Approximation Algorithm for the Minimum Hub Cover Set Problem

JOEL A. TREJO-SÁNCHEZ1, CANDELARIA E. SANSORES-PÉREZ2, JESÚS GARCÍA-DÍAZ3,4, AND JOSÉ ALBERTO FERNÁNDEZ-ZEPEDA5

1CONACYT—Center for Research in Mathematics, Carretera Sierra Papacal, Chuburna Puerto, Mérida 97302, Mexico
2Department of Basic Sciences and Engineering, Universidad del Caribe, Cancún 77528, Mexico
3Consejo Nacional de Ciencia y Tecnología, Mexico City 03940, Mexico
4Institute Nacional de Astrofisica, Optica y Electronica, Coordinacion de Ciencias Computacionales, Puebla 72840, Mexico
5Department of Computer Science, CICESE Research Center, Ensenada 22860, Mexico

Corresponding author: Candelaria E. Sansores-Pérez (csansores@ucaribe.edu.mx)

This work was supported in part by the National Council of Science and Technology, Mexico (CONACYT) under Grand Frontier Science Project 304006; and in part by the Department of Basic Sciences and Engineering, Universidad del Caribe.

ABSTRACT A subset $S \subseteq V$ of vertices of an undirected graph $G = (V, E)$ is a hub cover when for each edge $(u, v) \in E$, at least one of its endpoints belongs to $S$, or there exists a vertex $r \in S$ that is a neighbor of both $u$ and $v$. The problem of computing a minimum hub cover set in arbitrary graphs is NP-hard. This problem has applications for indexing large databases. This paper proposes $\Psi$-MHC, the first approximation algorithm for the minimum hub cover set in arbitrary graphs to the best of our knowledge. The approximation ratio of this algorithm is $\ln \mu$, where $\mu$ is upper bounded by $\min\{\frac{1}{2}(\Delta + 1)^2, |E|\}$ and $\Delta$ is the degree of $G$. The execution time of $\Psi$-MHC is $O((\Delta + 1)|E| + |S||V|)$. Experimental results show that $\Psi$-MHC far outperforms the theoretical approximation ratio for the input graph instances.

INDEX TERMS Approximation algorithms, heuristics, minimum hub cover set, optimization.

I. INTRODUCTION

Let $G = (V, E)$ be an undirected graph. A subset $S \subseteq V$ is a hub cover if for each edge $(u, v) \in E$ any of the following conditions hold.

1. $u \in S$
2. $v \in S$
3. $r \in S \land (r, u) \in E \land (r, v) \in E$

Computing a hub cover set has important applications in diverse areas, including query processing of large databases [1], [2], and graph isomorphism [3]. The smaller the cardinality of a hub cover set, the faster the query search in large databases [2]. The minimum hub cover set problem is computing the smallest cardinality hub cover set for a given undirected graph. Given an integer $k$, the hub cover set problem’s decision version consists of finding a hub cover set whose cardinality is at most $k$. Yelbay et al. proved that the minimum hub cover set problem is NP-hard in triangle-free graphs (graphs containing no cliques of size three). We extended this result to the case of graphs of girth 3 (graphs containing triangles).

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Alatas.

Notice the similarity between computing a minimum hub cover set and computing a minimum vertex cover set, a well-known NP-hard problem [4]. Recall that for a given undirected graph $G = (V, E)$, a vertex cover is a set $C \subseteq V$ such that for each edge $(u, v) \in E$, at least one vertex $u$ or $v$ belongs to $C$. Computing a hub cover set in triangle-free graphs is equivalent to computing a vertex cover set in such graphs.

There exist several algorithms dealing with the minimum vertex cover problem. Recently, Wang et al. [5] proposed an exact algorithm based on branch-and-bound that solves this problem in arbitrary graphs. Although they do not present an analysis of the algorithm’s execution time, their algorithm is time-consuming for graphs of small size ($< 1000$ vertices). Computing a vertex cover helps to solve some NP-hard problems in graphs. For instance, Cygan and Pilipczuk [6] used the best exact algorithm for the vertex cover to design an FPT exact exponential-time algorithm for a problem similar to the vertex cover.

Approximation algorithms [7] are polynomial-time algorithms that produce solutions approximate to the optimal in a fraction of the time required by an exact algorithm. Some interesting graph problems use approximation algorithms to deal with NP-hard problems in graphs. For the case of
the vertex cover, some heuristics exist to compute approximate solutions. Bhattacharya et al. [8] designed a \((2 + \epsilon)\)-approximate vertex cover in \(O(\log^3 n)\) time. Karakostas [9] provided a \((2 - o(1))\)-approximate minimum vertex cover for an arbitrary graph. Dinur et al. [10] proved that the problem of designing a polynomial-time approximation algorithm for the minimum vertex cover within a factor smaller than \(1.3606\) is \(NP\)-hard. Later, Khot and Regev [11] showed that approximating the minimum vertex cover with a factor of \((2 - \epsilon)\) is likely \(NP\)-hard for a constant \(\epsilon\).

There are only a few algorithms related to the minimum hub cover set problem. Yelbay et al. [12] proposed an approximation algorithm for this problem on planar graphs. Their algorithm transforms the planar graph \(G\) into a \(\lambda\)-outerplanar graph \(G'\). Then, it decomposes \(G'\) into a set of small outerplanar subgraphs. After that, it uses a solver to compute a minimum hub cover set into each subgraph of \(G'\). Finally, it merges the solutions. The approximation factor of their algorithm is \(\frac{\lambda + 1}{\lambda}\). Yelbay’s Ph.D. dissertation [13] provides further information about the minimum hub cover set. In particular, Yelbay also proved that computing a minimum hub cover set on triangle-free graphs is an \(NP\)-hard problem. In the present paper, we demonstrated that this problem is also \(NP\)-complete for graphs of girth 3.

