psi_T \le \tau(G) + p$. □

Proposition 5.19 (The improvers)**.** *$c_{13}, \dots, c_{16}$ and every refinement are valid covers; $|c_{14}| \le \min_{i\le 13} |c_i|$, $|c_{15}| \le \min_{i\le 14} |c_i|$, $|c_{16}| \le \min_{i\le 15} |c_i|$, and the refinement r_i of every candidate satisfies $|r_i| \le |c_i|$, so $|C| \le \min_{1\le i\le 16} |c_i|$. Every reduction pass of c_{15} returns a cover with at most $\tau(G) + p$ vertices, p its number of peels. If every vertex forced in a round of c_{16} lies in a common minimum cover $C^* \supseteq S$, the cover of that round has at most $|S| + 2\tau(G - S) - r(G - S) \le 2\tau(G) - |S| - r(G - S)$ vertices.*

Proof. A union of covers is a cover, and pruning keeps it one. c_{14} and c_{15} accept only covers that are not larger than their start, and a reduction pass of c_{15} is the procedure of c_{11} with the guided peel, so Theorem 5.18 applies. c_{16} returns the smallest cover of a pool that starts with $c_1, \dots, c_{15}$. A refinement move returns the complement of an independent set and is kept only when strictly smaller. If $S \subseteq C^*$, then $\tau(G - S) = \tau(G) - |S|$, and the residual solver, no larger than the pruned maximal-matching cover, returns at most $2\tau(G - S) - r(G - S)$ vertices by Theorem 5.3. □

Theorem 5.20 (Bounded-treewidth dynamic program: c_{12})**.** *If the elimination of c_{12} leaves an empty core, then c_{12}, and therefore C, is a minimum vertex cover. This holds in particular on every forest and every cycle.*

Proof. The elimination order and the bags $\{v\} \cup N_F(v)$ form a tree decomposition of G, and bucket elimination along it computes a maximum independent set exactly when no vertex is fixed; its complement is a minimum cover. A forest or a cycle is eliminated with bags of at most three vertices. □

Proposition 5.21 (Bipartite layers: c_9)**.** *Let K be a connected component of G with $m_K \ge 1$ edges.*

(i) *c_9 is a vertex cover of G, for every matching produced by the phases.*

(ii) *K is split into at most $\lfloor \log_2 m_K \rfloor + 1$ bipartite layers, and every edge of K lies in exactly one of them.*

(iii) *If K is bipartite, it forms a single layer. If moreover the last phase of* BipartiteCover *finds no augmenting path, then $c_9 \cap V(K)$ is a minimum vertex cover of K; in particular $|c_9| = \tau(G)$ when every component of G is bipartite and every solve ends in this way.*

(iv) On a bipartite graph the matching of BipartiteCover *after p phases has at least $\frac{p+1}{p+2}\nu$ edges, at least $\frac{9}{10}\nu$ for $p = 8$.*

Proof. (i) Let N be any matching of a bipartite graph with sides T and B, and Z the vertices reachable from the unmatched vertices of T by alternating paths, which leave T along non-matching edges and enter T along matching edges. Let xy be an edge with $x \in T$. If $x \notin Z$, then $x \in T \setminus Z$ is selected. If $x \in Z$ and $xy \notin N$, the path to x extends to y, so $y \in Z \setminus T$ is selected. If $x \in Z$ and $xy \in N$, then x is matched and was entered from its partner, so $y \in Z$. Hence $(T \setminus Z) \cup (Z \setminus T)$ is a cover of the layer. In an order of the layers, an edge of a layer either has an endpoint selected in an earlier layer or remains in the sub-layer that is solved, so every edge of K is covered.

(ii) Every vertex other than the start of its breadth-first tree is coloured after its parent, and it takes the colour that disagrees with at least half of its neighbours coloured so far. Charging each edge to its later-coloured endpoint, at least half of the edges join different colours, so layer t takes at least half of the m_t edges left and $m_{t+1} \le m_t/2$. Since $m_0 = m_K$, after $\lfloor \log_2 m_K \rfloor + 1$ layers fewer than one edge is left. The edges of a layer join different colours, so each layer is bipartite, and every edge leaves the set of remaining edges exactly once.

(iii) Let K have sides $X \ni s$ and Y, s the start of the colouring. By induction every coloured vertex of X has colour 0 and every coloured vertex of Y colour 1: the coloured neighbours of a new vertex all lie on the other side and share one colour, so the majority rule gives the colour of its side. So every edge of K joins different colours and K is one layer. When no augmenting path is left the matching is maximum and $Z \cap B$ contains no unmatched vertex, so the cover has one endpoint of every matched edge, $\nu(K)$ vertices, which is $\tau(K)$ by Kőnig's theorem [20].

(iv) A maximal matching has no augmenting path of length 1, and every Hopcroft–Karp phase raises the length of a shortest augmenting path by at least 2 [19], so after p phases every augmenting path has length at least $2(p+1)+1$. Then the symmetric difference with a maximum matching contains $\nu - |N|$ vertex-disjoint augmenting paths, each with at least $p+1$ edges of N, so $|N| \ge (p+1)(\nu - |N|)$ and $|N| \ge \frac{p+1}{p+2}\nu$. □

On a non-bipartite component the layers give no ratio of their own: a vertex can be selected in every layer through a different copy, and the orders only choose among the covers obtained. The guarantee of the ensemble on such graphs comes from the other strategies, and c_9 contributes when one of its orders happens to cover the odd cycles cheaply.

5.3. A combined certificate and the residue

Theorem 5.22 (Combined certificate)**.** *For every graph G with an edge,*

$$\frac{|C|}{\tau(G)} \;\le\; \beta(G) \;:=\; \min\Big\{2 - \frac{r}{\tau},\; \frac{2\nu}{\tau},\; \frac{n - \sum_v \frac{1}{\deg v + 1}}{\tau},\; \frac{\sqrt{2mW(G)}}{\tau},\; \beta_{\mathrm{cls}}(G)\Big\},$$

where $\beta_{\mathrm{cls}}(G) = 1$ if every component is a tree, path, cycle, complete graph, complete bipartite graph, regular bipartite graph or degree-separated bipartite graph, or if the core of c_{12} is empty, or if a variant of c_{11} needs no peel, or if every component is bipartite and the Hopcroft–Karp phases of c_9 end with a maximum matching; $\beta_{\mathrm{cls}}(G) = 11/6$ if $\Delta \le 3$; and $\beta_{\mathrm{cls}}(G) = 2$ otherwise.

Proof. $|C| \le |c_i|$ for every i (Proposition 5.19). The terms are Theorem 5.3 for c_2, Theorem 5.12, Lemma 5.8 for c_8, Theorem 5.15, Theorems 5.7 and 5.11 for c_5 and c_8, Theorem 5.14, Theorem 5.18, Theorem 5.20 and Proposition 5.21. □

The certificate $\beta(G)$ is computable from the degree sequence, a maximum matching and a lower bound on τ. Let $\mathcal{R} = \{G : \beta(G) > R_0\}$ be the *residue*. Since $\beta \le 2 - r/\tau$, every graph of $\mathcal{R}$ has $\tau > 10^9 r$ and hence more than 10^9 vertices; since $\beta \le 2\nu/\tau$, every graph of $\mathcal{R}$ is nearly Kőnig–Egerváry. On $\mathcal{R}$ the right tool is a certificate that reads the returned cover.

Lemma 5.23 (Slack identity)**.** *Let $y : E \to [0,1]$ be a fractional matching, $|y| = \sum_e y_e$, $\mathrm{def}_y(v) = 1 - \sum_{e \ni v} y_e$, and for a set C let $E(C)$ be the edges with both endpoints in C. Every vertex cover C satisfies*

$$|C| \;=\; |y| + \sigma_y(C), \qquad \sigma_y(C) := \sum_{e \in E(C)} y_e + \sum_{v \in C} \mathrm{def}_y(v) \;\ge\; 0,$$

so the excess of C over a minimum cover C^* is $|C| - \tau(G) = \sigma_y(C) - \sigma_y(C^*)$.

Proof. Summing $1 = \sum_{e \ni v} y_e + \mathrm{def}_y(v)$ over $v \in C$ gives $|C| = \sum_e y_e |e \cap C| + \sum_{v \in C} \mathrm{def}_y(v)$, and $|e \cap C|$ is 1 or 2, the latter exactly on $E(C)$, because C is a cover. □

Theorem 5.24 (A-posteriori certificate)**.** *Let $\nu_f(G)$ be the optimum of the linear relaxation and y a maximum fractional matching. For every graph and every cover C,*

$$\frac{|C|}{\tau(G)} \;\le\; \frac{|C|}{\nu_f(G)} \;=\; 1 + \frac{\sigma_y(C)}{\nu_f(G)} \;\le\; \frac{|C|}{\nu(G)},$$

and the three quantities on the right are computable in polynomial time. If $\tau(G) < 2\nu(G)/R_0$, in particular on $\mathcal{R}$, then $|C|/\nu(G) < (1 + 5.1 \cdot 10^{-10})\,|C|/\tau(G)$; and if $\tau(G) = \nu(G)$, as on every bipartite graph, then $|C| - \tau(G) = \sigma_M(C)$ for every maximum matching M.

