Theoretical Aspects of Generating Instances with Unique Solutions: Pre-assignment Models for Unique Vertex Cover

Takashi Horiyama[1], Yasuaki Kobayashi[1], Hirotaka Ono[2], Kazuhisa Seto[1], Ryu Suzuki[3]
[1]Faculty of Information Science and Technology, Hokkaido University, Japan
[2]Graduate School of Informatics, Nagoya University, Japan
[3]Graduate School of Information Science and Technology, Hokkaido University, Japan
horiyama@ist.hokudai.ac.jp, koba@ist.hokudai.ac.jp, ono@nagoya-u.jp, seto@ist.hokudai.ac.jp, suzuki-r1991@eis.hokudai.ac.jp

Abstract

The uniqueness of an optimal solution to a combinatorial optimization problem attracts many fields of researchers' attention because it has a wide range of applications, it is related to important classes in computational complexity, and an instance with only one solution is often critical for algorithm designs in theory. However, as the authors know, there is no major benchmark set consisting of only instances with unique solutions, and no algorithm generating instances with unique solutions is known; a systematic approach to getting a problem instance guaranteed having a unique solution would be helpful. A possible approach is as follows: Given a problem instance, we specify a small part of a solution in advance so that only one optimal solution meets the specification. This paper formulates such a "pre-assignment" approach for the vertex cover problem as a typical combinatorial optimization problem and discusses its computational complexity. First, we show that the problem is Σ_2^P-complete in general, while the problem becomes NP-complete when an input graph is bipartite. We then present an $O(2.1996^n)$-time algorithm for general graphs and an $O(1.9181^n)$-time algorithm for bipartite graphs, where n is the number of vertices. The latter is based on an FPT algorithm with $O^*(3.6791^\tau)$ time for vertex cover number τ. Furthermore, we show that the problem for trees can be solved in $O(1.4143^n)$ time.

Introduction

Preparing a good benchmark set is indispensable for evaluating the actual performance of problem solvers, such as SAT solvers, combinatorial optimization solvers, and learning algorithms. This is the reason that instance generation is a classical topic in the fields of AI and OR, including optimization and learning. Indeed, there are well-known benchmark sets, such as the TSPLIB benchmark set for the traveling salesperson problem (Reinelt 1991), the UCI Machine Learning Repository dataset for machine learning (Asuncion and Newman 2007), SATLIB for SAT (Hoos and Stützle 2000), and other benchmark sets for various graph optimization problems in the DIMACS benchmarks (DIMACS 2022). In these benchmarks, instances are generated from actual data or artificially generated. Instances generated from actual data are closely related to the application, so it is a very instance that should be solved in practice and is therefore suitable as a benchmark. On the other hand, it is not easy to generate sufficient instances because each one is hand-made, and there are confidentiality issues. Random generation is often used to compensate for this problem and has also been studied for a long time, especially how to generate good instances from several points of view, such "difficulty" control and variety.

In this paper, we consider generating problem instances with unique solutions. It is pointed out that the "unique solution" is related to the nature of intractability in SAT. Indeed, PPSZ, currently the fastest SAT algorithm in theory (Scheder 2022), is based on the unique k-SAT solver. Furthermore, a type of problem that determines whether or not it has a unique solution (e.g., UNIQUE SAT) belongs to a class above NP and coNP (Blass and Gurevich 1982), but the status of that class is still not well understood, where an exception is TSP; UNIQUE TSP is known to be Δ_2^p-complete (Papadimitriou 1984). Thus, it is desirable to have a sufficient number of instances with unique solutions in many cases, such as when problem instances with unique solutions are used as benchmarks or when we evaluate the performance of a solver for the uniqueness check version of combinatorial optimization problems. In other words, instances with unique solutions have an important role in experimental studies of computational complexity theory.

In this paper, we focus on VERTEX COVER problem as a representative of combinatorial optimization problems (also referred Drosophila in the field of parameterized algorithms (Downey and Fellows 2013)), propose a "pre-assignment" model as an instance generation model for this problem, and consider the computational complexity of instance generation based on this model. One of the crucial points in generating problem instances with unique solutions is that it is difficult for unweighted combinatorial optimization problems to apply approaches used in conventional generation methods. A simple and conventional way of generating instances is a random generation, but the probability that a randomly generated problem instance has a unique solution is low. Furthermore, although some combinatorial problems have randomized algorithms modifying an instance into an instance with a unique solution, they strongly depend on the structure of the problems, and it is non-trivial to apply them to other problems, including VERTEX COVER (Valiant and Vazirani 1986; Mulmuley, Vazirani, and Vazirani 1987).

Besides, the planted model used in SAT instance generation requires guaranteeing the non-existence of solutions other than the assumed solution, which is also a non-trivial task, or some planted instances are shown to be easy to solve (Krivelevich and Vilenchik 2006). Thus, generating instances with unique solutions requires a different approach than conventional instance generation methods.

A pre-assignment for *uniquification* in a problem means assigning an arbitrary part of the decision variables in the problem so that only one solution is consistent with the assignment. In SAT, for example, we select some of the Boolean variables and assign true or false to each so that only one truth assignment consistent with the pre-assignment satisfies the formula. In the case of VERTEX COVER, we select some vertices and assign a role to each, i.e., a cover vertex or a non-cover vertex, so that only one minimum vertex cover of the graph is consistent with the pre-assignment. Such a pre-assignment for uniquification enables the generation of problem instances with unique solutions to combinatorial optimization problems. In the case of VERTEX COVER, the graph obtained by the pre-assignment (see Observation 1 for the detail) has only one vertex cover with the minimum size, which can be used as an instance of UNIQUE VERTEX COVER problem. A strong point of the pre-assignment-based instance generation is that it transforms an instance of an ordinary problem into an instance of the uniqueness check version of the corresponding problem; if the ordinary version of the problem has a rich benchmark instance set, we can expect to obtain the same amount of instance set by applying the pre-assignment models. Indeed, many papers study instance generations of the ordinary combinatorial optimization problem as seen later; they can be translated into the results of instances with unique solutions.

