Approximating the Minimum Hub Cover Problem on Planar Graphs

Belma Yelbay[a,*], Ş. İlker Birbil[a], Kerem Bülbül[a], Hasan Jamil[b]

[a]*Sabancı University, Manufacturing Systems and Industrial Engineering, Orhanlı-Tuzla, 34956 Istanbul, Turkey.*
[b]*University of Idaho, Department of Computer Science, Moscow, Idaho, USA.*

Abstract

We study an approximation algorithm with a performance guarantee to solve a new $\mathcal{NP}$-hard optimization problem on planar graphs. The problem, which is referred to as the minimum hub cover problem, has recently been introduced to the literature to improve query processing over large graph databases. Planar graphs also arise in various graph query processing applications, such as; biometric identification, image classification, object recognition, and so on. Our algorithm is based on a well-known graph decomposition technique that partitions the graph into a set of outerplanar graphs and provides an approximate solution with a proven performance ratio. We conduct a comprehensive computational experiment to investigate the empirical performance of the algorithm. Computational results demonstrate that the empirical performance of the algorithm surpasses its guaranteed performance. We also apply the same decomposition approach to develop a decomposition-based heuristic, which is much more efficient than the approximation algorithm in terms of computation time. Computational results also indicate that the efficacy of the decomposition-based heuristic in terms of solution quality is comparable to that of the approximation algorithm.

Keywords: approximation algorithm, query processing, subgraph isomorphism, planar graph decomposition, minimum hub cover problem

1. Introduction

The optimization problem that we are interested in is called the *minimum hub cover (MHC)* problem. This problem has recently originated from a new representation used for query processing over large graph databases (Jamil, 2011; Yelbay et al., 2013). Many graph databases in query processing satisfy planarity condition that is common in diverse applications, such as; face recognition, fingerprint identification, hand posture recognition, image classification, object recognition, and so on. As an example, Figure 1 shows the planar graph representation of a finger print. Each node in the graph represents a finger ridge pattern and the edges are constructed according to the orientation of the ridges.

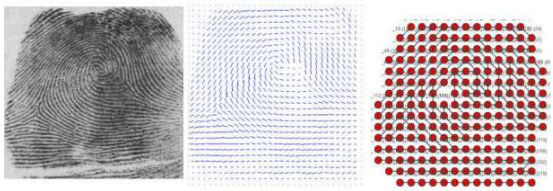
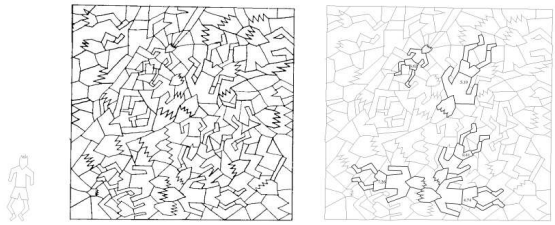


Figure 1: Graph representation of a fingerprint image (Neuhaus and Bunke, 2005).

A graph database stores a high volume of relational data coming from various sources including communication, social and biological networks. For instance, a chemical compound is a graph where each node represents an atom and each edge represents a chemical bond formed between two atoms. Graph query processing, also known as graph matching, is about querying the structural similarity between the nodes of a query graph and a database graph under a set of label constraints. Subgraph isomorphism problem, on the other hand, is to find whether a database graph includes a subgraph that is structurally similar to a given query graph. Figure 2 illustrates the relationship between graph query processing and subgraph isomorphism. Figures 2(a) and 2(b) are query and data graphs, respectively that capture two hand-drawn images of human figures. Figure 2(c) shows all subgraphs of Figure 2(b) that are isomorphic to Figure 2(a) and thus are embedded in the data graph. The goal of subgraph matching is to identify the isomorphic graphs in Figure 2(b) given the query graph in Figure 2(a) against the data graph in Figure 2(b). The MHC problem has been recently introduced as an alternate solution method to increase the efficiency of subgraph matching. Latest studies demonstrate that searching a graph query over the hubs obtained by solving the MHC problem improves the current techniques in graph query processing (Rivero and Jamil, 2014b,a).