Proof. Weak duality gives $\nu \le \nu_f \le \tau$, and Lemma 5.23 with $|y| = \nu_f$ gives the equality; ν_f is half a maximum matching of the bipartite double cover. If $\tau < 2\nu/R_0$, then $|C|/\nu < (2/R_0)|C|/\tau$ and $2/R_0 < 1 + 5.1 \cdot 10^{-10}$. If $\tau = \nu$, a maximum matching M is an optimal y, so $\sigma_M(C^*) = 0$. □

Theorem 5.24 makes the ratio of every returned cover checkable on the residue up to a factor $1 + 5.1 \cdot 10^{-10}$, and exactly on bipartite graphs, but it is not a uniform bound, because it depends on C. By Lemma 5.23, a ratio R_0 on a graph of $\mathcal{R}$ means $\sigma_y(C) \le R_0\tau - \nu_f$: the strategies must keep the violation of complementary slackness small. The minimum-degree rule of c_5 and c_8 never puts both endpoints of an edge into the cover when degrees are ordered across a bipartition (Theorem 5.11); the exact reductions of c_{11} and c_{15} never create slack (Theorem 5.18); and a $(1, 2)$-swap of c_{14} or c_{15} lowers $\sigma_y(C)$ by exactly one for every y. What is not proved is that together they keep $\sigma_y(C)$ below $R_0\tau - \nu_f$ on every graph of $\mathcal{R}$. If every fixed ratio below 2 is NP-hard and P ≠ NP, they cannot: Salvador then exceeds R_0 on infinitely many graphs, all in $\mathcal{R}$, and, since every minimal cover is within $1/(1-\delta)$ on graphs with $\alpha \le \delta n$, these graphs are YES-type instances of the Khot–Regev reduction, with an independent set of almost half the vertices, τ close to ν, large treewidth, and degrees neither ordered across a bipartition nor heterogeneous enough for Theorem 5.15.

5.4. The split graph of c_1

Lemma 5.25 (The component count)**.** *Let $M = (e_1, \dots, e_{|M|})$ be the maximal matching of c_1 and $F_k = \{e_1, \dots, e_k\}$. The projection π maps the edges of the split graph bijectively onto the edges of G. After the first k edges of M are split, the split graph has* $\mathrm{count}(k) = k + c(G - F_k)$ *components with an edge. Moreover* $\mathrm{count}(0) = r(G)$, $\mathrm{count}(|M|) \ge |M|$, *and* $\mathrm{count}(k+1) - \mathrm{count}(k) \in \{0, 1, 2\}$ *for every k.*

Proof. A split replaces each moved edge $(u, 0)z$ by $(u, 1)z$, so edges are neither created nor destroyed and their projections are unchanged; initially π is a bijection. The edges of M are disjoint, so after F_k is split every vertex u has at most one copy that carries edges not in F_k, namely $(u, 1)$ if u is an endpoint of F_k and $(u, 0)$ otherwise, and the edges of F_k are isolated. Identifying each vertex with that copy maps the edges not in F_k onto $G - F_k$ and preserves adjacency, which gives the formula. With $k = 0$ it is $r(G)$, and with $k = |M|$ the $|M|$ isolated edges alone give $\mathrm{count}(|M|) \ge |M|$. Removing the edge e_{k+1} from $G - F_k$ changes c by -1 when e_{k+1} forms a component by itself, by $+1$ when it is a bridge with an edge on both sides, and by 0 otherwise, so count grows by 0, 1 or 2. □

Lemma 5.26 ($|M| - 1 \le r' \le |M|$)**.** *Let G have no isolated vertices and let M be a maximal matching of G. Algorithm 3 computes* $\mathrm{count}(0), \dots, \mathrm{count}(|M|)$ *correctly, and the split graph it returns has r' components with an edge, where*

$$r(G) \;\le\; r' \;\le\; |M| \;\le\; \nu(G) \;\le\; \tau(G) \;\le\; 2|M| \qquad \text{and} \qquad r' \;\ge\; |M| - 1.$$

Consequently $\tau(G) \le 2\tau(G) - |M| \le \frac{3}{2}\tau(G)$ and $2\tau(G) - |M| \le 2\tau(G) - r' \le 2\tau(G) - |M| + 1$.

Proof. Every component of G with an edge xy contains an edge of M, because x or y is matched by maximality, so $r(G) \le |M|$; any matching is at most τ, and $V(M)$ is a cover, so $|M| \le \nu \le \tau \le 2|M|$. The union-find structure of Algorithm 3 starts from $G - F_{|M|}$, and adding e_{k+1} back joins two trees or none; the number of trees with an edge changes by $1 - a$, with $a \in \{0, 1, 2\}$ the number of joined trees that already had an edge, or by 0, which is exactly the change of $c(G - F_k)$. By Lemma 5.25, count starts at $r(G) \le |M|$, ends at a value of at least $|M|$, never decreases and grows by at most 2 per step, so it takes a value in $\{|M| - 1, |M|\}$, and the largest value not above $|M|$ is r', with $r(G) \le r'$. The last statements follow from $\tau/2 \le |M| \le \tau$. □

Proposition 5.27 (c_1)**.** *For every graph G without isolated vertices, c_1 is a minimal vertex cover, computed in $O((n+m)\,\alpha(n))$ time.*

Proof. The decoded set is completed by a greedy cover of the uncovered edges and pruned, so c_1 is a cover, minimal by Theorem 5.1. The time is part of Theorem 6.1. □

No bound in terms of τ is proved for c_1 alone, and on some small graphs of Section 7.11 it exceeds $\frac{3}{2}\tau$: the decoding may select, in a matched edge of the split graph, the endpoint that a minimum cover avoids. Its role in the ensemble is to supply a cover built from a different matching structure, and the bound of Theorem 5.3 for the returned cover comes from c_2.

5.5. A per-instance 3/2 certificate

By Lemma 5.26, the inequality $|C| \le 2\tau - |M|$, which would be the analogue of Theorem 5.3 with $|M|$ components in place of r, implies the ratio $3/2$, since $|M| \ge \tau/2$, and it implies $|C| \le 2\tau - r'$ for the split graph of c_1, since $r' \le |M|$. It involves the unknown optimum, so it cannot be tested while the algorithm runs. A lower bound on τ turns it into a test that can.

Proposition 5.28 (Per-instance certificate)**.** *Let C be a vertex cover of G, M the maximal matching of c_2, and $L \le \tau(G)$ any lower bound.*

(i) If $|C| \le \frac{3}{2}L$, then $|C| \le \frac{3}{2}\tau(G)$.

(ii) If $|C| \le 2L - |M|$, then $|C| \le 2\tau(G) - |M| \le \frac{3}{2}\tau(G)$.

(iii) The test of (ii) implies the test of (i): $2L - |M| \le \frac{3}{2}L$.

Proof. (i) and the first inequality of (ii) follow from $L \le \tau$; the second inequality of (ii) is $|M| \ge \tau/2$ (Lemma 5.26). For (iii), $2L - |M| \le \frac{3}{2}L$ is equivalent to $L \le 2|M|$, which holds because $L \le \tau \le 2|M|$. □

Proposition 5.28 is a certificate, not a guarantee: it never certifies a false ratio, but it can fail to certify a true one when L is weak. Test (ii) certifies the conjectured bound itself, and test (i) certifies only the ratio $3/2$. Both need a lower bound stronger than $|M|$: with $L = |M|$, test (ii) reads $|C| \le |M|$, which holds only when C is optimal and $\tau = |M|$. Suitable lower bounds that can be computed in polynomial time are the size of a Karp–Sipser matching [26], of a maximum matching, which equals τ on bipartite graphs by Kőnig's theorem [20] and is found there by the Hopcroft–Karp algorithm [19], a packing of vertex-disjoint cliques $K_1, \dots, K_t$, which gives $\tau \ge \sum_i (|K_i| - 1)$, and, on regular graphs, Hoffman's ratio bound [30]. On bipartite graphs the certificate is therefore exact.

Lemma 5.29 (The ratio $|M|/L$)**.** *Let L be a lower bound with $|M| \le L \le \tau(G)$, for instance the largest of $|M|$ and the bounds above. Then*

$$\frac{1}{2} \;\le\; \frac{|M|}{L} \;\le\; 1, \qquad \text{so the target } T = 2L - |M| = L\Big(2 - \frac{|M|}{L}\Big) \text{ satisfies } L \le T \le \tfrac{3}{2}L.$$

If a cover C passes the test $|C| \le T$, then $|C|/\tau(G) \le \min\{|C|/L,\ 2 - |M|/\tau(G)\} \le \frac{3}{2}$.

Proof. $|M| \le L$ gives $|M|/L \le 1$, and $L \le \tau \le 2|M|$ gives $|M|/L \ge 1/2$; hence $T \in [L, \frac{3}{2}L]$. If $|C| \le T$, then $|C|/\tau \le |C|/L$ because $L \le \tau$, and $|C| \le 2\tau - |M|$ by Proposition 5.28(ii). □

Lemma 5.29 bounds the target, not the cover. The value of $|M|/L$ also shows how demanding the test is on a given instance: when $|M|/L$ is close to $1/2$ the threshold is close to $\frac{3}{2}L$, and when $|M|/L = 1$ the test passes only if $|C| = L = \tau$. Section 7.11 reports it on every instance.