From this perspective, we formulate the pre-assignment problem to *uniquify* an optimal solution for the vertex cover problem instance. We name the problem PAU-VC, whose formal definition is given in the Models section. We consider three types of scenarios of pre-assignment: INCLUDE, EXCLUDE, and MIXED. Under these models, this study aims to determine the computational complexity of the pre-assignment for uniquification of VERTEX COVER.

Due to the space limitation, some proofs (marked $\star$) are omitted, which can be found in (Horiyama et al. 2023).

Our results. We investigate the complexity of PAU-VC for all pre-assignment models. As hardness results, we show:

1. PAU-VC is Σ_2^P-complete under any model,
2. PAU-VC for bipartite graphs is NP-complete under any model.

For the positive side, we design exact exponential-time algorithms and FPT algorithms for the vertex cover number. Let n be the number of the vertices and τ be the vertex cover number of G. Then, we present

3. an $O(2.1996^n)$-time algorithm, which works for any model,
4. an $O^*(3.6791^\tau)$-time algorithm under INCLUDE model,[1]
5. an $O^*(3.6791^\tau)$-time algorithm under MIXED and EXCLUDE models,

Due to the FPT algorithms, PAU-VC for bipartite graphs can be solved in $O(1.9181^n)$ time. Here, it should be noted that the fourth and fifth algorithms work in different ways, though they have the same running time.

The last results are for trees. Many graph problems are intractable for general graphs but polynomially solvable for trees. Most of such problems are just NP-complete, but some are even PSPACE-complete (Demaine et al. 2015). Thus, many readers might consider that PAU-VC for trees is likely solvable in polynomial time. On the other hand, not a few problems (e.g., NODE KAYLES) are intractable in general, but the time complexity for trees still remains open, and only exponential-time algorithms are known (Bodlaender, Kratsch, and Timmer 2015; Yoshiwatari et al. 2022). In the case of PAU-VC, no polynomial-time algorithm for trees is currently known. Instead, we give an exponential upper bound of the time complexity of PAU-VC for trees.

6. an $O(1.4143^n)$-time algorithm for trees under INCLUDE model, and
7. an $O(1.4143^n)$-time algorithm for trees under MIXED and EXCLUDE models.

These algorithms also work differently, though they have the same running time.

In the context of instance generation, these results imply that the pre-assignment approach could not be universally promising but would work well to generate instances with small vertex covers.

Related work. Instance generation in combinatorial (optimization) problems is well-studied from several points of view, such as controlling some attributes (Asahiro, Iwama, and Miyano 1996) and hard instance generation (Horie and Watanabe 1997), and some empirical studies use such generated instances for performance evaluation (Cha and Iwama 1995). Such instances are desirable to be sufficiently hard because they are supposed to be used for practical performance evaluation of algorithms for problems considered hard in computational complexity theory (e.g., (Sanchis 1995; Neuen and Schweitzer 2017; McCreesh et al. 2018)). Other than these, many studies present empirically practical instance generators (e.g., (Ullrich et al. 2018)). From the computational complexity side, it is shown that, unless NP = coNP, for most NP-hard problems, there exists no polynomial-time algorithm capable of generating all instances of the problem, with known answers (Sanchis 1990; Matsuyama and Miyazaki 2021).

A natural application of pre-assignment for uniquification is puzzle instance generation. In pencil puzzles such as SUDOKU, an instance is supposed to have a unique solution as the answer. Inspired by them, Demaine et al. (Demaine et al. 2018) define FCP (Fewest Clues Problem) type problems including FCP-SAT and FCP-SUDOKU. The FCP-SAT is defined as follows: Let ϕ be a CNF formula with a set of boolean variables X. Consider a subset $Y \subseteq X$ of variables and a partial assignment $f_Y : Y \to \{0, 1\}$. If there is a unique satisfying assignment $f_X : X \to \{0, 1\}$ extending

[1]The O^* notation suppresses polynomial factors in n.

f_Y, that is, $f_Y(x) = f_X(x)$ for $x \in Y$, we call variables in Y *clues*. The FCP-SAT problem asks, given a CNF formula ϕ and an integer k, whether ϕ has a unique satisfying assignment with at most k clues. They showed that FCP 3SAT, the FCP versions of several variants of the SAT problem, including FCP 1-IN-3 SAT, are Σ_2^P-complete. Also FCP versions of several pencil-and-paper puzzles, including SUDOKU are Σ_2^P-complete. Since the setting of FCP is suitable for puzzles, FCP versions of puzzles are investigated (Higuchi and Kimura 2019; Goergen et al. 2022).

The uniqueness of an (optimal) solution itself has been intensively and extensively studied for many combinatorial optimization problems. Some of them are shown to have the same time complexity to UNIQUE SAT (Hudry and Lobstein 2019a,b), while UNIQUE TSP is shown to be Δ_2^P-complete (Papadimitriou 1984). Juban (Juban 1999) provides the dichotomy theorem for checking whether there exists a unique solution to a given propositional formula. The theorem partitions the instances of the problem between the polynomial-time solvable and coNP-hard cases, where Horn, anti-Horn, affine, and 2SAT formulas are the only polynomial-time solvable cases.