The objective of the MHC problem is to cover all edges of a graph with the minimum number of vertices. Unlike the conventional meaning of covering, here, a selected vertex covers not only its incident edges but also the edges between its adjacent neighbors. For instance, in Figure 3, the edges that can be covered by vertex f are $(f,g),(f,h),(f,k)$ and (h,k) and the

*Corresponding Author
Email addresses: byelbay@sabanciuniv.edu (Belma Yelbay), sibirbil@sabanciuniv.edu (Ş. İlker Birbil), bulbul@sabanciuniv.edu (Kerem Bülbül), jamil@uidaho.edu (Hasan Jamil)



(a) (b) (c)

Figure 2: Example for subgraph isomorphism between two hand-drawn images and the resulting solution (Llados et al., 2001).

optimal solution is $\{a, c, f\}$. The formal definition of the MHC problem follows.

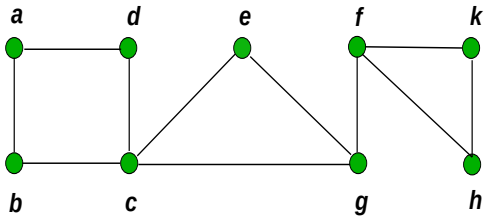


Figure 3: A sample graph for the minimum hub cover problem.

Definition 1. Let $G(V, E)$ be an undirected graph, where V is the set of vertices and E is the set of edges. Then, for a given graph G, a subset of the vertices $HC \subseteq V$ is a *hub cover* of G if for every edge $(i, j) \in E$, either $i \in HC$ or $j \in HC$ or there exists a vertex k such that $(i, k) \in E$ and $(j, k) \in E$ with $k \in HC$. The *MHC* problem is finding a hub cover that has the minimum number of vertices.

Our previous work (Yelbay et al., 2013) discusses the connection between the MHC concept and query processing and analyzes the computational cost of solving the MHC problem on a diverse set of graph databases. It demonstrates that the graph structure has an impact on the solvability of the problem and that the performance of exact methods for MHC on large-scale query graphs would be less than satisfactory in query processing. In this paper, we work with planar graphs for they appear in various graph query processing applications (Neuhaus and Bunke, 2005; Baloch and Krim, 2010). Our objective is to make use of an algorithm that gives a good approximation bound. Then, we aim at deriving fast heuristics from the approximation algorithm. As for the algorithm, we adapt the well-known decomposition technique introduced by (Baker, 1994). This is followed by our discussion on the implementation details of the algorithm. We also exploit its naturally parallel structure to improve the computational performance. We conduct a comprehensive numerical study to analyze the empirical performance of our algorithm. This study leads us to propose a decomposition-based heuristic that is much more efficient in terms of computation time. Our results indicate that the performance of this fast heuristic in terms of solution quality is comparable to that of the approximation algorithm. Thus, the proposed heuristic demonstrates a potential for practical applications of planar graph query processing.

In summary, we make the following research contributions: (i) a new approximation algorithm with a performance guarantee for solving the MHC problem on planar graphs; (ii) to the best of our knowledge, the first experimental analysis to test the empirical performance of Baker's planar graph decomposition algorithm; (iii) the first extensive numerical experiments on a set of planar MHC problems; (iv) a new fast decomposition-based heuristic inspired by our computational study.

2. Review of Related Literature

There is a dramatic increase in the use of graph databases, and hence, the need for efficient query processing thrives. However, the subgraph isomorphism problem is known to be $\mathcal{NP}$-complete (Cook, 1971). Therefore, researchers aim at developing approximation methods and fast heuristics. To solve the subgraph isomorphism problem, graph matching algorithms look for node structures that are identically connected or have identical labels. Then, those individual matches are consolidated to answer a query. Ullmann (1976) is one of the earliest studies related to subgraph isomorphism in the literature. The proposed algorithm incrementally constructs a search tree representing the appropriate mappings between a query and a database graph. It applies backtracking, if the current partial solution will not end up with a successful mapping. Since a search tree grows exponentially when its depth increases, the subsequent studies focus on developing new methods which prune the search space and increase the efficiency of query processing. Among these, we have studies applying index-based searching (Shang et al., 2008; Weber et al., 2012), graph decomposition methods (Lipets et al., 2009) and using new data structures and graph representations keeping topological data (Zhu et al., 2010). As an alternate approach, the algorithm proposed in (Yelbay et al., 2013; Rivero and Jamil, 2014b,a) prunes the search space to a few number of nodes in query graphs by solving a combinatorial optimization problem, which is coined by the authors as the MHC problem. They illustrate that solving the MHC problem in advance yields the least number of candidate matches, and hence, a significant reduction in computation time can be gained.

