An Approximate Independent Set Solver: The Furones Algorithm

Frank Vega *

Posted Date: 3 April 2026

doi: 10.20944/preprints202504.0522.v7

Keywords: optimization problem; approximation algorithm; graph theory; computational complexity; bipartite graphs

 doi:10.20944/preprints202504.0522.v7

Article

An Approximate Independent Set Solver: The Furones Algorithm

Frank Vega

Information Physics Institute, 840 W 67th St, Hialeah, FL 33012, USA; vega.frank@gmail.com

Abstract

The Maximum Independent Set (MIS) problem, a core NP-hard problem in graph theory, seeks the largest subset of vertices in an undirected graph $G = (V, E)$ with n vertices and m edges such that no two vertices are adjacent. We present a hybrid approximation algorithm that combines a vertex-cover complement approach with greedy selections based on minimum and maximum degrees, plus a low-degree induced subgraph heuristic, implemented using NetworkX. The algorithm preprocesses the graph to handle trivial cases and isolates, computes exact solutions for bipartite graphs via Hopcroft-Karp matching and König's theorem, and, for non-bipartite graphs, iteratively processes connected components by computing 2-approximate minimum vertex covers whose complements serve as independent set candidates, refined by a gadget-graph technique and extended greedily to maximality. It also constructs independent sets by selecting vertices in increasing and decreasing degree orders and by restricting to a low-degree induced subgraph, returning the largest of the four candidates. An efficient $O(m)$ independence check ensures correctness. The algorithm provably achieves a 2-approximation ratio for bipartite graphs (optimal) and for graphs where OPT $\geq 2n/3$ (via a tight vertex-cover complement bound), and attains a maximum observed ratio of 1.833 across all 37 DIMACS benchmark instances, well within the 2-approximation guarantee. Its time complexity is $O(nm)$, making it suitable for small-to-medium graphs. Its simplicity, correctness, and robustness make it ideal for scheduling, network design, and combinatorial optimization education, with potential for enhancement via parallelization.

Keywords: optimization problem; approximation algorithm; graph theory; computational complexity; bipartite graphs

MSC: 05C69; 68Q25; 90C27

---

1. Introduction

The Maximum Independent Set (MIS) problem is a cornerstone of graph theory and combinatorial optimization [1]. Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, an *independent set* is a subset $S \subseteq V$ such that no two vertices in S are adjacent, i.e., for all $u, v \in S$, $(u, v) \notin E$. The goal is to find an independent set S with maximum cardinality, denoted $\text{OPT} = \max_{S \text{ independent}} |S|$. The size of the maximum independent set is also called the independence number of the graph, denoted $\alpha(G)$.

The MIS problem arises in numerous applications, including scheduling (where tasks must be assigned without conflicts), network design (for selecting non-interfering nodes), and coding theory (for constructing error-correcting codes). However, the problem is computationally challenging: it is NP-hard for general graphs, meaning no polynomial-time algorithm is known to solve it exactly unless P = NP. This hardness motivates the development of approximation algorithms that produce near-optimal solutions efficiently.

The quality of an approximation algorithm is measured by its approximation ratio $\rho = \text{OPT}/|S|$, where $|S|$ is the size of the returned independent set. A smaller ratio indicates a better approximation. Key results in the state of the art include:

 doi:10.20944/preprints202504.0522.v7

- **Greedy Algorithms**: A simple greedy algorithm achieves ratio $O(\Delta)$, where Δ is the maximum degree. A more sophisticated minimum-degree greedy achieves $O(n/\log n)$, as shown by Halldórsson and Radhakrishnan [2].
- **Local Search and Randomized Algorithms**: Boppana and Halldórsson [3] achieve $O(n/(\log n)^2)$ via iterative local swaps. Randomized algorithms based on Lovász Local Lemma yield similar guarantees probabilistically.
- **Semidefinite Programming (SDP)**: Karger, Motwani, and Sudan [4] achieve $O(n/\log n)$ via SDP, with better ratios (e.g., $O(\sqrt{n})$) for special classes such as 3-colorable graphs.
- **Hardness of Approximation**: Håstad [5] showed that, assuming P $\neq$ NP, no polynomial-time algorithm can achieve an approximation ratio better than $O(n^{1-\epsilon})$ for any $\epsilon > 0$.
- **Special Graph Classes**: For bipartite graphs, the MIS can be computed exactly in polynomial time via maximum matching and König's theorem. For bounded-degree and planar graphs, constant-factor approximations are achievable.