Preliminaries

Notations. Let G be an undirected graph. Let $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. For $v \in V(G)$, we denote by $N_G(v)$ the set of neighbors of v in G, i.e., $N_G(v) = \{w \in V(G) : \{v, w\} \in E(G)\}$. We extend this notation to sets: $N_G(X) = \bigcup_{v \in X} N_G(v) \setminus X$. We may omit the subscript G when no confusion is possible. For $X \subseteq V(G)$, the subgraph of G obtained by deleting all vertices in X is denoted by $G - X$ or $G \setminus X$.

A *vertex cover* of G is a vertex set $C \subseteq V(G)$ such that for every edge in G, at least one end vertex is contained in C. We particularly call a vertex cover of G with cardinality k a *k-vertex cover* of G. The minimum cardinality of a vertex cover of G is denoted by $\tau(G)$. A set $I \subseteq V(G)$ of vertices that are pairwise non-adjacent in G is called an *independent set* of G. It is well known that I is an independent set of G if and only if $V(G) \setminus I$ is a vertex cover of G. A *dominating set* of G is a vertex subset $D \subseteq V(G)$ such that $V(G) = D \cup N(D)$, that is, every vertex in G is contained in D or has a neighbor in D. If a dominating set D is also an independent set of G, we call it an *independent dominating set* of G.

Models. In PAU-VC, given a graph G and an integer k, the goal is to determine whether G has a "feasible pre-assignment of size at most k". In this context, there are three possible models to consider: INCLUDE, EXCLUDE, and MIXED. Under INCLUDE model, a pre-assignment is defined as a vertex subset $U \subseteq V(G)$, and a pre-assignment $\tilde{U}$ is said to be *feasible* if there is a minimum vertex cover U^* of G (i.e., $\tau(G) = |U^*|$) such that $U \subseteq U^*$ and for every other minimum vertex cover U of G, $\tilde{U} \setminus U$ is nonempty. In other words, $\tilde{U}$ is feasible if there is a unique minimum vertex cover U^* of G that includes $\tilde{U}$ (i.e., $\tilde{U} \subseteq U^*$). Under EXCLUDE model, a pre-assignment is also defined as a vertex subset $\tilde{U} \subseteq V(G)$ as well as INCLUDE, and $\tilde{U}$ is said to be feasible if there is a unique minimum vertex cover U^* of G that excludes X (i.e., $U^* \subseteq V(G) \setminus \tilde{U}$). For these two models, the size of a pre-assignment is defined as its cardinality. Under MIXED model, a pre-assignment is defined as a pair of vertex subsets $(\tilde{U}_{\mathrm{in}}, \tilde{U}_{\mathrm{ex}})$, and $(\tilde{U}_{\mathrm{in}}, \tilde{U}_{\mathrm{ex}})$ is said to be feasible if there is a unique minimum vertex cover U^* of G that includes $\tilde{U}_{\mathrm{in}}$ and excludes $\tilde{U}_{\mathrm{ex}}$ (i.e., $\tilde{U}_{\mathrm{in}} \subseteq U^* \subseteq V(G) \setminus \tilde{U}_{\mathrm{ex}}$). We can assume that $\tilde{U}_{\mathrm{in}}$ and $\tilde{U}_{\mathrm{ex}}$ are disjoint, as otherwise the pre-assignment is trivially infeasible. The size of the pre-assignment $(\tilde{U}_{\mathrm{in}}, \tilde{U}_{\mathrm{ex}})$ is defined as $|\tilde{U}_{\mathrm{in}}| + |\tilde{U}_{\mathrm{ex}}|$ under the MIXED model.

Observation 1. *Let G be a graph and let $(\tilde{U}_{\mathrm{in}}, \tilde{U}_{\mathrm{ex}})$ be a feasible pre-assignment of G. Then, $G - (\tilde{U}_{\mathrm{in}} \cup N(\tilde{U}_{\mathrm{ex}}))$ has a unique minimum vertex cover of size $\tau(G) - |\tilde{U}_{\mathrm{in}} \cup N(\tilde{U}_{\mathrm{ex}})|$.*

In the following, we compare these three models.

Theorem 1. *If G has a feasible pre-assignment of size at most k in the* INCLUDE *model, then G has a feasible pre-assignment of size at most k under the* EXCLUDE *model.*

Proof. Let $\tilde{U} \subseteq V(G)$ be a feasible pre-assignment for G in the INCLUDE model. Let $U^* \subseteq V(G)$ be the unique minimum vertex cover of G such that $\tilde{U} \subseteq U^*$. Observe that $N(v) \setminus U^* \neq \emptyset$ for $v \in U^*$. This follows from the fact that if $N(v) \subseteq U^*$, $U^* \setminus \{v\}$ is also a vertex cover of G, which contradicts the minimality of U^*. For $v \in \tilde{U}$, we let v' be an arbitrary vertex in $N(v) \setminus U^*$ and define $\tilde{U}' \coloneqq \{v' : v \in \tilde{U}\}$. Note that v' and w' may not be distinct even for distinct $v, w \in \tilde{U}$. We claim that $\tilde{U}'$ is a feasible pre-assignment of G in the EXCLUDE model.