3. Approximating MHC on Planar Graphs

In this section, we first give the mathematical formulation of the MHC problem and then discuss a graph decomposition technique for planar graphs. This technique is used to develop both an approximation algorithm with a proven performance ratio and an efficient decomposition-based heuristic.

3.1. Problem Formulation

A careful reader would notice that the definition of the MHC problem resembles some well-known combinatorial optimization problems from the literature. This is indeed the case and

the similarity is more evident as we give its integer programming (IP) formulation:

$$\text{minimize} \quad \sum_{j \in V} x_j, \tag{1}$$

$$\text{subject to} \quad x_i + x_j + \sum_{k \in \mathcal{K}^{(i,j)}} x_k \geq 1, \qquad (i,j) \in E, \tag{2}$$

$$x_j \in \{0,1\}, \qquad j \in V, \tag{3}$$

where, x_j is a binary variable, which is equal to 1, if vertex j is selected. For $(i,j) \in E$, $\mathcal{K}^{(i,j)}$ denotes all those vertices $k \in V$ such that $(i,k) \in E$ and $(j,k) \in E$. The objective function (1) minimizes the number of selected vertices. Constraints (2) ensure that every edge is covered by at least one hub vertex in the cover. Finally, constraints (3) impose integrality on the variables.

The first related problem is the *set covering problem* (SCP), which is one of the oldest and the most studied combinatorial optimization problems. If an edge corresponds to an item, and a set is defined for each vertex whose elements are the edges covered by that vertex, then the connection between SCP and MHC can easily be established. The second related problem is the *minimum vertex cover* (MVC) problem, which is one of the well-known $\mathcal{NP}$-complete problems in the literature. Yelbay et al. (2013) use the equivalence of MHC and MVC in triangle-free graphs to prove the $\mathcal{NP}$-completeness of MHC for general graphs. Similarly, we can conclude that MHC on planar graphs is also $\mathcal{NP}$-complete based on the fact that MVC is $\mathcal{NP}$-complete when restricted to triangle-free planar graphs (Garey et al., 1976). The following corollary states this observation formally:

Corollary 3.1. *The MHC problem is $\mathcal{NP}$-complete on planar graphs.*

Proof. See Section E, page 7 in (Yelbay et al., 2013). □

A natural question at this point could be "Why not use the approximation algorithms proposed for these similar combinatorial problems?" Unlike the general algorithms, our algorithm is specialized for solving the MHC problem on planar graphs. Moreover, our approximation bound is not constant and can be improved by decreasing the number of subproblems to be solved at the expense of an increase in the computation time.

3.2. Approximation Algorithm with a Performance Guarantee

Let us first introduce our notation and the terminology used in the coming subsections and then introduce our algorithm. A graph G' is an induced subgraph of G if G' is isomorphic to a graph whose vertex set V' is a subset of the vertex set V of G and whose edge set E' consists of all those edges of G with both end vertices in V'. A *planar embedding* of a graph G is a special drawing of G in such a way that no edges cross each other. A graph is *planar* if and only if it has a planar embedding. A *face* of a planar graph is a region bounded by edges. A vertex of a planar graph is at *level* 1, if it is on the exterior face. A planar embedding is *k-level*, if it has no nodes of level greater than k. A graph is *outerplanar*, if it is a planar graph such that all of the vertices belong to the exterior face. A planar embedding is said to be *k-outerplanar*, if removing the vertices on the exterior face results in a $(k-1)$-outerplanar embedding.