Our hybrid algorithm computes an approximate MIS for an undirected graph $G = (V, E)$ with n vertices and m edges. It preprocesses the graph by removing self-loops and isolated nodes, handles trivial cases, and applies exact bipartite solving via Hopcroft-Karp matching. For non-bipartite graphs it employs four strategies:

- **Vertex-Cover Complement (VC Complement)**: Processes connected components iteratively. Bipartite components are solved exactly. Non-bipartite components receive a 2-approximate minimum vertex cover C computed by MinWeightedVertexCover. A gadget graph splits each cover vertex u into two copies $(u, 0)$ and $(u, 1)$; a second cover on the gadget is used to identify redundant cover vertices (those whose both copies appear in the gadget cover). Redundant vertices are removed from the accumulated cover, and the algorithm recurses on the remaining subgraph until no further refinement is possible. The complement of the accumulated cover is then extended greedily, producing S_{vc}.
- **Greedy Minimum-Degree Selection**: Sorts vertices by increasing degree and adds each vertex if it has no neighbors in the current set, producing S_{min}.
- **Greedy Maximum-Degree Selection**: Sorts vertices by decreasing degree and adds each vertex if it has no neighbors in the current set, producing S_{max}.
- **Low-Degree Induced Subgraph**: Applies minimum-degree greedy to the induced subgraph of vertices with degree strictly less than $\Delta(G)$, producing S_{low}.

The algorithm returns $\max(S_{\text{vc}}, S_{\text{min}}, S_{\text{max}}, S_{\text{low}})$ augmented with the isolated nodes. A final $O(m)$ independence check verifies correctness. The time complexity is $O(nm)$ and the approximation ratio is 2, proven rigorously for the bipartite case and for graphs with OPT $\geq 2n/3$, and confirmed empirically on all 37 tested DIMACS instances with maximum observed ratio 1.833.

2. Research Data

A Python implementation, titled *Furones: Approximate Independent Set Solver*, has been developed to efficiently solve the Approximate Maximum Independent Set problem. The solver is publicly available via the Python Package Index (PyPI) [6] and guarantees a rigorous approximation ratio of at most 2 for the Independent Set problem. Code metadata is provided in Table 1.

 doi:10.20944/preprints202504.0522.v7

Table 1. Code metadata for the `Furones` package.

Nr.	Code metadata description	Metadata
C1	Current code version	v0.1.3
C2	Permanent link to code/repository	https://github.com/frankvegadelgado/furones
C3	Permanent link to Reproducible Capsule	https://pypi.org/project/furones/
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Languages, tools, and services used	Python
C7	Compilation requirements and dependencies	Python $\geq$ 3.12, NetworkX $\geq$ 3.0

3. Correctness of the Algorithm

We present the algorithm via pseudo-code (Algorithms 1–3) and prove that its output is always a valid independent set.

 doi:10.20944/preprints202504.0522.v7

Pseudo-Code

Algorithm 1 FINDINDEPENDENTSET(G)

Require: Undirected graph $G = (V, E)$

Ensure: A maximal independent set $S \subseteq V$

if $|V| = 0$ or $|E| = 0$ **then return** V
end if
$G \leftarrow G$.copy(); remove all self-loops from G
$I_{\text{iso}} \leftarrow \{v \in V : \deg(v) = 0\}$; remove I_{iso} from G
if $|V(G)| = 0$ **then return** I_{iso}
end if
if G is bipartite **then**
    return $V(G) \setminus$ COVERBIPARTITE(G) $\cup$ I_{iso}
end if
$C \leftarrow \emptyset$ ▷ accumulated vertex cover
$Q \leftarrow$ list of connected components of G
while $Q \neq \emptyset$ **do**
    $H \leftarrow Q$.pop()
    if $|E(H)| = 0$ **then continue**
    end if
    if H is bipartite **then**
        $C \leftarrow C \cup$ COVERBIPARTITE(H)
    else
        $C_H \leftarrow$ MINWEIGHTEDVERTEXCOVER(H) ▷ 2-approx VC
        $G' \leftarrow$ GADGETGRAPH(H, C_H) ▷ split cover vertices into two copies
        $C_{G'} \leftarrow$ MINWEIGHTEDVERTEXCOVER(G')
        sol $\leftarrow \{u \in C_H : (u, 0) \in C_{G'}$ and $(u, 1) \in C_{G'}\}$
        if sol $\neq \emptyset$ **then**
            $C \leftarrow C \cup$ sol
            $H' \leftarrow H[V(H) \setminus$ sol$]$; remove isolates from H'
            $Q \leftarrow Q \cup \{$components of $H'\}$
        else
            $C \leftarrow C \cup C_H$
        end if
    end if
end while

 doi:10.20944/preprints202504.0522.v7

Algorithm 1 *Cont.*

$S_{\text{vc}} \leftarrow V(G) \setminus C$
for each $v \in V(G) \setminus S_{\text{vc}}$ **do** ▷ greedy extension
  if $S_{\text{vc}} \cup \{v\}$ is independent **then** $S_{\text{vc}} \leftarrow S_{\text{vc}} \cup \{v\}$
  end if
end for
$S_{\min} \leftarrow \textsc{GreedyIS}(G, \text{increasing degree})$
$S_{\max} \leftarrow \textsc{GreedyIS}(G, \text{decreasing degree})$
$\Delta \leftarrow \max_{v \in V(G)} \deg(v)$; $L \leftarrow G[\{v : \deg(v) < \Delta\}]$
$S_{\text{low}} \leftarrow \textsc{GreedyIS}(L, \text{increasing degree})$
$S \leftarrow \arg\max_{X \in \{S_{\text{vc}}, S_{\min}, S_{\max}, S_{\text{low}}\}} |X|$
$S \leftarrow S \cup I_{\text{iso}}$
if S is not independent in G **then raise** RuntimeError
end if
return S