As $v' \notin U^*$ for each $v \in \tilde{U}$, $U^* \subseteq V(G) \setminus \tilde{U}'$ holds. To see the uniqueness of U^* (under the pre-assignment $\tilde{U}'$ in EXCLUDE), suppose that there is a minimum vertex cover U of G with $U \neq U^*$ such that $U \subseteq V(G) \setminus \tilde{U}'$. For each $v' \in \tilde{U}'$, all vertices in $N(v')$ must be included in U, which implies that $v \in U$. Thus, we have $\tilde{U} \subseteq U$, contradicting the uniqueness of U^*. By $|\tilde{U}'| \leq |\tilde{U}|$, the theorem holds. □

The converse of Theorem 1 does not hold in general. Let us consider a complete graph K_n with n vertices. As $\tau(K_n) = n - 1$, exactly one vertex is not included in a minimum vertex cover of K_n. Under EXCLUDE model, a pre-assignment containing exactly one vertex is feasible for K_n. However, under INCLUDE model, any pre-assignment of at most $n - 2$ vertices does not uniquify a minimum vertex cover of K_n. This indicates that EXCLUDE model is "stronger" than INCLUDE model. The theorem below shows that EXCLUDE model and MIXED model are "equivalent".

Theorem 2 ($\star$). *A graph G has a feasible pre-assignment of size at most k under the* EXCLUDE *model if and only if G has a feasible pre-assignment of size at most k under the* MIXED *model.*

Basic observations. A class $\mathcal{G}$ of graphs is said to be *hereditary* if for $G \in \mathcal{G}$, every induced subgraph of G belongs to $\mathcal{G}$.

Lemma 1. *Let $\mathcal{G}$ be a hereditary class of graphs. Suppose that there is an algorithm $\mathcal{A}$ for computing a minimum vertex cover of a given graph $G \in \mathcal{G}$ that runs in time $T(n, m)$, where $n = |V(G)|$ and $m = |E(G)|$.[2] Then, we can check whether $G \in \mathcal{G}$ has a unique minimum vertex cover in time $O(\tau(G) \cdot T(n, m))$.*

Proof. We first compute an arbitrary minimum vertex cover U of G. If U is not a unique minimum vertex cover of G, G has another minimum vertex cover U' not containing v for some $v \in U$. As $v \notin U'$, we have $N(v) \subseteq U'$. To find such a minimum vertex cover for each $v \in U$, it suffices to compute a vertex cover of $G - (N(v) \cup \{v\})$ of size $\tau(G) - |N(v)|$, which can be done by using $\mathcal{A}$ as $G-(N(v)\cup\{v\}) \in \mathcal{G}$. □

Using Lemma 1, we have the following corollary.

Corollary 1. *Let G be a hereditary class of graphs. Suppose that there is an algorithm for computing a minimum vertex cover of a given graph $G \in \mathcal{G}$ that runs in time $T(n, m)$, where $n = |V(G)|$ and $m = |E(G)|$. Given a graph $G \in \mathcal{G}$ and vertex sets $\tilde{U}_{\text{in}}, \tilde{U}_{\text{ex}} \subseteq V(G)$, a pre-assignment $(\tilde{U}_{\text{in}}, \tilde{U}_{\text{ex}})$ is feasible for G under the* MIXED *model in time $O(\tau(G) \cdot T(n, m))$.*

Proof. If either $\tilde{U}_{\text{in}} \cap \tilde{U}_{\text{ex}} \neq \emptyset$ or $\tilde{U}_{\text{ex}}$ is not an independent set of G, the pre-assignment is trivially infeasible. Suppose otherwise. Observe that G has a minimum vertex cover U with $\tilde{U}_{\text{in}} \subseteq C \subseteq V(G) \setminus \tilde{U}_{\text{ex}}$ if and only if $G - (\tilde{U}_{\text{in}} \cup N_G(\tilde{U}_{\text{ex}}))$ has a vertex cover of size $\tau(G)-|\tilde{U}_{\text{in}}\cup N_G(\tilde{U}_{\text{ex}})|$. Therefore, we can check whether G has a unique minimum vertex cover U satisfying $\tilde{U}_{\text{in}} \subseteq U \subseteq V(G) \setminus \tilde{U}_{\text{ex}}$ using the algorithm in Lemma 1 as well. □

Complexity

This section is devoted to proving the complexity of PAU-VC. In particular, we show that PAU-VC is Σ_2^P-complete on general graphs and NP-complete on bipartite graphs.

Theorem 3 (⋆)**.** PAU-VC *is Σ_2^P-complete under any type of pre-assignment.*

Theorem 3 is shown by performing a polynomial-reduction from FCP 1-IN-3 SAT, which is known to be Σ_2^P-complete (Demaine et al. 2018).

We consider the complexity of PAU-VC on bipartite graphs. As observed in Theorem 2, it suffices to consider INCLUDE and MIXED models. For bipartite graphs, VERTEX COVER is equivalent to the problem of finding a maximum cardinality matching by König's theorem (see Chapter 2 in (Diestel 2012)), which can be computed in polynomial time by the Hopcroft-Karp algorithm (Hopcroft and Karp 1973). Combining this fact and Corollary 1, PAU-VC belongs to NP under both models.

To see the NP-hardness of PAU-VC, we first consider the MIXED model. We perform a polynomial-time reduction from INDEPENDENT DOMINATING SET on bipartite graphs. In INDEPENDENT DOMINATING SET, we are given a graph G and an integer k and asked whether G has an independent dominating of size at most k. This problem is NP-complete even on bipartite graphs (Corneil and Perl 1984).

From a bipartite graph G for INDEPENDENT DOMINATING SET, we construct a graph G' by adding, for each vertex $v \in V(G)$, a new vertex v' and an edge between v and v'.