This paper proposes \(\Psi\)-MHC, the first approximation algorithm for the minimum hub cover set in arbitrary graphs. This algorithm does not use a solver to execute subroutines. We organized the rest of the paper as follows. Section II presents the notation and some concepts related to \(\Psi\)-MHC. Section III describes \(\Psi\)-MHC. Section IV proves the correctness of \(\Psi\)-MHC and analyzes its execution time. Section V proves that the minimum hub cover set problem on arbitrary graphs is \(NP\)-complete. Section VI describes the experimental design. Section VII analyzes and discusses the experimental results. Finally, Section VIII presents some concluding remarks.

II. TERMINOLOGY AND NOTATION

Let \(G = (V, E)\) be an undirected graph. The order of \(G\) is \(|V|\), whereas the size of \(G\) is \(|E|\). The neighborhood of a vertex \(v \in V\), denoted by \(N(v)\), consists of all the vertices \(u \in V\) such that \((u, v) \in E\). The closed neighborhood of \(v\) is the set \(N[v] = N(v) \cup \{v\}\). The degree of any vertex \(v \in V\), denoted by \(\delta(v)\), consists of the cardinality of its neighborhood. The degree of \(G\), denoted by \(\Delta\), is the maximum value of \(\delta(v)\) for all \(v \in V\). A graph \(G\) is dense when \(|E| = O(|V|^2)\), whereas it is sparse when \(|E| = O(|V|)\). An edge \((x, y) \in E\) is a crossing-edge of vertex \(v\) if \(v \in N(x) \cap N(y)\). Let \(F(v)\) be the set of edges that vertex \(v\) can cover; specifically, \(F(v) = C(v) \cup I(v)\), where \(C(v)\) is the set of crossing-edges of \(v\) and \(I(v)\) is the set of edges incident to \(v\). In Fig. 1, \(C(a) = \{(b, d), (c, d)\}\), \(I(a) = \{(a, b), (a, c), (a, d)\}\), therefore, \(F(a) = \{(a, b), (a, c), (a, d), (b, d), (c, d)\}\). Let \(R(e)\) be the set of vertices that can cover edge \(e\). In Fig. 1, \(R((a, d)) = \{a, b, c, d\}\).

III. APPROXIMATION ALGORITHM FOR THE MINIMUM HUB COVER SET

This section describes \(\Psi\)-MHC. This algorithm receives as input an arbitrary undirected graph \(G = (V, E)\) and returns a hub cover set \(S \subseteq V\). \(\Psi\)-MHC follows a greedy strategy, so it includes in \(S\) the vertex that can cover the greatest number of uncovered edges in each iteration. After including a vertex in \(S\), the algorithm marks the edges covered by such vertex as cover.

The algorithm uses a max-priority queue \(Q\), based on a max-heap, to efficiently manage the number of uncovered edges in each set \(F(v)\) for all \(v\). We use the following variables during the description of the algorithm. We assume that each vertex has a unique identifier from the set \([1, \ldots, |V|]\). Some of the variables have additional information (named satellite data). We use a superscript to refer to the satellite data and emphasize this information in some parts of the algorithm.

- The set \(S\) stores the vertices of the computed hub cover set.
- The set \(F(v)\) contains all the edges vertex \(v\) can cover.
- The set \(R(e)\) contains all the vertices that can cover edge \(e\). Each vertex \(v\) stored in \(R(e)\) has edge \(e\) as satellite data, denoting it by \(v^e\).
- The Boolean variable \(cover(e)\) is true if some vertex covers the edge \(e\); otherwise, this variable is false.
- The list \(Adj(v)\) stores the set of vertices adjacent to vertex \(v\).
- The array \(E_r\) stores the number of uncovered edges of the set \(F(v)\), for all \(v \in V\), at the end of iteration \(r\).
- The array \(L_r\) stores, in a non-decreasing way, the set of vertices adjacent to vertex \(v\).
- The array \(L\) stores the output of algorithm \texttt{MERGE}.
- The arrays \(T\) and \(T'\) store vertices. Each vertex is attached to an edge as satellite data.
- The array \(U\) stores Boolean values. The value of \(U(v)\) is true if during the current iteration, at least one edge in \(F(v)\) was covered; otherwise, its value is false.
- The expression \(+e\) indicates the satellite-data edge associated to vertex \(v\).

The following functions and algorithms facilitate the description of \(\Psi\)-MHC.

- The algorithm \texttt{COUNTING-SORT} sorts the input array with at most \(|V|\) different values in \(O(|M|)\) time, where \(M\) is the size of the input array.
The algorithm **MERGE** takes as input two sorted arrays and returns a single sorted array in $O(M)$ time, where $M$ is the size of the output array.

The function **EXTRACT-MAX** extracts and removes the element with the highest priority in $Q$ in $O(\log |V|)$ time.

The function **BUILD-MAX-HEAP** constructs a max-heap $Q$ with the elements of array $E_t$ in $O(|V|)$ time.

The function **DECREASE-KEY** decreases in $Q$ the priority of an element $i$ to the value $E_t(i)$ in $O(\log |V|)$ time.

Algorithm 1 Pseudocode of INIT($G$)

**Require:** An undirected graph $G = (V, E)$.

**Ensure:** Initializes most of the variables of the set $\Psi$-MHC, computes the set $F(u)$ for each $u \in V$, and creates the max-priority queue $Q$.

