mathematics

MDPI

Article

An Exact Algorithm for Minimum Vertex Cover Problem

Luzhi Wang, Shuli Hu, Mingyang Li and Junping Zhou *

School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

* Correspondence: zhoujp877@nenu.edu.cn

Received: 8 May 2019; Accepted: 2 July 2019; Published: 6 July 2019

Abstract: In this paper, we propose a branch-and-bound algorithm to solve exactly the minimum vertex cover (MVC) problem. Since a tight lower bound for MVC has a significant influence on the efficiency of a branch-and-bound algorithm, we define two novel lower bounds to help prune the search space. One is based on the degree of vertices, and the other is based on MaxSAT reasoning. The experiment confirms that our algorithm is faster than previous exact algorithms and can find better results than heuristic algorithms.

Keywords: vertex cover; branch-and-bound; MaxSAT; exact algorithm

1. Introduction

In an undirected graph, $G = (V, E)$, where V is the set of vertices and E is the set of edges. A vertex cover (VC) is a subset $S \subseteq V$ in which each edge in G has at least one endpoint in S. The minimum vertex cover (MVC) problem is to find the minimum size of the vertex cover in a graph. The MVC problem is a typical NP-complete problem, which plays an important role in many practical applications, such as network security, parallel machine scheduling [1], financial networks [2], and economics [3]. The MVC problem is closely related to the well-known maximum clique (MC) problem. A minimum vertex cover of the graph G can be implemented by finding a maximum clique of the complementary graph of G (denoted by $\overline{G}$). Unfortunately, when G is large and sparse, searching for a maximum clique in $\overline{G}$ is very hard in practice, while it is usually relatively easy to find a minimum vertex cover in G. Thus, finding an algorithm that solves MVC directly is needed.

There exist a remarkable number of swarm intelligence algorithms for the MVC problem. For example, the work in [4] provided an algorithm for MVC using an ant colony optimization algorithm. The work in [5] gave an algorithm based on the simulated annealing algorithm. In addition to the above algorithms, many new algorithms have been developed. The work in [6] proposed a solving algorithm using DNA molecules. The work in [7] adopted a quantum algorithm to design the MVC algorithm. The work in [8] presented a deterministic data structure approximation ratio $(2 + \epsilon)$ to solve the MVC problem, where ϵ is a small constant. The work in [9] addressed a learning automaton-based algorithm. The work in [10] integrated reinforcement learning to solve the MVC problem.

Compared with the algorithms mentioned above, exact algorithms can prove the optimality of solutions. The studies on exact algorithms were mainly concentrated on fixed parametrized algorithms. We highlight the fixed parametrized algorithms described in [11–16]. Although these studies have made tremendous progress in theory, there is still a huge gap between theory and practice. To our best knowledge, SBMS [17] is the only empirically exact algorithm which is especially good at solving minimum weight vertex cover(MWVC). The Satisfiability problem (SAT) is to decide whether some Boolean formula is satisfiable or not. SBMS solves MWVC with efficient SAT solvers.

In this paper, we introduce a new exact algorithm, EMVC, for solving the MVC problem, which is based on the branch-and-bound (BnB) search scheme. In EMVC, we provide two approaches, *DegLB* and *SatLB*, to compute the lower bound. *DegLB* is based on the degrees of vertices, and *SatLB* is based on the MaxSAT reasoning. We carry out experiments on the DIMACS benchmark set which is a challenge for the Maximum Clique problem.The results show that EMVC is an extremely competitive exact solver.

In Section 2, we introduce some basic concepts related to the MVC problem. We propose the framework of the EMVC algorithm in Section 3. We also present three methods to calculate lower bounds to prune the searches in Section 4. Section 5 shows the experimental results of EMVC, along with experiments validating the effectiveness of the proposed novel ideas under the DIMACS Implementation Challenge. Finally, we give some conclusions and future works.