5.6. The conjectures

Conjecture 5.30 (Ensemble bound)**.** *For every graph G without isolated vertices, the cover C returned by Algorithm 1, with M the maximal matching of c_2, satisfies*

$$|C| \;\le\; 2\,\tau(G) - |M|, \qquad \textit{equivalently} \qquad |C| - \tau(G) \;\le\; \tau(G) - |M|.$$

Conjecture 5.31 (Universal $(2-\varepsilon_0)$-approximation)**.** *For every graph G, Algorithm 1 returns a vertex cover C with $|C| \le 1.999999999\,\tau(G)$.*

By Lemma 5.26, Conjecture 5.30 implies $|C| \le \frac{3}{2}\tau(G)$ and $|C| \le 2\tau(G) - r'$ on every graph, and hence Conjecture 5.31; by Corollary 5.4, Conjecture 5.31 is a theorem on every graph with $\tau \le 10^9 r$ and is open only on graphs with more than 10^9 vertices. Conjecture 5.30 would also give a lower bound on τ that is computable in polynomial time, $\tau \ge (|C| + |M|)/2$.

Conjecture 5.30 is a statement about the ensemble, not about any single strategy: on the small graphs of Section 7.11 both c_1 and c_2 alone violate it, on a graph with six vertices for the first and four for the second, while the returned cover satisfies it on every instance. The second form shows how demanding it is. The excess of C over the optimum must not exceed the gap $\tau - |M|$ between the optimum and the maximal matching. When $|M|$ is close to $\tau/2$, as in a graph made of disjoint triangles with one matched edge each, the conjecture asks only for the ratio $3/2$; when $|M| = \tau$, which happens whenever the maximal matching is maximum and $\tau = \nu$, as in many bipartite graphs, it asks for an *optimal* cover. If UGC holds in the form of [8] and P $\neq$ NP, then Conjecture 5.30 is false (Theorem 5.32), and since a polynomial-time algorithm that failed on finitely many graphs could be patched by a lookup table, it then fails on infinitely many graphs. Such graphs cannot be NO-type instances of the reduction of Khot and Regev: a graph with no independent set of more than δn vertices, $\delta \le 1/4$, has $\tau \ge (1-\delta)n$ and $|M| \le n/2$, so every cover satisfies $|C| - \tau \le \delta n \le (\frac{1}{2} - \delta)n \le \tau - |M|$. A counterexample must be a YES-type instance, with an independent set of almost half the vertices, on which the excess of C exceeds the gap $\tau - |M|$. The graphs of Section 7 are far smaller than the asymptotic instances of the reduction, so they test the conjecture, they do not prove it.

5.7. Conditional consequence for P versus NP

Theorem 5.32 (Conditional consequence)**.** *Assume that for every fixed $\varepsilon_0 > 0$ it is NP-hard to approximate Minimum Vertex Cover within $2 - \varepsilon_0$. If Conjecture 5.30 or Conjecture 5.31 is proved, then* P = NP*. The assumption holds if UGC holds [8], and in particular if the Unique Games Theorem reported in [12] holds; without any assumption, a proof of either conjecture would refute UGC unless* P = NP*.*

Proof. Algorithm 1 runs in linear time (Theorem 6.1). A proof of Conjecture 5.30 makes it a polynomial-time $\frac{3}{2}$-approximation, by Lemma 5.26, and a proof of Conjecture 5.31 makes it a polynomial-time 1.999999999-approximation. Both ratios are of the form $2 - \varepsilon_0$ with $\varepsilon_0 > 0$, so under the assumption the algorithm decides an NP-hard problem in polynomial time. The theorem of Khot and Regev [8] derives the assumption from UGC. □

Remark 5.33. Theorem 5.32 is a conditional implication, not a proof of P = NP. The theorem of [12] was released in October 2026 with Lean artifacts and had not been peer reviewed at the time of writing; we cite it as reported and have not checked the formalization. The ratio $3/2$ lies above the unconditional NP-hardness threshold $\sqrt{2} - \varepsilon$ of [6], so Conjecture 5.30 contradicts no hardness result that is known without UGC. The only bounds proved here for all graphs are 2 and $2 - r/\tau$ (Theorems 5.2 and 5.3), neither of which is a constant below 2.

6. Runtime Analysis

Theorem 6.1. *With the default parameters, every strategy $c_1, \ldots, c_{17}$ of Algorithm 1 runs in $O(n + m)$ time, except c_1 and c_7, which run in $O((n + m)\,\alpha(n))$ time, where α is the inverse Ackermann function. Hence the whole algorithm runs in $O((n + m)\,\alpha(n))$ time.*

Proof. Preprocessing and every pruning pass are single scans, $O(n + m)$. *Base strategies.* The matching of c_2, which is also the matching M of c_1, and the primal-dual scan of c_6 are single scans of the edges. In c_1 the union-find pass of Algorithm 3 makes $O(m)$ operations, $O((n + m)\alpha(n))$ with union by size and path compression [33]; the split graph has m edges and at most $2n$ nodes; the edges of M and the matched edges of H are ordered by counting sorts on positions, since the first endpoints within each class are distinct; the matching of H is one scan, and each of the three decodings, completions by a bucket queue and prunings is $O(n + m)$. The bucket queues of c_3, c_5 and the residual variant of c_8 receive $O(n + m)$ pushes, one per vertex and per degree decrement, and every pop costs $O(1)$; the minimum-bucket pointer of c_5 and c_8 moves down by at most one per deletion. In c_4 each auxiliary edge is inspected at most twice. In c_7 the spanning-forest core is built by union-find in $O((n + m)\alpha(n))$, the gadget is a forest with $O(n)$ nodes, and at the default parameter the Baker pass is its greedy baseline, which buckets the nodes by their three weights. The auxiliary graph of c_8 has $2m$ nodes in disjoint cycles and edges, solved by a dynamic program linear in its size. In c_9 a layer is built from the edges left in time linear in their number, and every layer takes at least half of them (Proposition 5.21), so all layers together cost $O(n + m)$; a solve of a layer runs a matching scan, at most $p = 8$ Hopcroft–Karp phases of linear cost [19] and two breadth-first searches, and each of at most $q = 6$ orders solves every layer once, so c_9 costs $O(pq(n + m))$. The dynamic programs of c_{10} and c_{12} and the folding of c_{11} charge every table entry and adjacency scan to work counters and stop at fixed multiples of $n + m$: 1024 times the gadget size for c_{10}, whose bags have at most six nodes, $256(n + m)$ for c_{12}, whose width is at most 10, and $4(n + m)$ for folding; the reductions of c_{11} run on bucket queues of residual degrees. *Improvers.* The union of c_{13} has at most n vertices. The local searches of c_{14} and c_{15} stop after $\max\{100(n + m), 20{,}000\}$ work per call, and the refinement c_{17} after $100(n + m)$ work per call, sorting included; each of the three rounds of c_{15} and of c_{16} costs $O(n + m)$, and the votes of c_{16} cost $O(\sum_i |c_i|) = O(n)$. Summing a constant number of these terms gives the claim. □

The constants are large: the work budgets alone allow a few hundred passes over the graph, and Section 7.14 measures how the running time grows in practice. Larger limits, such as a smaller ε for the Baker pass of c_7, unbounded refinement, or numbers of phases or orders of c_9 that grow with the graph, are available as options and give up the linear bound.

7. Reproducible Experiments

7.1. Setup

Every number in this section comes from running all the experiment scripts of the folder `car/` of the repository in quick mode, with the option `--quick`, using `Salvador` 0.1.6 from the repository, Python 3.12.3 and NetworkX 3.7 on Windows 11; the comparison with Hvala on the DIMACS benchmarks (Section 7.13) comes from a run of the command-line program of the package on those files. The scripts were written with Claude Opus 5.5 and Claude Fable 5.2 and run by the author. Table 2 lists the scripts, the options they were run with and their time, 2,807 seconds in total. Except for the small graphs of Part A, every instance is built and solved in a worker process with a time limit, and is skipped if it exceeds it; no instance was skipped. Instances are generated from a fixed seed and their name alone, so that every instance can be rebuilt independently of the others.

7.2. Parts A and B: small graphs and large adversarial graphs

Part A collects 298 small graphs whose optimum is known by Kőnig's theorem or by branch and bound: 202 graphs of order at most 7 from the graph atlas, 12 bipartite hill-climbing instances, 30 random general graphs with $n \le 12$, 25 random bipartite graphs, 16 bipartite grids, 12 relabelling stress instances and one bipartite obstruction. Every returned cover is valid and optimal; the part took 6.8 seconds.