1: $S \leftarrow \emptyset$
2: $Q \leftarrow \emptyset$
3: $T \leftarrow \emptyset$
4: $E' \leftarrow E$
5: for each $u \in V$ do
6:     $F(u) \leftarrow \emptyset$
7:     $L_v \leftarrow $COUNTING-SORT$(\text{Adj}(v), |V|)$
8: end for
9: for each $e = (x, y) \in E$ do
10:    $\text{cover}(e) \leftarrow \text{false}$
11:    $R(e) \leftarrow [x^e \cup y^e]$
12:    $L \leftarrow $MERGE$(L_x, L_y)$
13: for $i \leftarrow 1$ to $|L| - 1$ do
14:     if $L(i) = L(i + 1)$ then
15:         $R(e) \leftarrow R(e) \cup L(i)^e$
16: end if
17: end for
18: end for
19: $T \leftarrow \bigcup_{e \in E} R(e)$
20: $T' \leftarrow $COUNTING-SORT$(T, |V|)$
21: for $j \leftarrow 1$ to $|T'|$ do
22:     $F(T(j)) \leftarrow F(T(j)) \cup \{+e(T(j))\}$
23: end for
24: for $i \leftarrow 1$ to $|V|$ do
25:     $E_t(i) \leftarrow |F(i)|$
26: end for
27: $\text{BUILD-MAX-HEAP}(Q, E_t)$

Algorithm 2 presents the pseudocode of $\Psi$-MCH. This algorithm aims to cover all the edges of $E$ by systematically adding vertices to the hub cover set $S$. Each iteration of the while cycle of Lines 2-21 proceeds as follows.

First, since this algorithm follows a greedy approach, Lines 3-5, select the vertex $v$ that covers the greatest number of uncovered edges and adds it to $S$. After that, the cycle of Lines 10-19 marks as ‘covered’ each uncovered edge $e$ in the set $F(v)$. Then, for each new covered edge $e$, the cycle also marks $e$ in the set $F(u)$, where $u$ is a vertex that, like $v$, also covers $e$. Finally, Line 20 incorporates the changes produced during the previous cycle in $Q$. $\Psi$-MCH ends when the set $E'$ is empty; i.e., when the vertices in $S$ cover all the edges in $E$.

Algorithm 2 Pseudocode of $\Psi$-MCH

**Require:** An undirected graph $G = (V, E)$.

**Ensure:** A hub cover set $S$.

1: INIT($G$)
2: while $E' \neq \emptyset$ do
3:     $v \leftarrow $EXTRACT-MAX$(Q)$
4:     $S \leftarrow S \cup v$
5:     $\tau \leftarrow |S|$
6:     for each $u \in V$ do
7:         $U(u) \leftarrow \text{false}$
8:         $E_t(u) \leftarrow E_t - (u)$
9:     end for
10:    for each $e \in F(v)$ do
11:        if $\text{cover}(e) = \text{false}$ then
12:            $\text{cover}(e) \leftarrow \text{true}$
13:            $E' \leftarrow E' - e$
14:            for each $u \in R(e)$ do
15:                $E_t(u) \leftarrow E_t(u) - 1$
16:                $U(u) \leftarrow \text{true}$
17:            end for
18:        end if
19:    end for
20:    $\text{UPDATE-KEY}(Q, E_t, U)$
21: end while
22: Return $S$

Algorithm 3 presents the pseudocode of UPDATE-QUEUE. This algorithm aims to update the values stored in the max-priority queue $Q$. When more than $\lceil |V| / \log |V| \rceil$ elements of $Q$ require updates, this algorithm uses **BUILD-MAX-HEAP** to reconstruct $Q$; otherwise, it uses **DECREASE-KEY** to update the required elements of $Q$ one at a time. **UPDATE-QUEUE** runs in $O(|V|)$ time.

**IV. CORRECTNESS AND ANALYSIS OF $\Psi$-MHC**

This section presents the correctness and complexity proofs of $\Psi$-MCH. First, Lemma 1 proves that $\Psi$-MHC correctly computes a hub cover set $S$. Then, Lemma 3 shows that the approximation ratio of this algorithm is $\ln \mu$, where $\mu$ represents the cardinality of the largest set $F(v)$ for all $v \in V$. Finally, Lemma 6 establishes that the execution time of $\Psi$-MHC is $O((\Delta + 1)|E| + |S||V|)$.
Algorithm 3 Pseudocode of UPDATE-KEY($Q, E_t, U$)

Require: The queue $Q$ and the arrays $E_t$ and $U$.
Ensure: The queue $Q$ with its updated values.

1: $count \leftarrow 0$
2: for $i \leftarrow 1$ to $|V|$ do
3:     if $U(i) = true$ then
4:         $count \leftarrow count + 1$
5:     end if
6: end for
7: if $count \geq \lceil \frac{n}{log n} \rceil$ then
8:     BUILD-MAX-HEAP($Q, E_t$)
9: else
10:     for $i \leftarrow 1$ to $|V|$ do
11:         if $U(i)$ then
12:             DECREASE-KEY($Q, i, E_t(i)$)
13:         end if
14:     end for
15: end if
16: Return $Q$

Lemma 1: Given an arbitrary undirected graph $G = (V, E)$, Ψ-MHC computes a hub cover set $S \subseteq V$.