2. Basic Definition and Notation

Given an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices and $E = \{e_1, e_2, ..., e_m\}$ is a set of edges, a vertex cover (VC) is a subset $S \subseteq V$, such that each edge in G has at least one endpoint in S. An edge is said to be covered if at least one endpoint of the edge is in the vertex cover S. The minimum vertex cover (MVC) problem is to find the minimum size of vertex cover in a graph. For a vertex v in G, the neighborhood of v, denoted by $N(v)$, is the set containing all vertices adjacent to v, i.e., $N(v) = \{u \in V | (u, v) \in E\}$. The closed neighborhood of v is $N^*(v) = N(v) \cup \{v\}$. The degree of a vertex v, denoted by $d(v)$, is the number of edges incident to v. The density D of G is computed as $2 \times |E|/(|V| \times (|V| - 1))$. $G \setminus T$ is a sub-graph derived from G by removing all vertices of T and all edges with at least a vertex in T. An independent set I of G is a subset of vertices such that no two vertices in I are adjacent, i.e., for any $(u, v) \in I \times I$, $(u, v) \notin E$. The maximum independent set (MIS) problem is to find the maximum size of independent set I in a graph. A clique C is a subset of vertices of G in which every two vertices are adjacent, i.e., for any $(u, v) \in C \times C$, $(u, v) \in E$. The maximum clique (MC) problem consists of finding a clique with the maximum number of vertices. The complement graph of G is $\overline{G}$, $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) | u, v \in V \wedge u \neq v \wedge (u, v) \notin E\}$. A clique of G is an independent set of $\overline{G}$ and vice versa.

Example 1. *Figure 1 gives an undirected graph* $G = (V, E)$, *where* $V = \{v_1, v_2, v_3, v_4, v_5\}$ *and* $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5)\}$. *The minimum vertex cover is* $\{v_1, v_3, v_4\}$.

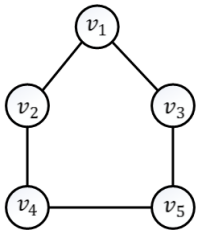

Figure 1. A graph of Example 1.

3. EMVC Algorithm for Solving the MVC Problem

In this section, we propose the EMVCalgorithm, which employs the branch-and-bound framework. The *EMVC* algorithm explores the search tree in a depth-first manner, which searches for the optimal solution by recursively branching on a vertex and thus creating two new nodes until there is no vertex in G. In other words, at first, the algorithm searches the complete space of solutions. When it branches on a vertex, the solution space is divided into a set of smaller subsets, and it obtains the relative upper and lower bound for each node to further reduce the search space. The outline of the algorithm is presented in Algorithm 1. The algorithm takes the graph G, the upper bound UB, and a growing partial vertex cover C as inputs. The upper bound UB, which is the overestimation of the size of the minimum vertex cover of the sub-graph, can be obtained by computing the size of the

minimum vertex cover found so far. When the size of the current partial vertex cover plus the the lower bound of the sub-graph is equal to or greater than UB, we prune the search and return UB as the size of the minimum vertex cover (Lines 1–2). In the algorithm, we use three methods to compute the lower bound of the sub-graph, including *DegLB*, *SatLB*, and *ClqLB*. The *DegLB* and the *SatLB* are the two new methods to calculate the lower bound, which we will describe in the next section. The *ClqLB* adopts the clique to figure out the lower bound [18]. The *ClqLB* decomposes the graph G into a set of disjoint cliques C_1, C_2, ..., C_r. Then, the lower bound of G is:

$$ClqLB(G) = \sum_{i=1}^{r} (|C_i| - 1) \tag{1}$$

where r is the number of cliques. If there is no vertex in G, the current growing partial vertex cover is the minimum vertex cover S_{min} (Lines 3–4). Otherwise, the EMVC extends the current minimum vertex cover by selecting a vertex v with the maximum degree and generates two branches, i.e., two relaxed graphs $G \setminus N^*(v)$ and $G \setminus v$ (Lines 5–7). Finally, EMVC repeats until it finds the best solution (Line 8).

Algorithm 1: The EMVC algorithm (G, UB, C).

Input: a graph $G = (V, E)$, an upper bound $UB = |V|$, and a growing partial vertex cover $C = \emptyset$.
Output: the size of the minimum vertex cover S_{min} of G.

1 **if** $|C| + max(DegLB(G), ClqLB(G), SatLB(G)) \geq UB$ **then**
2 **return** UB;
3 **if** $G = \emptyset$ **then**
4 **return** $|C|$;
5 Select a vertex v from V with the maximum degree;
6 $|C_1| \leftarrow$ EMVC$(G \setminus N^*(v), UB, C \cup N(v))$;
7 $|C_2| \leftarrow$ EMVC$(G \setminus v, min(UB, |C_1|), C \cup v)$;
8 **return** $min(|C_1|, |C_2|)$;

4. Two Novel Lower Bounds for the MVC Problem

The lower bound is an important aspect of the branch-and-bound algorithm because explicit enumeration is normally impossible due to the exponentially-increasing number of potential solutions. The use of the lower bound enables the branch-and-bound algorithm to search only parts of the solution space implicitly. Thus, designing a tight lower bound is crucial to enhance the efficiency of the branch-and-bound algorithm. In this section, we provide two novel methods to compute the lower bound.