Now, we show that G has an independent dominating set of size at most k if and only if G has a feasible pre-assignment of size at most k.

Theorem 4. PAU-VC *for bipartite graphs is NP-complete under the* MIXED *and* EXCLUDE *models.*

By modifying the proof of Theorem 4, we can show that the hardness holds also for INCLUDE model.

Theorem 5 (⋆)**.** PAU-VC *for bipartite graphs is NP-complete under the* INCLUDE *model.*

Exact Algorithms

General Graphs

In this section, we present exact algorithms for PAU-VC for general graphs. We first see exact exponential-time algorithms, and then see FPT algorithms for vertex cover number. As shown in Theorem 2, we only consider INCLUDE and EXCLUDE models.

Exponential-time algorithms. We first present an exact algorithm that utilizes an algorithm for UNIQUE VERTEX COVER (UVC). Let G be a graph with n vertices. We first fix a subset of vertices $U \subseteq V(G)$ for a pre-assignment, and then check if G has a unique minimum vertex cover under the pre-assignment U. This can be done by transforming G into G' as in Corollary 1: $G' := G - U$ for the INCLUDE model and $G' := G - U - N(U)$ for the EXCLUDE model. By applying this procedure for all subsets U, we can determine the answer of PAU-VC for G. The running time depends on the algorithm to solve UVC. We abuse the notation UVC(G') in Algorithm 1: UVC(G') returns true if and only if G' has a unique minimum vertex cover of size $\tau(G) - |U|$ for the INCLUDE MODEL and size $\tau(G) - |N(U)|$ for the EXCLUDE model. By Lemma 1, UVC can be solved in the same exponential order of running time as VERTEX COVER. Here, let $O^*(\alpha^n)$ be the running time of an exact exponential-time algorithm for VERTEX COVER. Note that Algorithm 1 applies the VERTEX COVER algorithm for G', which has at most $n - |U|$ vertices in any pre-assignment model. Thus, the total running time of Algorithm 1 is estimated as

$$\sum_{k=0}^{n} \binom{n}{k} O^*(\alpha^{n-k}) = \sum_{k=0}^{n} \binom{n}{k} O^*(\alpha^k) = O^*((\alpha+1)^n),$$

Algorithm 1: Algorithm using UVC routine

```
for k = 0, 1, ..., n do
    for all U ⊆ V with |U| = k do
        G' ← the graph obtained from G under pre-assignment on U
        if UVC(G') = true then
            Return U
```

[2] We assume that $n + m \leq T(n, m) \leq T(n', m')$ for $n \leq n'$ and $m \leq m'$.

by the binomial theorem. Since the current fastest exact exponential-time algorithm for VERTEX COVER runs in $O(1.1996^n)$ time (Xiao and Nagamochi 2017), Algorithm 1 runs in $O(2.1996^n)$ time.

Theorem 6. PAU-VC *for any model can be solved in* $O(2.1996^n)$ *time.*

FPT algorithms parameterized by vertex cover number. We now present FPT algorithms parameterized by vertex cover number. Throughout this subsection, we simply write $\tau \coloneqq \tau(G)$. Contrary to the previous subsection, algorithms for INCLUDE and EXCLUDE work differently.

INCLUDE Model. We first present the algorithm for INCLUDE, which is easier to understand. The idea of the algorithm is as follows. Let G be a graph. We first enumerate all the minimum vertex covers of G. Then, for every optimal vertex cover U^*, we find a minimum feasible pre-assignment. Namely, we fix a subset $U \subseteq U^*$ as INCLUDE vertices, and then check the uniqueness under U as Algorithm 1. We formally describe the algorithm in Algorithm 2.

Inside of the algorithm, we invoke an algorithm for computing a minimum vertex cover of G' parameterized by vertex cover number $\tau(G')$, and $O^*(\beta^{\tau(G')})$ denotes the running time. Since G' has a vertex cover at most $\tau - |U|$, the running time of Algorithm 2 is estimated as follows:

$$\sum_{U^* \in \mathcal{U}} \sum_{U \subseteq U^*} O^*(\beta^{\tau-|U|}) = |\mathcal{U}| \sum_{k=0}^{\tau} \binom{\tau}{k} O^*(\beta^{\tau-k}) = O^*(|\mathcal{U}|(\beta+1)^\tau). \quad (1)$$

Thus bounding $|\mathcal{U}|$ by a function of τ is essential. Actually, $|\mathcal{U}|$ is at most 2^τ by the following reason. For $\mathcal{U}$, we consider classifying the optimal vertex covers of G into two categories for a non-isolated vertex $v \in V(G)$: (1) $\mathcal{U}(v)$ is the set of optimal vertex covers of G containing v, and (2) $\mathcal{U}(\bar{v})$ is the set of optimal vertex covers of G not containing v. Note that any vertex cover in $\mathcal{U}(\bar{v})$ contains all the vertices in $N(v)$ by the requirement of a vertex cover; a vertex cover in $\mathcal{U}(v)$ (resp., $\mathcal{U}(\bar{v})$) reserves v (resp., $N(v)$) for vertex cover and can include at most $\tau - 1$ (resp., $\tau - |N(v)|$) vertices. We can further classify $\mathcal{U}(v)$ (resp., $\mathcal{U}(\bar{v})$) into $\mathcal{U}(vv')$ and $\mathcal{U}(v\bar{v}')$ (resp., $\mathcal{U}(\bar{v}v')$ and $\mathcal{U}(\bar{v}\bar{v}')$) similarly, which gives a branching with depth at most τ. Since every optimal vertex cover is classified into a leaf of the branching tree, and thus $|\mathcal{U}| \leq 2^\tau$ holds; the above running time is $O^*((2\beta+2)^\tau)$.