Proof: By contradiction. Assume that after executing Ψ-MHC on $G$, the generated set $S$ is not a hub cover set. This assumption implies that at least one edge $(x, y) \in E'$ is not covered. Notice that Line 13 of the algorithm only removes from $E'$ an edge if some vertex of $S$ has covered it; so Ψ-MHC can not end if $E' \neq \emptyset$; therefore, there exists a contradiction, and $S$ is a hub cover set.

The cost of the solution given by Ψ-MHC is $|S|$; i.e., one unit per vertex added to $S$. When Ψ-MHC adds a vertex $v_i$ to $S$ at iteration $i$, $v_i$ covers a set of edges $A_i \subseteq E$ for the first time; i.e., $A_i$ is the subset of uncovered edges of $F(v_i)$ just before the iteration $i$. Eq. 1 gives an expression for $A_i$.

$$A_i = F(v_i) - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v_i)) \quad (1)$$

Notice that $F(v_i) - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v_i))$ is the subset of edges of $F(v_i)$ that other vertices already covered in iterations 1 to $i - 1$. Eq. 1 assumes that Ψ-MHC added vertices $v_1, \ldots, v_{i-1}$ to $S$ before iteration $i$.

Let value of $c_e$ be the cost to cover edge $e \in A_i$. Eq. 2 computes the value of $c_e$

$$c_e = \frac{1}{|A_i|} \quad (2)$$

Lemma 2: The cost to cover all the edges of $F(v)$, for some $v \in V$, through the execution of Ψ-MHC is given by:

$$\sum_{e \in F(v)} c_e \leq H_{\lceil |F(v)| \rceil},$$

where $H_{\lceil |F(v)| \rceil}$ is the $\lceil |F(v)| \rceil$-th harmonic number.

Proof: Let $E_i(v) = |F(v) - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v))|$ be the number of uncovered edges in $F(v)$ at the end of iteration $i$. Let $k$ be the iteration at which Ψ-MHC covered the last edge of $F(v)$. Notice that $|F(v)| = E_0(v) \geq E_1(v) \geq \cdots \geq E_k(v) = 0$, and $(E_{i-1}(v) - E_i(v))$ is the number of edges in $F(v)$ that Ψ-MHC covered at iteration $i$. Then, Eq. 3 establishes the cost to cover all the edges in $F(v)$.

$$\sum_{e \in F(v)} c_e = \sum_{i=1}^{k} \frac{E_{i-1}(v) - E_i(v)}{[F(v)] - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v_i))] \quad (3)$$

Eq. 3 assumes that $|F(v)| - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v_i))] \leq |F(v)| - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v_i))$; otherwise, Ψ-MHC would select vertex $v_i$ instead of $v_i$. Then

$$\sum_{e \in F(v)} c_e \leq \sum_{i=1}^{k} \frac{E_{i-1}(v) - E_i(v)}{E_{i-1}(v)} = \sum_{i=1}^{k} \frac{1}{E_{i-1}(v)} = \sum_{i=1}^{k} \frac{1}{E_{i-1}(v)} = \sum_{j=1}^{i} \frac{1}{E_{i-1}(v)} = \sum_{j=1}^{i} \frac{1}{E_{j}}$$

therefore,

$$\sum_{e \in F(v)} c_e \leq H_{|F(v)|} \quad (5)$$

Lemma 3: Let $G = (V, E)$ be an undirected graph. Let $S^*$ be a minimum hub cover set in $G$, and let $S$ be the hub cover set computed by Ψ-MHC. Then, $|S| \leq H_\mu |S^*|$, where $\mu$ is the cardinality of the largest set $F(v)$ for all $v \in V$.

Proof: Let $v_i$ be the vertex added to $S$ at iteration $i$. Notice that Eq. 6 computes the value of $E_{i-1}(v_i)$ at the end of iteration $i - 1$.

$$E_{i-1}(v_i) = |F(v_i) - ((F(v_1) \cup \cdots \cup F(v_{i-1})) \cap F(v_i))| \quad (6)$$

Eq. 7 computes the cost to cover all the edges of $F(v_i)$ at iteration $i$.

$$\sum_{e \in F(v_i)} c_e = 1 \quad (7)$$

consequently,

$$|S| = \sum_{e \in E} c_e \quad (8)$$

Since the vertices of $S^*$ cover each edge $e \in E$ at least once, then

$$\sum_{v \in S^*} \sum_{e \in F(v)} c_e \geq \sum_{e \in E} c_e \quad (9)$$
Obtain Eq. 10 by substituting Eq. 8 in Eq 9,
\[ |S| \leq \sum_{v \in S^*} \sum_{e \in F(v)} c_e \] (10)

obtain Eq. 11 by substituting Eq. 5 in Eq 10,
\[ |S| \leq \sum_{s \in S^*} H_{F(v)} \] (11)

finally,
\[ |S| \leq |S^*| H_\mu \] (12)

where, \( \mu \) is the cardinality of the largest set \( F(v) \) for all \( v \in S^* \).

Lemma 4 proves that the value of \( \mu \) is bounded by \( \min\{\frac{1}{2}(\Delta + 1)^2, |E|\} \).

Lemma 4: Let \( \mu \) be the cardinality of the largest set \( F(v) \) for all \( v \in V \). Then, \( \mu \leq \min\{\frac{1}{2}(\Delta + 1)^2, |E|\} \).

Proof: Let \( \delta(v) \) be the degree of an arbitrary vertex \( v \in V \). Notice that the number of crossing edges \( C(v) \leq \binom{\delta(v)}{2} = \frac{\delta(v)^2 - \delta(v)}{2} \). Therefore, \( F(v) = C(v) + \delta(v) \leq \frac{\delta(v)^2 + \delta(v)}{2} \leq \frac{1}{2}(\Delta + 1)^2 \) since \( \delta(v) \leq \Delta \). Also, notice that \( F(v) \leq |E| \). Therefore, \( \mu \leq \min\{\frac{1}{2}(\Delta + 1)^2, |E|\} \).

