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

Frank Vega [1]

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

RESEARCH Gauge Freedom Journal 1(1), Article 004 (2026)
https://doi.org/10.65323/gfj.2026.004

GAUGE FREEDOM JOURNAL

Correspondence: Frank Vega (vega.frank@gmail.com)
Received: 27 April 2026 / *Accepted:* 19 May 2026
Published online: 29 May 2026

Abstract

We present **Hvala**, a linear-time ensemble approximation algorithm for Minimum Vertex Cover that combines a maximal-matching 2-approximation, a bucket-queue maximum-degree greedy, and the degree-1 weighted-reduction Hallelujah heuristic of a companion work, applies redundant-vertex pruning to each, and returns the smallest of the four resulting covers. We prove unconditionally that Hvala runs in $\mathcal{O}(n+m)$ time and space and attains worst-case approximation ratio $\rho \leq 2$ on every finite simple graph; conditional on the strict pointwise inequality of the Hallelujah heuristic, the bound improves to $|S| < 2 \cdot \mathrm{OPT}(G)$ on every graph. Empirically we evaluate Hvala on three independent studies totalling 252 instances — the 109-instance NPBench collection, 130 real-world graphs from the Network Data Repository (up to $\approx$ 2.5M vertices and $\approx$ 15M edges), and 13 hand-crafted adversarial graphs (crown graphs, odd cycles, high-girth cubic and bipartite-expander graphs) — separating certified ratios from ratios-to-best-known. On the 173 instances with a published reference value, the mean approximation ratio is 1.015 and the maximum is 1.192 (a single Sanchis instance); every reported ratio falls below 1.414. The natural open question this benchmark raises — not a theorem proved here — is whether Hvala admits a uniform $\sqrt{2} - \epsilon$ bound on broad restricted graph classes (bounded degree, treewidth, clique number, expander-like, or power-law families), since an all-graphs sub-$\sqrt{2}$ bound would, by Khot–Minzer–Safra, contradict P $\neq$ NP. The algorithm is publicly available via PyPI as the `hvala` package.

Keywords— Vertex Cover; Approximation Algorithm; Linear-Time Algorithm; Ensemble Heuristic; Graph Optimization

1 Introduction

The Minimum Vertex Cover problem asks, for an undirected graph $G = (V, E)$, for the smallest subset $S \subseteq V$ such that every edge of G has at least one endpoint in S. It is one of Karp's original 21 NP-complete problems [1] and underlies applications in wireless-network design, computational biology, scheduling, and VLSI.

Because exact minimum vertex covers cannot be computed in polynomial time unless P = NP, the problem has driven decades of work on approximation algorithms. The classical 2-approximation obtained by taking both endpoints of every edge of a maximal matching is folklore [2]; LP-based refinements by Karakostas [3] and Karpinski and Zelikovsky [4] reach factor $2 - \Theta(1/\sqrt{\log n})$, which is $2 - o(1)$ but does not match a constant $2 - \epsilon$. We summarise the inapproximability landscape, naming the precise hypothesis on which each lower bound rests, since the assumptions involved differ substantially in strength. From the hardness side, Dinur and Safra [5] ruled out ratio below 1.3606 for any polynomial-time algorithm *assuming only* P $\neq$ NP. Khot, Minzer and Safra (the line of work [6, 7, 8], henceforth *KMS*) proved the $\sqrt{2} - \epsilon$ inapproximability lower bound for any polynomial-time algorithm; we follow the convention of stating this bound *assuming* P $\neq$ NP (which is how the KMS line of work is usually interpreted, via their proof of the 2-to-2 Games Conjecture with imperfect completeness). Khot and Regev [9] proved that no polynomial-time algorithm achieves ratio below $2 - \epsilon$ for any fixed $\epsilon > 0$ *assuming the Unique Games Conjecture (UGC)* of Khot [10], a strictly stronger assumption than P $\neq$ NP which is currently open. In short: a polynomial-time algorithm with uniform constant ratio $\rho < \sqrt{2}$ for every graph G would, by KMS, contradict P $\neq$ NP; an unconditional $2 - \epsilon$ constant is open in both directions (no current technique reaches it; ruling it out requires UGC). We are careful in what follows to distinguish UGC-based barriers from P $\neq$ NP barriers, and not to invoke SETH (which is a fine-grained-complexity assumption about k-SAT running time, not the relevant assumption for approximation-hardness of minimum vertex cover).

Scope and contribution. Against this backdrop, this paper is deliberately modest in its theoretical claims and stays within rigorously provable territory. The contributions are:

1. A linear-time ensemble algorithm (**Hvala**, Algorithm 1) that wraps three complementary linear-time heuristics — (i) a maximal-matching 2-approximation, (ii) a bucket-queue max-degree greedy, and (iii) the Hallelujah degree-1 weighted-reduction heuristic [11] — inside a redundant-vertex pruning step, and returns the smallest resulting cover.

2. A *rigorous proof* (Theorem 5.4) that Hvala achieves worst-case approximation ratio $\rho \leq 2$ on every finite simple graph. The proof hinges on the maximal-matching component and is self-contained.

3. A *conditional* strict pointwise inequality $|S| < 2 \cdot \mathrm{OPT}(G)$ on every finite simple graph (Corollary 5.6), conditional on the companion paper [11], which we import explicitly as Hypothesis 5.5 and do not reprove. The Hallelujah heuristic's approximation ratio is asymptotic to 2 — strictly less than 2 on each graph, with supremum equal to 2 — so no constant strictly smaller than 2 bounds it uniformly; under Hypothesis 5.5, the pointwise strict inequality on each graph is preserved by the minimum-selection and pruning steps of Hvala. A reader who chooses not to grant Hypothesis 5.5 still has the unconditional ≤ 2 bound of Theorem 5.4.

4. An empirical evaluation on three independent experimental studies totalling 252 instances: 109 structured hard instances from the NPBench benchmark collection [12] (41 FRB hard random graphs with planted optima and 68 DIMACS clique-complement graphs, of which 66 have a certified maximum clique and the remaining 2 have only a best-known lower bound on the

clique number); 130 real-world large graphs from the Network Data Repository [13] (biological, social, collaboration, web, infrastructure, and scientific-computing networks, reaching up to 2,523,386 vertices and 15,245,729 edges; a public reference cover size is available for 51 of these, of which 29 are certified optima on tree-like components and 22 are best-known approximate values); and 13 hand-crafted adversarial instances (`experiment/hardest/`) covering crown graphs $K_{k,k} \setminus M$, odd cycles, and high-girth cubic / bipartite-expander graphs (Petersen, Heawood, Möbius–Kantor, Pappus, Desargues, and random $(3,3)$-biregular bipartite), all with certified OPT values, designed to stress-test the individual Hvala components against their textbook failure modes. We report solution quality, running time, and a breakdown by graph family. Throughout we separate *certified approximation ratios* (computed against certified OPT) from *ratios to best-known reference values*.

The remainder of the paper is organised as follows. Section 3 describes the Hvala algorithm in detail. Section 4 establishes the linear-time complexity. Section 5 contains the approximation-ratio analysis (the rigorous, unconditional ≤ 2 bound and, conditional on Hypothesis 5.5 imported from [11], the strict pointwise < 2 inheritance). Section 6 reports three experimental studies: the NPBench structured-hard-instance benchmark (Section 6.1), a real-world large-graph benchmark drawn from the Network Data Repository (Section 6.2), and a small adversarial benchmark targeting the known failure modes of degree-based and matching-based heuristics (Section 6.3). Section 7 discusses the empirical–theoretical gap, hardness barriers, and the prospects of Hvala as a candidate for refined analysis below the $\sqrt{2}$ threshold on restricted graph classes (Section 7.3); Section 8 concludes.

2 Research Data and Implementation

To facilitate reproducibility and community adoption, we developed the open-source Python package *Hvala: Approximate Vertex Cover Solver*, available via the Python Package Index (PyPI) [14]. This implementation encapsulates the full ensemble algorithm — including the maximal-matching 2-approximation, the bucket-queue max-degree greedy, the Hallelujah degree-1 weighted-reduction subroutine, and the redundant-vertex pruning post-processing step — while guaranteeing an approximation ratio at most 2 unconditionally (with the additional pointwise strict inequality < 2 on every finite simple graph holding conditional on the companion theorem of [11], see Hypothesis 5.5 and Corollary 5.6) through rigorous validation. The package integrates seamlessly with NetworkX for graph handling and supports both unweighted and weighted instances. Code metadata, including versioning, licensing, and dependencies, is detailed in Table 1.

Reproducibility bundle. The companion GitHub repository ships, in addition to the Python package, a full reproducibility bundle for the 252-instance study reported in Section 6: (i) the exact 109 NPBench instances (`experiment/npbench/benchmarks_npbench/` for the 41 FRB and `experiment/npbench/clique_complement_npbench/` for the 68 DIMACS clique-complements), the 130 Network Data Repository instances (`experiment/network/`), and the 13 adversarial instances (`experiment/hardest/`); (ii) the raw per-instance `batch_idemo` log files placed next to each input directory and named after it (`benchmarks_npbench.txt`, `clique_complement_npbench.txt`, `network.txt`, `hardest.txt`), each containing the parse time, algorithm time, and returned cover size of every instance; (iii) the exact benchmark commands, hardware specification (Intel Core i7-1165G7 @ 2.80 GHz, 32 GB RAM, single-threaded Python 3.12 with NetworkX 3.4.2, numpy $\geq$ 2.2.1, scipy $\geq$ 1.15.0), and graph-source URLs (NPBench at `https://roars.dev/npbench/`, the Network Data Repository at `https://networkrepository.com/`); (iv) the per-candidate ablation script

underlying Tables 8 and 9; and (v) a table-regenerator that reads the raw logs and the ablation summary and emits the LaTeX body rows for Tables 3–9. The exact contents and commands are documented in the repository `README.md`.

Nr.	Code metadata description	Metadata
C1	Current code version	v0.1.2
C2	Permanent link to code/repository used for this code version	https://github.com/frankvegadelgado/hvala
C3	Permanent link to Reproducible Capsule	https://pypi.org/project/hvala/
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python
C7	Compilation requirements, operating environments & dependencies	Python $\geq$ 3.12, NetworkX $\geq$ 3.4.2

Table 1: Code metadata for the HVALA package.

3 The Hvala Algorithm

3.1 Overview

Given a simple undirected graph $G = (V, E)$, Hvala first performs trivial preprocessing and then computes four candidate vertex covers. We make the input conventions explicit, since both the reference DIMACS parser and the algorithm itself impose them.

Input conventions and preprocessing. The reference `batch_idemo` parser (`hvala/parser.py`) reads DIMACS-format edge files of the form `p edge` n m followed by edge lines `e` u v. Vertex labels u, v are required to be *positive integers*: the parser raises an error on any non-positive integer label and ignores any edge line whose last two tokens are not parsable as positive integers, so non-integer labels (e.g., strings or floats) are not a supported input format; an upstream relabelling to consecutive positive integers $1, 2, \ldots, n$ is required for such inputs. Once parsed, the algorithm (`hvala/algorithm.py`) immediately performs two preprocessing steps on the parsed graph: (i) *self-loops* of the form (v, v) are removed; we treat their presence as a parsing artefact and not as a request to cover a degenerate edge — so for the purposes of vertex covering, the input is forced to be a simple graph. (ii) *Isolated vertices* (no incident edges after self-loop removal) are removed; isolated vertices contribute 0 to OPT and never to a minimum cover, so removing them preserves OPT. (iii) The DIMACS reader treats duplicate edge lines $\{u, v\}$ and $\{v, u\}$ as the same edge, so multi-edges are silently deduplicated. After preprocessing, the remaining graph is finite, simple, undirected, and without self-loops; this is the graph G_0 on which Theorem 5.4 and Corollary 5.6 are stated.

Hvala then computes four candidate vertex covers:

C_1 — **Maximal-matching cover.** Compute a maximal matching M of G and let $C_1 = \bigcup_{(u,v)\in M} \{u, v\}$. This is the classical 2-approximation of [2].

C_2 — **Bucket-queue max-degree greedy.** Repeatedly select a vertex of maximum current degree into the cover, removing it and its incident edges, until no edges remain. Implemented in linear total time using a bucket queue indexed by degree.

C_3 — **Hallelujah degree-1 reduction.** Build an auxiliary graph G' by splitting every vertex u of degree k into k auxiliary copies $(u, 0), \ldots, (u, k-1)$, each connected to exactly one of u's neighbours, and assigning weight $1/k$ to every such auxiliary vertex. G' has maximum degree at most 1 on the auxiliary side, so a minimum weighted vertex cover on G' is obtained by picking, per edge of G', the endpoint of smaller weight (with lexicographic tie-breaking). Projecting the selected auxiliary vertices (u, i) back to their original u yields a valid cover of G [11].

$\tilde{C}_4$ — **Pruned union.** After the three base candidates C_1, C_2, C_3 have each been individually pruned to obtain $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$ (see below), compute the union $\tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3$ and apply redundant-vertex pruning (Algorithm 6) to that union *once* to obtain the fourth candidate $\tilde{C}_4$. This convention — pruning the union of the already-pruned candidates — matches the reference implementation [14], and we use it consistently throughout the manuscript.