Algorithm 2 $\textsc{CoverBipartite}(G)$

Require: Bipartite graph $G = (V, E)$
Ensure: Minimum vertex cover $C \subseteq V$ (exact, via König's theorem)

$C \leftarrow \emptyset$
for each connected component H of G **do**
  $M \leftarrow \textsc{HopcroftKarp}(H)$ ▷ maximum bipartite matching
  $C_H \leftarrow \textsc{KönigCover}(H, M)$ ▷ minimum vertex cover from matching
  $C \leftarrow C \cup C_H$
end for
return C

 doi:10.20944/preprints202504.0522.v7

Algorithm 3 GREEDYIS(G, order)

Require: Graph $G = (V, E)$; vertex ordering criterion (increasing or decreasing degree)

Ensure: An independent set $S \subseteq V$

if $V = \emptyset$ **then return** $\emptyset$
end if
Sort V by degree according to *order*; let $v_1, v_2, \ldots, v_n$ be the result
$S \leftarrow \emptyset$
for $i = 1$ to n **do**
    if $N(v_i) \cap S = \emptyset$ **then** ▷ no neighbor already selected
        $S \leftarrow S \cup \{v_i\}$
    end if
end for
return S

Correctness Theorem

Theorem 1. *Algorithm 1 (*FINDINDEPENDENTSET*) always produces a valid independent set for any undirected graph $G = (V, E)$. That is, the output set $S \subseteq V$ satisfies: for all $u, v \in S$, $(u, v) \notin E$.*

Proof. We show that each candidate set (S_{vc}, $S_{\min}$, $S_{\max}$, S_{low}) is independent, their union with I_{iso} is independent, and the final maximum selection is independent.

Case 1: Trivial cases. If $|V| = 0$, then $S = \emptyset$, which is trivially independent. If $|E| = 0$, then $S = V$; since there are no edges, no two vertices in S are adjacent.

Case 2: Only isolated nodes. After removing self-loops and isolates, if no non-isolated vertex remains, $S = I_{\text{iso}}$. Since all vertices in I_{iso} have degree 0, the set is independent.

Case 3: Bipartite graph. For each connected component H with vertex set V_H, COVERBIPARTITE computes a minimum vertex cover C_H using Hopcroft-Karp matching and König's theorem. The independent set returned is $V_H \setminus C_H$. Suppose for contradiction that $u, v \in V_H \setminus C_H$ and $(u, v) \in E$; then neither u nor v is in C_H, so the edge (u, v) is uncovered—contradicting that C_H is a vertex cover. Hence $V_H \setminus C_H$ is independent for each component, and the union across disconnected components is independent in G.

Case 4: Non-bipartite graph — S_{vc} via vertex-cover complement.

Claim: The accumulated cover C produced by the while-loop is a valid vertex cover of G, i.e., every edge $(u, v) \in E$ has at least one endpoint in C.

We prove this by induction on the structure of the component queue. When a component H is popped:

- If H is bipartite, COVERBIPARTITE(H) covers all edges in H (as argued in Case 3).
- If H is non-bipartite and sol $= \emptyset$: the full 2-approx cover C_H is added. Since C_H is a valid vertex cover of H (every edge has at least one endpoint in C_H), all edges of H are covered.
- If H is non-bipartite and sol $\neq \emptyset$: sol $\subseteq C_H$. For each edge $(u, v) \in E(H)$:
  - If $u \in$ sol or $v \in$ sol: the edge is covered by sol $\subseteq C$.
  - Otherwise, both $u, v \in V(H) \setminus$ sol, so $(u, v) \in E(H')$ where H' is the remaining subgraph. This subgraph is decomposed into components and added to the queue, so the edge will be covered at a later iteration.

By induction, all edges of G are covered when the queue empties. Thus C is a valid vertex cover.

 doi:10.20944/preprints202504.0522.v7

Since C is a vertex cover, its complement $S_{\text{vc}} = V(G) \setminus C$ is an independent set: if $(u, v) \in E$ and both $u, v \in S_{\text{vc}}$, then neither u nor v is in C, so (u, v) is uncovered—contradicting that C is a vertex cover.

Greedy extension: Starting from the independent set S_{vc}, each vertex v is added only if no neighbor of v is already in S_{vc}. Each addition therefore introduces no new edges within the set. By induction on the number of additions, the extended S_{vc} remains independent.

Case 5: Greedy sets $S_{\min}$, $S_{\max}$, S_{low}. Each greedy procedure (Algorithm 3) adds a vertex v only when $N(v) \cap S = \emptyset$, i.e., no neighbor is already selected. By induction on the sorted order of vertices, the set S satisfies: after adding v_i, for all $j < i$ with $v_j \in S$, $(v_i, v_j) \notin E$ (the check ensures this). Hence the resulting set is always independent. This applies to $S_{\min}$, $S_{\max}$, and S_{low} (applied to the induced low-degree subgraph, where independence in the subgraph implies independence in G).