In this study, we make use of a general decomposition technique first proposed by Baker (1994). The technique can be applied to any planar graph whose planar embedding and the set of vertices in each level are known. In case they are not known, one of the algorithms in the literature can be applied to obtain a planar embedding (Hopcroft and R.Tarjan, 1974; Bienstock and Monma, 1990). With the proposed technique, given a planar embedding and a nonnegative number k, the planar graph is decomposed into a set of overlapping $(k+1)$-outerplanar graphs such that the union of the optimal solutions of those graphs gives a feasible solution to the original planar graph. The algorithm picks the best of these solutions as its approximation to the optimal hub cover. Figures 5 and 6 illustrate how the decomposition is applied to the problem shown in Figure 4 when $k = 2$. The unions of the optimal solutions of the subgraphs in Figure 5 and Figure 6 provide two different hub covers. The algorithm selects the solution with the minimum cardinality as an approximate solution. Optimal solutions may be obtained by modifying a dynamic programming algorithm proposed in (Baker, 1994). Yelbay (2014) discusses those modifications in detail. The complexity of the algorithm is $O(n8^k)$. If k is taken as $\lceil c \log n \rceil$, where c is a constant, then we obtain a polynomial time approximation algorithm for MHC with a performance guarantee of $(k+1)/k$ – see Proposition 3.1. We refer to Baker (1994) for the details of the complexity analysis. Optimal solutions may also be obtained by solving the IP formulation (1)-(3) of each subproblem by using an off-the-shelf solver like `CPLEX`. In this paper, we prefer to use `CPLEX` for ease of implementation.

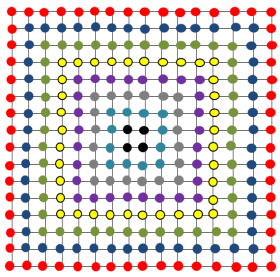
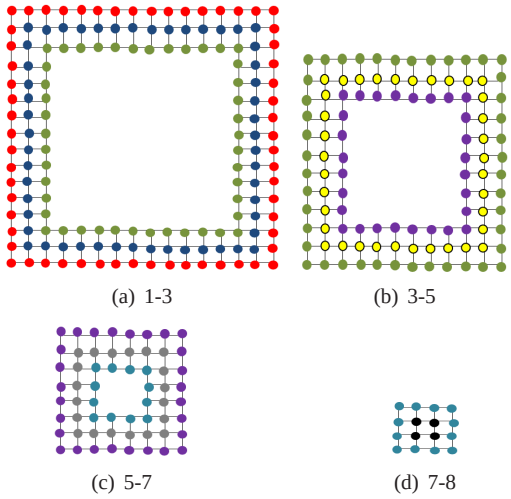


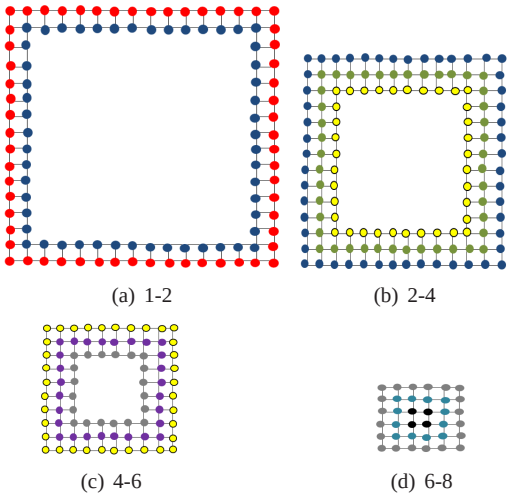
Figure 4: An 8-level planar graph embedding. Each level is represented by a different color and the vertices with the same colors lie in the same level.

The steps of the proposed decomposition and the solution approach are detailed in Algorithm 1. The algorithm takes a planar embedding of a graph and the decomposition parameter k as input and returns an approximate hub cover HC^{approx}. Let S^i_j be a $(k+1)$-outerplanar graph induced by levels $jk+i$ to $(j+1)k+i$ and $\overline{S^i_j}$ is the optimal solution of S^i_j. For instance, Figures 5(a)-5(d) demonstrate $S^1_0, S^1_1, S^1_2, S^1_3$ for $i = 1$. In lines 5 to 13, for each partition, Algorithm 1 iterates as follows: In line 8, a subgraph lying between boundary levels $jk+i$ and $(j+1)k+i$ is obtained. Then, in lines 9 and 10, the IP formulation of that subgraph is solved and the solution is added



(a) 1-3 (b) 3-5 (c) 5-7 (d) 7-8

Figure 5: The overlapping 3-outerplanar graphs when $i = 1$ and $k = 2$.



(a) 1-2 (b) 2-4 (c) 4-6 (d) 6-8

Figure 6: The overlapping 3-outerplanar graphs when $i = 2$ and $k = 2$.

to the current partial feasible solution of partition i denoted as HC^i. When the algorithm exits the inner loop, a feasible MHC is obtained for the partition i. After iterating for all partitions, in line 14, the solution with minimum cardinality is selected as an approximate hub cover.