The candidates C_1, C_2, C_3 are first *each* individually passed through redundant-vertex pruning to obtain $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$; the fourth candidate $\tilde{C}_4$ is then obtained by a final pruning of $\tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3$, and the algorithm returns the smallest among $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3, \tilde{C}_4$. Note that C_1 is included as a *worst-case safety net*: its value is guaranteed to be at most $2 \cdot \mathrm{OPT}$, and since the algorithm returns $\min(|\tilde{C}_1|, |\tilde{C}_2|, |\tilde{C}_3|, |\tilde{C}_4|) \le |\tilde{C}_1| \le |C_1|$, this guarantee propagates to the final output regardless of how C_2, C_3, and the pruned-union candidate behave.

3.2 Main Algorithm

Algorithm 1: Hvala: FindVertexCover(G)

Input: Simple undirected graph $G = (V, E)$
Output: Vertex cover $S \subseteq V$ satisfying $|S| \le 2 \cdot \mathrm{OPT}(G)$

Remove self-loops and isolated vertices from G;
if $E(G) = \emptyset$ **then**
  return $\emptyset$;
end
Build adjacency table adj$[v] = N_G(v)$ for every $v \in V$;
`// Three base heuristics (all linear time)`
$C_1 \leftarrow$ MaximalMatchingVC(G) ; `// Algorithm 2`
$C_2 \leftarrow$ BucketDegreeGreedy(adj) ; `// Algorithm 3`
$C_3 \leftarrow$ HallelujahReduction(G) ; `// Algorithm 4; [11]`
`// Individual pruning of every base candidate (overwrite in place)`
foreach $i \in \{1, 2, 3\}$ **do**
  $\tilde{C}_i \leftarrow$ PruneRedundant(adj, C_i);
end
`// Pruned union of the already-pruned candidates`
$\tilde{C}_4 \leftarrow$ PruneRedundant(adj, $\tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3$) ; `// Algorithm 6`
return $\arg\min_{i \in \{1,2,3,4\}} |\tilde{C}_i|$;

3.3 Subroutines

Algorithm 2: MaximalMatchingVC(G)

Input: Graph G
Output: Vertex cover C_1

$M \leftarrow$ a maximal matching of G;
$C_1 \leftarrow \emptyset$;
foreach $(u, v) \in M$ **do**
  $C_1 \leftarrow C_1 \cup \{u, v\}$;
end
return C_1;

Algorithm 3: BucketDegreeGreedy(adj)

Input: Adjacency table adj of a graph $G = (V, E)$
Output: Vertex cover C_2

$\deg[v] \leftarrow |\text{adj}[v]|$ for all $v \in V$;
$\Delta \leftarrow \max_v \deg[v]$;
Create buckets $B[0], B[1], \ldots, B[\Delta]$, double-ended queues;
foreach $v \in V$ **do**
  append v to $B[\deg[v]]$;
end
$C_2 \leftarrow \emptyset$; *removed* $\leftarrow \emptyset$;
for $d = \Delta$ ***down to*** 1 **do**
  while $B[d] \neq \emptyset$ **do**
    pop v from the front of $B[d]$;
    if $v \in$ *removed* ***or*** $\deg[v] \neq d$ **then**
      continue;
    end
    $C_2 \leftarrow C_2 \cup \{v\}$; *removed* $\leftarrow$ *removed* $\cup \{v\}$;
    foreach $u \in \text{adj}[v] \setminus$ *removed* **do**
      $\deg[u] \leftarrow \deg[u] - 1$;
      append u to $B[\deg[u]]$;
    end
  end
end
return C_2;

Algorithm 4: HallelujahReduction(G) — the degree-1 weighted reduction of [11]

Input: Graph $G = (V, E)$
Output: Vertex cover C_3

Build an auxiliary graph $G' = (V', E')$:;
foreach $u \in V$ *with degree* $k > 0$ *and neighbours* $v_0, v_1, \ldots, v_{k-1}$ **do**
  Add auxiliary vertex (u, i) and edge $\{(u, i), v_i\}$ to G' for every $i = 0, \ldots, k-1$;
  Set $w((u, i)) \leftarrow 1/k$;
end
$D_{\text{uw}} \leftarrow$ MinVCDegree1(G', uniform weights);
$D_{\text{w}} \leftarrow$ MinVCDegree1(G', w);
$S_{\text{uw}} \leftarrow \{u : (u, i) \in D_{\text{uw}} \text{ for some } i\} \cup (D_{\text{uw}} \cap V)$;
$S_{\text{w}} \leftarrow \{u : (u, i) \in D_{\text{w}} \text{ for some } i\} \cup (D_{\text{w}} \cap V)$;
return smaller of S_{uw} and S_{w};

Algorithm 5: MinVCDegree1(G', w) — exact weighted VC on a max-degree-1 graph

Input: Graph G' of maximum degree 1; weight w
Output: Minimum weighted vertex cover D

$D \leftarrow \emptyset$; *visited* $\leftarrow \emptyset$;
foreach $v \in V(G')$ *with* $v \notin$ *visited* **do**
  if $\deg(v) = 1$ **then**
    $u \leftarrow$ unique neighbour of v;
    if $u \notin$ *visited* **then**
      if $w(v) < w(u)$ ***or*** *(*$w(v) = w(u)$ ***and*** $v < u$*)* **then**
        $D \leftarrow D \cup \{v\}$;
      else
        $D \leftarrow D \cup \{u\}$;
      end
      visited $\leftarrow$ *visited* $\cup \{v, u\}$;
    end
  end
end
return D;

Algorithm 6: PruneRedundant(adj, C) — linear-time redundant-vertex pruning

Input: Adjacency table adj of G and a vertex cover C of G
Output: A vertex cover $C' \subseteq C$ of G

$L \leftarrow$ a fixed list copy of C (iteration order is immaterial for the proof);
foreach $v \in L$ **do**
  if *every* $u \in \text{adj}[v]$ *is currently in* C **then**
    $C \leftarrow C \setminus \{v\}$;
  end
end
return C;

4 Complexity Analysis

Theorem 4.1 (Linear-time and linear-space)**.** *Hvala runs in $\mathcal{O}(n+m)$ time and $\mathcal{O}(n+m)$ space, where $n = |V|$ and $m = |E|$.*

Proof. Preprocessing and construction of the adjacency table are $\mathcal{O}(n+m)$.

Maximal matching can be computed in $\mathcal{O}(n+m)$ by the greedy linear-time procedure that scans edges once and adds every edge both of whose endpoints are still unmatched. Building C_1 from M is $\mathcal{O}(|M|) \leq \mathcal{O}(m)$.

Bucket-queue max-degree greedy. Each vertex is inserted into a bucket at most once per decrement of its degree; across the whole execution the total number of bucket insertions is bounded by $\sum_v \deg(v) = 2m$, and each insertion/removal is $\mathcal{O}(1)$. The outer loop over d performs $\mathcal{O}(\Delta) \leq \mathcal{O}(n)$ constant-time bucket checks. Total time: $\mathcal{O}(n+m)$.

Hallelujah reduction. The auxiliary graph G' has exactly $2m$ vertices and m edges (one edge per edge of G, with auxiliary vertices added on both endpoints when both are of positive degree). MinVCDegree1 visits every vertex once and its single neighbour once, hence runs in $\mathcal{O}(|V(G')| + |E(G')|) = \mathcal{O}(n+m)$.

Pruning. For each $v \in C$, checking whether all neighbours are in C is $\mathcal{O}(\deg(v))$; summed over all $v \in C \subseteq V$ the total work is at most $\sum_{v \in V} \deg(v) = 2m$. Each pruning call is therefore $\mathcal{O}(n+m)$, and the algorithm performs a constant number of pruning calls.

Space is dominated by the adjacency table and the auxiliary graph G', both $\mathcal{O}(n+m)$. □

5 Approximation Ratio Analysis

We now establish the worst-case approximation guarantees of Hvala, in two stages. First, a self-contained, *unconditional* proof that Hvala always returns a cover of size at most $2 \cdot \mathrm{OPT}$ (Theorem 5.4); this is the baseline guarantee and depends on no external result. Second, an inheritance argument (Corollary 5.6) showing that, *conditional on Hypothesis 5.5 imported from the companion paper [11]*, Hvala satisfies the strict pointwise inequality $|S| < 2 \cdot \mathrm{OPT}(G)$ on every finite simple graph G, mirroring the analogous property proved for the Hallelujah heuristic in [11]. Both statements are needed: Theorem 5.4 gives the absolute ≤ 2 bound unconditionally, while Corollary 5.6 records that, under the imported hypothesis, the inequality is in fact strict on each graph. We are careful not to confuse the conditional pointwise strict inequality with a uniform improvement: as discussed in Remark 5.7 below, the value of $\sup_G |S|/\mathrm{OPT}(G)$ is not determined by the present analysis, and we record it as an open problem.

Throughout this section, $G = (V, E)$ is a finite simple undirected graph without self-loops, and $\mathrm{OPT}(G)$ denotes the size of a minimum vertex cover of G. Isolated vertices contribute 0 to OPT, so removing them (as the algorithm does in preprocessing) leaves OPT unchanged.

5.1 A Lemma about Redundant-Vertex Pruning

Lemma 5.1 (Pruning preserves validity and never increases size)**.** *Let C be a vertex cover of G and let $C' = \textsc{PruneRedundant}(\mathrm{adj}, C)$. Then $C' \subseteq C$ and C' is also a vertex cover of G.*

Proof. That $C' \subseteq C$ is clear from the procedure (it only ever removes elements).

We prove by induction on the iteration count that the invariant "C is a vertex cover of G" holds throughout PruneRedundant.

Base case. At the start, C is a vertex cover of G by hypothesis.

Inductive step. Suppose the invariant holds just before iteration i, at which we are considering vertex $v \in L$. Two cases.

Case 1: v is not removed at iteration i. Then C is unchanged, and the invariant is preserved trivially.

Case 2: v is removed at iteration i. The removal condition is that every neighbour u of v in G is currently in C. Consider any edge $e = (x, y) \in E$:

- If e is not incident to v, neither of its endpoints is touched; the inductive hypothesis says some endpoint of e was in C before iteration i, and both endpoints remain unaffected, so the property survives.
- If $e = (v, u)$ is incident to v, then by the removal condition, u is in C just before v is removed, and u is not the vertex being removed, so u remains in C after the removal. Hence e is still covered by u.

Thus the invariant is preserved.

Since the loop terminates after a finite number of iterations, the invariant holds at the end, and C' is a vertex cover of G. □

It is instructive, though not logically necessary for what follows, to note the following strengthening: once a vertex v is removed, no neighbour of v can subsequently be removed, because the "all neighbours currently in C" test would fail (with v itself missing from C). In particular, after PruneRedundant, for every edge (v, u), at most one of v, u has been removed. This reinforces Lemma 5.1.

5.2 Validity of the Base Candidate Covers

Before turning to the worst-case ratio analysis we record two short, self-contained lemmas establishing that the second and third base candidates — C_2, produced by BucketDegreeGreedy (Algorithm 3), and C_3, produced by HallelujahReduction (Algorithm 4) — are vertex covers of the input graph. These lemmas are used in the proof of Theorem 5.4 in Section 5.3 below. (The maximal-matching candidate C_1 produced by Algorithm 2 is a folklore vertex cover: it contains, for every edge of a maximal matching M, both endpoints, and every edge of G must share an endpoint with some edge of M by maximality of M.)

Lemma 5.2 (Validity of C_2: bucket-queue max-degree greedy)**.** *Let $G = (V, E)$ be a finite simple undirected graph. The cover $C_2 =$ BucketDegreeGreedy(adj) produced by Algorithm 3 is a vertex cover of G.*

Proof. We use the following two state variables maintained by Algorithm 3: the set *removed* $\subseteq V$ of vertices that have been added to C_2, and, for every $v \in V$, the integer $\deg[v]$, which the algorithm updates to equal $|\{u \in \mathrm{adj}[v] : u \notin \textit{removed}\}|$ throughout execution (this is the standard correctness invariant of the bucket-queue greedy implementation, maintained by the bookkeeping in the body of the inner loop).

We claim that, at termination of Algorithm 3, every vertex $v \in V$ satisfies the disjunction

$$v \in C_2 \quad \text{or} \quad \deg[v] = 0 \quad \text{(equivalently, every neighbour of } v \text{ in } G \text{ lies in } C_2). \tag{$\star$}$$

To see that ($\star$) holds, fix any $v \in V$ and consider the outer loop variable d when it reaches the value $\deg[v]$ at termination (note that $\deg[v]$ is monotone non-increasing, so the value of $\deg[v]$ at

the moment its bucket is examined is well-defined and an upper bound on its terminal value). Two cases arise depending on whether v is ever added to C_2.

Case 1: v is added to C_2 at some iteration. Then $v \in C_2$ and $(\star)$ is satisfied by the first disjunct.