Part B uses graphs with up to 5,000 vertices, where we report $|C|/L$ for a certified lower bound $L \le \tau$, the largest of: a Hopcroft–Karp maximum matching on bipartite graphs, which gives $L = \tau$ by Kőnig's theorem [20, 19]; the NetworkX minimum maximal matching and a Karp–Sipser matching [26]; a greedy packing of vertex-disjoint cliques K_i, with $\tau \ge \sum_i(|K_i| - 1)$; and a spectral bound, $\tau \ge n - \lfloor n(1 - \delta/\mu_{\max})\rfloor$ with δ the minimum degree and $\mu_{\max}$ the largest Laplacian eigenvalue, which on regular graphs is Hoffman's bound [30]; $\mu_{\max}$ is computed by Lanczos iteration, inflated by its residual and capped by $\max_{uv}(\deg u + \deg v)$. Table 3 gives the results.

Script	Options used by `--quick`	Measures	Time (s)
`car_ratio.py`	`--max-n 5000 --timeout 120`	approximation ratio, Parts A and B	107.2
`car_strategies.py`	`--max-n 5000`	which strategy wins; improvement chain	126.8
`car_c11_variants.py`	`--max-n 5000`	the six variants of c_{11}	98.9
`car_c10_bags.py`		bags and exactness of c_{10}, Part G	123.8
`car_c1_components.py`	`--max-n 5000`	the split graph and the ensemble bound, Part I	156.5
`car_c1_fix.py`		the bound for single strategies, Part I	1.1
`car_c9_layers.py`		the bipartite layers, Part J	6.2
`car_regular.py`		regular graphs, Part F	16.3
`car_bipartite.py`	`--max-n 1000`	bipartite graphs, Part C	143.7
`car_yes_instances.py`	`--max-n 1280 --exact-time 20`	Khot–Regev YES instances, Part D	634.5
`car_mechanism.py`	`--exact-time 20`	certificates and mechanism, Part E	208.2
`car_triangle_free.py`	`--max-n 2000 --exact-time 20 --timeout 120`	triangle-free graphs, Part H	889.1
`car_scaling.py`	`--max-n 8000`	running time against $n + m$	286.2
`car_ratio_timing.py`		time of each strategy on Part A	8.2

Table 2: The experiment scripts run in quick mode, the options added to `--quick`, and their time. `--max-n` caps the number of vertices, `--exact-time` the seconds given to the integer program that proves τ, and `--timeout` the limit per instance.

Family	Max n	Max ratio u.b.	Bound	Matching-only	Max time (s)
Random 12-regular	5,000	1.1668	spectral	1.566	4.67
Random 6-regular	5,000	1.1418	spectral	1.400	3.05
Random 3-regular	5,000	1.0726	spectral	1.234	1.93
Sparse Erdős–Rényi (deg. 4)	5,000	1.0639	Karp–Sipser	1.254	2.26
Watts–Strogatz small-world	5,000	1.0539	clique packing	1.417	2.57
Barabási–Albert scale-free	5,000	1.0078	Karp–Sipser	1.354	3.01
Complete bipartite	800	1.0000	exact	1.000	32.56
Crown graph	800	1.0000	exact	1.000	30.20
Double star + bridge	2,002	1.0000	exact	1.000	0.72

Table 3: Part B: certified ratio upper bound $|C|/L$ on the 11 instances of nine families, the lower bound that attained it, the bound the minimum maximal matching alone would give, and the time of the slowest instance.

The certified ratio never exceeds 1.1668, on the random 12-regular graph with 5,000 vertices, where the cover has 3,757 vertices and the spectral bound certifies $\tau \geq 3{,}220$; with the matching alone it would be 1.566. The part took 99 seconds. With Part A, the exact or certified ratio never exceeds 1.17.

7.3. Which strategy wins

The strategy report runs the families of Parts A and B up to $n = 5{,}000$, 297 instances, and checks every candidate. Among the base strategies, c_{12}, which starts from the best of $c_1, \ldots, c_{11}$, attains the smallest base cover on all 297 instances and is the only one to do so on two; c_{11} attains it on 295, c_{10} on 291, c_9 on 287, c_5 and c_8 on 284, c_3 on 276, c_1 on 275, c_4 on 265, c_7 on 264, and c_2 and c_6 on 251. Among the strategies that use earlier covers, c_{11} strictly improved the best of $c_1, \ldots, c_{10}$ on 6 instances, by 183 vertices in total, c_{12} the best of $c_1, \ldots, c_{11}$ on 2, by 3, the local search c_{14} the best of $c_1, \ldots, c_{13}$ on 5, by 113, and c_{15} the best of $c_1, \ldots, c_{14}$ on 4, by 32, of which 28 came from its swap searches and 4 from its reductions; c_{10} and c_{16} never improved their reference, c_{17} improved the best of $c_1, \ldots, c_{16}$ on 1 instance, by 1, and the union c_{13} was larger than the best earlier cover on 43 instances and never smaller. Table 4 gives the improvement chain on the six large random graphs.

Instance (n = 5,000)	c_1	**Best** $c_2..c_8$	c_9	c_{11}	c_{12}	c_{14}	c_{15}	**search**	**reduce**	**forced**	L	c_{17}
Random 3-regular	3,118	2,838	2,886	2,772	2,771	2,767	2,762	2	3	1,437	2,498	2,762
Random 6-regular	3,582	3,355	3,429	3,332	3,332	3,288	3,278	10	0	2,349	2,499	3,277
Random 12-regular	4,012	3,826	3,943	3,810	3,810	3,769	3,757	12	0	3,401	2,498	3,757
Sparse Erdős–Rényi	2,743	2,640	2,716	2,600	2,600	2,598	2,598	0	0	2,110	2,442	2,598
Watts–Strogatz	3,715	3,605	3,661	3,569	3,567	3,545	3,540	4	1	2,981	2,491	3,540
Barabási–Albert	2,580	2,467	2,581	2,465	2,465	2,465	2,465	0	0	2,027	2,446	2,465

Table 4: Improvement chain on the six large random graphs: c_1, the best of $c_2, \ldots, c_8$, c_9, c_{11}, c_{12} (here c_{10} equals its reference, the graphs being non-bipartite), c_{14}, c_{15} (with the vertices removed by its swap searches and its reductions), the vertices forced in the last round of c_{16} with its matching lower bound L, and the returned cover c_{17}.

On these graphs the minimum-degree strategies c_5 and c_8 give the best independent cover, the reductions of c_{11} remove 2 to 66 more vertices, and the local search of c_{14}, which starts from the best of all base strategies, removes up to 44; consensus forcing fixes 29% to 68% of the vertices but finds no smaller cover. The exact strategies certify their own optimality often: c_{12} has an empty core on 287 of the 297 instances, c_{11} needs no peel on 223, c_{10} has an empty core on 109 of the 116 instances whose components are all bipartite, and the phases of c_9 end with a maximum matching on every bipartite instance; on the large random families the core of c_{12} holds most of the vertices and c_{11} peels up to half of them. The six variants of c_{11} (peel `max`, `ref` or `min`, with or without folding) were compared with the best of $c_1, \ldots, c_{10}$ on the same 297 instances: folding raises the instances that need no peel from 168 to 223; `ref` and `ref+fold` are never worse and save 117 and 132 vertices; `max+fold` is worse on 2 instances and saves 80; and `min`, which does not satisfy $|S| \leq \tau + p$, is worse on 6. The ensemble uses `max+fold` and `ref+fold`.

7.4. Ablation

Every strategy sees the covers found before it, so the contribution of each group of strategies can be read off one run (Table 5).

The base strategies $c_1, \ldots, c_{10}$ already solve 106 of the 108 bipartite graphs, and the reductions of c_{11} solve the other two. On the Khot–Regev YES instances and on the mechanism instances the base strategies are weak, and the reductions of c_{11} and the local search of c_{14} do most of the work: together they reduce the total excess on the YES instances with known optimum from 140 to 12. Reduce-and-search, consensus forcing and the refinement still lower the total excess on the mechanism instances, from 21 to 16, and the refinement makes one more cover optimal. No strategy ever made the returned cover larger, as the guards of Algorithm 1 guarantee.

7.5. Part C: bipartite graphs

On bipartite graphs $\tau = \nu$, so Theorem 5.12 gives only the ratio 2, while by Theorem 5.24 the measured ratio is the exact a-posteriori certificate. Part C runs 108 bipartite graphs of 15 families with up to 1,000 vertices and shuffled labels, and computes $\tau = \nu$ by Hopcroft–Karp for measurement only (Table 6).

Experiment	**Inst.**	**Measure**	**Best** $c_1..c_{10}$	$+c_{11}$	$+c_{12}, c_{13}$	$+c_{14}$	$+c_{15}$	$+c_{16}$	**Final** c_{17}
Bipartite graphs (Part C)	108	optimal	106	108	108	108	108	108	108
		total excess	3	0	0	0	0	0	0
Mechanism (Part E), τ known	33	optimal	21	27	27	28	28	28	29
		total excess	55	42	42	21	20	18	16
Khot–Regev YES (Part D), τ known	18	optimal	1	8	8	13	15	15	15
		total excess	140	56	56	12	9	9	9
Khot–Regev YES (Part D), all	38	cover − planted	−503	−673	−673	−800	−820	−820	−828
Regular graphs (Part F), general	26	total cover	3,978	−25	0	−31	−6	0	0
Large random graphs (strategy report)	6	total cover	18,731	−183	−3	−113	−32	0	−1

Table 5: Ablation. Each column adds strategies to the previous ones; "optimal" counts covers of size τ, "total excess" is $\sum(|C| - \tau)$, and the rows "total cover" give the size of the best of $c_1, \ldots, c_{10}$ and then the change made by each addition.