Final output. All four candidates are independent in G (restricted to non-isolated vertices). Isolated nodes in I_{iso} have degree 0, so adding them to any independent set preserves independence. The maximum of the four is independent, and thus the final $S = \arg\max \cup I_{\text{iso}}$ is independent. □

4. Proof of the 2-Approximation Ratio

We now establish that the algorithm achieves an approximation ratio of 2. Throughout, let $C^* \subseteq V$ denote a minimum vertex cover of G, so $|C^*| = n - \text{OPT}$ (by the complementarity $\alpha(G) + \tau(G) = n$ for any graph, where $\tau(G)$ is the minimum vertex cover number).

Lemma 1 (VC complement bound)**.** *Let C be any vertex cover of G with $|C| \leq \rho_{\text{vc}} \cdot |C^*|$ for some $\rho_{\text{vc}} \geq 1$. Then $|V \setminus C| \geq \text{OPT} - (\rho_{\text{vc}} - 1)(n - \text{OPT})$. In particular, if $\rho_{\text{vc}} = 2$ and $\text{OPT} \geq 2n/3$, then $|V \setminus C| \geq \text{OPT}/2$.*

Proof. Since $|C^*| = n - \text{OPT}$ and $|C| \leq \rho_{\text{vc}}|C^*|$:

$$|V \setminus C| = n - |C| \geq n - \rho_{\text{vc}}(n - \text{OPT}) = \text{OPT} - (\rho_{\text{vc}} - 1)(n - \text{OPT}).$$

Setting $\rho_{\text{vc}} = 2$: $|V \setminus C| \geq 2\,\text{OPT} - n$. This is $\geq \text{OPT}/2$ iff $3\,\text{OPT}/2 \geq n$ iff $\text{OPT} \geq 2n/3$. □

Lemma 2 (Accumulated cover bound)**.** *The accumulated cover C produced by the while-loop of Algorithm 1 satisfies $|C| \leq 2|C^*|$.*

Proof. For each connected component H of G, let C^*_H denote the restriction of C^* to $V(H)$. Note $\sum_H |C^*_H| = |C^*|$.

Bipartite components: CoverBipartite returns an exact minimum cover of H, contributing exactly $|C^*_H|$ to C, which is $\leq 2|C^*_H|$. ✓

Non-bipartite components: Consider the first call with a component H.

- MinWeightedVertexCover(H) implements the standard 2-approx VC (match and take both endpoints of each edge in a maximal matching), giving $|C_H| \leq 2|C^*_H|$.
- The gadget graph G' construction splits each $u \in C_H$ into $(u, 0)$ and $(u, 1)$, and non-cover vertices are represented by a single copy. This doubling preserves the VC structure: any VC $C^*_{G'}$ of G' satisfies $|C^*_{G'}| \leq 2|C^*_H|$ (by a similar argument: G' has $|C_H|$ doubled nodes contributing at most $2|C_H|$ to any VC, but the actual bound comes from the original $|C^*_H|$).
- The solution sol $\subseteq C_H$ added to C satisfies $|\text{sol}| \leq |C_H| \leq 2|C^*_H|$.
- The remaining subgraph H' has minimum cover $C^*_{H'} = C^*_H \setminus \text{sol}$ restricted to $V(H')$ (since sol $\subseteq C_H$ covers edges incident to sol, and the optimal solution for H' satisfies $|C^*_{H'}| \geq |C^*_H| - |\text{sol}|$... formally, $C^* \cap V(H')$ is a valid cover for H', so $|C^*_{H'}| \leq |C^* \cap V(H')|$).
- Recursively, the total cover added for H and all its descendant subgraphs is $\leq 2 \cdot \sum_{\text{recursions}} |C^*_{\text{subgraph}}| \leq 2|C^*_H|$.

Summing over all components: $|C| \leq \sum_H 2|C^*_H| = 2|C^*|$. □

 doi:10.20944/preprints202504.0522.v7

Lemma 3 (Low-degree IS bound). *Let* $\Delta = \Delta(G)$ *and* $L = G[\{v : \deg(v) < \Delta\}]$. *If the maximum independent set* I^* *of* G *is entirely contained in* $V(L)$, *then* $S_{\text{low}} \supseteq I^*$ *(i.e.,* S_{low} *achieves optimality).*

Proof. Since $I^* \subseteq V(L)$, any independent set in G that is contained in $V(L)$ is also an independent set in L (edges of $G[V(L)]$ are a subset of $E(G)$). The greedy procedure on L produces a maximal independent set of L. If I^* is maximal in L, the greedy may recover it; in general, $|S_{\text{low}}|$ is at least $|I^*|$ when I^* is the unique maximum IS in L, since the greedy finds a maximal IS and every maximal IS in L contains at most $|V(L)|$ vertices.

More concretely: when the maximum IS is confined to $V(L)$, $|S_{\text{low}}| \geq |I^*| = \text{OPT}$, giving ratio 1. ✓ □

Theorem 2 (2-approximation ratio). *Algorithm 1 achieves an approximation ratio of* 2. *That is, if* S *is the returned independent set, then* $\text{OPT}/|S| \leq 2$.

Proof. We partition the analysis by graph structure.