Case 2: v is never added to C_2. Then $v \notin$ *removed* throughout execution. Recall that v is enqueued into bucket $B[d_0]$ at initialisation (where d_0 is its original degree) and, every time a neighbour of v is added to *removed*, $\deg[v]$ is decremented and v is re-enqueued into bucket $B[\deg[v]]$ at the new (lower) value. The outer loop iterates $d = \Delta, \Delta - 1, \ldots, 1$. Since the iteration order is from high d down to 1, and since v has been enqueued into each bucket it has ever been associated with, the algorithm will examine v at the moment its current $\deg[v]$ equals the loop variable d for some $d \geq 1$ *unless* all such enqueueings happen to satisfy $\deg[v] \neq d$ at the time of dequeue, which by the bookkeeping rule occurs exactly when $\deg[v]$ has dropped below the bucket index between enqueue and dequeue (the standard "stale-entry skip" line $\deg[v] \neq d$). The remaining possibility is that $\deg[v]$ reaches 0, in which case v is enqueued into bucket $B[0]$, but the outer loop never reaches $d = 0$ (it stops at $d = 1$). In either sub-case — stale-skip at every positive d, or reaching $\deg[v] = 0$ before being examined at a positive d — the terminal value of $\deg[v]$ is necessarily 0, since by the maintained invariant $\deg[v]$ counts non-removed neighbours of v, and any uncovered neighbour would have forced $\deg[v] \geq 1$ to persist through some positive bucket index where v's own dequeue would have triggered its insertion into C_2 — contradicting the hypothesis of Case 2. Thus $\deg[v] = 0$ at termination, i.e. every neighbour of v in G has been added to C_2, so $(\star)$ is satisfied by the second disjunct.

Property $(\star)$ implies that C_2 is a vertex cover: for any edge $(u, w) \in E$, applying $(\star)$ to u shows that either $u \in C_2$ or every neighbour of u — in particular w — lies in C_2. Hence at least one of u, w lies in C_2, so the edge is covered. Since this holds for every edge of G, C_2 is a vertex cover of G. □

Lemma 5.3 (Validity of C_3: Hallelujah degree-1 reduction)**.** *Let $G = (V, E)$ be a finite simple undirected graph. The cover $C_3 = \textsc{HallelujahReduction}(G)$ produced by Algorithm 4 is a vertex cover of G.*

Proof. We first establish that the auxiliary graph $G' = (V', E')$ constructed by Algorithm 4 has the following two properties: *(P1)* every auxiliary vertex $(u, i) \in V'$ has degree exactly 1 in G'; and *(P2)* for every original edge $(u, v) \in E$, there is an edge in G' of the form $\{(u, i), (v, j)\}$ for some indices i, j.

Proof of (P1). By construction, the outer `foreach` loop of Algorithm 4 processes every $u \in V$ of positive original degree k_u, creating k_u auxiliary vertices $(u, 0), \ldots, (u, k_u - 1)$ and adding, for each $i \in \{0, \ldots, k_u - 1\}$, the single edge $\{(u, i), v_i\}$ to G', where v_i is the i-th neighbour of u at the time u is processed (i.e. the i-th neighbour of u that has not yet been replaced by its own auxiliary copies). When such a v_i is later processed by the loop, the original v_i is removed from G' and the edge $\{(u, i), v_i\}$ is replaced by an edge $\{(u, i), (v_i, j)\}$ for some j (specifically, the index j such that (u, i) is the j-th neighbour of v_i at the time v_i is processed). In particular, throughout the construction, the auxiliary vertex (u, i) has exactly one incident edge: either $\{(u, i), v_i\}$ (immediately after u is processed but before v_i is) or $\{(u, i), (v_i, j)\}$ (after v_i is also processed). Hence at termination of the construction, every auxiliary vertex has degree exactly 1.

Proof of (P2). Fix any original edge $(u, v) \in E$, and assume without loss of generality that u is processed before v. When u is processed, v is still an unreplaced original vertex, so it appears as some v_i in u's neighbour list at that moment, and the edge $\{(u, i), v\}$ is added to G'. When v is later processed, the original vertex v is removed, and for every current neighbour w of v — including (u, i) — the edge $\{w, v\}$ is replaced by an edge $\{w, (v, j)\}$ for some index j. In particular, the edge $\{(u, i), v\}$ is replaced by $\{(u, i), (v, j)\}$, which lies in G' at termination.

Now consider Algorithm 5 applied to G' (called by Algorithm 4 as MinVCDegree1). On a graph G' in which every vertex has degree at most 1 — such as the one constructed above, by (P1) — the edge set of G' decomposes into a disjoint set of edges (a matching), plus possibly some isolated vertices. Algorithm 5 scans every vertex $v \in V(G')$ once and, when $\deg(v) = 1$ and neither v nor its unique neighbour u has yet been visited, adds exactly one of v, u to the output set D (the one with smaller weight, breaking ties lexicographically). Both v and u are then marked as visited and never revisited. Hence for every edge $e \in E(G')$, at least one endpoint of e is in D: D is a vertex cover of G'.

Finally, Algorithm 4 returns S_{w} or S_{uw} (whichever is smaller), where each is obtained from the corresponding $D \subseteq V(G')$ by the map-back rule

$$S \;:=\; \{u \in V \,:\, (u, i) \in D \text{ for some } i\} \cup (D \cap V).$$

We claim that for the choice of D used in either branch, the resulting S is a vertex cover of G. Fix any edge $(u, v) \in E$. By (P2), G' contains some edge $\{(u, i), (v, j)\}$. Since D is a vertex cover of G', at least one of (u, i) and (v, j) lies in D. By the map-back rule, the original endpoint corresponding to the chosen auxiliary — u or v — lies in S. Hence the edge (u, v) of G is covered by S. Since this holds for every edge of G, both S_{uw} and S_{w} are vertex covers of G, and so is $C_3 = \textsc{HallelujahReduction}(G)$, defined as the smaller of the two. □

5.3 The Rigorous $\rho \leq 2$ Bound

Theorem 5.4 (Worst-case 2-approximation)**.** *For every finite simple undirected graph G, the output S of FindVertexCover(G) (Algorithm 1) is a vertex cover of G satisfying*

$$|S| \;\leq\; 2 \cdot \mathrm{OPT}(G).$$

Proof. Let G_0 be the graph after preprocessing (self-loops and isolated vertices removed). As noted above, $\mathrm{OPT}(G_0) = \mathrm{OPT}(G)$, and any vertex cover of G_0 is a vertex cover of G. We work with G_0 in what follows.

Step 1: C_1 is a vertex cover of G_0 of size at most $2 \cdot \mathrm{OPT}(G_0)$.

Let M be the maximal matching computed in Algorithm 2. Since M is maximal, every edge $e \in E(G_0)$ shares a vertex with some edge of M — otherwise $M \cup \{e\}$ would be a larger matching, contradicting maximality. Therefore, at least one endpoint of e lies in $C_1 = \bigcup_{(u,v)\in M} \{u, v\}$, i.e. C_1 is a vertex cover of G_0.

Let C^* be any minimum vertex cover of G_0, so $|C^*| = \mathrm{OPT}(G_0)$. Since the edges of M are pairwise vertex-disjoint, and each of these edges must be covered by C^*, distinct edges of M contribute distinct vertices to C^* (one endpoint each). Hence $|C^*| \geq |M|$. Consequently,

$$|C_1| \;=\; 2|M| \;\leq\; 2|C^*| \;=\; 2 \cdot \mathrm{OPT}(G_0).$$

Step 2: Pruning C_1 does not increase its size.

By Lemma 5.1 applied to C_1, the pruned $\tilde{C}_1 = \textsc{PruneRedundant}(\mathrm{adj}, C_1)$ is still a vertex cover of G_0, and $\tilde{C}_1 \subseteq C_1$ implies $|\tilde{C}_1| \leq |C_1|$. Combining with Step 1:

$$|\tilde{C}_1| \;\leq\; |C_1| \;\leq\; 2 \cdot \mathrm{OPT}(G_0).$$

Step 3: The output S satisfies $|S| \leq |\tilde{C}_1|$.

By construction, the algorithm returns $S = \arg\min_{i\in\{1,2,3,4\}} |\tilde{C}_i|$. Hence $|S| \leq |\tilde{C}_1|$, provided $\tilde{C}_1$ is a vertex cover so that the minimum is well-defined over valid covers — and it is, by Step 2.

The other three candidates $\tilde{C}_2, \tilde{C}_3, \tilde{C}_4$ are also valid covers: $\tilde{C}_2$ is obtained by Lemma 5.1 applied to C_2, which is a vertex cover by Lemma 5.2; $\tilde{C}_3$ is obtained by Lemma 5.1 applied to C_3, which is a vertex cover by Lemma 5.3; and $\tilde{C}_4$ is obtained by Lemma 5.1 applied to the union $\tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3$, which is a vertex cover as a superset of the (already valid) vertex cover $\tilde{C}_1$.

Combining Steps 1–3, $|S| \leq 2 \cdot \mathrm{OPT}(G_0) = 2 \cdot \mathrm{OPT}(G)$. □

The constant 2 in Theorem 5.4 is a *uniform worst-case* bound on $|S|/\mathrm{OPT}(G)$ that holds unconditionally over all finite simple graphs. Achieving a strictly smaller *uniform constant* $2 - \epsilon$ with a simple combinatorial algorithm is a well-known open problem, because an unconditional constant $2 - \epsilon$ would improve over the best known $2 - \Theta(1/\sqrt{\log n})$ [3] and is UGC-hard [9]. We do not claim such an improvement here. What we do obtain, conditional on a theorem of the companion paper [11] which we will formalise as Hypothesis 5.5 in Section 5.4 below, is a weaker but non-trivial statement: the inequality $|S| \leq 2 \cdot \mathrm{OPT}(G)$ is in fact *strict* on every particular graph.

5.4 Inheritance of the Pointwise Strict Inequality from Hallelujah (Conditional on the Companion Theorem)

The strict pointwise inequality stated in this subsection is *not* proved in the present paper. It is imported, as an external dependency, from the companion paper [11]. To make the dependency explicit and auditable, we record the imported statement as a named hypothesis and clearly mark every result that relies on it as conditional on this hypothesis. The unconditional uniform bound $|S| \leq 2 \cdot \mathrm{OPT}(G)$ of Theorem 5.4 is not affected by this dependency: a reader who chooses not to accept Hypothesis 5.5 below still obtains the full ≤ 2 guarantee with a self-contained proof.

Hypothesis 5.5 (Hallelujah Pointwise Strict Inequality (HPSI), imported from [11])**.** *For every finite simple undirected graph G, the degree-1 weighted-reduction heuristic of [11] (our component $C_3 = \textsc{HallelujahReduction}(G)$) returns a vertex cover satisfying*

$$|C_3| \;<\; 2 \cdot \mathrm{OPT}(G).$$

That is, the inequality is strict on every particular graph. The proof of this statement is given in [11] and is not reproduced here.

We adopt Hypothesis 5.5 only as a named import from the companion paper. The companion paper additionally establishes that the supremum of $|C_3|/\mathrm{OPT}(G)$ over all finite simple graphs equals 2 — the Hallelujah ratio is *asymptotic* to 2, that is, for every $\epsilon > 0$ there exists a graph G_ϵ on which $|C_3|/\mathrm{OPT}(G_\epsilon) > 2 - \epsilon$ — but we use only the pointwise strict inequality of Hypothesis 5.5 below. Readers are referred to [11] for the full proofs.

Corollary 5.6 (Strict pointwise inequality for Hvala, conditional on Hypothesis 5.5)**.** Conditional on Hypothesis 5.5*, for every finite simple undirected graph G, the output S of Algorithm 1 satisfies*

$$|S| \;<\; 2 \cdot \mathrm{OPT}(G).$$

In particular, conditional on Hypothesis 5.5, the inequality $|S| \leq 2 \cdot \mathrm{OPT}(G)$ of Theorem 5.4 is strict on every finite simple graph. The corollary does not assert any uniform constant strictly smaller than 2: we leave open whether $\sup_G |S|/\mathrm{OPT}(G)$ equals 2 or is strictly smaller.

Proof. Assume Hypothesis 5.5. Let $\tilde{C}_3 = \textsc{PruneRedundant}(\mathrm{adj}, C_3)$. By Lemma 5.1, $\tilde{C}_3$ is a vertex cover of G with $|\tilde{C}_3| \leq |C_3|$. By Hypothesis 5.5, $|C_3| < 2 \cdot \mathrm{OPT}(G)$. Hence $|\tilde{C}_3| < 2 \cdot \mathrm{OPT}(G)$. Since $S = \arg\min_{i \in \{1,2,3,4\}} |\tilde{C}_i|$, we have $|S| \leq |\tilde{C}_3| < 2 \cdot \mathrm{OPT}(G)$. The conclusion holds whenever Hypothesis 5.5 does, and the corollary makes no claim in the absence of that hypothesis. □

Remark 5.7 (On the supremum of the Hvala ratio)**.** A natural follow-up question is whether the strict pointwise inequality of Corollary 5.6 extends to a uniform constant $\rho < 2$ such that $|S| \leq \rho \cdot \mathrm{OPT}(G)$ for every G. We emphasise that we do *not* establish such a bound, nor do we establish the matching statement $\sup_G |S|/\mathrm{OPT}(G) = 2$. The companion paper [11] exhibits a family of graphs G_ϵ on which $|C_3|/\mathrm{OPT}(G_\epsilon) > 2 - \epsilon$, so the Hallelujah candidate alone is asymptotic to 2; however, this does *not* imply that the minimum-selection output $|S| = \min_i |\tilde{C}_i|$ is also asymptotic to 2 on the same family, because on G_ϵ another candidate (typically $\tilde{C}_1$, $\tilde{C}_2$, or $\tilde{C}_4$) may achieve a strictly smaller ratio. Constructing an explicit family G_k on which *every* candidate $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3, \tilde{C}_4$ simultaneously approaches ratio 2 is, to the best of our knowledge, open: it would in particular require defeating the maximal-matching candidate $\tilde{C}_1$, the bucket-queue max-degree greedy $\tilde{C}_2$, and the union-pruning candidate $\tilde{C}_4$ on a common family. We therefore record the supremum of the Hvala ratio as an open problem and confine our rigorous claims to (i) the unconditional uniform ≤ 2 bound of Theorem 5.4 and (ii) the pointwise strict < 2 inequality of Corollary 5.6, which is itself conditional on Hypothesis 5.5.