Family	**Inst.**	**Max** n	**Opt.** $c_1..c_{10}$	**Opt.** c_9	**Optimal**	**Max ratio**	**Exc.** $c_1..c_{10}$	**Excess**	c_9 **exact**	c_{10} **exact**	c_{12} **exact**
Random bipartite (balanced)	18	1,000	17	17	18	1.0000	2	0	17	0	0
Random bipartite (1:2)	18	1,000	18	18	18	1.0000	0	0	18	0	0
Random d-regular bipartite	18	1,000	18	18	18	1.0000	0	0	18	0	0
Power-law bipartite	6	1,000	6	6	6	1.0000	0	0	6	0	3
Double cover $G \times K_2$	12	1,000	12	12	12	1.0000	0	0	12	0	0
Hidden perfect matching	12	1,000	12	12	12	1.0000	0	0	9	0	3
Random tree	6	1,000	6	6	6	1.0000	0	0	6	6	6
2D grid, 70% of edges	6	961	5	5	6	1.0000	1	0	5	2	3
2D grid	2	961	2	2	2	1.0000	0	0	2	0	0
3D grid	2	1,000	2	2	2	1.0000	0	0	2	0	0
Even cycle	2	1,000	2	2	2	1.0000	0	0	1	2	2
Complete bipartite	2	1,000	2	2	2	1.0000	0	0	2	0	0
Crown graph	2	1,000	2	2	2	1.0000	0	0	2	0	0
Hypercube	1	256	1	1	1	1.0000	0	0	1	0	0
Hypercube, 50% of edges	1	256	1	1	1	1.0000	0	0	1	0	0
All	**108**	**1,000**	**106**	**106**	**108**	**1.0000**	**3**	**0**	**102**	**10**	**17**

Table 6: Part C: the returned cover on 108 bipartite graphs against $\tau = \nu$; "Opt." and "Exc." count optimal instances and total excess of the best of $c_1, \ldots, c_{10}$, of c_9 and of the returned cover; "c_9 exact" counts the instances on which the phases of c_9 end with a maximum matching, and the last two columns the empty cores of c_{10} and c_{12}.

The ensemble returned a minimum cover on all 108 graphs, 26,523 vertices in total. The phases of c_9 end with a maximum matching on 102 graphs, where c_9 is optimal by Proposition 5.21(iii); it is optimal on 106, and on the other two, a random balanced bipartite graph and a thinned grid, it is two and one vertices above τ. Every regular bipartite graph, double cover, tree, grid, cycle, complete bipartite graph, crown graph and hypercube was solved exactly, as Theorem 5.11 predicts for the 19 regular and degree-separated ones. The part took 143 seconds.

7.6. Part D: YES-type instances of the Khot–Regev reduction

A Unique Games instance with labels $[k]$ is generated on a random d-regular graph with N vertices and a planted labelling L, each edge carrying a permutation π_{vw} with $\pi_{vw}(L(w)) = L(v)$ (random, or a shift modulo k in the linear variant), and a fraction η of the edges receive a random constraint. Every vertex v becomes a block of s subsets $x \subseteq [k]$ drawn from the p-biased measure, and (v, x) and (w, y) are joined when x and $\pi_{vw}(y)$ are disjoint [8]. The dictator set $\{(v, x) : L(v) \in x\}$ spans edges only on violated constraints, and removing one endpoint of each leaves an independent set whose complement is a *planted cover* of about $(1 - p)n$ vertices. The base family has $k = 4$, $p = 0.45$, $\eta = 0$, $d = 4$, $s = 16$; nine more change one parameter. With $N \in \{40, 80\}$ and two seeds this gives 38 instances with 640 to 1,280 vertices. The lower bound is the larger of the linear relaxation and the bound of an integer program (HiGHS, 20 seconds), which proved τ on 18 instances (Table 7).

Family	**Inst.**	**≤ planted**	**τ known**	**Opt. $c_1..c_{10}$**	**Optimal**	**Max exact**	**Max cert.**
Base family	4	3	2	0	2	1.0000	1.0612
Label size $k = 3$	4	4	1	0	1	1.0000	1.0731
Label size $k = 6$	4	4	4	0	4	1.0000	1.0000
Bias $p = 0.35$	4	4	0	0	0	–	1.1740
Bias $p = 0.49$	4	4	4	0	4	1.0000	1.0000
Noise $\eta = 0.02$	4	3	2	0	1	1.0089	1.0687
Noise $\eta = 0.1$	4	4	1	0	0	1.0029	1.0381
Constraint degree $d = 8$	4	3	2	1	2	1.0000	1.0810
Samples per block $s = 32$	2	1	0	0	0	–	1.1140
Linear UG, Max-2-Lin($\mathbb{Z}_k$)	4	3	2	0	1	1.0148	1.0427
All	**38**	**33**	**18**	**1**	**15**	**1.0148**	**1.1740**

Table 7: Part D: 38 sampled Khot–Regev YES instances. "≤ planted" counts covers no larger than the planted one; "Max exact" is the largest $|C|/\tau$ where τ is known; "Max cert." the largest $|C|/L$.

The cover is no larger than the planted one on 33 instances and strictly smaller on 28. Where τ is known it is optimal on 15 of 18, with mean ratio 1.0015 and maximum 1.0148, on a linear-constraint instance (342 against 337; planted 347); the certified ratio never exceeds 1.174. The base strategies are optimal on only one of the 18, and the ablation of Table 5 shows that the reductions of c_{11} and the local search of c_{14} close almost all of the gap. The part took 634 seconds, mostly in the integer programs. These instances are far from the asymptotic regime of the reduction, so they show only that Salvador stays within 1.5% of the optimum where it is known.

7.7. Part E: certificates and mechanism

Part E runs 40 instances of ten families with 140 to 320 vertices: sampled and fixed-weight Khot–Regev YES instances, regular graphs with a planted independent set, random regular graphs and thinned grids. It computes τ by an integer program, the linear relaxation, the Nemhauser–Trotter decomposition [34], the best term of $\beta(G)$ in Theorem 5.22, the degree bias of a maximum independent set, and the best of the base strategies and the union, $\min_{i \le 13} |c_i|$ (Table 8).

The a-priori certificate $\beta(G)$ is below R_0 on all 40 instances, always through the Caro–Wei term of c_8, with maximum 1.908, and no instance lies in $\mathcal{R}$. Maximum independent sets favour low-degree vertices (mean bias 0.663, exactly 0.5 on regular graphs), which is the choice of c_5 and c_8. The best of $c_1, \dots, c_{13}$ is 1.02% above the optimum on average and optimal on 27 of the 33 instances with known τ; the improvers improve it on 5 and leave a mean excess of 0.49%, with 29 of 33 covers optimal. The hardest family is the fixed-weight Khot–Regev family with $k = 5$, where the optimum is proved on three of the four instances, two of these covers are not optimal, and the largest ratio is 1.094. No returned cover admits a $(1, 2)$-swap. The Nemhauser–Trotter kernel covers 74% of the vertices on average, so the linear

Family	τ known	Opt.	Max exact	Max cert.	Max β	Bias	Base exc.
KR sampled $k = 4$	4/4	3	1.0054	1.0054	1.798	0.858	0.87%
KR sampled $k = 6$	4/4	4	1.0000	1.0000	1.902	0.933	0.00%
KR fixed $k = 4, t = 2$ (bip.)	4/4	4	1.0000	1.0000	1.908	0.618	0.00%
KR fixed $k = 5, t = 2$	3/4	1	1.0938	1.0938	1.741	0.552	9.66%
KR fixed $k = 6, t = 3$ (bip.)	4/4	4	1.0000	1.0000	1.734	0.786	0.00%
Planted regular $\alpha = 0.45$, $d = 6$	4/4	4	1.0000	1.0000	1.575	0.594	0.00%
Planted regular $\alpha = 0.48$, $d = 10$	4/4	4	1.0000	1.0000	1.743	0.631	0.00%
Random regular $d = 3$	2/4	1	1.0112	1.0409	1.404	0.500	0.56%
Random regular $d = 6$	0/4	0	–	1.1989	1.557	0.500	–
Thinned grid (70%, bip.)	4/4	4	1.0000	1.0000	1.450	0.660	0.00%
All	**33/40**	**29**	**1.0938**	**1.1989**	**1.908**	**0.663**	**1.02%**

Table 8: Part E: "Max β" is the largest a-priori certificate of Theorem 5.22, always its Caro–Wei term; "Bias" the mean probability that a vertex of a maximum independent set has smaller degree than a vertex outside it; "Base exc." the mean relative excess of $\min_{i \le 13} |c_i|$ over τ.

relaxation fixes little, and its persistency is respected on all 40 instances. The mean ratio is 1 at both sizes $N = 10$ and $N = 20$ on six families; on the sampled family with $k = 4$ it is 1 and 1.0027, on the fixed-weight family with $k = 5$ it is 1.073 and 1.017, on the random 3-regular graphs 1.006 and 1.038, and on the random 6-regular graphs the certified bound grows from 1.093 to 1.162, where the integer program no longer proves τ and the bound comes from the weak relaxation. The part took 207 seconds.