Lemma 5: \( |\bigcup_{v \in V} R(v)| = |\bigcup_{v \in E} R(e)\big| \leq (\Delta + 1)|E| \)

Proof: First, compute \( |R(e)| \) for an arbitrary edge \( e = (x, y) \). Notice that both \( \delta(x) \) and \( \delta(y) \) are less than or equal to \( \Delta \). Both vertices \( x \) and \( y \) can cover \( e \), and at most other \( \Delta - 1 \) vertices can also cover \( e \) since they can be present in both \( \text{Adj}(x) \) and \( \text{Adj}(y) \). Therefore, \( |R(e)| \leq \Delta + 1 \) and \( |\bigcup_{v \in E} R(e)\big| \leq (\Delta + 1)|E| \).

Notice that INIT, in Lines 19-23, computes all the sets \( F(v) \) for all \( v \in V \) by sorting \( \bigcup_{e \in E} R(e) \) and then extracting one satellite-data edge from each vertex of this set. Therefore, \( |\bigcup_{v \in V} F(v)\big| = |\bigcup_{e \in E} R(e)\big| \leq (\Delta + 1)|E| \).

Lemma 6: The execution time of \( \Psi \)-MHC is
\( O((\Delta + 1)|E| + |S||V|) \).

Proof: The most time consuming part of INIT is computing the sets \( R(e) \) for all \( e \in E \) (the cycle of Lines 9-18). By Lemma 5, \( |\bigcup_{e \in E} R(e)\big| \leq (\Delta + 1)|E| \) and INIT requires constant time to compute each element in this set; thus, INIT requires \( O((\Delta + 1)|E|) \) time.

For the remaining portion of \( \Psi \)-MHC, the procedure to cover all edges of \( E \) also takes \( (\Delta + 1)|E| \) time. Notice that Line 15 decreases in one unit the value of \( E_i(u) \), and the algorithm only can end when all variables \( E_i(u) \) for all \( u \in V \) are equal to zero. UPDATE-KEY (Line 20) requires \( O(|V|) \) time per iteration, and the algorithm performs \( |S| \) iterations.

Overall, \( \Psi \)-MHC requires \( O((\Delta + 1)|E| + |S||V|) \) time.

V. NP-HARDNESS OF THE MINIMUM HUB COVER SET PROBLEM

Yelbay [13] proved that the minimum hub cover set problem is NP-hard for triangle-free graphs. This section proves that the decision version of this problem is NP-complete, even if the input graph has triangles. The proof’s idea consists of reducing the 3-SAT problem to the decision version of the hub cover set problem.

Let \( \phi \) be a Boolean formula in conjunctive normal form with \( n \) Boolean variables and \( m \) clauses (containing exactly three different Boolean variables). The 3-satisfiability problem (3-SAT) determines if an assignment exists that makes \( \phi \) true. The 3-SAT problem is NP-complete [14].

Lemma 7: The decision version of the hub cover set problem is in the NP class.

Proof: Let \( G = (V, E) \) be an undirected graph and let \( k \) be a positive integer. Let the pair \( G, k \) be an instance of the decision hub cover set problem. Assume that \( S' \in V \) is a certificate for such an instance. It is possible to use a modification of \( \Psi \)-MHC to verify this certificate. In this algorithm, instead of extracting vertices from \( Q \), extract them from \( S' \) and verify that every edge is covered. Such an algorithm runs in polynomial time, as shown in Section IV. Therefore, the decision hub cover set problem is in the NP class.

Lemma 8: The 3-SAT problem can be polynomially reduced to the hub cover set of size \( k \).

Proof: Let \( \phi \) be an arbitrary instance of the 3-SAT problem with \( n \) Boolean variables \( U = \{u_1, \ldots, u_n\} \) and \( m \) clauses \( C = \{c_1, \ldots, c_m\} \). In each clause \( c_j = (b_{j,1} \lor b_{j,2} \lor b_{j,3}) \), \( b_{j,i} \in U \) for \( 1 \leq i \leq 3 \) and \( 1 \leq j \leq m \). Each variable \( b_{j,i} \) can be non-negated or negated (denoted by \( \bar{b}_{j,i} \)) in \( c_j \). We present an algorithm to reduce \( \phi \) to an instance of a hub cover set of size \( k = n + 2m \).

Construct the undirected graph \( G_\phi = (V, E) \), as follows.

The set \( V = \{V_U \cup V_D\} \), where
\[ V_U = \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\} \]
\[ V_D = \{V_{D_1} \cup V_{D_2} \cup \ldots \cup V_{D_m}\}, \quad \text{and} \quad V_{D_j} = \{a_{j,1}, a_{j,2}, a_{j,3}\} \text{ for } 1 \leq j \leq m. \]

The set \( E = \{E_U, E_D, E_{UD}\} \), where
\[ E_U = \bigcup_{j=1}^{n} \{(u_i, \bar{u}_{i})\} \]
\[ E_D = \bigcup_{j=1}^{m} \{(a_{j,1}, a_{j,2}), (a_{j,2}, a_{j,3}), (a_{j,1}, a_{j,2})\} \]
\[ E_{UD} = \bigcup_{j=1}^{m} \{(b_{j,1}, a_{j,1}), (b_{j,2}, a_{j,2}), (b_{j,3}, a_{j,3})\} \]

For a clause \( c_j \in C \), let \( C_{c_j} \) be the subgraph of \( G_\phi \) induced by the vertices \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \cup \{a_{j,1}, a_{j,2}, a_{j,3}\} \). Figure 2 illustrates such a subgraph.

