# Graph Covering and Subgraph Problems

Gabrovšek, Peter and Mihelič, Jurij

**Abstract:** ***Combinatorial optimization problems and graph theory play an important role in several fields including logistics, artificial intelligence, software engineering, etc. In this paper, we explore a subset of combinatorial graph problems which we placed into two groups, namely, graph covering and subgraph problems. Here, the former include well-known dominating set and vertex cover problem, and the latter include clique and independent set problem. For each problem we give its definition, report state-of-the-art exact algorithms as well as heuristic and approximation algorithms, and list real-world applications if we were able to obtain any. At the end of the paper, we also relate the problems among each other.***

**Index Terms**: ***combinatorial optimization, graph covering, subgraph problems, exact algorithms, approximation algorithms***

## 1. Introduction

Graphs are often used by computer scientists and other researchers for modeling various problems from practice. In particular, one can find them in areas such as social network analysis, wireless networks, gaming industry, advertising, bioinformatics [8, 26, 23, 27].

In this paper, we review several popular graph problems and their variants. We include *covering problems* where the goal is to select either a subset of vertices or edges of the graph in order to "cover" the whole graph. Additionally, we also include problems, sometimes called *subgraph problems*, where the goal is to either select such a subset of vertices or edges of the graph that all the selected objects are either adjacent or not-adjacent to each other. We also discuss various relations among the problems.

Manuscript received May, 2019.

The authors are with the Faculty of Computer and Information Science, University of Ljubljana, Slovenia (e-mail: peter.gabrovsek@fri.uni-lj.si).

For each problem, we first present its formal definition and give its hardness of solving. Afterward, we also list several exact, heuristic and approximation algorithms for solving it.

Our survey methodology is as follows. First we investigated definitions in several generic sources such as online compendiums[1], encyclopedias[2] and books [1, 13, 14]. Afterward, we searched for the research papers on the focused graph problems using Google scholar. Our goal was to supplement each problem with references to exact, heuristics and approximation algorithms. Moreover, we also supplemented our descriptions with real-world applications. Finally, we also collected a list of reductions among the selected problems.

Let us present a few basic notions used in the rest of the paper. An *undirected graph* $G = (V, E)$ consist of a set of vertices $V$ and a set of edges $E \subseteq V \times V$. We also use $n = |V|$ for the number of vertices in the graph and $m = |E|$ for the number of edges in the graph. An example of a graph is shown in Figure 1.

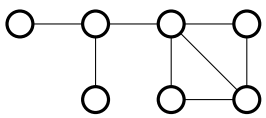

Figure 1: An example of a graph

To describe the asymptotic complexity of algorithms a well-known Oh-notation is used. We write $f(n) = O(g(n))$ and say that a function $f(n)$ is bounded from above by $g(n)$ when there exists two constants $c > 0$ and $n_0 > 0$ such that for all $n > n_0$ it holds that

$$|f(n)| \leq c \cdot |g(n)|.$$

For exponential time algorithms we oftentimes also use the $O^*$ notation, which is defined as

$$O^*(c^n) = O(c^n \cdot poly(n)),$$

[1] https://www.nada.kth.se/ viggo/problemlist/
[2] https://www.wikipedia.org/

where $poly(n)$ in a polynomial in $n$. This notation allows us to ignore the polynomial factor in the overall exponential time complexity.

In the paper, we also mention several polynomial-time approximation algorithms. A problem may exhibit an approximation algorithm that guarantees a constant approximation ratio. We also say that such problems belong to the class APX ("approximable") [1].

The rest of the paper is organized as follows. In section 2 we describe and analyze graph covering problems. Afterward, in section 3 we describe and analyze subgraph problems. In section 4 we summarize the paper by showing several relations among the problems described in the paper. Finally, we conclude the paper in Section 5.

## 2. Graph Covering Problems

In this section, we focus on graph covering problems where the task is to select some objects, i.e., either vertices or edges, in such a way that the selected objects cover the whole graph, i.e., either all vertices or all edges of the graph. For both objects that are covering and objects that are covered, we have two possibilities, thus altogether we have four different covering problems presented in Table 1.

| | | what is covered | |
|---|---|---|---|
| | | $V$ | $E$ |
| what covers | $S \subseteq V$ | dominating set | vertex cover |
| | $S \subseteq E$ | edge cover | edge dominating set |

Table 1: Graph covering problems

In particular, the problems are

- dominating set: vertices cover vertices,
- vertex cover: vertices cover edges,
- edge cover: edges cover vertices, and
- edge dominating set: edges cover edges.

In the rest of this section, we explore each of the above-listed problems.