7.8. Part F: regular graphs of increasing degree

For every $d \in \{3, 4, 5, 6, 8, 10, 12, 16, 20, 24, 32, 48, 64\}$, $n = 200$ and two seeds, Part F builds a random d-regular graph and a random d-regular bipartite graph. On the 26 bipartite graphs every cover is optimal, and c_5, c_8 and c_9 are each exact on all of them alone, as Theorem 5.11 requires for c_5 and c_8. Table 9 reports the general graphs, with the spectral lower bound, and compares with $1 + 2\sqrt{d-1}/d$, the value of Hoffman's bound $n/\tau \le 1 + |\lambda_{\min}|/d$ when $|\lambda_{\min}| = 2\sqrt{d-1}$, which random regular graphs attain asymptotically [35].

d	Max ratio u.b.	Mean ratio u.b.	$\lvert C\rvert/n$	L/n	vs. estimate	$1 + \frac{2\sqrt{d-1}}{d}$
3	1.0769	1.0721	0.557	0.520	2.083	1.943
4	1.1101	1.1064	0.598	0.540	1.947	1.866
5	1.1339	1.1339	0.635	0.560	1.783	1.800
6	1.1466	1.1379	0.660	0.580	1.639	1.745
8	1.1639	1.1605	0.705	0.608	1.468	1.661
10	1.1562	1.1523	0.738	0.640	1.367	1.600
12	1.1527	1.1478	0.757	0.660	1.293	1.553
16	1.1606	1.1491	0.790	0.688	1.209	1.484
20	1.1549	1.1509	0.820	0.713	1.171	1.436
24	1.1507	1.1434	0.838	0.733	1.139	1.400
32	1.1523	1.1383	0.865	0.760	1.104	1.348
48	1.1250	1.1211	0.902	0.805	1.076	1.286
64	1.1145	1.1145	0.925	0.830	1.063	1.248

Table 9: Part F: random d-regular graphs, $n = 200$. "vs. estimate" is $|C|$ over the estimate $n(1 - 2\ln d/d)$ of τ [36], asymptotic in d and not a bound.

The certified ratio never exceeds 1.1639 ($d = 8$) and does not grow with d, far below the bound $2 - 2/(d + 1)$ of Theorem 5.9; the covers approach the estimate of τ as d grows. The part took 14 seconds of solving time.

7.9. Part G: the bags and the exactness of c_{10}

Part G runs c_{10} alone, with the pruned cover of the NetworkX minimum maximal matching as reference, on the 108 bipartite graphs of Part C, records the bags and the core, and computes the minor-min-width lower bound on the treewidth of the gadget (Table 10); it also runs c_{10} on 285 random bipartite graphs with at most 14 vertices and compares it by brute force with every minimum cover.

Family	Inst.	Max bag	Exact	$\lvert c_{10}\rvert/\tau$	Ref./τ	Max tw LB	Time (s)
Random bipartite (balanced)	18	3	0	1.649	1.808	69	6.85
Random bipartite (1:2)	18	4	0	1.856	1.989	46	6.40
Random d-regular bipartite	18	4	0	1.693	1.866	41	3.81
Power-law bipartite	6	3	0	1.084	1.841	17	0.78
Double cover $G \times K_2$	12	3	0	1.660	1.858	30	2.16
Hidden perfect matching	12	4	0	1.461	1.793	30	2.05
Random tree	6	2	6	1.000	1.775	1	0.50
2D grid, 70% of edges	6	6	2	1.110	1.897	5	2.07
2D grid	2	4	0	1.697	1.976	7	0.43
3D grid	2	3	0	1.941	1.996	15	0.44
Even cycle	2	3	2	1.000	2.000	2	0.30
Complete bipartite	2	4	0	2.000	2.000	5	26.88
Crown graph	2	5	0	2.000	2.000	6	37.21
Hypercube	1	3	0	1.953	1.953	19	0.13
Hypercube, 50% of edges	1	3	0	1.578	1.828	12	0.08
All	**108**	**6**	**10**				**90.08**

Table 10: Part G: c_{10} alone. "Exact" counts empty cores; $\lvert c_{10}\rvert/\tau$ and "Ref./τ" are means; "Max tw LB" the largest lower bound on the treewidth of the gadget.

No bag exceeded six nodes. The core is empty on 10 instances (trees, even cycles and two thinned grids), where the cover is optimal; on 84 others the treewidth bound is at least 6, so a core is unavoidable under the cap whatever the elimination order. Where the core is not empty, c_{10} improves its reference on every family except the dense complete bipartite and crown graphs and the hypercube. On the 285 small graphs the core was empty on 270, and on all of them the cover was a minimum cover with the fewest doubly covered edges, as Proposition 5.17 states. The part took 90 seconds, 64 of them on the four dense graphs.

7.10. Part H: triangle-free graphs

Part H runs triangle-free graphs: named graphs (Petersen, Clebsch, Hoffman–Singleton and others), Mycielski, Andrásfai, Kneser $K(n,k)$ with $n < 3k$, generalized Petersen graphs, middle-third circulants, small random graphs of the triangle-free process, and an adversarial hill climb that maximises the ratio against the optimum; and larger graphs: Khot–Regev YES instances on triangle-free constraint graphs, the triangle-free process with average degree 4 to 32, $G(n,p)$ with one edge of every triangle removed, random regular graphs with triangles switched out, Frankl–Rödl graphs, and large Mycielski, Andrásfai and circulant graphs. Where τ is not proved, the true ratio lies in $[\lvert C\rvert/\tau_{\text{upper}}, \lvert C\rvert/L]$ (Table 11).

Family	Inst.	Max n	τ known	Optimal	Max exact	Max lower	Max cert.
Small families with known optimum[a]	161	64	161	161	1.0000	1.0000	1.0000
Khot–Regev on triangle-free constraint graphs[b]	19	1,280	3	2	1.0016	1.0356	1.2003
Large Mycielski M_8–M_{10}	3	767	3	3	1.0000	1.0000	1.0000
Large Andrásfai graphs	4	1,199	1	1	1.0000	1.0000	1.2165
Large middle-third circulants	4	2,000	1	1	1.0000	1.0000	1.2174
Frankl–Rödl $FR(q,t)$	2	1,024	0	0	–	1.0000	1.1938
Triangle-free process, avg. deg. 4–32	4	2,000	0	0	–	1.0000	1.6970
$G(n,p)$, triangles removed, deg. 4–16	3	2,000	0	0	–	1.0000	1.5150
Random d-regular, triangle-free, d = 3–12	5	2,000	0	0	–	1.0000	1.1694
Random bipartite (control)	1	1,996	1	1	1.0000	1.0000	1.0000
All	**206**	**2,000**	**170**	**169**	**1.0016**	**1.0356**	**1.6970**

Table 11: Part H: 206 triangle-free graphs. "Max lower" and "Max cert." are the largest certified lower and upper bounds on the ratio. [a]Named, Mycielski, Andrásfai, Kneser, generalized Petersen, circulant, random ($n \le 40$) and hill-climb instances. [b]Ten variants as in Part D.

All 161 small instances are solved optimally, and the hill climb finds no ratio above 1. Among the 45 large instances the optimum was proved on 9, and the cover is optimal on 8; the exception is a Khot–Regev instance with $p = 0.49$ (621 against 620, ratio 1.0016). On the Khot–Regev triangle-free instances the certified upper bound is at most 1.200. On the triangle-free process and the $G(n, p)$ graphs the only lower bound is the relaxation, about $n/2$, and the certified upper bound grows with the average degree to 1.70 at degree 32, while the lower bound stays at 1: the intervals are wide because the relaxation is weak. The certified bound is at most 1.5 on 203 of the 206 instances. These figures do not contradict the hardness of the class. By the triangle-removal argument recalled in the introduction, a polynomial-time $(2 - \varepsilon)$-approximation on triangle-free graphs would give one on all graphs, so under UGC triangle-free graphs are as hard as general graphs in the worst case [11, 8]. The ratios measured here, optimal on every small instance and certified at most 1.5 on 203 of the 206 instances, describe the families generated in this part, not the worst case; the hard instances of the reduction are far larger, and if UGC holds and P $\neq$ NP, then for every fixed $\varepsilon > 0$ there are triangle-free graphs on which the cover of Algorithm 1 exceeds $(2 - \varepsilon)\tau$. The part took 888 seconds.

7.11. Part I: the bound $2\tau - |M|$

Part I tests Conjecture 5.30 and the structure of the split graph of c_1. Its first script runs the ensemble on the 297 non-adaptive instances of Parts A and B, 291 of them with a known optimum, and checks $|C| \le 2\tau - |M|$ for the returned cover; its second script checks the same bound, and $2\tau - r'$, for single strategies on 311 small graphs with a known optimum, the 286 non-adaptive graphs of Part A and 25 hand-made stress graphs on which a leaf-for-hub choice is expensive (paths, stars, double stars, spiders, caterpillars, brooms and $K_{2,k}$). On the six graphs without a known optimum, the bound is decided by the certificate of Proposition 5.28(ii) (Table 12).