Algorithm 1

1: **Input:** A planar embedding of G, the vertices lying in each level, and k
2: **Output:** HC^{approx}
3: $HC^i \leftarrow \emptyset$ for each $i \in \{1, 2, \cdots, k\}$
4: $HC^{approx} \leftarrow \emptyset$
5: **for** $i := 1$ **to** k **do**
6: $\quad j \leftarrow 0$
7: $\quad$ **while** $(j+1)k + i$th level of G is available **do**
8: $\quad\quad$ Obtain S^i_j induced by levels $jk + i$ to $(j+1)k + i$
9: $\quad\quad$ Solve IP model (1)-(3) for S^i_j and obtain $\bar{S}^i_j$
10: $\quad\quad$ $HC^i = HC^i \cup \bar{S}^i_j$
11: $\quad\quad$ $j \leftarrow j + 1$
12: $\quad$ **end while**
13: **end for**
14: **Return** $HC^{approx} \leftarrow HC^p = \arg\min\{|HC^i| \,|1 \le i \le k\}$

The decomposition technique guarantees a feasible solution which is within a factor of $(k+1)/k$ from the optimal solution for a given $k \geq 1$. Proposition 3.1 gives a formal proof of this statement.

Proposition 3.1. *Algorithm 1 finds an approximate hub cover for a planar graph which is at most* $(k+1)/k$ *optimal.*

Proof. With the decomposition approach, the boundary levels of the $(k+1)$-outerplanar graphs, i.e., the overlapping levels, partition the graph into k pieces. Let V_i be the set of all vertices in the overlapping levels for each i, $1 \le i \le k$. Since the decomposition partitions the graph into k pieces, there exists at least one partition i such that at most $1/k$ of the vertices in HC^{opt} are included in V_i, where HC^{opt} is the optimal MHC in G. For each i, the union over j of the solutions gives a hub cover for the whole graph. Since only the vertices in V_i are counted twice, the cardinality of the solution is at most as follows:

$$|HC^{opt}| \le \bigcup_j \bar{S}^i_j \le |HC^{opt}| + |HC^{opt}|/k \le (k+1)|HC^{opt}|/k. \quad (4)$$

This completes the proof. □

To illustrate the set of vertices V_i in the overlapping levels, observe that in Figure 4, V_1 and V_2 are the vertices lying in levels 1, 3, 5, 7 and 2, 4, 6, 8, respectively, when $k = 2$.

3.3. Computational Considerations

Notice that the decomposition technique splits the problem into a set of subproblems that are independent from each other. This structure of the algorithm enables us to use a parallel implementation to solve the subproblems concurrently. Such an

implementation not only saves a considerable amount of computation time but it also allows handling extremely large problems for which even storing the graph in computer memory is a big burden.

Algorithm 1 generates feasible solutions that are obtained by taking the union of the optimal solutions of the subproblems. We observe that, if the subproblems have alternate optimal solutions, then the cardinality of the feasible solution set found by Algorithm 1 may not be unique. Depending on the alternate optimal solution selected for each subgraph, the union, that is the cardinality of the solution set, may change. Therefore, we added a subroutine to decrease the cardinality of the solution by decreasing the double coverages in the levels between two neighboring subproblems. The subroutine checks the optimal solution of the subproblem j and then perturbs the objective function coefficients of the neighboring subproblem $(j+1)$ before solving it. The objective function coefficients of the variables that are optimal in the jth subproblem are set to $1-\epsilon$ in the $(j+1)$th subproblem, where ϵ is a small non-negative number between 0 and 1. The subroutine helps neighboring subproblems generate similar optimal solutions, if there exists such an optimal solution.

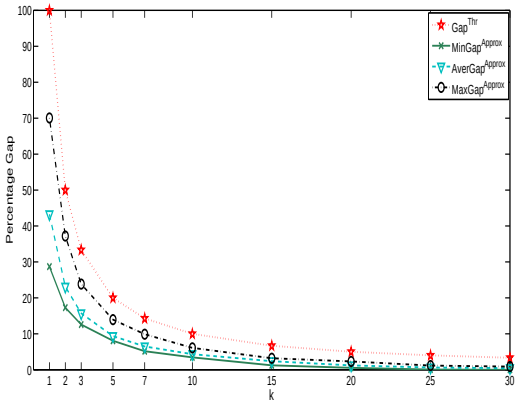
Next comes the fast heuristic that we mentioned in Section 1. The computation time of the approximation algorithm increases with k. As an alternate approach, we propose a *decomposition-based heuristic* which selects a partition i randomly among k different partitions. Then, we solve the subproblems resulting from partition i and take the union of the optimal solutions of those subproblems. The decomposition-based heuristic does not guarantee a performance ratio but it provides a feasible solution whose computation time is $1/k$ of that of the approximation algorithm.