4.1. DegLB: The Degree-Based Lower Bound for MVC

The first lower bound *DegLB* uses the degrees of vertices to compute. We select the vertex with the largest degree, name it as v'_1, and update the degree of other vertices. Then, we select the largest vertex among the remaining vertices and name it v'_2, and so on. We find a vertex v'_i such that $d(v'_1) + d(v'_2) + ... + d(v'_i) \geq |E|$ and $d(v'_1) + d(v'_2) + ... + d(v'_{i-1}) < |E|$. Then, the lower bound of G of the MVC problem is defined as follows.

$$DegLB(G) = \lfloor i + \frac{|E'|}{d(v'_{i+1})} \rfloor, \quad v'_{i+1} \in H \tag{2}$$

where $G \setminus \{v'_1, v'_2, ..., v'_i\} = (V', E')$.

Since the number of edges that i vertices covered is equal to or smaller than $|E|$, there may exist some edges that are not covered. Thus, we need at least $\frac{|E'|}{d(v'_{i+1})}$ vertices to cover $|E'|$.

Example 2. *Figure 1 illustrates an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5)\}$. In G, we sort the vertices with dynamic degrees. We find that $d(v_1) = 2$, so we choose v_1 as v'_1 and update other vertices. At present, $d(v_2) = 1, d(v_3) = 1, d(v_4) = 2, d(v_5) = 2$. Then, we choose v_4 as v'_2 and update the degrees as $d(v_2) = 0, d(v_3) = 0, d(v_5) = 1$. We choose v_5 as v'_3, and we find that $d(v'_1) + d(v'_2) + d(v'_3) \geq |E|$ and $d(v'_1) + d(v'_2) < |E|$. Thus, $i = 3$. We select three vertices v'_1, v'_2, and v'_3, where $|E| = 5$. Then, the sub-graph $G \setminus \{v'_1, v'_2, v'_3\} = (V', E')$, where $V' = \{v_2, v_3\}$ and $E' = \emptyset$. The lower bound of G obtained by DegLB is three, where $i = 3, |E'| = 0$.*

4.2. SatLB: The Max-SAT-Based Lower Bound for MVC

Before describing the lower bound *SatLB*, we concentrate on specifying some notions. A variable x may take a value of zero (*false*) or one (*true*). A literal l is a variable x or its negation $\overline{x}$. A clause is defined as a disjunction of literals, e.g., $c = l_1 \vee l_2 \vee ... \vee l_k$. A unit clause is a clause containing only one literal. A conjunctive normal form (CNF) formula F is a conjunction of clauses, denoted as $F = c_1 \wedge c_2 \wedge ... \wedge c_m$. A truth assignment is a map that assigns each variable a value. A clause is satisfied *iff* at least one literal takes the value *true*. The partial MaxSAT formula is a conjunction of hard and soft clauses, where the hard clause is a clause that must be satisfied, while a soft clause is a clause that may or may not be satisfied. The partial MaxSAT problem is to determine a truth assignment satisfying all hard clauses and maximizing the number of satisfied soft clauses. A subset of soft clauses is called an inconsistent subset *iff* the subset together with all hard clauses results in a contradiction (an empty clause).

Unit propagation (UP), which is based on unit clauses, is a successful technique widely used in SAT and MaxSAT. Since each clause needs to be satisfied, the single literal composed of the unit clause must be *true*. Thus, if a set of clauses contains the unit clause l, the other clauses are simplified by the application of the following UP rule:

- Every clause (other than the unit clause itself) containing l is removed;
- In every clause that contains $\overline{l}$, this literal $\overline{l}$ is deleted.

In the following, we will introduce a lower bound of the MVC problem with MaxSAT reasoning. At first, we reduce a graph into a MaxSAT instance. Given a graph $G = (V, E)$, G can be partitioned into a set of cliques $\{C_1, C_2, ..., C_m\}$, where $C_1 \cup C_2 \cup ... \cup C_m = V$ and $C_i \cap C_j = \emptyset$ $(i \neq j)$. The graph can be encoded into a MaxSAT instance as follows. (1) Each vertex v_i can be encoded into a variable x_i. (2) Each pair of non-connected vertices $(v_i, v_j) \notin E$ can be represented by a hard clause $\overline{x_i} \vee \overline{x_j}$. (3) A soft clause $x_i \vee x_j \vee ... \vee x_k$ is added for each clique $C_i = \{x_i, x_j, ..., x_k\}$.