Although it is impossible to improve bound 2^τ on the number of minimum vertex covers of G,[3] we can still improve the running time. The idea is to adopt a smaller set of optimal vertex covers instead of the whole set of the optimal vertex covers. Here, we focus on $\mathcal{U}(\boldsymbol{v})$, where $\boldsymbol{v}$ is a sequence of symbols representing a vertex or its negation, as discussed above. Since $\boldsymbol{v}$ has a history of branching, it reserves a set of vertices as a part of an optimal vertex cover in $\mathcal{U}(\boldsymbol{v})$. Let $U(\boldsymbol{v})$ denote the corresponding vertex set (i.e., the vertices appearing positively $\boldsymbol{v}$). Then, $\tau - |U(\boldsymbol{v})|$ vertices are needed to cover the edges in $G - U(\boldsymbol{v})$.

We now assume that $G - U(\boldsymbol{v})$ forms a collection of isolated edges. In such a case, we can exploit the structure of an optimal vertex cover without branching to leaves by the following argument. To uniquify a minimum vertex cover of G, we need to specify exactly one of two endpoints of each isolated edge. In Algorithm 2, we fix a target vertex cover U^* at line 2, choose a subset of U^* for pre-assignment INCLUDE at line 4, and check if it ensures the uniqueness of an optimal vertex cover. Instead of fixing a target vertex cover U^*, we focus on $U(\boldsymbol{v})$ with k' isolated edges, where $|U(\boldsymbol{v})|+k' = \tau$ holds (as otherwise G has no vertex cover extending $U(\boldsymbol{v})$ of size τ). That is, a pre-assignment with size k ($\geq k'$) on $U(\boldsymbol{v})$ with k' isolated edges must form k' end points of the isolated edges (i.e., one of $2^{k'}$ choices) and a $(k-k')$-subset of $U(\boldsymbol{v})$ (one of $\binom{|U(\boldsymbol{v})|}{k-k'}$). Thus, for all the pre-assignments on $U(\boldsymbol{v})$ plus k' isolated edges, the uniqueness check takes

$$2^{k'} \sum_{k''=0}^{\tau-k'} \binom{\tau-k'}{k''} \cdot O^*(\beta^{\tau-k'-k''}) = 2^{k'} \cdot O^*((\beta+1)^{\tau-k'}) = O^*((\beta+1)^\tau).$$

To obtain such $G - U(\boldsymbol{v})$ (i.e., a graph consisting of isolated edges), we do branching by v with a degree at least 2. If no vertex with a degree at least 2 is left, then $G - U(\boldsymbol{v})$ forms a collection of isolated edges. We now estimate the size $f(\tau)$ of a branching tree. If v with a degree at least 2 is in the vertex cover, the remaining graph has a vertex cover with at least $\tau - 1$ vertices. If v is not in the vertex cover, $N(v)$ should be included in a vertex cover; the remaining graph has a vertex cover with at least $\tau - 2$. Thus we have

$$f(\tau) \leq f(\tau-1) + f(\tau-2).$$

This recurrence inequality leads to $f(\tau) \leq \phi^\tau$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio. Overall, the running time of the revised Algorithm 2 becomes $O^*((\phi(\beta+1))^\tau)$. Since the current fastest vertex cover algorithm runs in $O^*(1.2738^\tau)$ time (Chen, Kanj, and Xia 2010), we can solve PAU-VC under INCLUDE model in $O^*(3.6791^\tau)$ time.

Theorem 7. PAU-VC *under* INCLUDE *model can be solved in* $O^*(3.6791^\tau)$ *time.*

Algorithm 2: FPT algorithm for INCLUDE

1: Enumerate all the optimal vertex covers of G, and let $\mathcal{U}$ be the collection of them.
2: **for all** $U^* \in \mathcal{U}$ **do**
3: $\quad$ Sol$(U^*) \leftarrow U^*$ ▷ *Current min. pre-asgmt. for* U^*
4: $\quad$ **for all** $U \subseteq U^*$ with $|U| < \tau$ **do**
5: $\quad\quad$ $G' \leftarrow$ the graph obtained from G under pre-assignment on U
6: $\quad\quad$ **if** $|U| <$ |Sol(U^*)| and UVC(G') = true **then**
7: $\quad\quad\quad$ Sol$(U^*) \leftarrow U$
8: **Return** Sol(U^*) with the minimum size for $U^* \in \mathcal{U}$

[3] If G is a disjoint union of $n/2$ isolated edges, G has exactly 2^τ minimum vertex covers.

EXCLUDE Model. We next consider EXCLUDE model. A straightforward extension of Algorithm 2 changes line 4 as:

$$\textbf{for all } \tilde{U} \subseteq V \setminus U^* \text{ with } |\tilde{U}| = k \textbf{ do} \tag{2}$$

In line 5, G' is defined from $\tilde{U}$ as in the EXCLUDE model. Although this change still guarantees that the algorithm works correctly, it affects the running time. Equation (1) becomes

$$|\mathcal{U}| \sum_{k=0}^{n-\tau} \binom{n-\tau}{k} O^*(\beta^{\tau-k}) = O^*(|\mathcal{U}|\beta^{\tau}(1+1/\beta)^{n-\tau}),$$

which is no longer to be fixed parameter tractable with respect to τ. To circumvent the problem, we show the following lemma.