Two remarks clarify the interplay between Theorem 5.4 and Corollary 5.6.

First, Theorem 5.4 is *not* rendered redundant by Corollary 5.6. The theorem gives a uniform, self-contained, unconditional, non-strict ≤ 2 bound whose proof does not depend on [11] or on Hypothesis 5.5 at all. The corollary strengthens this to a strict inequality on each particular graph, but only *conditionally*: its proof relies entirely on Hypothesis 5.5, imported from the companion paper's analysis of the Hallelujah reduction [11] and not reproved in this paper. Readers who wish to audit Hvala against only the simplest, self-contained assumptions therefore retain the full ≤ 2 guarantee independently of any companion result.

Second, the strict inequality in Corollary 5.6 is pointwise only: it does not provide a uniform constant $2 - \epsilon$, and no such constant is known for either the Hallelujah component or Hvala. Establishing a uniform $2 - \epsilon$ constant for any simple polynomial-time algorithm would improve over [3] and, under the Unique Games Conjecture, is not possible [9, 10]. For the Hallelujah component alone, the statement "the ratio tends to 2" holds in the precise sense proved in [11]: on every finite graph $|C_3|/\mathrm{OPT}(G)$ is strictly below 2, but by choosing progressively harder graphs one can make it arbitrarily close to 2. Whether the analogous asymptotic statement holds for the Hvala output $|S|/\mathrm{OPT}(G)$ — equivalently, whether $\sup_G |S|/\mathrm{OPT}(G) = 2$ — is left open, as discussed in Remark 5.7.

5.5 Other Candidates

We have used C_1 and C_3 in the proofs; the roles of C_2 and $\tilde{C}_4$ are complementary rather than load-bearing:

- C_2 (bucket-queue max-degree greedy) has no general worst-case ratio better than $\Theta(\log \Delta)$ (Johnson's classical bound), but it is very strong on near-regular and clique-like graphs and is included because, on those families, it is frequently optimal or near-optimal. Its presence in the minimum cannot worsen the bound.
- $\tilde{C}_4$ is the pruning of the union of the *already-pruned* candidates, $\tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3$, matching the reference implementation [14]. Because $|\tilde{C}_4| \leq |\tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3|$ may be larger or smaller than each individual $|\tilde{C}_i|$ for $i \in \{1, 2, 3\}$, its role is best understood as occasionally exploiting structural overlaps between the three pruned base heuristics that per-candidate pruning alone cannot resolve.

Neither C_2 nor $\tilde{C}_4$ is required for the bounds of Theorem 5.4 or Corollary 5.6.

6 Experimental Validation

We evaluate Hvala on three independent experimental studies totalling 252 instances. Section 6.1 reports results on 109 structured hard instances of the public NPBench collection [12]; Section 6.2 reports results on 130 real-world large graphs drawn from the Network Data Repository [13], reaching millions of vertices; and Section 6.3 reports results on 13 hand-crafted adversarial instances (`experiment/hardest/`) targeting the textbook failure modes of degree- and matching-based heuristics. Throughout all three sections we carefully distinguish *certified approximation ratios*, defined as $|S|/\mathrm{OPT}(G)$ where $\mathrm{OPT}(G)$ is a mathematically certified minimum vertex cover size, from *ratios to a best-known reference value*, defined as $|S|/r(G)$ where $r(G)$ is a published best-known cover size whose optimality is not certified. All three studies use the same implementation of Algorithm 1 on commodity hardware (Intel Core i7-1165G7 at 2.80 GHz, 32 GB RAM, single-threaded Python 3.12 with NetworkX 3.4.2).

Timing reporting and confidence.

Every per-instance time reported in the result tables of Sections 6.1–6.3 is the algorithm wall-clock time of a *single* `batch_idemo` run, extracted verbatim from the raw log files shipped with the reproducibility bundle (`benchmarks_npbench.txt`, `clique_complement_npbench.txt`, `network.txt`, and `hardest.txt`). To give a per-benchmark sense of the spread of those per-instance times without committing to a multi-trial repeat (which would not change the cover sizes and is dominated by the same Python/NetworkX overheads on every run), Table 2 reports the median, interquartile range (IQR), mean, standard deviation, minimum, and maximum of the algorithm-time-per-instance distribution within each benchmark. We emphasise that this is across-instance spread, not within-instance repeated-trial spread; it gives a confidence interval on the *typical* per-instance time of a given benchmark family rather than on a single instance.

Table 2: Per-benchmark distribution of single-run algorithm time per instance, in milliseconds (and cumulative total in seconds), extracted from the four `batch_idemo` log files shipped with the reproducibility bundle. The four benchmarks are NPBench-FRB (Section 6.1), NPBench-DIMACS clique-complement (Section 6.1), Real-world (Section 6.2), and Adversarial (Section 6.3). Q1 and Q3 are the first and third quartiles; IQR = Q3−Q1.

Benchmark	**n**	**median** (ms)	**IQR** (ms)	**mean** (ms)	**stddev** (ms)	**min** (ms)	**max** (ms)	**total** (s)
FRB	41	2,947.03	2,930.81	2,868.12	2,740.86	217.19	16,836.65	117.59
DIMACS	68	475.80	2,366.12	1,785.12	2,679.40	2.94	14,283.61	121.39
Real-world	130	2,524.19	44,024.64	44,094.24	96,928.26	0.00	756,988.20	5,732.25
Adversarial	13	1.01	5.63	11.08	26.98	0.00	98.03	0.14
Combined (all 252)	252	992.78	6,450.80	23,695.94	72,643.97	0.00	756,988.20	5,971.38

Two observations on Table 2 guide the interpretation of the timing numbers in the per-instance tables below. First, the algorithm time has a strongly right-skewed distribution within every benchmark — on the real-world study, the median is ≈ 2.5 s but the mean is ≈ 44 s and the maximum is ≈ 757 s (12.62 min on `ca-coauthors-dblp`). The bulk of cumulative time is therefore concentrated in a small number of very-large instances, and a single rerun of the largest one or two instances would dominate any IQR estimate computed across re-runs of the whole benchmark. Second, on a per-benchmark scale, the median-to-Q3 spread is ≈ 1 order of magnitude on each of NPBench-DIMACS, real-world, and adversarial, and $\approx 2.5\times$ on NPBench-FRB; we report individual

per-instance times to two-decimal-place ms or two-decimal-place seconds (as in the existing result tables), so the practical precision of our timing claims is bounded both by the per-benchmark variability summarised here and by the typical 5–15% wall-clock noise of single-threaded Python on Windows.

6.1 Experiment 1: Structured Hard Instances (NPBench)

6.1.1 Setup

We evaluate Hvala on 109 vertex-cover instances of the public NPBench collection [12], comprising two families for which a reference cover size is publicly available. We classify the 109 instances into two reporting categories that we keep *separate* when summarising performance:

1. **41 FRB hard instances** (NPBench Section "Vertex Cover instances", originally from Ke Xu's benchmark repository), with known minimum vertex cover sizes ranging from 420 to 3900. We treat the FRB reference values as *mathematically certified optima* as distributed by NPBench, since these instances are constructed with the planted optimum recorded by the benchmark provider; the resulting ratios are therefore *certified approximation ratios*.

2. **68 DIMACS clique-complement instances** (NPBench Section "Clique complement graphs"), constructed as the complements of the DIMACS Second Implementation Challenge maximum-clique instances. The optimum vertex cover of the complement equals $n - \omega(G)$, where $\omega(G)$ is the maximum clique size of the original graph; we use the maximum-clique values compiled on Mascia's DIMACS benchmark page [15]. For 66 of these 68 instances, $\omega(G)$ is a *certified* value, so the reported ratio is a certified approximation ratio. For the two instances `C500.9` and `C1000.9`, in contrast, the clique number is only known to be a best-known lower bound ($\omega \geq 57$ and $\omega \geq 68$ respectively); the values shown are therefore the best-known upper bounds on OPT and the reported ratio is a *ratio to a best-known reference value* (and is a lower bound on the true approximation ratio). These two instances are marked with † in the result tables.

In summary, of the 109 NPBench instances, 107 yield a certified approximation ratio (41 FRB + 66 DIMACS with certified ω) and 2 yield a ratio to a best-known reference value (the †-flagged DIMACS instances). Cumulative solve time over all 109 instances is 238.98 seconds, broken down as 117.59 s on the 41 FRB instances and 121.39 s on the 68 DIMACS clique-complements, as recorded in the raw `batch_idemo` log files shipped with the reproducibility bundle (see Section 2).

6.1.2 Results

Table 3 reports, for every FRB instance, the known optimum, the cover size produced by Hvala, the wall-clock solve time, and the approximation ratio. Tables 4 and 5 report the same quantities for the DIMACS clique-complement instances, split alphabetically into two halves so that each fits on a standard page.

Table 3: Hvala on the 41 FRB vertex-cover instances of NPBench [12].

Instance	Known OPT	Hvala size	Time	Ratio
frb30-15-1	420	428	223.8ms	1.019
frb30-15-2	420	429	232.4ms	1.021
frb30-15-3	420	427	217.2ms	1.017
frb30-15-4	420	429	219.6ms	1.021
frb30-15-5	420	427	223.6ms	1.017
frb35-17-1	560	569	1.75s	1.016
frb35-17-2	560	570	414.5ms	1.018
frb35-17-3	560	568	466.5ms	1.014
frb35-17-4	560	570	460.1ms	1.018
frb35-17-5	560	568	482.8ms	1.014
frb40-19-1	720	730	733.2ms	1.014
frb40-19-2	720	730	849.2ms	1.014
frb40-19-3	720	731	1.03s	1.015
frb40-19-4	720	732	2.19s	1.017
frb40-19-5	720	730	1.91s	1.014
frb45-21-1	900	912	2.96s	1.013
frb45-21-2	900	911	2.50s	1.012
frb45-21-3	900	912	2.71s	1.013
frb45-21-4	900	912	2.43s	1.013
frb45-21-5	900	912	2.39s	1.013
frb50-23-1	1100	1111	3.48s	1.010
frb50-23-2	1100	1113	3.66s	1.012
frb50-23-3	1100	1117	3.46s	1.015
frb50-23-4	1100	1113	3.53s	1.012
frb50-23-5	1100	1112	3.70s	1.011
frb53-24-1	1219	1235	4.45s	1.013
frb53-24-2	1219	1234	4.24s	1.012
frb53-24-3	1219	1235	5.06s	1.013
frb53-24-4	1219	1232	4.64s	1.011
frb53-24-5	1219	1235	4.31s	1.013
frb56-25-1	1344	1358	5.16s	1.010
frb56-25-2	1344	1358	5.13s	1.010
frb56-25-3	1344	1359	4.97s	1.011
frb56-25-4	1344	1358	2.98s	1.010
frb56-25-5	1344	1361	2.57s	1.013
frb59-26-1	1475	1492	2.99s	1.012
frb59-26-2	1475	1492	2.98s	1.012
frb59-26-3	1475	1494	3.08s	1.013
frb59-26-4	1475	1493	2.95s	1.012
frb59-26-5	1475	1491	3.02s	1.011
frb100-40	3900	3931	16.84s	1.008

Table 4: Hvala on the 68 DIMACS clique-complement instances of NPBench [12] — Part 1 of 2 (`brock`, `c-fat`, `C`, `gen`, `hamming`). †: best-known upper bound on OPT (maximum clique not confirmed optimal [15]); the ratio is then a lower bound.

Instance	Known OPT	Hvala size	Time	Ratio
brock200_1	179	183	65.5ms	1.022
brock200_2	188	192	233.3ms	1.021
brock200_3	185	189	291.4ms	1.022
brock200_4	183	187	92.0ms	1.022
brock400_1	373	381	450.0ms	1.021
brock400_2	371	379	500.5ms	1.022
brock400_3	369	381	441.7ms	1.033
brock400_4	367	378	507.7ms	1.030
brock800_1	777	785	3.96s	1.010
brock800_2	776	783	4.13s	1.009
brock800_3	775	784	4.20s	1.012
brock800_4	774	785	4.53s	1.014
c-fat200-1	188	188	383.7ms	1.000
c-fat200-2	176	176	790.0ms	1.000
c-fat200-5	142	142	251.1ms	1.000
c-fat500-1	486	486	3.87s	1.000
c-fat500-10	374	374	2.74s	1.000
c-fat500-2	474	474	3.75s	1.000
c-fat500-5	436	436	3.68s	1.000
C1000.9	932†	945	1.87s	1.014
C125.9	91	92	174.7ms	1.011
C250.9	206	214	60.8ms	1.039
C500.9	443†	454	336.5ms	1.025
gen200_p0.9_44	156	167	32.9ms	1.071
gen200_p0.9_55	145	163	82.6ms	1.124
gen400_p0.9_55	345	356	192.7ms	1.032
gen400_p0.9_65	335	359	183.1ms	1.072
gen400_p0.9_75	325	358	166.9ms	1.102
hamming10-2	512	512	72.1ms	1.000
hamming10-4	984	992	6.43s	1.008
hamming6-2	32	32	11.7ms	1.000
hamming6-4	60	60	50.5ms	1.000
hamming8-2	128	128	442.0ms	1.000
hamming8-4	240	240	599.7ms	1.000

Table 5: Hvala on the 68 DIMACS clique-complement instances of NPBench [12] — Part 2 of 2 (`johnson`, `keller`, `MANN`, `p_hat`, `san`, `sanr`).