Then, we compute the inconsistent subsets of the transformed MaxSAT instance using the UP approach. Specifically, given a CNF formula $F = c_1 \wedge c_2 \wedge ... \wedge c_m$, we use a stack ST to store all unit clauses including the ones generated by UP rule, where c_i $(1 \leq i \geq m)$ is the clause (hard clause or soft clause) in the transformed MaxSAT and m is the number of hard clauses plus the soft clauses in the transformed MaxSAT. In the beginning, ST stores all unit clauses in F, then we recursively pop out a unit clause from ST and apply the UP rule, which may produce new unit clauses, until no more unit clauses in the stack or an empty clause are produced. The soft clauses generating the empty clause using the UP rule build an inconsistent subset. After finding an inconsistent subset, the soft clauses in the subset are removed from the formula F, and we continue to find other inconsistent subsets in the remaining F until no inconsistent subsets can be found.

UP can be enhanced by failed literal detection [19]. A literal l is called a failed literal in the formula F if the UP rule working on $F \wedge l$ produces an empty clause. When both l and $\overline{l}$ are failed literals, the union of the soft clauses used to generate the two empty clauses constitutes an inconsistent subset. For the MaxSAT instance encoding an MVC instance, one does not detect whether or not a negative literal has failed, because a variable only has a positive occurrence in a soft clause.

After obtaining the number of inconsistent subsets, we can compute the lower bound of G of the MVC problem as follows.

Theorem 1. *Let $G = (V, E)$ be a graph. If G can be partitioned into k disjoint cliques and there are s inconsistent subsets in the transformed MaxSAT instance, then the lower bound of G for the MVC problem is $SatLB = |V| - k + s$.*

Proof. Since $|S_{MVC}| = |V| + |S_{MIS}|$, we can compute the lower bound of the MVC problem after obtaining the upper bound of the MISproblem, where S_{MVC} and S_{MIS} are the solutions of the MVC and MIS problems, respectively. If G can be partitioned into k disjoint cliques, selecting one vertex from each clique constitutes an independent set. k is the upper bound of the MIS problem. Assuming that an instance contains k soft clauses, the upper bound is less than or equal to $k - s$ if s disjoint inconsistent clause sets can be detected. After using MaxSAT reasoning, we can find s inconsistent subsets, which can further decrease the upper bound of MIS to $k - s$. Thus, the lower bound of the MVC problem is $|V| - k + s$. □

Example 3. *Let us also employ Figure 1 to illustrate how to obtain a lower bound using the SatLB method based on MaxSAT reasoning. We first encode the graph in Figure 1 into a partial MaxSAT formula. Since the graph can be partitioned into three disjoint cliques $\{v_1, v_2\}$, $\{v_3, v_5\}$, and $\{v_4\}$, the transformed partial MaxSAT formula is composed of hard clauses $c_1 = \overline{x_1} \vee \overline{x_2}, c_2 = \overline{x_1} \vee \overline{x_3}, c_3 = \overline{x_2} \vee \overline{x_4}, c_4 = \overline{x_3} \vee \overline{x_5}, c_5 = \overline{x_4} \vee \overline{x_5}$ and soft clauses $c'_1 = x_1 \vee x_2$, $c'_2 = x_3 \vee x_5$, and $c'_3 = x_4$. Then, the CNF formula $F = c_1 \wedge c_2 \wedge ... \wedge c_5 \wedge c'_1 \wedge c'_2 \wedge c'_3$, and the stack ST is initialized to c'_3. Then, based on Theorem 1, let c'_3 become unit propagation. Thus, x_4 is true. Then, we remove x_4 from hard clauses. Notice that hard clauses must be satisfied. Then, x_2 and x_5 are false. At this point, the soft clauses change. $c'_1 = x_1$, and $c'_2 = x_3$. When x_1 is true, to satisfied hard clauses, x_2 and x_3, are false. There is one conflict at this point because all soft clauses cannot be satisfied at the same time. Therefore, the upper bound of the independent set is narrowed to a size of $3 - 1$. Thus, the lower bound of MVC is $5 - 3 + 1 = 3$.*