4. Numerical Experiments

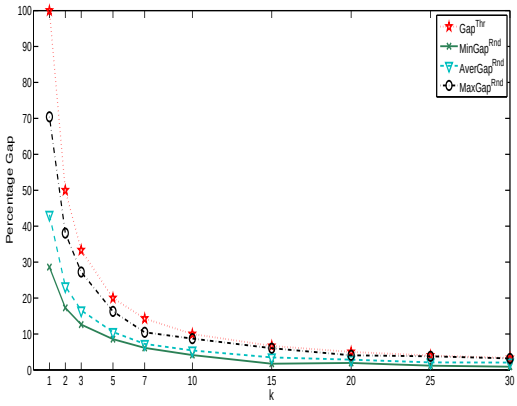
Approximation algorithms provide solutions with proven performance guarantees for computationally intractable problems. However, the bounds suggested by the theory are usually quite conservative. In this section, we conduct a set of experiments to compare the theoretical bound $(1+1/k)$ against the empirical performance of Algorithm 1. We also test how well the decomposition-based heuristic performs.

Before delving into the details, let us define the instances and the experimental setup. The proposed approximation algorithm and the decomposition-based heuristic were tested on synthetically generated planar graphs with known planar embeddings. Our problem set includes 20 planar graphs with different sizes from small to large. The numbers of vertices and edges range from several thousands to a million. The number of levels, on the other hand, ranges from 100 to 5,000. Optimal IP solutions were obtained by `IBM ILOG CPLEX Optimization Studio 12.6` running on a personal computer with Intel Xeon CPU E5-2630 and 64 GB of RAM. The upper limit on the solution time is set to 3,600 seconds for the `CPLEX` solver. The batch processing of the instances is carried out through C++ scripts. We used C++ libraries named Boost Asio and Thread to execute the algorithm in parallel.

Figure 7 shows how the empirical and theoretical performances of the approximation algorithm and the decomposition-based heuristic change with k. The theoretical performance of the approximation algorithm improves with increasing k and the optimality gap approaches 0 as k tends to infinity. It also demonstrates that the optimality gap of the approximation algorithm is far better than the theoretical gap $1/k$. For each value of k, we plot the minimum, average and maximum optimality gap observed over all instances versus the theoretical approximation ratio. These figures depict that when we increase k, both the empirical and theoretical performances of the algorithms get close to each other. Therefore, the rate of overestimation decreases considerably for large k. The results also indicate that even though the decomposition-based heuristic does not prove a theoretical performance bound, the optimality gaps are lower than the theoretical gap provided by the approximation algorithm. However, the maximum optimality gaps of the decomposition-based heuristic are slightly larger than that of the approximation algorithm.



(a) Approximation Algorithm



(b) Decomposition-based Heuristic

Figure 7: Observed vs. theoretical approximation gaps obtained by both the approximation algorithm and the decomposition-based heuristic.

Figure 8 compares the performances of `CPLEX` and the approximation algorithm in terms of solution time for different

values of k. The approximation algorithm can return a feasible solution with much less computational effort for many k values compared to CPLEX. Recall that k determines the number of levels in each subproblem so it affects the subproblem size. As expected, the empirical performance of the algorithm in terms of solution quality increases with k at the expense of high computation times. Therefore, it is very critical to determine the best value of k. The value of k should be large enough for good approximation but it should be less than a threshold value not to exceed the solution time of CPLEX. Figure 8 indicates that for small size instances, CPLEX outperforms the approximation algorithm when k is larger than 7. Those instances are solved to optimality within the time limit. Therefore, we especially focus on large problems for which CPLEX could not find an optimal solution within the time limit. Figure 8 demonstrates that for k values larger than 20, the solution time for CPLEX is less than that of the approximation algorithm. Overall, $k = 20$ seems like a compromise value for this set of instances.