Cover	**Graphs with τ**	$> 2\tau - \lvert M\rvert$	$> 2\tau - r$	$> \frac{3}{2}\tau$	**Optimal**	**Max exact ratio**	**Certified**
Returned cover c_{17}	291	**0**	**0**	**0**	**291**	**1.0000**	**6 of 6**
c_1 alone	311	13	0	3	295	1.6667	–
c_2 alone	311	18	0	3	270	1.6667	–

Table 12: Part I: the bound $2\tau - |M|$ for the returned cover on the instances of Parts A and B, and for c_1 and c_2 alone on the small graphs. "Certified" counts the graphs without a known optimum on which $|C| \le 2L - |M|$.

The returned cover satisfies the bound on every instance, whereas c_1 and c_2 alone violate it on 13 and 18 small graphs; the smallest counterexamples have six vertices for c_1 ($\tau = 3$, $|M| = 3$, $|c_1| = 4$) and four for c_2. The bound is therefore a property of the ensemble, as Conjecture 5.30 states, and not of either strategy. On 154 of the graphs with a known optimum the maximal matching has $|M| = \tau$, so the conjecture requires an optimal cover; the returned cover was optimal on all of them, and c_1 on 141. Over the 297 instances $|M|/L$ lies between 0.500 and 1.000, as Lemma 5.29 requires, with mean 0.843 and median 1.000, and on 65% of them $|M|/L \ge 0.75$, so that the threshold $2 - |M|/L$ is at most 1.25. The split graph has $r' = |M|$ components on 282 of the 297 instances and $r' = |M| - 1$ on the other 15, as Lemma 5.26 allows.

Instance	n	m	L	**Bound**	$\lvert M\rvert$	$2L - \lvert M\rvert$	$\lvert C\rvert$	$\lvert M\rvert/L$	$2 - \lvert M\rvert/L$	$\lvert C\rvert/L$
Random 3-regular	5,000	7,500	2,575	spectral	2,239	2,911	2,762	0.870	1.130	1.0726
Random 6-regular	5,000	15,000	2,870	spectral	2,340	3,400	3,277	0.815	1.185	1.1418
Random 12-regular	5,000	30,000	3,220	spectral	2,399	4,041	3,757	0.745	1.255	1.1668
Barabási–Albert	5,000	14,991	2,446	Karp–Sipser	1,821	3,071	2,465	0.744	1.256	1.0078
Sparse Erdős–Rényi	4,908	9,891	2,442	Karp–Sipser	2,072	2,812	2,598	0.848	1.152	1.0639
Watts–Strogatz	5,000	15,000	3,359	clique packing	2,498	4,220	3,540	0.744	1.256	1.0539

Table 13: The six graphs without a known optimum. Every one is certified, $|C|/L \le 2 - |M|/L$, that is $|C| \le 2L - |M|$, which proves $|C| \le 2\tau - |M| \le \frac{3}{2}\tau$ on that graph (Proposition 5.28 and Lemma 5.29).

All six are certified (Table 13), with margins between 0.04 and 0.25 below the threshold; the smallest margins occur on the random regular graphs, where the spectral bound is the only good lower bound.

7.12. Part J: the bipartite layers

Part J runs the ensemble on 325 small graphs with a known optimum: the 286 non-adaptive graphs of Part A and 39 hand-made non-bipartite graphs and mixtures on which the layers matter (odd cycles, wheels, complete graphs, the Petersen and dodecahedral graphs, circulants, Mycielski graphs, triangular and king's grids, chains of odd cycles, and bipartite graphs with a few odd edges). On the 113 bipartite graphs c_9 is a single layer, its phases end with a maximum matching, and it is optimal on all of them, as Proposition 5.21(iii) states. On the 212 non-bipartite graphs it uses two or three layers, never more than $\lfloor \log_2 m \rfloor + 1$, which is attained on the smallest graphs, and it is optimal on 203; its largest ratio is $3/2$, on a graph of the atlas with five vertices. It is never smaller than every other candidate on these graphs, smaller than c_2 on 36 and larger on 7, and its refinement attains the final size on 324 of the 325. The returned cover is optimal on all 325.

7.13. Part K: DIMACS clique complements, compared with Hvala

On the 68 DIMACS clique-complement instances of NPBench [37], whose optimum is $n - \omega$ with the clique numbers compiled by Mascia [38] (best-known values for `C500.9` and `C1000.9`), we compare the figures reported for Hvala [14] with one run of Salvador 0.1.6 at the default parameters. Table 14 summarises by family.

Family	Inst.	OPT	Hvala	Salvador	Wins	Opt. Hvala	Opt. Salvador	Max ratio H.	Max ratio S.
`brock`	12	5,317	5,407	5,370	12	0	0	1.033	1.025
`c-fat`	7	2,276	2,276	2,276	0	7	7	1.000	1.000
`C`	4	1,672	1,705	1,681	4	0	1	1.039	1.007
`gen`	5	1,306	1,403	1,314	5	0	3	1.124	1.026
`hamming`	6	1,956	1,964	1,958	1	5	5	1.008	1.002
`johnson`	4	672	672	672	0	4	4	1.000	1.000
`keller`	2	909	921	911	2	0	1	1.013	1.003
`MANN`	4	3,192	3,202	3,195	3	1	2	1.007	1.001
`p_hat`	10	5,151	5,187	5,152	9	1	9	1.015	1.001
`san`	10	2,573	2,751	2,621	10	0	4	1.192	1.051
`sanr`	4	1,106	1,115	1,107	4	0	3	1.013	1.006
All	**68**	**26,130**	**26,603**	**26,257**	**50**	**18**	**39**	**1.192**	**1.051**

Table 14: Part K: total cover sizes by family, number of instances on which Salvador is strictly smaller ("Wins"), optimal instances and largest ratio to the optimum, for Hvala and Salvador.

Salvador is never worse than Hvala, strictly better on 50 instances, optimal on 39 against 18, with mean ratio 1.0054 against 1.0253 and maximum 1.051 (`san200_0.9_3`, 164 against 156) against 1.192; the largest gains are on `san400_0.9_1` (300 against 348), `gen400_p0.9_75` (325 against 358), `san200_0.9_1` (130 against 155) and `gen400_p0.9_65` (335 against 359), all optimal; the total cover size is 26,257 against 26,603. It cannot be worse by construction (Theorem 5.2), since Hvala's candidates are among its own. The cumulative algorithm time is 233 seconds against 121, the slowest instance being `p_hat700-1` (23 seconds). The two sets of times come from separate runs, so the comparison is indicative only.

7.14. Running time

The script `car_scaling.py` times the whole call on eleven families at $n = 1{,}000$, 2,000, 4,000 and 8,000 and fits $\log t$ against $\log(n + m)$ (Table 15).

On all eleven families the fitted exponent is between 1.02 and 1.09 with $R^2 \geq 0.999$, every doubling multiplies the time by 1.89 to 2.18, and the time per element is 122 to 252 microseconds. At $n = 8{,}000$ the largest share of the time goes to the improvers: on the Erdős–Rényi family with average degree 16, 3.59 of the 10.46 seconds are spent in c_{17} and 2.78 in c_{15}. Four sizes over a factor of eight cannot prove an asymptotic exponent. On the small graphs of Part A the whole algorithm took 7.03 seconds for 442 evaluations, of which 5.50 seconds in the calls of the local search (the step c_{14} and the searches inside c_{15}) and 1.06 in c_{10}; the slowest call took 0.10 seconds.

8. Conclusion

Salvador 0.1.6 is a linear-time ensemble for Minimum Vertex Cover with seventeen strategies, the base strategies first and the improvers last. It always returns a valid cover (Theorem 4.1) of size at most $2\tau - r$ (Theorem 5.3), so

Family	Max n	Max m	Slope	R^2	μs/(n+m), first	last	Max doubling	Max time (s)
Random 3-regular	8,000	12,000	1.058	1.000	137	156	2.14	3.13
Random 6-regular	8,000	24,000	1.083	1.000	135	161	2.16	5.14
Random 12-regular	8,000	48,000	1.059	1.000	131	149	2.17	8.33
Erdős–Rényi, avg. deg. 4	7,861	15,934	1.048	0.999	139	153	2.14	3.63
Erdős–Rényi, avg. deg. 16	8,000	63,718	1.089	1.000	122	146	2.17	10.46
Barabási–Albert, $m = 3$	8,000	23,991	1.069	1.000	133	153	2.14	4.91
Watts–Strogatz, $k = 6$	8,000	24,000	1.052	1.000	127	144	2.16	4.60
2D grid	7,921	15,664	1.020	0.999	243	252	2.09	5.94
Random tree	8,000	7,999	1.062	1.000	207	236	2.16	3.78
Random bipartite, avg. deg. 6	7,992	24,000	1.089	1.000	208	251	2.18	8.04
Triangle-free process, avg. deg. 4	7,832	16,000	1.056	1.000	136	152	2.15	3.63

Table 15: Running time of Salvador on graphs of doubling size: least-squares slope of $\log t$ against $\log(n + m)$ and its R^2 (a linear algorithm has slope 1), time per element at the smallest and largest size, largest ratio $t(2n)/t(n)$, and time at the largest size.

its ratio is at most 1.999999999 on every graph with $\tau \le 10^9 r$, in particular on every graph with at most $10^9 + 1$ vertices, and every strategy runs in $O((n + m)\,\alpha(n))$ time (Theorem 6.1). The per-class theorems of Section 5.2 prove a minimum cover on forests, cycles, complete and complete bipartite graphs and every regular or degree-separated bipartite graph, on every bipartite graph on which the phases of c_9 reach a maximum matching, ratio $2 - 2/(d + 1)$ on d-regular graphs, $11/6$ on subcubic graphs, and a ratio below 1.999999999 on every graph outside a residue $\mathcal{R}$ of nearly Kőnig–Egerváry graphs with more than 10^9 vertices, on which the a-posteriori certificate of Theorem 5.24 is exact up to a factor $1 + 5.1 \cdot 10^{-10}$.