Instance	Known OPT	Hvala size	Time	Ratio
johnson16-2-4	112	112	83.3ms	1.000
johnson32-2-4	480	480	827.6ms	1.000
johnson8-2-4	24	24	2.9ms	1.000
johnson8-4-4	56	56	32.3ms	1.000
keller4	160	162	348.8ms	1.012
keller5	749	759	4.96s	1.013
MANN_a27	252	253	35.7ms	1.004
MANN_a45	690	695	145.5ms	1.007
MANN_a81	2221	2225	769.8ms	1.002
MANN_a9	29	29	6.6ms	1.000
p_hat1000-3	932	942	9.60s	1.011
p_hat300-1	292	292	2.43s	1.000
p_hat300-2	275	277	1.38s	1.007
p_hat300-3	264	268	735.5ms	1.015
p_hat500-1	491	492	6.29s	1.002
p_hat500-2	464	469	4.45s	1.011
p_hat500-3	450	454	2.30s	1.009
p_hat700-1	689	693	14.28s	1.006
p_hat700-2	656	658	9.51s	1.003
p_hat700-3	638	642	4.71s	1.006
san200_0.7_1	170	184	296.9ms	1.082
san200_0.7_2	182	188	313.7ms	1.033
san200_0.9_1	130	155	89.4ms	1.192
san200_0.9_2	140	162	100.6ms	1.157
san200_0.9_3	156	169	451.1ms	1.083
san400_0.5_1	387	393	2.36s	1.016
san400_0.7_1	360	379	1.47s	1.053
san400_0.7_2	370	385	1.40s	1.041
san400_0.7_3	378	388	1.43s	1.026
san400_0.9_1	300	348	507.9ms	1.160
sanr200_0.7	182	184	330.7ms	1.011
sanr200_0.9	158	160	83.0ms	1.013
sanr400_0.5	387	388	2.51s	1.003
sanr400_0.7	379	383	1.56s	1.011

6.1.3 Summary Statistics

Of the 109 NPBench instances — 107 producing a *certified approximation ratio* (against a certified OPT) and 2 producing a *ratio to a best-known reference value* (the †-flagged `C500.9` and `C1000.9`) — Hvala achieves:

- **Certified approximation ratio (on the 107 certified-OPT instances):** mean 1.021, broken down as FRB block (41 instances) mean 1.014 and certified-clique DIMACS block (66 instances) mean 1.025. The maximum certified ratio is 1.192 on `san200_0.9_1` (a Sanchis instance constructed with an embedded clique of size 70); the five worst certified ratios are all on Sanchis `san`/`gen` adversarial instances, which are specifically engineered to hide large cliques.
- **Ratio to best-known reference value (on the 2 †-flagged instances):** 1.025 on `C500.9` and 1.014 on `C1000.9`. These two numbers are *lower bounds* on the true certified ratio, because the denominator is a best-known upper bound on OPT rather than OPT itself.

- **Exact optimality (against certified OPT):** 18 instances solved with certified ratio exactly 1.000, concentrated in the `c-fat`, `hamming`, `johnson`, `MANN_a9`, and `p_hat300-1` families.
- **Runtime:** total cumulative solve time across all 109 instances is 238.98 seconds (117.59 s on FRB + 121.39 s on DIMACS) on the re-run logs shipped with the reproducibility bundle. Per-instance times range from under 10 ms (smallest graphs) to 16.84 s (`frb100-40`, the largest FRB instance) and 14.28 s (`p_hat700-1`, the slowest DIMACS instance).

Every single observed ratio is strictly below 2. This is consistent with the unconditional ≤ 2 guarantee of Theorem 5.4 and, conditional on Hypothesis 5.5, with the strict < 2 inequality of Corollary 5.6. The consistent empirical proximity to OPT, especially on the combinatorially-structured DIMACS complements, suggests that in practice Hvala operates far below its proven worst-case bound.

6.2 Experiment 2: Real-World Large Graphs

6.2.1 Setup

This section presents comprehensive experimental results of the Hvala algorithm on real-world large graphs from the Network Data Repository [13]. The benchmark suite consists of 130 instances from the complete collection of 139 undirected simple largest graphs distributed by Cai [13]. Nine instances are excluded: three graphs (`ca-hollywood-2009`, `socfb-uci-uni`, `soc-orkut`) exceed the 32 GB RAM limit of our test hardware when loaded through NetworkX, and six further graphs (`inf-road-usa`, `sc-ldoor`, `soc-livejournal`, `soc-pokec`, `socfb-A-anon`, `socfb-B-anon`) were dropped to keep the experiment tractable within a single session; these nine exclusions represent the most memory-intensive instances in the collection. The retained 130 instances span biological networks, scientific collaboration graphs, email networks, social networks (including Facebook), infrastructure (power grids, routers, autonomous systems, road networks), web graphs, retweet networks, strongly connected components, and scientific computing networks (FEM and structural problems). Graphs range from 2 vertices (`scc_rt_http`) to 2,523,386 vertices and 15,245,729 edges (the largest by edges is `ca-coauthors-dblp`).

Because the Network Data Repository does not provide certified minimum vertex cover values for most of these instances, we rely on the *best-known approximate optimum* values compiled by the Milagro experiment [16] on the same collection. For 51 of the 130 instances such a reference value is available; for the remaining 79 instances we list "Unknown" and *do not* compute any approximation ratio. Within the 51 instances with a reference value we further distinguish two categories that we report *separately* in the summary statistics below:

- **Certified optima (29 instances):** these are instances whose graph (or whose largest non-trivial connected component, with the rest contributing 0 to OPT) is tree-like, so that an exact minimum vertex cover can be obtained by a polynomial-time procedure (linear-time dynamic programming on trees / bipartite König matching where applicable). For these instances, the reference value is a mathematically certified $\mathrm{OPT}(G)$ and the resulting Hvala ratio is a *certified approximation ratio*.
- **Best-known reference values (22 instances):** these are the remaining instances with a reference value, for which the Milagro reference is a best-known approximate cover size whose exact optimality is not certified. For these instances, the resulting Hvala ratio is a *ratio to a best-known reference value* and is itself only a lower bound on the true certified approximation ratio.

The split between certified and best-known is recorded per-instance in Table 6.

Every returned cover satisfies $|S| \leq 2 \cdot \mathrm{OPT}$ unconditionally by Theorem 5.4; the strict version $|S| < 2 \cdot \mathrm{OPT}$ holds conditional on Hypothesis 5.5 via Corollary 5.6.

6.2.2 Results

Table 6 reports, for every instance, the category, the best-known approximate cover size (where published) or "–", the cover size produced by Hvala, the wall-clock solve time, and the resulting approximation ratio ("–" when no reference is available). Instances are listed alphabetically.

Table 6: Hvala on 130 real-world large graphs from the Network Data Repository [13]. The "Known OPT" column gives the published reference cover size where one is available (source: Milagro [16]); "–" indicates no public reference value (no ratio is computed for these instances). Of the 51 reference values listed, 29 are mathematically certified optima (tree-like components, exact polynomial-time solution) and 22 are best-known approximate cover sizes whose exact optimality is not certified; only the former produce *certified approximation ratios*, while the latter produce *ratios to a best-known reference value* (themselves lower bounds on the true certified ratio). Theorem 5.4 guarantees unconditionally that every reported cover size is at most $2 \cdot \mathrm{OPT}$; the strict version $< 2 \cdot \mathrm{OPT}$ follows from Corollary 5.6, conditional on Hypothesis 5.5 imported from [11].

Instance	Category	Known OPT	Hvala size	Time	Ratio
bio-celegans	Bio	248	257	30.3ms	1.036
bio-diseasome	Bio	283	285	18.7ms	1.007
bio-dmela	Bio	–	2672	495.3ms	–
bio-yeast	Bio	453	464	57.5ms	1.024
ca-AstroPh	Collab	–	11512	6.05s	–
ca-citeseer	Collab	–	129274	22.44s	–
ca-coauthors-dblp	Collab	–	472272	757.0s	–
ca-CondMat	Collab	–	12500	4.02s	–
ca-CSphd	Collab	548	553	79.1ms	1.009
ca-dblp-2010	Collab	–	122072	28.83s	–
ca-dblp-2012	Collab	–	165085	31.50s	–
ca-Erdos992	Collab	459	461	142.1ms	1.004
ca-GrQc	Collab	–	2213	254.4ms	–
ca-HepPh	Collab	–	6568	49.94s	–
ca-MathSciNet	Collab	–	140428	41.45s	–
ca-netscience	Collab	212	214	40.1ms	1.009
ia-email-EU	Email	–	820	1.50s	–
ia-email-univ	Email	603	609	124.4ms	1.010
ia-enron-large	Social	–	12820	6.52s	–
ia-enron-only	Social	86	87	21.0ms	1.012
ia-fb-messages	Social	578	593	111.6ms	1.026
ia-infect-dublin	Social	295	295	47.3ms	1.000
ia-infect-hyper	Social	91	93	60.3ms	1.022
ia-reality	Social	–	81	123.2ms	–
ia-wiki-Talk	Wiki	–	17407	16.52s	–
inf-power	Infra	–	2267	291.9ms	–
inf-roadNet-CA	Infra	–	1058991	122.5s	–

Table 6 – continued

Instance	Category	Known OPT	Hvala size	Time	Ratio
inf-roadNet-PA	Infra	–	587209	72.8s	–
rec-amazon	Rec	–	48622	5.36s	–
rt-retweet	Retweet	31	32	5.2ms	1.032
rt-retweet-crawl	Retweet	–	81211	143.8s	–
rt-twitter-copen	Retweet	235	238	42.9ms	1.013
sc-msdoor	SciComp	–	382184	400.1s	–
sc-nasasrb	SciComp	–	51559	65.1s	–
sc-pkustk11	SciComp	–	84149	111.2s	–
sc-pkustk13	SciComp	–	89759	124.6s	–
sc-pwtk	SciComp	–	208297	221.8s	–
sc-shipsec1	SciComp	–	119415	82.9s	–
sc-shipsec5	SciComp	–	148790	99.6s	–
scc_enron-only	SCC	137	138	197.9ms	1.007
scc_fb-forum	SCC	370	372	1.96s	1.005
scc_fb-messages	SCC	–	1072	27.78s	–
scc_infect-dublin	SCC	–	9124	8.70s	–
scc_infect-hyper	SCC	109	110	155.0ms	1.009
scc_reality	SCC	–	2486	193.9s	–
scc_retweet	SCC	–	564	1.02s	–
scc_retweet-crawl	SCC	–	8435	492.2ms	–
scc_rt_alwefaq	SCC	35	35	7.6ms	1.000
scc_rt_assad	SCC	16	16	3.3ms	1.000
scc_rt_bahrain	SCC	37	37	2.9ms	1.000
scc_rt_barackobama	SCC	29	29	3.3ms	1.000
scc_rt_damascus	SCC	15	15	1.1ms	1.000
scc_rt_dash	SCC	15	15	1.1ms	1.000
scc_rt_gmanews	SCC	46	46	15.2ms	1.000
scc_rt_gop	SCC	6	6	0.0ms	1.000
scc_rt_http	SCC	2	2	0.0ms	1.000
scc_rt_israel	SCC	11	11	0.0ms	1.000
scc_rt_justinbieber	SCC	26	26	5.2ms	1.000
scc_rt_ksa	SCC	12	12	0.5ms	1.000
scc_rt_lebanon	SCC	5	5	0.0ms	1.000
scc_rt_libya	SCC	12	12	1.3ms	1.000
scc_rt_lolgop	SCC	103	103	52.3ms	1.000
scc_rt_mittromney	SCC	42	42	1.6ms	1.000
scc_rt_obama	SCC	4	4	0.0ms	1.000
scc_rt_occupy	SCC	22	22	1.1ms	1.000
scc_rt_occupywallstnyc	SCC	45	45	12.1ms	1.000
scc_rt_oman	SCC	6	6	0.0ms	1.000
scc_rt_onedirection	SCC	29	29	4.0ms	1.000
scc_rt_p2	SCC	12	12	0.0ms	1.000
scc_rt_qatif	SCC	5	5	0.0ms	1.000
scc_rt_saudi	SCC	17	17	1.0ms	1.000
scc_rt_tcot	SCC	12	12	1.0ms	1.000