Lemma 2. *Suppose two subsets $\tilde{U}_1$ and $\tilde{U}_2$ of $V(G)$ have the same neighbors, i.e., $N(\tilde{U}_1) = N(\tilde{U}_2)$. If a set $U \subseteq V \setminus \tilde{U}_1$ is an optimal vertex cover of G, $U \cap \tilde{U}_2 = \emptyset$ holds.*

Proof. Note that $N(\tilde{U}_1)(= N(\tilde{U}_2))$ and $U_1 \cup U_2$ are disjoint. Suppose that a set U with $\tilde{U}_1 \cap U = \emptyset$ is an optimal vertex cover of G. Then, $N(\tilde{U}_1) \subseteq U$ holds, because otherwise the edges incident with $\tilde{U}_1$ are never covered. Since $N(\tilde{U}_1)$ $(= N(\tilde{U}_2))$ covers also the edges incident with $\tilde{U}_2$, $U \setminus \tilde{U}_2$ $(\supseteq N(\tilde{U}_2))$ is also a vertex cover. By the optimality of U, $U \setminus \tilde{U}_2$ must be U, which implies $U \cap \tilde{U}_2 = \emptyset$. □

Lemma 2 implies that for two subsets $\tilde{U}_1$ and $\tilde{U}_2$ of $V(G)$ with $N(\tilde{U}_1) = N(\tilde{U}_2)$, excluding $\tilde{U}_1$ and excluding $\tilde{U}_2$ have the same effect as including $N(\tilde{U}_1)$ $(= N(\tilde{U}_2))$ for optimal vertex covers. Namely, for an optimal vertex cover U, the "cost" of including U' $(\subseteq U)$ is interpreted as the minimum $|\tilde{U}|$ such that $U' \subseteq N(\tilde{U})$. The notion of costs enables us to transform a pre-assignment of INCLUDE into one of EXCLUDE; we can utilize the same framework of Algorithm 2 by installing a device to compute the costs. We thus consider obtaining the costs efficiently. The cost of U' is the optimal value of the following set cover problem: The elements to be covered are the vertices in U', and a set to cover elements is $u \in V \setminus U$, which consists of $N(u)$. By executing a standard dynamic programming algorithm for the set cover problem for the ground set U once, we obtain the cost to cover every subset of U in $O^*(2^{|U|})$ time (also see e.g., (Fomin and Kratsch 2010)).

Algorithm 3: Algorithm for SET COVER

Input: $(X, \mathcal{S})$, where $X = \{1, 2, \ldots, n\}$ is a ground set (i.e., the set of elements to be covered) and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is a collection of subsets of X.

Output: For $S \subseteq X$, $\mathrm{cost}[S; j]$ is the minimum cardinality of a subset of $\{S_1, S_2, \ldots, S_j\}$ that covers $S \subseteq X$. If S cannot be covered by $\{S_1, S_2, \ldots, S_j\}$, cost is set to ∞.

1: **for all** $S \subseteq X$ **do**
2: **for** $j = 1, \ldots, n$ **do**
3: $\mathrm{cost}[S; j] \leftarrow \min\{\mathrm{cost}[S; j-1], \mathrm{cost}[S \setminus S_j; j-1] + 1\}$

Algorithm 4: FPT algorithm for EXCLUDE

1: Enumerate all the optimal vertex covers. Let $\mathcal{U}$ be the set of them.
2: **for all** optimal vertex cover U^* **do**
3: Apply Algorithm 3 for $(U^*, \{N(u) \mid u \in V(G) \setminus U^*\})$.
4: $\mathrm{Sol}(U^*) \leftarrow \mathbf{SC}(U^*)$
5: **for all** $U \subseteq U^*$ with $|\mathbf{SC}(U)| < |\mathrm{Sol}(U^*)|$ **do**
6: $G' \leftarrow$ the graph obtained from G under pre-assignment on $\mathbf{SC}(U)$ (EXCLUDE)
7: **if** $|U| < |\mathrm{Sol}(U^*)|$ and UVC(G') = true **then**
8: $\mathrm{Sol}(U^*) \leftarrow \mathbf{SC}(U)$
9: **Return** $\mathrm{Sol}(U^*)$ with the minimum size for $U^* \in \mathcal{U}$

Note that Algorithm 3 for $(X, \mathcal{S})$ computes the optimal cost to cover S (denoted $\mathrm{cost}[S]$) but can easily modified to compute an optimal solution $\mathbf{SC}[S]$ for every $S \subseteq X$ in the same running time.

Algorithm 4 is the algorithm for EXCLUDE model. It works in a similar framework to Algorithm 2, but it equips the device to transform INCLUDE pre-assignment into EXCLUDE pre-assignment at line 3, which solves the set cover problem $(U^*, \{N(u) \mid u \in V(G) \setminus U^*\})$ in advance. Line 4 fixes a subset U as vertices to be included in U^* by pre-assigning some vertices in $V(G) \setminus U^*$ EXCLUDE. To link U to such vertices in $V(G) \setminus U^*$, we use the solutions of the set cover problem $(U^*, \{N(u) \mid u \in V(G) \setminus U^*\})$, whose correctness is guaranteed by Lemma 2. The other structures are essentially equivalent to those of Algorithm 2.