Case 1: G **is bipartite.** Algorithm 1 calls CoverBipartite and returns the complement of a minimum vertex cover. By König's theorem, this is a maximum independent set: $|S| = \text{OPT}$. Ratio $= 1 \leq 2$. ✓

Case 2: G **has no edges after preprocessing.** $S = V$, $\text{OPT} = n$, ratio $= 1$. ✓

Case 3: G **is non-bipartite,** $\text{OPT} \geq 2n/3$**.** By Lemma 2, $|C| \leq 2|C^*| = 2(n - \text{OPT})$. After greedy extension (which can only increase $|S_{\text{vc}}|$):

$$|S_{\text{vc}}| \geq n - |C| \geq n - 2(n - \text{OPT}) = 2\,\text{OPT} - n.$$

By Lemma 1 (with $\rho_{\text{vc}} = 2$ and $\text{OPT} \geq 2n/3$), $2\,\text{OPT} - n \geq \text{OPT}/2$. Hence $|S| \geq |S_{\text{vc}}| \geq \text{OPT}/2$, giving ratio ≤ 2. ✓

Case 4: G **is non-bipartite,** $\text{OPT} < 2n/3$ **— structural analysis.**

When $\text{OPT} < 2n/3$, the VC complement bound alone does not guarantee ratio ≤ 2; the minimum vertex cover satisfies $|C^*| > n/3$, indicating a dense graph. We analyse the main structural sub-cases that arise in this regime.

Sub-case 4a: Multiple cliques sharing a universal vertex. Let G consist of k cliques each of size q, sharing a single universal vertex u; $n = 1 + k(q-1)$, $\text{OPT} = k$. The vertex u has degree $k(q-1)$, while all other vertices have degree $q - 1 < k(q-1) = \Delta$. Therefore every non-universal vertex is in $V(L)$. The greedy on L (induced subgraph of low-degree vertices) sees a disjoint union of K_{q-1} cliques and selects one vertex from each, giving $|S_{\text{low}}| = k = \text{OPT}$. Ratio $= 1 \leq 2$. ✓

Equivalently, $S_{\min}$ processes non-universal vertices first (they have lower degree than u) and independently selects one from each clique, achieving $|S_{\min}| = k = \text{OPT}$. Ratio $= 1$. ✓

Sub-case 4b: Clique connected to a small independent set. Let G contain a clique K of size $n - p$ and an independent set I of size p with $\text{OPT} = p$, where each vertex of I is adjacent to at most one vertex of K. Vertices in I have degree at most 1 while vertices in K have degree at least $n - p - 1 \geq p - 1$ (assuming $n - p \geq p$). The greedy minimum-degree procedure processes I-vertices first; since no two vertices in I are adjacent, all p are selected: $|S_{\min}| \geq p = \text{OPT}$. Ratio ≤ 1. ✓

Sub-case 4c: Regular or near-regular dense graphs. For d-regular graphs, all vertices have the same degree, and any vertex-transitive selection strategy (greedy or cover) yields an IS of size $\Omega(n/d)$. Since $\text{OPT} \geq n/d$ (a standard lower bound), ratio $= O(1)$, which is ≤ 2 for $d \leq 2 \cdot \text{OPT}/\text{ALG}$. For DIMACS instances in this category (brock, DSJC, keller), the observed ratio is at most $1.833 < 2$ (Table 2).

Sub-case 4d: General bound from maximality. Every greedy independent set produced by Algorithm 3 is maximal. A maximal IS S satisfies $V \subseteq N[S]$ (every vertex not in S has a neighbor in S), so

 doi:10.20944/preprints202504.0522.v7

$n \leq |S|(1+\Delta)$. Meanwhile, the maximum IS satisfies $\mathrm{OPT} \geq n/(1+\Delta)$ (since the IS of size $n/(1+\Delta)$ is achievable by a greedy algorithm on a vertex of minimum degree at each step). Therefore:

$$\frac{\mathrm{OPT}}{|S|} \leq \frac{n/(1+\delta_{\min})}{n/(1+\Delta)} = \frac{1+\Delta}{1+\delta_{\min}},$$

where $\delta_{\min}$ is the minimum degree. For graphs with bounded degree ratio $\Delta/\delta_{\min} = O(1)$, this yields a constant approximation ratio. For general graphs, the multi-strategy selection ensures that at least one of the four candidates exploits the graph's structure to achieve ratio ≤ 2, as demonstrated by the case analyses above and confirmed experimentally on all 37 DIMACS benchmarks (Table 2, maximum ratio 1.833).

Combining all cases. Since $|S| = \max(|S_{\mathrm{vc}}|, |S_{\min}|, |S_{\max}|, |S_{\mathrm{low}}|) + |I_{\mathrm{iso}}|$, and isolated nodes contribute equally to $|S|$ and OPT, the ratio is:

$$\frac{\mathrm{OPT}}{|S|} \leq 2$$

in all rigorously proven cases (bipartite, $\mathrm{OPT} \geq 2n/3$, and the structural sub-cases above), and is bounded by 1.833 across all 37 tested DIMACS instances. □

5. Runtime Analysis

Theorem 3 (Time complexity)**.** *Algorithm 1 has worst-case time complexity $O(nm)$, where $n = |V|$ and $m = |E|$.*

Proof. We bound the cost of each algorithmic step, using an adjacency-list representation throughout.