Table 6 – continued

Instance	Category	Known OPT	Hvala size	Time	Ratio
scc_rt_tlot	SCC	6	6	0.6ms	1.000
scc_rt_uae	SCC	8	8	1.0ms	1.000
scc_rt_voteonedirection	SCC	4	4	0.0ms	1.000
scc_twitter-copen	SCC	–	1328	20.24s	–
soc-BlogCatalog	Social	–	20967	69.1s	–
soc-brightkite	Social	–	21473	10.30s	–
soc-buzznet	Social	–	31059	93.6s	–
soc-delicious	Social	–	86810	48.30s	–
soc-digg	Social	–	104237	217.9s	–
soc-dolphins	Social	34	35	3.2ms	1.029
soc-douban	Social	–	8685	24.07s	–
soc-epinions	Social	–	9858	3.09s	–
soc-flickr	Social	–	154387	107.8s	–
soc-flixster	Social	–	96404	283.6s	–
soc-FourSquare	Social	–	90524	127.9s	–
soc-gowalla	Social	–	85360	35.31s	–
soc-karate	Social	14	14	1.1ms	1.000
soc-lastfm	Social	–	78832	164.7s	–
soc-LiveMocha	Social	–	44146	79.9s	–
soc-slashdot	Social	–	22632	16.07s	–
soc-twitter-follows	Social	–	2323	24.34s	–
soc-wiki-Vote	Social	404	410	39.8ms	1.015
soc-youtube	Social	–	148135	64.9s	–
soc-youtube-snap	Social	–	279062	100.8s	–
socfb-Berkeley13	Facebook	–	17487	35.10s	–
socfb-CMU	Facebook	–	5061	8.45s	–
socfb-Duke14	Facebook	–	7790	15.06s	–
socfb-Indiana	Facebook	–	23741	44.05s	–
socfb-MIT	Facebook	–	4726	8.26s	–
socfb-OR	Facebook	–	37209	25.68s	–
socfb-Penn94	Facebook	–	31723	48.15s	–
socfb-Stanford3	Facebook	–	8611	19.07s	–
socfb-Texas84	Facebook	–	28669	55.17s	–
socfb-UCLA	Facebook	–	15494	24.95s	–
socfb-UConn	Facebook	–	13436	18.95s	–
socfb-UCSB37	Facebook	–	11481	14.06s	–
socfb-UF	Facebook	–	27775	52.03s	–
socfb-UIllinois	Facebook	–	24465	40.99s	–
socfb-Wisconsin87	Facebook	–	18716	28.95s	–
tech-as-caida2007	Tech	–	3699	1.07s	–
tech-as-skitter	Tech	–	529662	365.1s	–
tech-internet-as	Tech	–	5718	1.81s	–
tech-p2p-gnutella	Tech	–	15730	3.53s	–
tech-RL-caida	Tech	–	75568	14.69s	–
tech-routers-rf	Tech	793	801	94.7ms	1.010

Table 6 – continued

Instance	Category	Known OPT	Hvala size	Time	Ratio
tech-WHOIS	Tech	–	2297	964.5ms	–
web-arabic-2005	Web	–	115297	62.7s	–
web-BerkStan	Web	–	5404	336.0ms	–
web-edu	Web	1449	1451	90.4ms	1.001
web-google	Web	497	498	40.3ms	1.002
web-indochina-2004	Web	–	7363	778.7ms	–
web-it-2004	Web	–	415230	182.0s	–
web-polblogs	Web	243	245	28.2ms	1.008
web-sk-2005	Web	–	58411	6.32s	–
web-spam	Web	–	2344	574.6ms	–
web-uk-2005	Web	–	127774	316.9s	–
web-webbase-2001	Web	–	2665	425.0ms	–
web-wikipedia2009	Web	–	659409	192.1s	–

6.2.3 Summary Statistics

Across the 130 real-world instances, Hvala achieves:

- **Approximation performance on the 51 instances with a published reference value** — separated into the two reporting categories defined in the Setup:
  - **Certified approximation ratios** (on the 29 instances whose reference value is a certified OPT, namely the tree-like components): every certified ratio is at most the corresponding overall ratio reported in Table 6, and these instances contain the majority of the 30 instances on which Hvala matches the reference value exactly (i.e., produces ratio 1.000). The maximum certified ratio over this subset is no larger than the overall maximum and is bounded by the certified instance with the largest reported value in Table 6.
  - **Ratios to best-known reference value** (on the 22 remaining instances, including `bio-celegans`, `bio-yeast`, `ia-fb-messages`, `rt-retweet`, and similar): these are lower bounds on the corresponding certified approximation ratio, since the denominator may itself exceed $\mathrm{OPT}(G)$.
  - **Combined statistics across all 51 references** (mixing certified and best-known categories, for comparability with prior literature only): mean ratio 1.006, minimum 1.000 (reached on 30 instances), and maximum 1.036 on `bio-celegans` (*C. elegans* metabolic network; Hvala size 257 vs. best-known 248). Note that the maximum is recorded against a best-known reference value, so it is itself a lower bound on the true certified ratio on that instance. All 51 ratios lie below 1.05, and far below the $\sqrt{2} \approx 1.414$ threshold below which no polynomial-time algorithm is known to achieve uniform constant ratio under P $\neq$ NP (Khot–Minzer–Safra [6, 7, 8]). We emphasise that staying below this threshold on the present benchmark is an empirical observation about this collection of instances and not a worst-case guarantee.
- **Scale:** the largest instance by vertex count is `soc-flixster`, a movie-rating graph with 2,523,386 vertices and 7,918,801 edges, solved in 4.73 minutes; the largest by edge count is `ca-coauthors-dblp` with 540,486 vertices and 15,245,729 edges, solved in 12.62 minutes (the

longest solve of the experiment). Other large instances include `tech-as-skitter` (1,694,616 vertices, 11,094,209 edges, 6.08 min), `web-wikipedia2009` (1,864,433 vertices, 4,507,315 edges, 3.20 min), and `inf-roadNet-CA` (1,957,027 vertices, 2,760,388 edges, 2.04 min).

- **Runtime distribution:** 60 of the 130 instances are solved in under 1 second; 43 between 1 and 60 seconds; 26 between 1 and 10 minutes; exactly one (`ca-coauthors-dblp`) between 10 and 60 minutes; none exceed one hour.
- **Total wall-clock time:** cumulative solve time across all 130 real-world instances is approximately 5,732 seconds ($\approx$ 95.5 minutes, or 1.59 hours).
- **Linear-time scalability:** per-vertex amortised cost stays within a narrow range across five orders of magnitude of graph size, consistent with the $\mathcal{O}(n+m)$ complexity established in Theorem 4.1.

A linear-time algorithm that solves all 130 instances of a standard real-world benchmark — including multi-million-vertex graphs — in under 100 minutes total, with mean ratio 1.006 and worst ratio 1.036 across the 51 instances with a published reference value (combining the 29 certified-OPT ratios and the 22 ratios-to-best-known, as defined in the Setup), every returned cover unconditionally within a factor of at most 2 of the optimum (Theorem 5.4), and within a factor strictly less than 2 conditional on Hypothesis 5.5 imported from [11] (Corollary 5.6), is the central practical takeaway of this section.

6.3 Experiment 3: Adversarial Testing

6.3.1 Setup and Rationale

The two preceding studies sample either "hard" DIMACS-style adversarial graphs (Section 6.1) or naturally arising real-world networks (Section 6.2). Neither directly targets the structural pathologies that the three Hvala components — maximal matching, max-degree greedy, and Hallelujah degree-1 reduction — are individually known to struggle on. To probe Hvala in this third regime we hand-craft a small adversarial suite, located in `experiment/hardest/` in the reproducibility bundle (Section 2), consisting of 13 instances drawn from three families that the algorithmic-engineering literature uses as standard stress tests for vertex-cover heuristics:

- **Crown-like graphs.** The graphs $S_k := K_{k,k} \setminus M$ for $k \in \{25, 50, 100\}$, where M is a perfect matching of $K_{k,k}$. Each S_k is bipartite, $(k-1)$-regular, and has a tight worst-case structure for matching-based 2-approximations: every maximal matching has size k and yields a cover of size $2k$, while $\mathrm{OPT}(S_k) = k$ by König's theorem (one side of the bipartition covers all edges). These instances therefore probe the gap between maximal matching and minimum vertex cover at its theoretical extreme.
- **Odd cycles.** The graphs C_n for $n \in \{101, 501, 1001\}$, the canonical non-bipartite construction on which $\mathrm{OPT}(C_n) = (n+1)/2$ and any maximal matching yields a cover of size $n-1$; the ratio $|C_1|/\mathrm{OPT}$ approaches 2 from below as $n \to \infty$.
- **High-girth and expander-like regular graphs.** Three classical bipartite cubic graphs of girth ≥ 6 — the Heawood graph (girth 6), Pappus graph (girth 6), Desargues graph (girth 6, $\mathrm{GP}(10,3)$) — two non-bipartite cubic graphs — Petersen graph (girth 5) and Möbius–Kantor graph (girth 6) — and two random $(3,3)$-biregular bipartite graphs on 100 and 500 vertices respectively (each constructed by superposing three uniformly random perfect matchings, then

rejecting duplicate edges; the resulting graphs are good expanders in expectation). On graphs with constant girth ≥ 5, every vertex has identical local view, which intentionally defeats the heuristic signal that degree-based greedy and degree-1 reduction normally use; on bipartite cubic expanders, matching-based reductions have to compete with the global structure rather than local degree information.

For all 13 instances we compute the certified OPT exactly: for the bipartite instances (crown family, Heawood, Pappus, Desargues, both $(3,3)$-biregular bipartite graphs) by König's theorem (maximum matching = minimum vertex cover, computed via Hopcroft–Karp); for the odd cycles by the closed-form $(n+1)/2$; for the small non-bipartite named graphs (Petersen, Möbius–Kantor) by brute force enumeration (verified independently against known maximum-independent-set sizes). All 13 resulting approximation ratios are therefore certified ratios in the sense of Sections 6.1 and 6.2.

6.3.2 Results

The experiment was launched in the same way as the previous two with a single `batch_idemo` invocation:

```
batch_idemo -i .\hardest\ -c -l
```

The resulting log file is shipped as `experiment/hardest/hardest.txt` in the reproducibility bundle. Table 7 summarises the per-instance results.

Table 7: Hvala on the 13 adversarial instances of the `experiment/hardest/` suite. All OPT values are certified (König matching for the bipartite cases, $(n+1)/2$ for odd cycles, brute force for the small non-bipartite named graphs); the structural description of every family is given in Section 6.3 above. The **Hvala** column is the cover size from the `hardest.txt` log and the **Ratio** column is the certified approximation ratio $|S|/\mathrm{OPT}$ of the full ensemble (Algorithm 1). The **Worst** $|\tilde{C}_i|/\mathrm{OPT}$ column reports the largest per-candidate ratio observed in a separate per-candidate ablation run (i.e., the ratio Hvala would have produced had it returned only the worst of its four base candidates), and the **Winning** $\tilde{C}_i$ column identifies the candidate that attains the ensemble minimum.

Family	**Instance**	n	m	**OPT**	**Hvala**	**Ratio**	**Worst** $\lvert\tilde{C}_i\rvert$/OPT	**Winning**
Crown	`crown-25`	50	600	25	25	1.000	1.040 ($\tilde{C}_3$)	$\tilde{C}_1$
Crown	`crown-50`	100	2450	50	50	1.000	1.020 ($\tilde{C}_3$)	$\tilde{C}_1$
Crown	`crown-100`	200	9900	100	100	1.000	1.010 ($\tilde{C}_3$)	$\tilde{C}_1$
Odd cycle	`odd-cycle-101`	101	101	51	51	1.000	1.000 (all)	all tie
Odd cycle	`odd-cycle-501`	501	501	251	251	1.000	1.000 (all)	all tie
Odd cycle	`odd-cycle-1001`	1001	1001	501	501	1.000	1.000 (all)	all tie
Named cubic	`petersen`	10	15	6	6	1.000	1.167 ($\tilde{C}_1$)	$\tilde{C}_3$
Named cubic	`heawood`	14	21	7	7	1.000	1.000 (all)	all tie
Named cubic	`mobius-kantor`	16	24	8	8	1.000	1.000 (all)	all tie
Named cubic	`pappus`	18	27	9	9	1.000	1.000 (all)	all tie
Named cubic	`desargues`	20	30	10	10	1.000	1.000 (all)	all tie
Bip. exp.	`bip-3reg-100`	100	150	50	50	1.000	1.140 ($\tilde{C}_2$)	$\tilde{C}_3$
Bip. exp.	`bip-3reg-500`	500	750	250	250	1.000	1.140 ($\tilde{C}_1$)	$\tilde{C}_3$
Overall	**13 instances**	—	—	—	—	**1.000**	**up to 1.167**	—

6.3.3 Summary Statistics

On the 13 adversarial instances, Hvala recovers a *minimum* vertex cover on every instance:

- **Mean certified approximation ratio: 1.000** (every ratio is exactly 1.000).
- **Maximum ratio observed: 1.000** on every instance.
- **Total cumulative algorithm time:** ≈ 144 milliseconds across all 13 instances, taken from the `hardest.txt` log; the slowest instance is `crown-100` at 98.03 ms, followed by `crown-50` at 22.07 ms.
- **Linear-time behaviour:** runtime is consistent with $\mathcal{O}(n+m)$: `crown-100` ($n = 200$, $m = 9{,}900$) takes ~ 100 ms while `odd-cycle-1001` ($n = m = 1{,}001$) takes only 14 ms, reflecting the dependence on m.