The guarantees come from different strategies on different graphs, and so does the quality measured in Section 7. The minimum-degree rules of c_5 and c_8 give the best independent covers on regular and random graphs, the bipartite solvers of c_9 and c_{10} and the reductions of c_{11} make the bipartite graphs exact, and the local searches and reductions of the improvers close most of the remaining gap on the Khot–Regev YES instances. On the 68 DIMACS clique complements the ensemble is never worse than Hvala and optimal on 39 instances against 18. The bound $|C| \le 2\tau - |M|$, which would give the ratio $3/2$, held for the returned cover on every instance with a known optimum on which it was tested and was certified on the six without one, while c_1 and c_2 alone violate it on small graphs.

We do not claim that Salvador is a $3/2$-approximation. The bound is a conjecture (Conjecture 5.30); the experiments concern graphs of at most 5,000 vertices in quick mode, most of those with a known optimum are small, and a proof would refute UGC unless P = NP, and would imply P = NP if the theorem reported in [12] holds (Theorem 5.32). What can be guaranteed today is the per-instance certificate: whenever $|C| \le \frac{3}{2}L$ for a computable lower bound L, the cover is proved to be within ratio $3/2$ on that graph. Future work includes sharper lower bounds, larger Khot–Regev YES-type instances, and an analysis of the layers of c_9 on non-bipartite graphs. The implementation is freely available as the `Salvador` package on PyPI [13].

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 Salvador, and used Claude Opus 5.5 and Claude Fable 5.2 to implement them by vibe coding and to improve the readability of this manuscript. 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/salvador` (release `v0.1.6`); the experiments of Section 7 are reproduced by running the scripts of `car/` with `--quick`.

References

[1] R. M. Karp, Reducibility Among Combinatorial Problems, in: 50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art, Springer, Berlin, Germany, 2010, pp. 219–241. doi:10.1007/978-3-540-68279-0_8.

[2] C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Courier Corporation, North Chelmsford (MA), 1998.

[3] G. Karakostas, A Better Approximation Ratio for the Vertex Cover Problem, ACM Transactions on Algorithms 5 (4) (2009) 1–8. doi:10.1145/1597036.1597045.

[4] M. Karpinski, A. Zelikovsky, Approximating Dense Cases of Covering Problems, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26, Providence, Rhode Island, 1996, pp. 147–164.

[5] I. Dinur, S. Safra, On the Hardness of Approximating Minimum Vertex Cover, Annals of Mathematics 162 (2005) 439–485. doi:10.4007/annals.2005.162.439.

[6] S. Khot, D. Minzer, M. Safra, Pseudorandom Sets in Grassmann Graph Have Near-Perfect Expansion, in: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science, Paris, France, 2018, pp. 592–601. doi:10.1109/FOCS.2018.00062.

[7] 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.

[8] 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.

[9] 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.

[10] E. Halperin, Improved Approximation Algorithms for the Vertex Cover Problem in Graphs and Hypergraphs, SIAM Journal on Computing 31 (5) (2002) 1608–1623. doi:10.1137/S0097539700381097.

[11] É. Bonnet, P. Rzążewski, An 11/6-approximation algorithm for vertex cover on string graphs, in: O. Aichholzer, H. Wang (Eds.), 41st International Symposium on Computational Geometry (SoCG 2025), Vol. 332 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, pp. 24:1–24:15. doi:10.4230/LIPIcs.SoCG.2025.24.

[12] 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 (Oct. 2026).
URL https://github.com/openai/math

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

[14] F. Vega, An Approximate Solution to the Minimum Vertex Cover Problem: The Hvala Algorithm, Gauge Freedom Journal 1 (1) (2026). doi:10.65323/gfj.2026.004.
URL https://www.gaugefreedom.org/article/v1-i1-004-the-hvala-algorithm/

[15] F. Vega, Hvala: Approximate Vertex Cover Solver, https://pypi.org/project/hvala (2026).

[16] F. Vega, An approximate solution to the minimum vertex cover problem: the Hallelujah algorithm, International Journal of Parallel, Emergent and Distributed Systems, published online 9 May 2026 (2026). doi:10.1080/17445760.2026.2660724.

[17] R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, in: Analysis and Design of Algorithms for Combinatorial Problems, Vol. 25 of Annals of Discrete Mathematics, Elsevier, 1985, pp. 27–46, http://www.cs.technion.ac.il/~reuven/PDF/vc_lr.pdf.

[18] B. S. Baker, Approximation algorithms for NP-complete problems on planar graphs, Journal of the ACM 41 (1) (1994) 153–180. doi:10.1145/174644.174650.

[19] 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.

[20] D. Konig, Graphen und Matrizen, Matematikai es Fizikai Lapok 38 (1931) 116–119.

[21] M. Yannakakis, F. Gavril, Edge Dominating Sets in Graphs, SIAM Journal on Applied Mathematics 38 (3) (1980) 364–372. doi:10.1137/0138030.

[22] J. Chen, I. A. Kanj, W. Jia, Vertex cover: Further observations and further improvements, Journal of Algorithms 41 (2) (2001) 280–301. doi:10.1006/jagm.2001.1186.

[23] L. Chang, W. Li, W. Zhang, Computing a near-maximum independent set in linear time by reducing-peeling, in: Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD), 2017, pp. 1181–1196. doi:10.1145/3035918.3035939.

[24] D. V. Andrade, M. G. C. Resende, R. F. Werneck, Fast local search for the maximum independent set problem, Journal of Heuristics 18 (4) (2012) 525–547. doi:10.1007/s10732-012-9196-4.

[25] S. Cai, K. Su, C. Luo, A. Sattar, NuMVC: An efficient local search algorithm for minimum vertex cover, Journal of Artificial Intelligence Research 46 (2013) 687–716. doi:10.1613/jair.3907.

[26] R. M. Karp, M. Sipser, Maximum matchings in sparse random graphs, in: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, 1981, pp. 364–375. doi:10.1109/SFCS.1981.21.
[27] Y. Caro, New Results on the Independence Number, Technical report, Tel Aviv University (1979).
[28] V. K. Wei, A Lower Bound on the Stability Number of a Simple Graph, Technical Memorandum 81-11217-9, Bell Laboratories (1981).
[29] P. Hall, On Representatives of Subsets, Journal of the London Mathematical Society s1-10 (1) (1935) 26–30. doi:10.1112/jlms/s1-10.37.26.
[30] A. J. Hoffman, On eigenvalues and colorings of graphs, in: B. Harris (Ed.), Graph Theory and its Applications, Academic Press, New York, 1970, pp. 79–91.
[31] D. S. Johnson, Approximation Algorithms for Combinatorial Problems, Journal of Computer and System Sciences 9 (3) (1974) 256–278. doi:10.1016/S0022-0000(74)80044-9.
[32] P. Alimonti, V. Kann, Some APX-completeness Results for Cubic Graphs, Theoretical Computer Science 237 (1–2) (2000) 123–134. doi:10.1016/S0304-3975(98)00158-3.
[33] 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.
[34] G. L. Nemhauser, L. E. Trotter, Vertex Packings: Structural Properties and Algorithms, Mathematical Programming 8 (1) (1975) 232–248. doi:10.1007/BF01580444.
[35] J. Friedman, A Proof of Alon's Second Eigenvalue Conjecture and Related Problems, Vol. 195 of Memoirs of the American Mathematical Society, American Mathematical Society, Providence, RI, 2008. doi:10.1090/memo/0910.
[36] A. M. Frieze, T. Łuczak, On the Independence and Chromatic Numbers of Random Regular Graphs, Journal of Combinatorial Theory, Series B 54 (1) (1992) 123–132. doi:10.1016/0095-8956(92)90070-E.
[37] T. Nguyen, T. Bui, NP-Complete Benchmark Instances, https://roars.dev/npbench/, vertex cover DIMACS clique complements (2025).
[38] F. Mascia, The Maximum Clique Problem – DIMACS Benchmark Set, https://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark, compiled clique-number values for DIMACS Second Implementation Challenge instances (2015).