Notice that the time required to transform the 3-SAT instance \( \phi \) into an instance \( G_\phi \) of a hub cover set of size \( k \) is polynomial in \( n, m \) since \( |V_U| = 2n, |V_D| = 4m, |E_U| = n, |E_D| = 3m, \) and \( |E_{UD}| = 4m \).

Now, we prove that a hub cover set of size exactly \( k = n + 2m \) exists in \( G_\phi \) if and only if \( \phi \) is satisfiable.

First, assume that there exists a satisfiable assignment for \( \phi \). Let \( S_H \in V_U \) be such an assignment, where \( |S_H| = n \).

We need to show that for each graph \( C_{c_j} \subseteq G_\phi \), there exists

![Figure 2](image-url)

FIGURE 2. Graph \( C_{c_j} \subseteq G_\phi \) is the one induced by the vertices \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \cup \{a_{j,1}, a_{j,2}, a_{j,3}\} \).
a hub cover set containing exactly two vertices from the set \( \{a_{j,1}, a_{j,2}, a_{j,2}', a_{j,3}\} \) and at least one vertex from the set \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \) (otherwise, \( \phi \) is not satisfiable). There are three cases to consider:

- **Case 1.** Only one vertex from the set \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \) belong to \( S_U^{L} \). There are two subcases:
  - **Subcase 1.** Only one vertex from the set \( \{b_{j,1}, b_{j,3}\} \in S_U^{L} \). Without loss of generality, assume that \( b_{j,1} \in S_U^{L} \). Then, both vertices \( \{a_{j,2}, a_{j,3}\} \) must also be in \( S_U^{L} \) to cover all the edges in \( C_j \).
  - **Subcase 2.** The vertex \( b_{j,2} \in S_U^{L} \). Then, both vertices \( \{a_{j,1}, a_{j,3}\} \) must also be in \( S_U^{L} \) to cover all the edges in \( C_j \).

- **Case 2.** Exactly two vertices from the set \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \) belong to \( S_U^{L} \). There are two subcases:
  - **Subcase 1.** Both vertices \( \{b_{j,1}, b_{j,3}\} \in S_U^{L} \). Then, only one of the following three sets \( \{a_{j,1}, a_{j,2}', a_{j,2}, a_{j,3}\} \), \( \{a_{j,2}, a_{j,3}\} \) must also be in \( S_U^{L} \) to cover all the edges in \( C_j \).
  - **Subcase 2.** Vertex \( b_{j,2} \in S_U^{L} \) and only one vertex from the set \( \{b_{j,1}, b_{j,3}\} \) is in \( S_U^{L} \). Without loss of generality, assume that \( b_{j,1} \in S_U^{L} \). Then, vertex \( a_{j,3} \) and one from the set \( \{a_{j,1}, a_{j,2}\} \) must also be in \( S_U^{L} \) to cover all the edges in \( C_j \).

- **Case 3.** The three vertices from the set \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \) belong to \( S_U^{L} \). Then, only one of the following three sets \( \{a_{j,1}, a_{j,2}', a_{j,2}, a_{j,3}\} \), \( \{a_{j,2}, a_{j,3}\} \) must also be in \( S_U^{L} \) to cover all the edges in \( C_j \).

In Cases 1 to 3, each clause \( c_j \in C \) contains exactly two vertices from the set \( \{a_{j,1}, a_{j,2}, a_{j,2}', a_{j,3}\} \). Therefore, there exists a hub cover set of size exactly \( k = n + 2m \) in \( G_\phi \).

Now, assume that there exists a hub cover set \( S \) in \( G_\phi \) of size exactly \( k = n + 2m \). We need to show that it is possible to obtain a satisfiable assignment from \( S \). Assume by contradiction that it is not possible to obtain a satisfiable assignment from \( S \).

Notice that the set \( S \) contains at least \( n \) vertices from \( V_U \); otherwise, there exists one edge from \( E_U \) that is not covered. By the first part of this proof, each subgraph \( C_{c_k} \) contains exactly two vertices from the set \( \{a_{j,1}, a_{j,2}, a_{j,2}', a_{j,3}\} \) in \( S \).

Since \( \phi \) is unsatisfiable, there exists at least one clause \( c_k \in C \) that is unsatisfiable. Thus, no vertex from the set \( \{b_{j,1}, b_{j,2}, b_{j,3}\} \) belongs to \( S \). Then, the hub cover set \( S \) requires to include at least three vertices from the set \( \{a_{j,1}, a_{j,2}, a_{j,2}', a_{j,3}\} \) to cover all the edges in \( C_j \), generating a hub cover set of size \( n + 2m + 1 \), which contradicts the initial assumption. Therefore, the hub cover set \( S \) represents a satisfiable assignment to \( \phi \), and the 3-SAT can be polynomially reduced to the hub cover set of size \( k = n + 2m \).

**Theorem 1:** Let \( G = (V, E) \) be an undirected graph and let \( k \) be a positive integer. The problem of computing a hub cover set of size at most \( k \) is in the NP-complete class.

**Proof:** The proof is a consequence of Lemmas 7 and 8.

## VI. EXPERIMENTAL DESIGN

This section presents a set of numerical experiments that evaluates the quality of the minimum hub cover set generated by \( \Psi \)-MHC. Section VI-A presents a linear programming formulation of the MHC problem. Section VI-B describes the graph instances used by the experiments. Section VI-C presents the metrics to compare the quality of \( \Psi \)-MHC against Gurobi.