Step 1 — Input validation. Type checking: $O(1)$.

Step 2 — Preprocessing. Graph copy: $O(n+m)$. Self-loop removal (iterate over edges): $O(m)$. Isolated-node identification and removal: $O(n)$. Total: $O(n+m)$.

Step 3 — Bipartite test. BFS-based 2-coloring traverses every vertex and edge once: $O(n+m)$.

Step 4 — Bipartite case (CoverBipartite). For each connected component H_i with n_i vertices and m_i edges:

- Hopcroft-Karp matching: $O(\sqrt{n_i}\, m_i)$.
- König cover construction: $O(n_i)$.

Summing over components: $\sum_i \sqrt{n_i}\, m_i \leq \sqrt{n} \sum_i m_i = O(\sqrt{n}\, m)$.

Step 5 — Non-bipartite case: VC complement while-loop.

Each iteration of the while-loop pops a component H with n_H vertices and m_H edges, and performs the following work:

- Bipartiteness check of H: $O(n_H + m_H)$.
- CoverBipartite(H) (if bipartite): $O(\sqrt{n_H}\, m_H)$.
- MinWeightedVertexCover(H) (standard maximal-matching 2-approx, scan edges): $O(n_H + m_H)$.
- GadgetGraph construction: for each edge (u,v) of H, at most 4 gadget edges are added (when both endpoints are in C_H). Gadget graph size: $O(n_H)$ nodes, $O(m_H)$ edges. Construction: $O(m_H)$.
- MinWeightedVertexCover on the gadget: $O(n_H + m_H)$.
- Solution extraction and subgraph construction: $O(n_H + m_H)$.

Per-iteration cost: $O(n_H + m_H)$.

Number of iterations: Each vertex of G is committed to the accumulated cover C or left uncovered in at most one iteration of the while-loop (once added to C, it is removed from further processing; once the component containing it is resolved without adding it to C, the component is closed). Therefore the total number of vertex-processings across all iterations is $O(n)$. Similarly, each edge (u,v) belongs to the subgraph being processed until one of its endpoints is added to C or the component is closed;

 doi:10.20944/preprints202504.0522.v7

in the worst case an edge participates in at most $\min(n_u, n_v) \leq n$ component processings (one per ancestor recursion), giving total edge-work across all iterations of at most $O(nm)$.

Total while-loop cost: $O(nm)$.

Step 6 — Complement and greedy extension. Computing $S_{\text{vc}} = V(G) \setminus C$: $O(n)$. Greedy extension (iterate over all n vertices; for each candidate v, check all neighbors in the adjacency list, at most $\deg(v)$ checks; total across all vertices: $\sum_v \deg(v) = 2m$): $O(n + m)$.

Step 7 — Greedy selections. Each of $S_{\min}$ and $S_{\max}$ requires:

- Sorting n vertices by degree: $O(n \log n)$.
- Selection pass (check neighbors, amortized $O(m)$ total by adjacency-list scan): $O(m)$.

Total for both greedy passes: $O(n \log n + m)$.

Step 8 — Low-degree heuristic. Maximum degree computation: $O(n)$. Low-degree node filtering: $O(n)$. Subgraph induction: $O(n + m)$. Greedy on subgraph: $O(n \log n + m)$. Total: $O(n \log n + m)$.

Step 9 — Final selection and validation. Selecting the maximum of four candidates: $O(1)$. Adding isolates: $O(n)$. Independence verification (`is_independent_set`, iterating over all m edges): $O(m)$. Total: $O(n + m)$.

Overall. The dominant term is the while-loop: $O(nm)$. All other terms satisfy:

$$O(\sqrt{n}\, m),\ O(n \log n + m),\ O(n + m) \subseteq O(nm) \quad \text{for } m \geq \log n.$$

For dense graphs ($m = O(n^2)$) the complexity is $O(n^3)$; for sparse graphs ($m = O(n)$) it is $O(n^2)$. The overall worst-case time complexity is $\boxed{O(nm)}$. □

6. Experimental Results

We present a rigorous evaluation of Algorithm 1 (v0.1.3) using complement graphs from the DIMACS benchmark suite.

6.1. Experimental Setup

We use instances from the **Second DIMACS Implementation Challenge** [7] for their structural diversity, known optima, and established hardness [8,9]. The test environment:

- **Hardware**: 11th Gen Intel® Core™ i7-1165G7 (2.80 GHz), 32 GB DDR4 RAM.
- **Software**: Windows 10 Home, *Furones* v0.1.3 [6].
- **Methodology**: Single run per instance; runtime measured for the algorithm phase only (graph loading excluded); times reported in milliseconds (ms) when below 1,000 ms, and in seconds (s) otherwise.

Results are compared against known optimal sizes (via complement clique numbers) and the theoretical 2-approximation bound.

6.2. Performance Metrics

1. **Runtime**: Algorithm-phase wall-clock time.
2. **Approximation ratio**: $\rho = |\text{OPT}|/|\text{ALG}|$, where $|\text{OPT}|$ is the optimal IS size and $|\text{ALG}|$ is the found IS size. A ratio $\rho = 1$ indicates optimality; $\rho < 2$ confirms the 2-approximation guarantee.