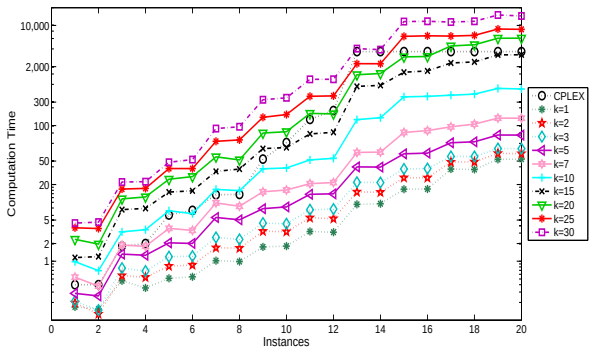
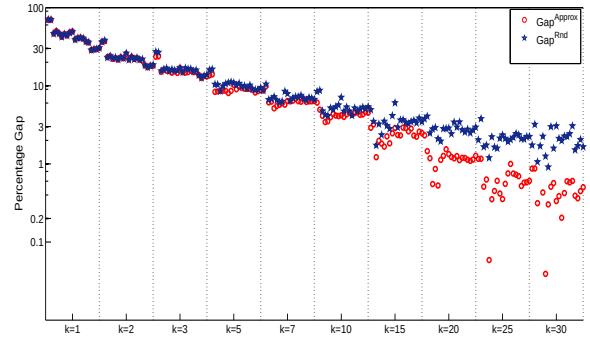
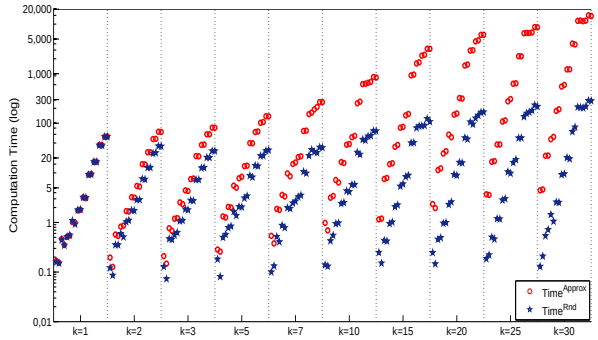


Figure 8: Computation times of the approximation algorithm and CPLEX.

Figure 9 compares the performances of the approximation algorithm and the decomposition-based heuristic in terms of both solution quality and time. Despite the fact that our heuristic does not guarantee a performance bound, the results demonstrate that the optimality gaps could be lower than the theoretical gap. As seen in Figure 9(a), the solution quality is comparable to that of the approximation algorithm. Since the number of feasible solutions computed by the approximation algorithm increases with k, we need to invest much more computational effort for the approximation algorithm than for the decomposition-based heuristic. Therefore, as seen in Figure 9(b), the approximation algorithm is clearly outperformed by the decomposition-based heuristic in terms of solution time. The decomposition-based heuristic returns a feasible solution whose solution time is $1/k$ that of the approximation algorithm.



(a) Optimality gaps.



(b) Computation time.

Figure 9: Percentage gaps and computation times of the approximation algorithm and the decomposition-based heuristic.

5. Conclusion

In this study, we analyzed the MHC problem on planar graphs. The problem is $\mathcal{NP}$-hard, and hence, solving the problem to optimality is computationally intractable especially for large instances. Therefore, we have proposed a decomposition-based approximation algorithm to identify heuristic solutions of certifiable quality. The algorithm uses a planar graph decomposition technique introduced in (Baker, 1994) to partition a planar graph into smaller subgraphs with manageable sizes. This approach always returns a feasible solution even for large-scale problems.

We investigated the empirical performance of the algorithm extensively. Our computational results demonstrate that the empirical performance of the algorithm is far better than its theoretical performance. Alternatively, we proposed a decomposition-based heuristic without a proven performance bound. This heuristic obtains comparable results relative to the approximation algorithm in terms of solution quality. Moreover, its solution time is on average several times less than that of the approximation algorithm. We discussed that the solution quality is affected by the particular optimal solution of a subproblem selected among various alternate optimal solutions. Since it is very time consuming to evaluate all combinations of the optimal solutions, finding a combination that is good enough for a particular application is an interesting question that we plan to address in our future research.