### A. MHC LINEAR PROGRAMMING MODEL

Given an undirected graph \( G = (V, E) \), Eq. 13-15 [12] represents the integer linear programming formulation to compute the minimum hub cover set, ILP-MHC.

\[
\text{minimize} \sum_{v \in V} x_v, \quad (13)
\]

subject to

\[
x_u + x_v + \sum_{w \in K(u, v)} x_w \geq 1, \quad (u, v) \in E, \quad (14)
\]

\[
x_v \in \{0, 1\}, \quad v \in V \quad (15)
\]

The function in Eq. 13 minimizes the number of selected vertices in the hub cover set. In this equation, the binary variable \( x_v \) is equal to one only if vertex \( v \) belongs to the hub cover set. The constraint in Eq. 14 ensures that every edge \( (u, v) \in E \) is covered by at least one vertex in the hub cover set. For \( (u, v) \in E \), the set \( K(u, v) \) denotes all the vertices \( w \in V \) such that \( (u, w) \in E \) and \( (v, w) \in E \); i.e., \( K(u, v) = R(u, v) \setminus \{u, v\} \). Finally, Eq. 15 restricts each variable \( x_v \) to be binary.

We used Gurobi, a typical solver in the industry and academia, to optimally solve ILP-MHC. The design of Gurobi exploits multicore architectures to parallelize computations. It solves several types of problems like linear programming (LP), mixed-integer programming (MIP), and quadratic programming (QP). It uses advanced algorithms and heuristics like simplex, parallel barrier with crossover, concurrent and sifting, and cutting planes. However, similar to other problems in the NP-hard class, solving optimally large-scale MHC instances prove very difficult, even for a solver.

### B. GRAPH INSTANCES

We generated and divided our set \( I \) of 40 graphs instances using the Erdős–Rényi model, into four subsets of ten graphs, denoted by \( I_{2k}, I_{25k}, I_{50k}, \) and \( I_{100k} \).

The instances used are diverse enough to allow us to show the performance and quality of our algorithm. In particular, such instances include dense and sparse graphs whose order ranges from 10 to 1,000 nodes and size from 18 to 100,000 edges.

- **Set \( I_{2k} \).** The graph instances in this set contain few vertices. The order varies from 10 to 100 in increments of 10, and the size from 18 to 2000. With these graphs, we aim to verify that the observed performance of \( \Psi \)-MHC, outperforms the approximation ratio. Gurobi could obtain the minimum hub cover set only for this set of graphs.
TABLE 1. Gap percentage between the result produced by $\psi$-MHC and the best result from Gurobi within time limit. The symbol $*$ indicates that the cardinality of the hub cover set obtained by $\psi$-MHC is smaller than the one computed by Gurobi.

| Set | $I_{2k}$ | $I_{25k}$ | $I_{50k}$ | $I_{100k}$ |
|-----|---------|---------|---------|---------|
| $G_1$ | 0% | 8.7% | 5.2% | 13.3% |
| $G_2$ | 0% | 6.7% | 12.2% | 10.6% |
| $G_3$ | 9.1% | 8% | 11.9% | 10.7% |
| $G_4$ | 7.7% | 10.1% | 10.8% | 2.2% |
| $G_5$ | 7.1% | 9.1% | 7% | $*$ |
| $G_6$ | 12.5% | 7.7% | 8.5% | 0% |
| $G_7$ | 11.8% | 6.9% | 10.3% | $*$ |
| $G_8$ | 10.5% | 9.4% | 7.2% | 0.8% |
| $G_9$ | 5.5% | 8.3% | 3.2% | $*$ |
| $G_{10}$ | 9.5% | 8.1% | 5.5% | $*$ |

FIGURE 3. Empirical and observed performance according to the approximation ratio of $\psi$-MHC and the execution times required to solve $I_{2k}$, both as a function of the number of vertices.

C. METRICS

We use the following metrics to evaluate the performance of $\psi$-MHC.

- The observed performance compares the theoretical upper bound of $\Psi$-MHC (proved in Lemma 3) against its empirical performance. It is worth mentioning that it is not “feasible” to obtain the cardinality of the minimum hub cover set in most of the graphs in $\mathcal{I}$ within the time limit. Thus, we only compare the observed performance on the instances of $I_{2k}$.

24999 for $I_{50k}$, and between 488 and 99810 for $I_{100k}$. With these graphs, we aim to evaluate the performance of $\psi$-MHC.

Let $|S|_{G_i}$ and $|G_i|$ be the cardinalities of the hub cover sets obtained in graph $G_i \in \mathcal{I}$ by $\Psi$-MHC and Gurobi with no more than 10,000 seconds, respectively. Let $|S^*|_{G_i}$ be the cardinality of the minimum hub cover set in $G_i$. Notice that Gurobi computes the minimum hub cover set only if $G_i \in I_{2k}$. In such case, $|G_i|=|S^*|_{G_i}$; for the remaining instances $|G_i|\geq|S^*|_{G_i}$.
J. A. Trejo-Sánchez et al.: Approximation Algorithm for Minimum Hub Cover Set Problem

VOLUME 10, 2022

The quality of the solution compares the cardinality of the hub cover set obtained by Ψ-MHC versus the one obtained by Gurobi.

The execution time provides information about the time required by Ψ-MHC to compute a hub cover set against the one required by Gurobi.