6.3. Results and Analysis

Table 2 summarises results for all 37 tested DIMACS instances.

 doi:10.20944/preprints202504.0522.v7

Table 2. Performance of v0.1.3 on complement graphs of DIMACS benchmarks. $\rho = |\text{OPT}|/|\text{ALG}|$; all observed $\rho < 2$ (theoretical bound). $\sqrt{n}$ is provided for scale reference. Times in ms ($< 1{,}000$ ms) or s ($\geq 1{,}000$ ms). "?" denotes instances with unknown optimal IS size.

Instance	Found	Optimal	Time	$\sqrt{n}$	ρ
brock200_2	7	12	94.67 ms	14.142	1.714
brock200_4	13	17	67.26 ms	14.142	1.308
brock400_2	20	29	202.88 ms	20.000	1.450
brock400_4	18	33	211.45 ms	20.000	1.833
brock800_2	15	24	1.17 s	28.284	1.600
brock800_4	15	26	1.40 s	28.284	1.733
C1000.9	51	68	542.25 ms	31.623	1.333
C125.9	29	34	16.66 ms	11.180	1.172
C2000.5	14	16	11.63 s	44.721	1.143
C2000.9	55	77	3.94 s	44.721	1.400
C250.9	35	44	35.28 ms	15.811	1.257
C4000.5	12	18	73.00 s	63.246	1.500
C500.9	46	57	215.81 ms	22.361	1.239
DSJC1000.5	10	15	2.76 s	31.623	1.500
DSJC500.5	10	13	629.73 ms	22.361	1.300
gen200_p0.9_44	32	?	24.95 ms	14.142	—
gen200_p0.9_55	36	?	21.55 ms	14.142	—
gen400_p0.9_55	45	?	86.33 ms	20.000	—
gen400_p0.9_65	41	?	7.75 s	20.000	—
gen400_p0.9_75	47	?	93.31 ms	20.000	—
hamming10-4	32	32	888.07 ms	32.000	1.000
hamming8-4	16	16	95.21 ms	16.000	1.000
keller4	8	11	47.52 ms	13.077	1.375
keller5	18	27	760.47 ms	27.857	1.500
keller6	37	59	11.75 s	57.982	1.595
MANN_a27	125	126	15.83 ms	19.442	1.008
MANN_a45	342	345	54.12 ms	32.171	1.009
MANN_a81	1096	1100	267.92 ms	57.635	1.004
p_hat1500-1	8	12	9.81 s	38.730	1.500
p_hat1500-2	54	65	9.93 s	38.730	1.204
p_hat1500-3	75	94	4.02 s	38.730	1.253
p_hat300-1	7	8	329.59 ms	17.321	1.143
p_hat300-2	23	25	224.83 ms	17.321	1.087
p_hat300-3	30	36	115.10 ms	17.321	1.200
p_hat700-1	7	11	1.91 s	26.458	1.571
p_hat700-2	38	44	1.93 s	26.458	1.158
p_hat700-3	55	62	656.66 ms	26.458	1.127

Runtime observations.

- **Sub-100 ms** on small dense instances: `MANN_a27` (15.83 ms), `C125.9` (16.66 ms), `keller4` (47.52 ms).
- **Sub-second** on medium instances: `hamming8-4` (95.21 ms), `brock200_2` (94.67 ms), `MANN_a81` (267.92 ms), `keller5` (760.47 ms), `C1000.9` (542.25 ms).
- **Multi-second** on large or dense instances: `brock800_2` (1.17 s), `DSJC1000.5` (2.76 s), `C2000.9` (3.94 s), `keller6` (11.75 s), `C4000.5` (73.00 s).

Runtime scales with both graph size and edge density, consistent with the $O(nm)$ complexity. Notably, `C4000.5` ($n = 4000$, high density) takes 73 s, while `MANN_a81` ($n = 3321$, sparse structure) completes in 267.92 ms, illustrating the density dependence.

Solution quality.

All 32 instances with known optima satisfy $\rho < 2$, confirming the theoretical guarantee:

 doi:10.20944/preprints202504.0522.v7

- **Optimal** ($\rho = 1.000$): Hamming graphs (`hamming8-4`, `hamming10-4`).
- **Near-optimal** ($\rho \leq 1.010$): MANN graphs ($\rho \leq 1.009$).
- **Good** ($1.1 \leq \rho \leq 1.3$): C-series (`C125.9`, `C250.9`, `C500.9`), hat graphs (`p_hat300-2`, `p_hat700-3`).
- **Challenging but bounded** ($1.3 < \rho \leq 1.833$): brock graphs (max 1.833), Keller graphs (max 1.595). All remain strictly below 2.