### 2.1 Dominating Set Problem

A *dominating set* of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that every vertex in $V$ is either in $S$ or is adjacent to at least one vertex of $S$, i.e.

$$\forall v \in V \setminus S \; \exists (u, v) \in E : u \in S$$

We also say that vertices of $D$ cover (or dominate) vertices of the graph. An example of a dominating set is shown in Figure 2. Here the vertices marked with red cover all vertices of the given graph.

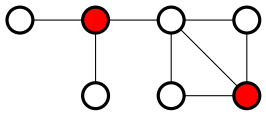

Figure 2: An example of a dominating set

A minimum dominating set of a graph is a dominating set with the smallest number of vertices. The size of such minimum dominating set is oftentimes called *domination number* and is denoted with $\gamma(G)$. The problem of finding a minimum dominating set is $\mathcal{NP}$-hard [14].

Several variants of the problem exist. For example, a *connected dominating set* is a dominating set that is also a connected component. A minimum connected dominating set is equivalent to a *maximum leaf spanning tree*, where the goal is to find a spanning tree in a graph with the highest number of leaves [1].

Various dominating set problems appear in social networks, mobile networks, sensor networks [23, 15], etc. Dominating sets algorithms can also be used to solve other combinatorial optimization problems, for example, k-center problem [21].

**Algorithms.** An exact algorithm, which finds a minimum dominating set by exhaustively enumerating all vertex subsets, has a computational time complexity of $O^*(2^n)$.

If we tackle the problem with basic use of Measure and Conquer method the running time improves to $O^*(1.5259^n)$ [13]. The fastest algorithm we were able to find has a running time of $O^*(1.4969^n)$ [25] and also uses a Measure and Conquer method with additional edge case analysis and optimization techniques.

Minimum dominating set problem is in not the class $APX$ which means that the problem does not have an approximation algorithm with a constant approximation guarantee. In particular, the quality depends on the problem size [16] and is approximable within the factor of $1 + \log n$.

### 2.2 Vertex Cover Problem

A *vertex cover* of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that at least one endpoint of each edge is in $S$, i.e., $u \in S \vee v \in S$ for all $(u, v) \in E$. An example of a vertex cover is shown in Figure 3.

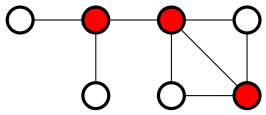

Figure 3: An example of a vertex cover

In the minimum vertex cover problem, the goal is to find a vertex cover with the smallest number of vertices. The size of such set is called *vertex cover number* and denoted with $\tau(G)$.

The vertex cover problem has been used for sequence alignment in biology and biochemistry [3].

**Algorithms.** A trivial exact algorithm finds a minimum vertex cover by exhaustive enumeration. It checks whether a vertex has to be in the solution or not. The computational time complexity of this approach is $O^*(2^n)$.

The vertex cover is one of the most popular fixed-parameter tractable (FPT) problems. Currently the best FPT algorithm finds a minimum vertex cover in $O(1.28^k + k \cdot n)$ time, where $k$ represents the size of the solution, i.e., $k = |S|$ [4].

Unlike dominating set, vertex cover is in the APX class, which means that constant-factor approximation algorithms exist for the problem. A simple greedy algorithm based on maximal matching guarantees an approximation factor of $2$. Vertex cover cannot be approximated within a factor lower than $1.36$ for arbitrary graphs [6]. By using semidefinite-programming formulation of vertex cover with the addition of antipodal points, we can achieve the approximation factor of $2 - \Theta\left(\frac{1}{\sqrt{\log(n)}}\right)$ [17]. For other algorithms see also [20, 21].

### 2.3 Edge Cover Problem

An *edge cover* of a graph $G = (V, E)$ is a set of edges $C$ such that each vertex in $G$ is incident with at least one edge in $C$, i.e., $\forall v \in V, \exists e \in E, v \in e$. An example of the edge cover is shown in Figure 4.

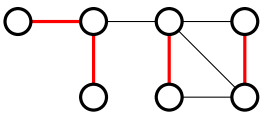

Figure 4: An example of an edge cover

A minimum edge cover of a graph is an edge cover with the smallest amount of edges possible. The *edge covering number* $\rho(G)$ is the size of the minimum edge cover.

**Algorithms.** The smallest edge cover of a graph can be found in polynomial time [18]. At first, we find a maximal matching which we then greedily extend to the solution. Thus the computational complexity of such an algorithm is $O(n^2 m)$.