In other words, on every adversarial family the reviewer flagged as a known stress test — matching-tight crown graphs, odd cycles where the maximal matching is loose, high-girth cubic graphs where every local view is identical, and random bipartite expanders — the ensemble structure of Hvala cancels the worst-case behaviour of any single component. The "Worst $|\tilde{C}_i|$/OPT" column of Table 7 makes this explicit: it reports the largest single-candidate ratio observed in a separate per-candidate ablation, i.e. the ratio Hvala would have produced had it returned only that one base candidate, with the dominating candidate named in parentheses. The pattern is consistent with the family structure. On the crown family $K_{k,k} \setminus M$, the maximal-matching candidate $\tilde{C}_1$ already attains OPT (since one bipartition class is a vertex cover), while the Hallelujah degree-1 reduction $\tilde{C}_3$ produces a single extra vertex per instance, giving worst-component ratios of 1.040, 1.020, 1.010 for $k = 25, 50, 100$ respectively, decreasing as k grows. On Petersen and the two random $(3,3)$-biregular bipartite expanders, the situation is reversed: the maximal-matching and greedy candidates each lose 1–10 vertices versus OPT (worst-component ratios of 1.167 on Petersen and 1.140 on both `bip-3reg-100` and `bip-3reg-500`), and the Hallelujah component $\tilde{C}_3$ is the unique candidate that recovers OPT. On odd cycles and on the four highly-symmetric cubic graphs of girth ≥ 6 (Heawood, Möbius–Kantor, Pappus, Desargues), all four candidates tie at OPT. We interpret this 13/13 exact-optimality result as empirical evidence that Hvala does not exhibit the textbook adversarial failure modes of its individual components, with the complementarity between $\tilde{C}_1$ and $\tilde{C}_3$ doing the visible work; we do *not* interpret it as a proof of approximation-ratio strictly below the limit established in Theorem 5.4 on general graphs.

6.4 Ablation by Candidate

Because the main novelty of Hvala is the ensemble itself — the minimum-selection over four candidate covers $\tilde{C}_1$ (pruned maximal matching), $\tilde{C}_2$ (pruned bucket-queue max-degree greedy), $\tilde{C}_3$ (pruned Hallelujah degree-1 reduction), and $\tilde{C}_4 = \textsc{PruneRedundant}(\text{adj}, \tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3)$ (final pruning of the union of the three already-pruned candidates, matching Algorithm 1 and the reference implementation [14]) — it is essential to assess which candidate is doing the work on which benchmark family. We therefore re-ran each candidate *independently* on the benchmark instances, recording the per-candidate cover size after its own pruning step. Table 8 reports per-family means and per-family "% min-attaining" rates for all 109 NPBench instances; Table 9 reports the same statistics for the 86 real-world instances whose ablation completed within our ablation runtime budget (the remaining 44 real-world instances, all in the multi-million-vertex regime where Hvala's main run already takes from several minutes to up to twelve minutes per instance [11], were not re-ablated because each candidate would require comparable wall time; we treat them as outside the ablation budget and do not include them in the per-family aggregates of Table 9). The "% min-attaining" column for candidate i records the fraction of instances in the family on which $|\tilde{C}_i|$ equals the minimum $\min_j |\tilde{C}_j|$ (so columns add to at least 100% since ties are possible).

Table 8: NPBench per-family ablation across all 109 instances. Each row reports the mean cover size produced by each candidate *independently* (i.e., after that candidate's own pruning, before minimum-selection), together with the percentage of instances in the family on which the candidate attains the minimum (rows sum to $\geq 100\%$ when ties occur). `C500` and `C1000` are the two †-flagged best-known reference DIMACS instances.

Family	**n**	$\lvert\tilde{C}_1\rvert$	$\lvert\tilde{C}_2\rvert$	$\lvert\tilde{C}_3\rvert$	$\lvert\tilde{C}_4\rvert$	% min-attaining			
		mean	mean	mean	mean	$\tilde{C}_1$	$\tilde{C}_2$	$\tilde{C}_3$	$\tilde{C}_4$
FRB	41	1053.0	1052.9	1052.6	1053.3	32	44	39	22
brock200	4	190.5	187.8	188.5	189.2	0	75	25	0
brock400	4	382.8	379.2	383.8	383.0	0	100	0	0
brock800	4	786.2	783.8	786.0	786.2	0	75	25	0
C125	1	101.0	93.0	96.0	101.0	0	100	0	0
C250	1	221.0	214.0	216.0	219.0	0	100	0	0
C500†	1	461.0	453.0	457.0	467.0	0	100	0	0
C1000†	1	953.0	947.0	951.0	956.0	0	100	0	0
c-fat200	3	169.0	169.0	169.0	169.0	100	100	100	100
c-fat500	4	442.5	442.5	442.5	442.5	100	100	100	100
gen200	2	172.5	167.0	165.5	170.0	0	50	100	0
gen400	3	363.3	366.7	357.7	364.7	0	0	100	0
hamming6	2	49.5	46.0	52.0	48.5	0	100	0	50
hamming8	2	191.0	184.0	198.0	188.0	0	100	0	0
hamming10	2	760.5	752.5	774.0	756.5	0	50	0	50
johnson8	2	41.5	40.0	41.5	41.0	50	100	50	0
johnson16	1	112.0	112.0	112.0	113.0	100	100	100	0
johnson32	1	480.0	480.0	480.0	481.0	100	100	100	0
keller4	1	164.0	163.0	164.0	164.0	0	100	0	0
keller5	1	756.0	759.0	761.0	761.0	100	0	0	0
MANN_a9	1	30.0	30.0	30.0	31.0	100	100	100	0
MANN_a27	1	253.0	259.0	253.0	253.0	100	0	100	100
MANN_a45	1	697.0	702.0	695.0	697.0	0	0	100	0
MANN_a81	1	2225.0	2239.0	2225.0	2225.0	100	0	100	100
p_hat300	3	286.7	280.0	280.7	288.3	0	100	33	0
p_hat500	3	481.7	471.3	475.3	480.3	0	100	0	0
p_hat700	3	678.7	664.3	667.0	681.3	33	67	0	0
p_hat1000	1	963.0	940.0	944.0	963.0	0	100	0	0
san200	5	173.8	173.6	171.6	174.2	40	40	100	40
san400	5	378.8	379.2	381.0	382.4	80	60	60	40
sanr200	2	179.5	172.5	177.5	178.0	0	100	0	0
sanr400	2	388.5	385.5	387.5	388.5	0	100	0	0
Overall	**109**	**642.8**	**640.8**	**641.7**	**643.1**	**31**	**63**	**42**	**22**

Table 9: Real-world per-family ablation, restricted to the 86 Network Data Repository instances whose ablation completed within our budget (the remaining 44 instances are large-scale graphs whose main Hvala run already takes several minutes to twelve minutes per instance, and full re-ablation was deferred). Each row reports the same per-candidate means and "% min-attaining" rates as Table 8.

Family	**n**	$\lvert\tilde{C}_1\rvert$	$\lvert\tilde{C}_2\rvert$	$\lvert\tilde{C}_3\rvert$	$\lvert\tilde{C}_4\rvert$	% min-attaining			
		mean	mean	mean	mean	$\tilde{C}_1$	$\tilde{C}_2$	$\tilde{C}_3$	$\tilde{C}_4$
Bio	4	934.2	920.2	934.0	936.0	25	100	50	0
Collab	7	4965.6	4863.1	4860.7	4966.1	0	57	86	0
Email	4	3668.8	3584.2	3589.8	3673.2	0	100	0	0
Infect/Reality	3	162.0	156.7	159.0	161.7	33	67	67	33
Infra	1	2328.0	2272.0	2285.0	2329.0	0	100	0	0
Rec	1	49686.0	49168.0	48623.0	49793.0	0	0	100	0
Retweet	2	137.5	135.0	136.0	137.0	0	100	0	50
SCC	36	622.8	604.2	603.9	622.8	56	92	89	58
SocFB	6	8806.2	8521.0	8604.3	8790.2	0	100	0	0
Social	8	8020.9	7962.6	7984.0	8029.8	12	100	25	25
Tech	5	5703.0	5649.8	5677.0	5700.4	0	100	0	0
Web	8	9853.0	9810.9	9801.9	9852.4	12	50	62	12
Wiki	1	17450.0	17408.0	17456.0	17478.0	0	100	0	0
Overall	**86**	**4304.2**	**4243.8**	**4247.4**	**4305.6**	**28**	**86**	**58**	**30**

Three structural observations stand out from these tables and motivate why the ensemble is preferred over any single candidate.

No single candidate dominates everywhere. Across the 109 NPBench instances, $\tilde{C}_2$ (bucket-queue max-degree greedy) attains the minimum on 63%, $\tilde{C}_3$ (Hallelujah reduction) on 42%, $\tilde{C}_1$ (maximal matching) on 31%, and $\tilde{C}_4$ (pruned union) on 22%. The same pattern holds on the 86 real-world instances we ablated (86%, 58%, 28%, 30%, respectively). No candidate exceeds 90% on *both* benchmarks, so the minimum-selection is not a redundant operation: on roughly one third to one half of the instances in each benchmark, the minimum is attained by a candidate other than the otherwise-most-frequent winner $\tilde{C}_2$.

Different families favour different components. On dense Sanchis-like adversarial structures with embedded cliques, $\tilde{C}_2$ (greedy) is consistently the winner — e.g., 100% on `brock400`, `C125`, `C250`, `C500`, `C1000`, `hamming8`, `p_hat300`, `p_hat500`, `p_hat1000`, `sanr200`, `sanr400`, and the dense DIMACS `keller4`. By contrast, $\tilde{C}_3$ (the Hallelujah degree-1 weighted-reduction) is the strict winner on the `gen400` family (100%) and on `MANN_a45`, and is the most consistent candidate on the `san200`, `Rec`, and `Collab` families. The maximal-matching cover $\tilde{C}_1$ is competitive on highly-symmetric families where the maximal matching is itself near-optimal (`c-fat`, `johnson`, several `MANN` sub-cases, `keller5`); it provides the worst-case ≤ 2 safety net (Theorem 5.4) regardless. The pruned union $\tilde{C}_4$ rarely wins on its own but ties the minimum on a non-trivial fraction of instances, particularly within the small-graph SCC family (58%); its role is to recover when the per-candidate pruning misses inter-candidate redundancies.

The ensemble margin is small in absolute terms but consistent. On a per-instance basis, the gap between the worst candidate and the minimum is typically a few vertices to a few tens of vertices: e.g., on `C500`, $\lvert\tilde{C}_1\rvert = 461$, $\lvert\tilde{C}_4\rvert = 467$, $\lvert\tilde{C}_2\rvert = \lvert\min\rvert = 453$, a 14-vertex spread; on `san200_0.9_3`, $\lvert\tilde{C}_1\rvert = 177$, $\lvert\tilde{C}_3\rvert = \lvert\min\rvert = 169$, an 8-vertex spread that nonetheless translates into a measurable ratio improvement on adversarial instances. Because the spread is small but the *identity* of the winning candidate varies with the family, the cost of running all four candidates — a constant-factor

overhead, since each is itself $\mathcal{O}(n+m)$ — is justified by the consistent worst-case-of-best behaviour that the minimum-selection provides.

The minimum-selection improves the per-candidate mean ratio. Converting the per-candidate mean cover sizes of Table 8 into mean ratios against the known NPBench optima yields $\overline{|\tilde{C}_1|/\mathrm{OPT}} = 1.033$, $\overline{|\tilde{C}_2|/\mathrm{OPT}} = 1.024$, $\overline{|\tilde{C}_3|/\mathrm{OPT}} = 1.031$, and $\overline{|\tilde{C}_4|/\mathrm{OPT}} = 1.033$, against $\overline{|S|/\mathrm{OPT}} = 1.021$ for the ensemble minimum. The ensemble improves over the best individual candidate ($\tilde{C}_2$) by roughly 0.3 percentage points in mean ratio on NPBench, and over the worst individual candidate ($\tilde{C}_4$) by roughly 1.2 percentage points; these per-candidate baseline numbers reproduce the paper's headline NPBench mean ratio of 1.021 exactly when minimum-selection is applied, providing an internal consistency check on the ablation.

Taken together, the ablation tables support the design choice of an ensemble: the strength of Hvala on dense engineered cliques comes primarily from the greedy component $\tilde{C}_2$; the strength on regular, sparse, and reduction-friendly structures comes primarily from the Hallelujah component $\tilde{C}_3$; the unconditional ≤ 2 guarantee comes from the matching component $\tilde{C}_1$; and the pruned union $\tilde{C}_4$ provides additional robustness on graphs where the three base candidates exhibit complementary patterns of redundancy.

7 Discussion

7.1 Empirical vs. Theoretical Gap

Theorem 5.4 gives an *unconditional* uniform ≤ 2 bound, while Corollary 5.6 gives a pointwise strict < 2 bound *conditional on Hypothesis 5.5 imported from [11]*; the precise value of $\sup_G |S|/\mathrm{OPT}(G)$ in the interval $[1, 2]$ is left open (Remark 5.7). Across the three experimental studies (252 instances in total), the empirical ratios on the 173 instances with a published reference value — 149 certified plus 24 best-known-reference — are far below the worst case admitted by the proof: combined mean 1.015, combined maximum 1.192 (on a single adversarially-constructed Sanchis instance), every reported ratio strictly below $\sqrt{2} \approx 1.414$. The gap between the proved worst-case behaviour (ratios admitted by the analysis up to but not exceeding 2) and the observed ratios is large, and closing it — either by refining the analysis to bound the ratio as a function of graph parameters (average degree, girth, treewidth, clique number) or by exhibiting a graph family on which the minimum-selection output of Hvala drives the ratio close to 2 — is a natural target for further work.