5. Experimental Results

In this section, we carry out extensive experiments to test the performance of our algorithm EMVC on DIMACS instances obtained from NetworkRepository [20]. We compare EMVC with the exact algorithm SBMS and heuristic algorithm FastVC. FastVC is the most competitive heuristic solver for MVC so far [21]. SBMS is the unique exact algorithm for solving the MVC problem, which encodes MVC into SAT [17]. In the experiment, the cutoff time of all solvers was set to 1000 seconds. For the heuristic solver FastVC, we report the best solution (best) and average time (avgt) in 10 runs. For exact solvers, we recorded the time running on each instance and the size of the minimum vertex cover of each instance, denoted as S_m. For the sake of space, we only show the results on graphs with more than 200 vertices and densities greater than 0.7. This is because dense graphs are more challenging for the MVC problem. We implemented our algorithm in Java and compiled it using JDK1.7. All the experiments were run on CentOS Linux, with a 3.1 GHZ CPU and 64 GB of memory.

Experimental results on the classical random benchmark are shown in Table 1. Table 1 illustrates the comparison of the three algorithms, where "↑" means the heuristic algorithms did not find the optimal value. In the table, the runtimes are in seconds, and "-" means timeout or out of memory. From the table, we see that EMVC could find the optimum for most graphs, while SBMS failed on 27 instances; FastVC could not obtain the optimum on four instances. In addition, although the runtime of EMVC was more than FastVC, FastVC could not ensure the optimum result, and EMVC could solve these instances in a reasonable time. In general, these results indicate that EMVC is an extremely competitive exact solver.

Table 1. Comparison with different algorithms.

Instance	S_m	EMVCTime	SBMSTime	FastVCBest	FastVCAvgt
brock400-1	393	1.58	-	393	0.02
brock400-2	392	1.54	-	392	0.03
brock400-3	393	1.55	-	393	0.02
brock400-4	393	1.55	360.50	393	0.01
C250-9	245	0.15	14.11	245	0.01
C500-9	495	1.41	-	495	0.10
C1000-9	994	8.89	-	994	1.81
C2000-9	1994	78.33	-	1994	4.69
DSJC500-5	487	177.73	-	487	4.15
DSJC1000-5	-	-	-	986	48.91
gen400-p0-9-55	392	0.37	72.99	392	<0.01
gen400-p0-9-65	393	0.38	-	393	<0.01
gen400-p0-9-75	394	0.37	46.59	394	<0.01
hamming10-2	1022	3.50	-	1022	0.27
hamming10-4	1004	13.55	-	1004	0.05
hamming8-2	254	0.48	-	254	0.01
johnson32-2-4	465	0.64	-	465	0.03
keller5	745	2.38	-	745	0.02
keller6	3298	186.54	-	3298	1.45
MANN-a27	375	0.23	10.57	375	0.01
MANN-a45	1032	3.04	-	1032	0.12
MANN-a81	3318	114.15	-	3318	1.57
p-hat1000-3	989	177.51	-	991 ↑	0.03
p-hat1500-3	-	-	-	1489	35.87
p-hat300-1	292	0.56	93.12	292	<0.01
p-hat300-3	291	0.62	154.61	291	0.01
p-hat500-3	490	4.34	-	493↑	0.14
p-hat700-3	-	-	-	638	0.01
san1000	933	10.25	-	933	0.02
san200-0-7-1	191	0.29	-	194 ↑	0.02
san400-0-5-1	368	0.75	-	368	0.01
san400-0-7-1	389	0.44	-	390↑	0.02
san400-0-7-2	385	0.44	231.90	385	<0.01
san400-0-7-3	381	0.41	132.99	381	<0.01
san400-0-9-1	395	0.45	-	395	0.01
sanr400-0-7	392	2.43	-	392	0.06

6. Summary and Future Work

In this paper, we proposed a new exact algorithm EMVC for MVC. In the algorithm, we defined two tight lower bounds to reduce the search space. Extensive experiments showed that our algorithm achieved good performance. In the future, we will study variants of the reduction rules to improve the performance of the MVC solver.

Author Contributions: Software, L.W. ; methodology, L.W.; writing, original draft preparation, L.W. and J.Z.; writing, review and edit, L.W., S.H., M.L. and J.Z.