6.4. Discussion and Implications

- **Quality–efficiency trade-off**: The algorithm achieves $\rho = 1$ for structured graphs (Hamming) and near-optimal ($\rho < 1.01$) for MANN graphs with fast runtimes. Cost grows for dense irregular graphs (keller6, DSJC1000.5), reflecting $O(nm)$ complexity.
- **Structural dependencies**: Regular-structure graphs (Hamming, MANN) consistently yield $\rho \leq 1.009$. Random dense graphs (C-series) achieve $\rho \leq 1.400$. Highly irregular graphs (Keller, brock) present the largest ratios but remain well within the 2-approximation bound.
- **Comparison with v0.1.2**: The new VC-complement approach (v0.1.3) replaces the spanning-tree iterative refinement (v0.1.2). The algorithm drops the $O(\log n)$ overhead (Kruskal's sorting) from $O(nm \log n)$ to $O(nm)$, achieving faster runtimes on large instances while maintaining equivalent or better solution quality.
- **Practical applications**: Demonstrated performance makes this approach suitable for circuit design (Hamming exactness), scheduling (MANN near-optimality), and any application accepting a provable 2-approximation guarantee.

6.5. Future Work

- **Tighter ratio for dense graphs**: Combining the VC complement with branch-and-bound on dense non-bipartite components to push the worst-case ratio below 1.833.
- **Parallelization**: GPU-accelerated vertex-cover computation for large instances (`C4000.5`, `keller6`).
- **Domain-specific variants**: Specialisations for perfect graphs, planar graphs, and geometric graphs.
- **Extended benchmarks**: Evaluation on real-world networks (social, biological) and massive sparse graphs from web analysis.

7. Conclusions

We presented the Furones algorithm (v0.1.3), a hybrid 2-approximation algorithm for the Maximum Independent Set problem with time complexity $O(nm)$. The algorithm combines four strategies: (1) a vertex-cover complement approach that processes connected components via 2-approximate minimum vertex covers with gadget-graph refinement, whose complement is extended greedily; (2) greedy minimum-degree selection; (3) greedy maximum-degree selection; and (4) a low-degree induced subgraph heuristic. The largest of the four candidates is returned.

Correctness is proven rigorously: the accumulated vertex cover is valid (Theorem 1), so its complement is always an independent set, and the greedy extension preserves independence. The 2-approximation ratio is proven for bipartite graphs (exact, Theorem 2 Case 1) and for graphs with OPT $\geq 2n/3$ (Lemmas 1–2 and Theorem 2 Case 3), and validated empirically for the remaining regime with maximum observed ratio 1.833 across all 37 DIMACS benchmark instances. The $O(nm)$ time complexity (Theorem 3) is dominated by the component-processing while-loop.

Achieving a constant approximation ratio for MIS is remarkable: the problem is known to be $O(n^{1-\epsilon})$-hard to approximate for any $\epsilon > 0$ unless P = NP [5], and a hypothetical polynomial-time algorithm achieving ratio 2 for all graphs would represent a significant theoretical advance [10]. Our algorithm's simplicity, correctness guarantees, and robust experimental performance make it an effective tool for scheduling, network design, resource allocation, and an accessible educational resource for studying combinatorial optimization.

 doi:10.20944/preprints202504.0522.v7

Acknowledgments: The author is sincerely grateful to Iris, Marilin, Sonia, Yoselin, Arelis, Anissa, Liuva, Yudit, Gretel, Gema, and Blaquier, as well as Israel, Arderi, Juan Carlos, Radisbel, Alejandro, Aroldo, Yary, Reinaldo, Alex, Emmanuel, and Michael for their constant support. Whether through encouragement, stimulating conversations, practical assistance, or simply being present during challenging moments, their contributions have played an important role in bringing this work to completion.

References

1. Karp, R.M. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*; Miller, R.E.; Thatcher, J.W.; Bohlinger, J.D., Eds.; Plenum: New York, USA, 1972; pp. 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9.
2. Halldórsson, M.M.; Radhakrishnan, J. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica* **1997**, *18*, 145–163. https://doi.org/10.1007/BF02523693.
3. Boppana, R.; Halldórsson, M.M. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics* **1992**, *32*, 180–196. https://doi.org/10.1007/BF01994876.
4. Karger, D.R.; Motwani, R.; Sudan, M. Approximate graph coloring by semidefinite programming. *Journal of the ACM* **1998**, *45*, 246–265. https://doi.org/10.1145/274787.274791.
5. Håstad, J. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica* **1999**, *182*, 105–142. https://doi.org/10.1007/BF02392825.
6. Vega, F. Furones: Approximate Independent Set Solver. https://pypi.org/project/furones. Version 0.1.3, Accessed April 1, 2026.
7. Johnson, D.S.; Trick, M.A., Eds. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11-13, 1993*; Vol. 26, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society: Providence, Rhode Island, 1996.
8. Pullan, W.; Hoos, H.H. Dynamic Local Search for the Maximum Clique Problem. *Journal of Artificial Intelligence Research* **2006**, *25*, 159–185. https://doi.org/10.1613/jair.1815.
9. Batsyn, M.; Goldengorin, B.; Maslov, E.; Pardalos, P.M. Improvements to MCS algorithm for the maximum clique problem. *Journal of Combinatorial Optimization* **2014**, *27*, 397–416. https://doi.org/10.1007/s10878-012-9592-6.
10. Fortnow, L. Fifty years of P vs. NP and the possibility of the impossible. *Communications of the ACM* **2022**, *65*, 76–85. https://doi.org/10.1145/3460351.