D. COMPUTING ENVIRONMENT

We performed these experiments on an Ubuntu server Intel(R) Xeon(R) Gold 5222 CPU with 20 cores and 40GB RAM. We implemented ILP-MHC in Gurobi and executed it with only one core.

VII. ANALYSIS OF EXPERIMENTAL RESULTS

This section analyses and discusses the hub cover sets generated by Ψ-MHC on the instances described in section VI-B in three different aspects.

A. OBSERVED PERFORMANCE

Fig. 3a depicts the cardinalities of the hub cover sets generated by Gurobi, Ψ-MHC, and the theoretical approximation ratio \(|G|_G, |S|_G, \text{and } H_\mu|S^\gamma|_G\), respectively for each graph \(G_i \in \mathcal{I}\). Recall that \(\mu\) is bounded by \(\min\{\frac{1}{2}(\Delta + 1)^2, |E|\}\), where \(\Delta\) is the degree of \(G_i\). Observe that \(|S|_G\) is very close to \(|S^\gamma|_G\) for each \(G_i\), and \(|S|_G\) is closer to \(|S^\gamma|_G\) than to \(H_\mu|S^\gamma|_G\). Therefore, the empirical performance of Ψ-MHC far outperforms the approximation ratio for instances in \(\mathcal{I}_{25k}\).

Fig. 3b presents the execution times required by Ψ-MHC and Gurobi to compute their corresponding hub cover sets for each \(G_i\). Notice that for the minor instances, both Gurobi and Ψ-MHC are similar; however, when the number of edges surpasses 1000, Gurobi requires several minutes to obtain the minimum hub cover set, whereas the execution time required by Ψ-MHC remains practically constant.

B. QUALITY OF THE SOLUTION

Gurobi could not obtain the minimum hub cover set for large instances. For such instances, we compared the solution generated by Gurobi with no more than 10,000 seconds versus the one generated by Ψ-MHC. Fig. 4a, 5a, and 6a plot the cardinality of the solution generated by Ψ-MHC and Gurobi for \(\mathcal{I}_{25k}, \mathcal{I}_{50k}, \text{and } \mathcal{I}_{100k}\), respectively.
Table 1 provides detailed the gap percentage between the cardinality of the hub cover obtained by $\Psi$-MHC and the one computed by Gurobi within the time limit.

In general, the results show that the solution obtained by $\Psi$-MHC is quite close to the one of Gurobi, particularly for small and large instances; e.g., in some instances in $T_2$ and in $T_{1000}$, both Gurobi and $\Psi$-MHC obtained the same cardinality. Even more, in some large instances $\Psi$-MHC obtained a smaller hub cover than the one obtained by Gurobi within the time limit.

C. EXECUTION TIME OF THE ALGORITHMS

Fig. 4b, 5h, and 6b plot the execution times required by both $\Psi$-MHC and Gurobi to compute hub cover sets. Observe that the cardinality obtained by Gurobi represents the hub cover set obtained after 10,000 seconds. In contrast, $\Psi$-MHC computes good hub cover sets in a fraction of the time required by Gurobi. The results in these plots show that the execution times of $\Psi$-MHC are much shorter than those of Gurobi, with reasonably good quality solutions, or even better, as depicted by Fig. 6a.

VIII. CONCLUDING REMARKS

This paper proposes $\Psi$-MHC, the first approximation algorithm for the minimum hub cover set problem in arbitrary graphs. We show that the approximation ratio of $\Psi$-MHC is $\ln \mu$, where $\mu$ is bounded by $\min\{\frac{1}{2}(\Delta + 1)^2, |E|\}$. The time complexity of $\Psi$-MHC is $O((\Delta + 1)|E| + |S||V|)$.

We provided experimental results showing the observed performance of $\Psi$-MHC. Experimentation empirically shows that $\Psi$-MHC far outperforms the expected approximation ratio in a fraction of the time required by Gurobi to obtain approximate solutions. In addition, in some instances, $\Psi$-MHC is close to the best solution obtained by the solver.

We seek to design better approximation bounds for computing the minimum hub cover set as future research work.

REFERENCES

[1] H. Jamil, “Computing subgraph isomorphic queries using structural unification and minimum graph structures,” in Proc. ACM Symp. Appl. Comput. (SAC), 2011, pp. 1053–1058.
[2] M. R. Garey and D. S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness,” vol. 26, ser. Computer Science and Computational Theory. San Francisco, CA, USA: Freeman, 1979.
[3] C. R. Rivero and H. M. Jamil, “On isomorphic matching of large disk-resident graphs using an XQuery engine,” in Proc. IEEE 30th Int. Conf. Data Eng. Workshops, Mar. 2014, pp. 20–27.
[4] D. S. Johnson, M. R. Garey, and S. A. Cook, “The complexity of theorem-proving procedures,” in Proc. 3rd Annu. ACM Symp. Theory Comput. (STOC), 1971, pp. 151–158.
[5] M. Cygan and M. Pilipczuk, “Split vertex deletion meets vertex cover: New fixed-parameter and exact exponential-time algorithms,” Inf. Process. Lett., vol. 113, nos. 5–6, pp. 179–182, Mar. 2013.
[6] S. Bhattacharya, M. Henzinger, and D. Nanongkai, “Fully dynamic approximate maximum matching and minimum vertex cover in $O(\log^3 n)$ worst case update time,” in Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms, Jan. 2017, pp. 470–489.
[7] “A better approximation ratio for the vertex cover problem,” in Proc. Int. Colloq. Automata, Lang., Program. Cham, Switzerland: Springer, 2005, pp. 1043–1050.