Funding: This work was fully supported by the National Natural Science Foundation of China under Grant No. 61370156, No. 61403076, and No. 61403077; the Large-scale Scientific Instrument and Equipment Sharing Project of Jilin Province (20150623024TC-03); and the Fundamental Research Funds for the Central Universities (2412019FZ050).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hong, W.; Wang, Z. Improved Approximation Algorithm for the Combination of Parallel Machine Scheduling and Vertex Cover. *Int. J. Found. Comput. Sci.* **2017**, *28*, 977–992. [CrossRef]
2. Boginski, V.; Butenko, S.; Pardalos, P.M. Mining market data: A network approach. *Comput. Oper. Res.* **2006**, *33*, 3171–3184. [CrossRef]
3. Wu, Q.; Hao, J.K. A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.* **2015**, *242*, 693–709. [CrossRef]
4. Shyu, S.J.; Yin, P.Y.; Lin, B.M.T. An Ant Colony Optimization Algorithm for the Minimum Weight Vertex Cover Problem. *Ann. Oper. Res.* **2004**, *131*, 283–304. [CrossRef]
5. Xu, X.; Ma, J. An efficient simulated annealing algorithm for the minimum vertex cover problem. *Neurocomputing* **2006**, *69*, 913–916. [CrossRef]
6. Wang, Z.; Huang, D.; Pei, R. Solving the minimum vertex cover problem with DNA molecules in Adleman-Lipton model. *J. Comput. Theor. Nanosci.* **2014**, *11*, 521–523. [CrossRef]
7. Chang, W.-L.; Ren, T.-T.; Feng, M. Quantum algorithms and mathematical formulations of biomolecular solutions of the vertex cover problem in the finite-dimensional hilbert space. *IEEE Trans. Nanobiosci.* **2015**, *14*, 121–128. [CrossRef] [PubMed]
8. Bhattacharya, S.; Henzinger, M.; Nanongkai, D. Fully dynamic approximate maximum matching and minimum vertex cover in O (log3 n) worst case update time. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, Barcelona, Spain, 16–19 January 2017; pp. 470–489.
9. Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; Song, L. Learning combinatorial optimization algorithms over graphs. *arXiv* **2017**, arXiv:1704.01665.
10. Mousavian, A.; Rezvanian, A.; Meybodi, M.R. Solving minimum vertex cover problem using learning automata. *arXiv* **2013**, arXiv: 1311.7215.
11. Katrenič, J. A faster FPT algorithm for 3-path vertex cover. *Inf. Process. Lett.* **2016**, *116*, 273–278. [CrossRef]
12. Stege, U.; Fellows, M.R. An Improved Fixed Parameter Tractable Algorithm for Vertex Cover. In *Technical Report/Departement Informatik*; ETH: Zürich, Switzerland, 1999; Volume 318.
13. Niedermeier, R.; Rossmanith, P. On efficient fixed-parameter algorithms for weighted vertex cover. *J. Algorithms* **2003**, *47*, 63–77. [CrossRef]
14. Iwata, Y.; Oka, K.; Yoshida, Y. Linear-time FPT algorithms via network flow. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, Portland, OR, USA, 5–7 January 2014; pp. 1749–1761.
15. Cygan, M.; Pilipczuk, M. Split Vertex Deletion meets Vertex Cover: New fixed-parameter and exact exponential-time algorithms. *Inf. Process. Lett.* **2013**, *113*, 179–182. [CrossRef]
16. Abu-Khzam, F.N.; Langston, M.A.; Suters, W.H. Fast, effective vertex cover kernelization: A tale of two algorithms. In Proceedings of the ACS/IEEE International Conference on Computer Systems & Applications, Cairo, Egypt, 3–6 January 2005.
17. Xu, H.; Kumar, T.K.S.; Koenig, S. A New Solver for the Minimum Weighted Vertex Cover Problem. In Proceedings of the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming, Banff, AB, Canada, 29 May –1 June 2016.
18. Tomita, E.; Sutani, Y.; Higashi, T.; Takahashi, S.; Wakatsuki, M. A simple and faster branch-and-bound algorithm for finding a maximum clique. In Proceedings of the International Workshop on Algorithms and Computation, Haka, Bangladesh, 10–12 February 2010; pp. 191–203.
19. Li, C.M.; Fang, Z.; Xu, K. Combining MaxSAT Reasoning and Incremental Upper Bound for the Maximum Clique Problem. In Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 4–6 November 2013.

20. Rossi, R.; Ahmed, N. The Network Data Repository with Interactive Graph Analytics and Visualization. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
21. Cai, S.; Lin, J.; Luo, C. Finding a small vertex cover in massive sparse graphs: Construct, local search, and preprocess. *J. Artif. Intell. Res.* **2017**, *59*, 463–494. [CrossRef]