### 2.4 Edge Dominating Set Problem

An *edge dominating set* of a graph $G = (V, E)$ is a subset $D \subseteq E$ such that every edge not in $D$ is adjacent to at least one edge in $D$, i.e.,

$$\forall (u, v) \in E \; \exists w \in V : (u, w) \in E \vee (v, w) \in E.$$

An example of edge dominating set is shown in Figure 5.

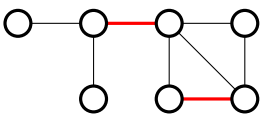

Figure 5: An example of an edge dominating set

A minimum edge dominating set is an edge dominating set of the smallest size. The size of the minimum edge dominating set is denoted with $\beta(G)$.

**Algorithms.** A trivial exact algorithm for edge dominating set has a computational time complexity of $O(2^m m)$. It converts the graph to its line graph and calculates a minimum dominating set on it. A line graph is a graph where on one hand vertices represent edges of the original graph, but on the other hand, the edges represent which edges in the original graph are adjacent. The best known exact algorithm uses Measure and Conquer method and has a time complexity of $O(1.3160^n n)$ [28].

A simple approximation algorithm ensures an approximation factor of 2: find any maximal matching. Edge dominating set cannot be approximated by a factor lower than $7/6$, but an algorithm with an approximation factor of $3/2$ exists [5].

## 3. Subgraph Problems

In this section we consider subgraph problems which are listed in table 2. These problems primarily search for the maximum subset of either vertices or edges, depending on the problem.

| connectivity \ what is in solution | V | E |
|---|---|---|
| D | independent set | matching |
| C | clique | edge clique |

Table 2: Properties of subgraph problems. In the connectivity dimension C means connected, D means disconnected

### 3.1 Independent Set Problem

An *independent set* of a graph [13] (also called stable set, coclique or anticlique) is a subset $C \subseteq V$ of vertices in a graph, no two of which are adjacent, i.e.,

$$\forall u, v \in C : (u, v) \notin E.$$

An example of an independent set is shown in Figure 6.

The goal of the maximum independent set problem is to find an independent set of the largest size possible. The independence number $\alpha(G)$ represents the size of the maximum independent set on $G$.

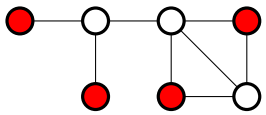

Figure 6: An example of an independent set

**Algorithms.** An exhaustive enumeration algorithm uses a naive brute force to examine every possible vertex subset and checks whether it is an independent set or not and has a computational time complexity of $O(n^2 2^n)$.

There exists a simple algorithm that uses a Measure and Conquer method with computational time complexity of $O(1.2210^n)$ [12]. The fastest algorithm we were able to find has a time complexity of $O(1.1996^n)$ and polynomial space [29] which also uses a Measure and Conquer method.

The maximum independent set problem does not have a polynomial algorithm which would produce a solution with a constant approximation ratio. Although the problem is not constant approximable it is approximable within $\frac{n}{(\log n)^2}$ [1].

### 3.2 Matching Problem

A *matching* (also called independent edge set) of a graph is a subset of edges without common endpoints. An example is shown in Figure 7.

The minimum maximal matching problem searches for the smallest matching in a graph that cannot be increased. We know other problems on matching, like maximal matching, perfect matching, etc., but they can be usually calculated in polynomial time or they apply on special cases of graphs so we will not consider them in this paper.

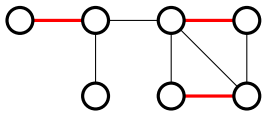

Figure 7: An example of a matching

**Algorithms.** The naive algorithm for minimum maximal matching has a computational time complexity of $O(2^n k)$, enumerates all maximal matchings, and chooses the one of minimum size.

Minimum maximal matching is approximable within the factor of 2. If we consider a solution size there exists an algorithm which has a computational time complexity of $O^*(2.61813^k)$ [10]. On the other hand, an algorithm that considered the size of the input and uses Branch & Reduce has a computational complexity of $O(1.4082^n)$ [11].

### 3.3 Clique Problem

A *clique* of a graph is a $C \subseteq V$ subset of vertices such that every two distinct vertices in the clique are adjacent, i.e.,

$$\forall u, v \in C : (u, v) \in E.$$

An example of clique is shown in Figure 8.

Maximum clique is the largest subgraph of fully connected vertices. The size of the maximum clique is denoted by $\omega(G)$.

Clique problem has a lot of applications in the real world. Cliques are used in social networks [7, 19], bioinformatics [2, 24], chemistry [22], etc.