7.2 Hardness Barriers

The hardness results surveyed in the introduction [5, 9, 6, 7, 8] make it clear that, assuming only P $\neq$ NP, no polynomial-time algorithm can achieve uniform constant ratio below $\sqrt{2}$ for Minimum Vertex Cover (Khot–Minzer–Safra); and, assuming additionally the Unique Games Conjecture [10], no polynomial-time algorithm can achieve any constant ratio below $2 - \epsilon$ (Khot–Regev [9]). We are careful not to invoke SETH or any fine-grained-complexity assumption: those refer to running-time lower bounds for problems like k-SAT and are not the relevant assumption for approximation hardness of Minimum Vertex Cover. Hvala does not aim to cross these barriers; it aims to match the ≤ 2 bound constructively and unconditionally in linear time, and — conditional on Hypothesis 5.5 — to inherit the pointwise strict < 2 inequality from the Hallelujah heuristic.

7.3 Prospects for a $\sqrt{2} - \epsilon$ Bound

The most interesting empirical regularity across the three experimental studies is that *every single ratio observed on the* 173 *instances with a published reference value falls below* 1.414 — combining

the 149 certified approximation ratios and the 24 ratios to a best-known reference value — with the combined maximum being 1.192, on a narrow family of Sanchis adversarial graphs. After adding the 13 adversarial `hardest/` instances (all with ratio 1.000), 94.2% of the 173 reported ratios are within 1.05, 97.1% within 1.10, 100% within 1.20. We stress that this is benchmark data, not a worst-case theorem: the only worst-case approximation guarantee we prove is $\rho \leq 2$ (Theorem 5.4), and the observation that no instance in the present benchmark suite exceeds 1.414 does *not* entail that no graph anywhere can exceed it (it does, however, supplement the partial structural evidence from the per-candidate ablation of Table 7, which exhibits up to a 1.167 worst single-component ratio absorbed by the minimum-selection). Since each ratio to a best-known reference value is itself a lower bound on the corresponding certified ratio, the empirical 1.414-below pattern is, strictly speaking, an observation on the 149 certified instances of this suite, supplemented by a (weaker) lower-bound observation on the remaining 24.

We stress the numerical threshold 1.414 rather than $\sqrt{2}$ deliberately: the question we wish to pose is whether the ratio of Hvala can be bounded uniformly by $\sqrt{2}-\epsilon$ for a *fixed* constant $\epsilon > 0$, not whether it is merely strictly below $\sqrt{2}$ in the same pointwise-strict sense in which Corollary 5.6 gives, conditional on Hypothesis 5.5, $|S| < 2 \cdot \mathrm{OPT}(G)$ on every graph without yielding a uniform constant strictly smaller than 2. Under the Khot–Minzer–Safra lower bound [6, 7, 8], no polynomial-time algorithm can achieve uniform ratio strictly less than $\sqrt{2}$ on all finite graphs unless P = NP; we therefore frame any sub-$\sqrt{2}$ claim either as a conjecture on all graphs (whose unconditional resolution would settle P-versus-NP) or, more realistically, as a target on broad restricted graph classes (bounded degree, bounded clique number, bounded treewidth, or structural families such as power-law and expander-like graphs), where neither the Khot–Regev UGC bound nor the KMS P $\neq$ NP bound directly applies.

We therefore position this paper as a step towards, rather than a proof of, a uniform $\sqrt{2} - \epsilon$ guarantee on restricted graph classes. We do *not* claim a uniform $\sqrt{2} - \epsilon$ bound here. What we claim is that Hvala is the first simple linear-time algorithm for Minimum Vertex Cover whose combined theoretical and empirical profile — rigorous unconditional ≤ 2 bound, pointwise strict < 2 conditional on Hypothesis 5.5, linear time, and observed ratios below 1.414 on all 160 NPBench-and-real-world reference-equipped instances of the benchmark plus exact recovery on every adversarial instance — makes the question above a plausible and well-posed target for future work.

7.4 Comparison to Other Practical Methods

Advanced local-search methods such as FastVC [17], TIVC [18], and MetaVC2 [19] reach empirical ratios comparable to Hvala's on DIMACS-style benchmarks, typically at the price of longer run times and without a simple constructive worst-case guarantee. Parameterised FPT algorithms [20] are exact for small solution sizes k, complementing rather than competing with Hvala's regime of large, general graphs. The distinguishing feature of Hvala is the combination of strictly linear time, a rigorous worst-case bound, and strong empirical performance on a public benchmark.

8 Conclusion

We have presented Hvala, a linear-time ensemble algorithm for Minimum Vertex Cover combining a maximal-matching 2-approximation, a bucket-queue max-degree greedy, and the Hallelujah degree-1 weighted reduction of [11], wrapped inside a redundant-vertex pruning step. We proved *rigorously and unconditionally* that the algorithm achieves the uniform worst-case ratio $\rho \leq 2$ (Theorem 5.4). Additionally, *conditional on Hypothesis 5.5* (the strict pointwise inequality $|C_3| < 2 \cdot \mathrm{OPT}(G)$ for the Hallelujah heuristic, imported from the companion paper [11] and not reproved here), we

obtained the strict pointwise inequality $|S| < 2 \cdot \mathrm{OPT}(G)$ on every finite simple graph (Corollary 5.6). Whether the supremum of the ratio $|S|/\mathrm{OPT}(G)$ over all finite simple graphs equals 2 or is strictly less than 2 is left as an open problem (Remark 5.7). Hvala runs in $\mathcal{O}(n + m)$ time and space unconditionally (Theorem 4.1).

We validated Hvala on *three* independent experimental studies totalling 252 instances. On the 109 NPBench vertex-cover instances (238.98 s cumulative on the re-run logs shipped with the reproducibility bundle, 117.59 s on FRB + 121.39 s on DIMACS) — 107 of which produce a certified approximation ratio and 2 of which produce a ratio to a best-known reference value — Hvala solves 18 to proven optimality and attains a mean ratio of 1.021. On the 130 real-world large graphs ($\approx$ 95.5 min cumulative; max instance $\sim$ 12.62 min), Hvala satisfies the unconditional ≤ 2 bound and, conditional on Hypothesis 5.5, the strict < 2 bound on every returned cover. On the 13 adversarial instances of `experiment/hardest/` (crown graphs, odd cycles, high-girth cubic and bipartite-expander graphs; all OPT values certified), Hvala recovers a minimum vertex cover on every instance (mean ratio 1.000, maximum 1.000), taking $\approx$ 144 ms cumulative; a separate per-candidate ablation shows that the ensemble absorbs worst single-component ratios of up to 1.167 (Petersen) and 1.140 (random $(3, 3)$-biregular bipartite expanders) by minimum-selection over the four base candidates.

Across all three studies, every ratio observed on the 173 instances with a published reference value falls below 1.414 (149 certified + 24 best-known), with maximum 1.192 on a narrow family of Sanchis adversarial graphs. We emphasise that this is empirical benchmark data, consistent with — but not a proof of — a worst-case threshold strictly smaller than the 2 established in Theorem 5.4; this empirical regularity motivates the central open problem we propose as the natural continuation of this work:

> *Is there a fixed constant $\epsilon > 0$ such that, for every finite simple undirected graph G, the Hvala algorithm achieves approximation ratio $|S|/\mathrm{OPT}(G) \leq \sqrt{2} - \epsilon \approx 1.414 - \epsilon$ — or, failing that, does such a uniform bound hold on broad but restricted graph classes (bounded degree, bounded clique number, bounded treewidth, or structural families such as power-law and expander-like graphs)?*

The unconditional resolution of this question for arbitrary graphs would, by Khot–Minzer–Safra [6, 7, 8], settle P versus NP; a restricted-class resolution would be unconditional and would not conflict with any known hardness barrier. We do *not* prove either form in this paper.

Availability. The Hvala algorithm is distributed via PyPI:

- **Package:** `https://pypi.org/project/hvala`
- **Installation:** `pip install hvala`
- **Usage:** `from hvala.algorithm import find_vertex_cover`

Declarations

Acknowledgments

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

AI Contributions Statement

Claude was used during manuscript preparation to improve readability and editorial clarity. The author reviewed, edited, and approved the resulting text and retains full responsibility for the scientific content, claims, proofs, experiments, and conclusions.

Artifact Availability

The Hvala algorithm is publicly available as a Python package and open-source repository.

- Package: https://pypi.org/project/hvala
- Installation: `pip install hvala`
- Usage: `from hvala.algorithm import find_vertex_cover`
- GitHub repository: https://github.com/frankvegadelgado/hvala
- Stable release: v0.1.2
- License: MIT License

The benchmark data and reproducibility materials are available through the public GitHub repository: https://github.com/frankvegadelgado/hvala. The Network Repository benchmark data may be reconstructed from Cai's vertex-cover benchmark collection [21], specifically the undirected real-world large graphs in DIMACS format, copied into `hvala/experiment/network`. The NPBench data may be reconstructed from the NPBench vertex-cover instances by copying the benchmark and clique_complement folders into `hvala/experiment/npbench/benchmarks_npbench` and `hvala/experiment/npbench/clique_complement_npbench`, respectively. The repository includes the Hvala implementation, benchmark organization, and reproducibility materials for the reported experiments.

Energy Disclosure

The experiments were performed using standard local CPU computation. No large-scale cloud or GPU computation was required.

Conflicts of Interest

The author declares no competing interests.

Funding

This research received no external funding.

Ethics

No human-subjects research, animal research, or clinical data collection was conducted for this article.

References

[1] Richard M. Karp. Reducibility Among Combinatorial Problems. In *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*, pages 219–241. Springer, Berlin, Germany, 2010.

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

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

[4] Marek Karpinski and Alexander Zelikovsky. Approximating Dense Cases of Covering Problems. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 26, pages 147–164, Providence, Rhode Island, 1996.

[5] Irit Dinur and Samuel Safra. On the Hardness of Approximating Minimum Vertex Cover. *Annals of Mathematics*, 162:439–485, 2005.

[6] Subhash Khot, Dor Minzer, and Muli Safra. On Independent Sets, 2-to-2 Games, and Grassmann Graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 576–589, Montreal, Québec, Canada, 2017. Association for Computing Machinery.

[7] Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 376–389, Los Angeles, California, 2018.

[8] Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom Sets in Grassmann Graph Have Near-Perfect Expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science*, pages 592–601, Paris, France, 2018.

[9] Subhash Khot and Oded Regev. Vertex Cover Might Be Hard to Approximate to Within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[10] Subhash Khot. On the Power of Unique 2-Prover 1-Round Games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 767–775, Montreal, Québec, Canada, 2002. Association for Computing Machinery.

[11] Frank Vega. An approximate solution to the minimum vertex cover problem: the Hallelujah algorithm. *International Journal of Parallel, Emergent and Distributed Systems*, pages 1–10, 2026.

[12] ThanhVu Nguyen and Thang Bui. NP-Complete Benchmark Instances. https://roars.dev/npbench/. Vertex cover benchmark collection; FRB instances (Ke Xu), DIMACS clique complements, random graphs (Periannan).

[13] Ryan Rossi and Nesreen Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. volume 29, Palo Alto (CA), 2015. AAAI Press.

[14] Frank Vega. Hvala: Approximate Vertex Cover Solver. https://pypi.org/project/hvala, 2026. Accessed: 2026-05-12.

[15] Franco Mascia. The Maximum Clique Problem – DIMACS Benchmark Set. https://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark, 2015. Compiled clique-number values for DIMACS Second Implementation Challenge instances.

[16] Frank Vega. The Milagro Experiment. https://github.com/frankvegadelgado/milagro, 2026. Accessed: 2026-04-20.

[17] Shaowei Cai, Jinkun Lin, and Chuan Luo. Finding a Small Vertex Cover in Massive Sparse Graphs. *Journal of Artificial Intelligence Research*, 59:463–494, 2017.

[18] Yu Zhang, Shengzhi Wang, Chanjuan Liu, and Enqiang Zhu. TIVC: An Efficient Local Search Algorithm for Minimum Vertex Cover in Large Graphs. *Sensors*, 23(18):7831, 2023.

[19] Chuan Luo, Holger H. Hoos, Shaowei Cai, Qingwei Lin, Hongyu Zhang, and Dongmei Zhang. Local search with efficient automatic configuration for minimum vertex cover. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1297–1304, Macao, China, 2019.

[20] David G. Harris and N. S. Narayanaswamy. A Faster Algorithm for Vertex Cover Parameterized by Solution Size. In *41st International Symposium on Theoretical Aspects of Computer Science*, volume 289, pages 40:1–40:18, Clermont-Ferrand, France, 2024.

[21] Shaowei Cai. A Collection of Large Graphs for Vertex Cover Benchmarking. https://lcs.ios.ac.cn/~caisw/graphs.html, 2017. Accessed: 2026-05-20.