In the numerical experiments, we used a set of planar graphs with given planar embeddings. Alternatively, one can imple-

ment the polynomial algorithm discussed in (Kammer, 2007) to obtain a planar embedding with the minimum number of levels for each planar graph. Since the complexity depends only on the number of levels ($k + 1$) in each subproblem and not on the number of levels of the original problem, starting with a planar embedding with the smallest number of levels does not change the theoretical complexity of the algorithm or the approximation bound. Even so, using a planar embedding with the minimal number of levels may still affect the computation time. Clearly, decreasing the number of levels in the original problem decreases the number of subproblems for a given k. However, this is achieved at the expense of an increase in the number of vertices in each subproblem. Therefore, it is important to note that we may end up solving fewer but harder subproblems with more variables.

The decomposition algorithm, by its nature, is amenable to a parallel implementation. In this study, we have used a straightforward shared-memory implementation of the algorithm that helped us save significant computation time. In fact, it is possible with the proposed approach to partition a graph and make use of multiple memory locations in a network. This is of interest to those practitioners, who deal with huge-scale graphs for their problems that are difficult to manage on a single computer. Therefore, obtaining computational results in a distributed computing environment is also in our future research agenda.

References

Baker, B., 1994. Approximation algorithms for $\mathcal{NP}$-complete problems. Journal of the Association for Computing Machinery 41, 153–180.

Baloch, S., Krim, H., 2010. Object recognition through topo-geometric shape models using error-tolerant subgraph isomorphisms. IEEE Transactions on Image Processing 19, 1191–1200.

Bienstock, D., Monma, C., 1990. On the complexity of embedding planar graphs to minimize certain distance measures. Algorithmica 5, 93–109.

Cook, S., 1971. The complexity of theorem-proving procedures. In: 3rd ACM Symposium on Theory of Computing. Ohio, pp. 151–158.

Garey, M., Johnson, D., Stockmeyer, L., 1976. Some simplified np-complete graph problems. Theoretical Computer Science 1, 237–267.

Hopcroft, J., R.Tarjan, 1974. Efficient planarity testing. Journal of the ACM 21, 549–568.

Jamil, H. M., 2011. Computing subgraph isomorphic queries using structural unification and minimum graph structures. In: SAC. pp. 1053–1058.

Kammer, F., 2007. Determining the smallest k such that G is k-outerplanar. Lecture Notes in Computer Science 4698, 359–370.

Lipets, V., Vanetik, N., Gudes, E., 2009. Subsea: an efficient heuristic algorithm for subgraph isomorphism. Data Min. Knowl. Discov. 19, 320–350.

Llados, J., Marti, E., Villanueva, J., 2001. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 1137–1143.

Neuhaus, M., Bunke, H., 2005. A graph matching based approach to fingerprint classification using directional variance. Audio-and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science 3546, 191–200.

Rivero, C., Jamil, H. M., 2014a. On isomorphic matching of large disk resident graphs using an xquery engine, international Workshop on Graph Data Management: Techniques and Applications.

Rivero, C. R., Jamil, H. M., 2014b. Exact subgraph isomorphism using graphlets and minimum hub covers, work-in-process.

Shang, H., Zhang, Y., Lin, X., Yu, J., 2008. Taming verification hardness: An efficient algorithm for testing subgraph isomorphism. In: Journal Proceedings of the VLDB Endowment. Vol. 1. Auckland, New Zealand, pp. 364–375.

Ullmann, J., 1976. An algorithm for subgraph isomorphism. Journal of the ACM 23, 31–42.

Weber, M., Liwicki, M., Dengel, A., 2012. Faster subgraph isomorphism detection by well-founded total order indexing. Pattern Recognition Letters 33, 2011–2019.

Yelbay, B., 2014. Minimum hub cover problem: Solution methods and applications. Ph.D. thesis, Sabanci University.

Yelbay, B., Ş. İ. Birbil, Bülbül, K., Jamil, H. M., 2013. Trade-offs computing minimum hub cover toward optimized graph query processing. `http://arxiv.org/abs/1311.1626`.

Zhu, K., Zhang, Y., Lin, X., Zhu, G., Wang, W., 2010. A novel and efficient framework for finding subgraph isomorphism mappings in large graphs. In: 15th International Conference on Database Systems for Advanced Applications. Tsukuba, Japan, pp. 140–154.