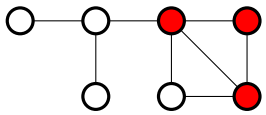

Figure 8: An example of a clique

**Algorithms.** Exhaustive enumeration algorithm checks all the combinations of vertices if they construct a clique and has a computational time complexity of $O(2^n)$. Currently, the best-known algorithm that runs in $O(1.888^n)$ uses a combination of analysis of local structures, independent set, and dynamic programming.

A clique problem cannot be approximated within the approximation ratio of a constant factor. This means that a clique problem does not belong to $APX$ class. Although it is not constant approximable it is approximable within $\Omega\left(\left(\frac{\log n}{\log \log n}\right)^2\right)$ and polynomial time [9].

### 3.4 Edge Clique Problem

For the completeness of Table 2 we also consider edge clique problem which has not been formally defined as such.

A *edge clique* is a set of edges in which every edge is adjacent to another edge in edge clique An example of an edge clique is shown in Figure 9. An edge clique is either a star or a triangle.

Maximum edge clique problem searches for the largest edge clique, i.e., a subset of edges that contain a vertex of the highest degree or they form a triangle.

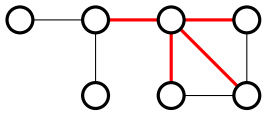

Figure 9: An example of an edge clique

**Algorithms.** An exact algorithm for edge clique is trivial and runs in linear time $O(n)$. We have to find a vertex $v$ of the highest degree and the solution are the edges incident to the vertex $v$.

## 4. Relations Among Problems

In this section, we discuss various interesting relations among the problems presented in this paper. See also Figure 10 for the graph of problem relations, where each node contains a problem and a labeled edge represents the relation. The relations among problems are as explained in the next list.

1. A dominating set in line graph $L(G)$ corresponds to an edge dominating set in $G$ [14].
2. An independent set is also a dominating set if and only if it is a maximal independent set.
3. An independent set of a graph $G$ corresponds to a clique of the graph's complement $\overline{G}$.
4. A set of vertices is a vertex cover if and only if its complement is an independent set [14].
5. Endpoints of any maximal matching are vertex cover [14].

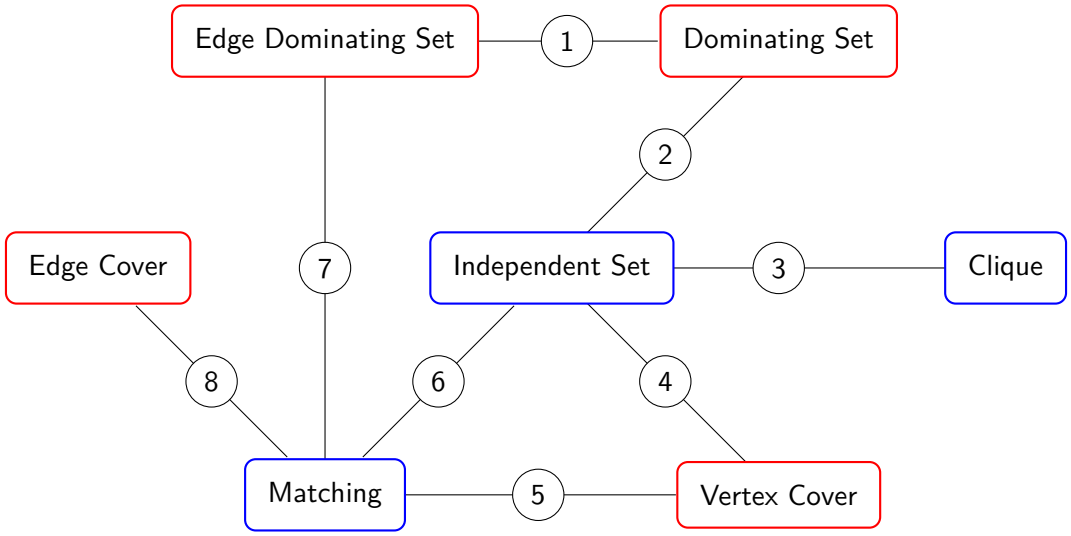


Figure 10: Relations among considered problems. Problems in red are graph covering problems, problems in blue are subgraph problems.

6. An independent set in line graph $L(G)$ corresponds to a matching in $G$.

7. Size of minimum maximal matching is equal to the size of the minimum edge dominating set.

8. A perfect matching (if it exists) is a minimum edge cover.

## 5. Conclusion

In this paper we surveyed several related combinatorial problems form the field of graph theory. Our focus was on covering problem where the goal is to cover some objects, e.g., to cover all vertices of a graph with a subset of vertices, as well as on subgraph problems, where the goal is to find a special subgraph of a given graph, e.g. a clique or matching.

Our goal was also to list several exact and heuristic algorithms for the problems. Interestingly, many exact algorithms are based on the Measure & Conquer algorithm design method, which is similar to Branch & Reduce method with an additional subproblem analysis and then reducing the size of the problem accordingly.

Finally, we also gathered relations among the problems giving us insight into how the problems can be transformed into each other. We believe that such a survey gives a nice introduction into the field of combinatorial graph problems.

## References

[1] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[2] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.

[3] James Cheetham, Frank Dehne, Andrew Rau-Chaplin, Ulrike Stege, and Peter J Taillon. Solving large fpt problems on coarse-grained parallel machines. *Journal of Computer and System Sciences*, 67(4):691–706, 2003.

[4] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *International Symposium on Mathematical Foundations of Computer Science*, pages 238–249. Springer, 2006.

[5] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of edge dominating set problems. *Journal of Combinatorial Optimization*, 11(3):279–290, 2006.

[6] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.

[7] Patrick Doreian and Katherine L Woodard. Defining and locating cores and boundaries of social networks. *Social networks*, 16(4):267–293, 1994.

[8] Uéverton dos Santos Souza, Frances Rosamond, Michael R Fellows, Fábio Protti, and Maise Dantas

da Silva. The flood-it game parameterized by the vertex cover number. *Electronic Notes in Discrete Mathematics*, 50:35–40, 2015.

[9] Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225, 2004.

[10] Henning Fernau. Edge dominating set: Efficient enumeration-based exact algorithms. In *International Workshop on Parameterized and Exact Computation*, pages 142–153. Springer, 2006.

[11] Fedor V Fomin, Serge Gaspers, and Saket Saurabh. Branching and treewidth based exact algorithms. In *International Symposium on Algorithms and Computation*, pages 16–25. Springer, 2006.

[12] Fedor V Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and conquer: a simple o (2 0.288 n) independent set algorithm. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 18–25. Society for Industrial and Applied Mathematics, 2006.

[13] Fedor V Fomin and Dieter Kratsch. *Exact exponential algorithms*. Springer Science & Business Media, 2010.

[14] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[15] Peng Jiang, Jun Liu, Feng Wu, Jianzhong Wang, and Anke Xue. Node deployment algorithm for underwater sensor networks based on connected dominating set. *Sensors*, 16(3):388, 2016.

[16] David S Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

[17] George Karakostas. A better approximation ratio for the vertex cover problem. In *International Colloquium on Automata, Languages, and Programming*, pages 1043–1050. Springer, 2005.

[18] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001.

[19] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[20] Jurij Mihelič and Borut Robič. Algoritmi za problem najmanjšega vozliščnega pokritja. In *Zbornik trinajste mednarodne elektrotehniške in računalniške konference ERK 2004*, pages 115–118. IEEE Region 8, Slovenska sekcija IEEE, 2004.

[21] Jurij Mihelič and Borut Robič. Solving the k-center problem efficiently with a dominating set algorithm. *Journal of computing and information technology*, 13(3):225–234, 2005.

[22] Nicholas Rhodes, Peter Willett, Alain Calvet, James B Dunbar, and Christine Humblet. Clip: similarity searching of 3d databases using clique detection. *Journal of chemical information and computer sciences*, 43(2):443–448, 2003.

[23] A Sasireka and AH Nandhu Kishore. Applications of dominating set of graph in computer networks. *International Journal of Engineering Sciences & Research Technology (IJESRT), Vol. No. 3, Issue No*, 1:170–173, 2014.

[24] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1):S136–S144, 2002.

[25] Johan MM Van Rooij and Hans L Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011.

[26] Feng Wang, Erika Camacho, and Kuai Xu. Positive influence dominating set in online social networks. In *International Conference on Combinatorial Optimization and Applications*, pages 313–321. Springer, 2009.

[27] Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 7–14. Citeseer, 1999.

[28] Mingyu Xiao and Hiroshi Nagamochi. A refined exact algorithm for edge dominating set. *Theoretical Computer Science*, 560:207–216, 2014.

[29] Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017.

**Peter Gabrovšek** received his master degree in Computer Science from the University of Ljubljana in 2017. Currently, he is with the Laboratory of Algorithmics, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, as an assistant and PhD student. His research interests include algorithmics, mathematics, and neural networks.

**Jurij Mihelič** received his doctoral degree in Computer Science from the University of Ljubljana in 2006. Currently, he is with the Laboratory of Algorithmics, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, as an assistant professor. His research interests include algorithm engineering, combinatorial optimization, system software, and programming languages.