The running time is estimated as follows. We apply Algorithm 3 for each U^*, which takes $O^*(2^{\tau})$ time. For each $U \subseteq U^*$ with $|\mathbf{SC}(U)| < |\mathrm{Sol}(U^*)|$, we obtain G', whose vertex cover has size at most $|U^*| - |U|$. By these, we can estimate the running time as

$$\sum_{U^* \in \mathcal{U}} \left(O^*(2^{\tau}) + \sum_{U \in U^*} O^*(\beta^{\tau - |U|}) \right)$$
$$= |\mathcal{U}| \left(O^*(2^{\tau}) + \sum_{k=0}^{\tau} \binom{\tau}{k} O^*(\beta^k) \right) = O^*(|\mathcal{U}|(\beta+1)^{\tau}),$$

because $\beta > 1$. By $|\mathcal{U}| \leq 2^{\tau}$, we can see that PAU-VC under EXCLUDE model can be solved in $O^*(4.5476^{\tau})$ time.

Here, we further try to improve the running time as INCLUDE model. Instead of an optimal vertex cover U^*, we focus on $U(\boldsymbol{v})$ in the branching argument of the previous subsection, where $G - U(\boldsymbol{v})$ forms a collection of k' edges. Since almost the same argument holds, lines 5−7 are executed at most $2^{|U|}$, where U is an INCLUDE pre-assignment (i.e., an implicit EXCLUDE pre-assignment) for an optimal vertex cover $U(\boldsymbol{v})$ plus a set of k' end vertices of the isolated edges in $G - U(\boldsymbol{v})$. Thus, what we need to consider newly for EXCLUDE model is how we efficiently get the set cover solution (i.e., vertices to be excluded) of every U for $U(\boldsymbol{v})$.

We explain how we resolve this. Suppose that $K(\boldsymbol{v})$ is the set of isolated edges in $G - U(\boldsymbol{v})$, and we then define the following set cover problem: The elements to be

covered are the vertices in $U(\boldsymbol{v})$, and a set to cover elements is $u \in V(G) \setminus (U(\boldsymbol{v}) \cup V(K(\boldsymbol{v})))$, which consists of $N(u)$. By applying Algorithm 3 for $(U(\boldsymbol{v}), \{N(u) \mid u \in V(G) \setminus (U(\boldsymbol{v}) \cup V(K(\boldsymbol{v})))\})$, we obtain for every $U' \subseteq U(\boldsymbol{v})$, $\mathbf{SC}'(U')$ as a minimum set to cover U', where we use $\mathbf{SC}'$ instead of $\mathbf{SC}$ to distinguish. Let U^* be an optimal vertex cover including $U(\boldsymbol{v})$, and let $K_1 := U^* \cap V(K(\boldsymbol{v}))$ and $K_2 := V(K(\boldsymbol{v})) \setminus U^*$. We claim that for $U \subseteq U^*$ with $N(K_2) \subseteq U$, $\mathbf{SC}'(U \setminus N(K_2)) \cup K_2$ is an optimal solution for U of the set cover problem $(U^*, \{N(u) \mid u \in V(G) \setminus U^*\})$. In fact, an optimal solution for U must contain K_2 to cover K_1, which can be covered only by K_2. The remaining vertices of the optimal solution are in $V(G) \setminus (U(\boldsymbol{v}) \cup V(K(\boldsymbol{v})))$ and optimally cover $U \setminus N(K_2)$; this is the requirement of $\mathbf{SC}'(U \setminus N(K_2))$. Thus, the solutions of the set cover problems can be obtained in the same running time. Thus, we can apply the same branching rule as INCLUDE model, which leads to Theorem 8.

Theorem 8. PAU-VC *under* EXCLUDE *or* MIXED *model can be solved in* $O^*(3.6791^\tau)$ *time.*

Since the vertex cover number of a bipartite graph is at most $n/2$, PAU-VC for bipartite graphs under any model can be solved in $O(1.9181^n)$ time.

Trees

In this section, we give an exponential upper bound $O(1.4143^n)$ of the time complexity of PAU-VC for trees.

We first give a generic recursive algorithm for PAU-VC of trees in Algorithm 5. Here, "generic" means that it describes the common actions for INCLUDE and EXCLUDE models. Because of this, lines 10 and 13 are described in ambiguous ways, which will be given later. The algorithm is so-called a divide and conquer algorithm. Lines from 2 to 5 handle the base cases. The following easy lemma might be useful.

Lemma 3. *A star with at least two leaves (i.e., $K_{1,l}$ with $l \geq 2$) has the unique optimal vertex cover. A star with one leaf (i.e., a single edge) has two optimal vertex covers.*

At line 9, choose a dividing point $v \in V(T)$ as a pre-assigned vertex, and after that, apply the algorithm to the divided components. Here, notice that the chosen vertex may not be included in any minimum feasible pre-assignment for T. This is the reason why we try all possible v, which guarantee that at least one v is correct, though it causes recursive calls for many subtrees of T. By using memoization, we can avoid multiple calls for one subtree; the number of subtrees appearing in the procedure dominates the running time. In the following subsections, we estimate an upper bound on the number of subtrees that may appear during the procedure in Algorithm 5 for each pre-assignment model.

In the INCLUDE Model, we slightly change lines 9 and 10 in Algorithm 5 as follows:

for all $v \in V(T)$ with $d(v) > 1$ **do**
  Let $T_1, \ldots, T_i$ be the connected components of $T - \{v\}$.

This change is safe by the following lemma.

Algorithm 5: Exact algorithm for trees

```
1: function PAU-TREE(T)        ▷ Return (τ(T), opt(T))
2:     Compute an optimal vertex cover of T
3:     if T has a unique optimal vertex cover then
4:         Return (τ(T), 0)
5:     else if T is a single edge then
6:         Return (1, 1)
7:     else
8:         k ← |V(T)|
9:         for all v ∈ V(T) do ▷ v is chosen for a pre-assignment
10:            Let T_1, ..., T_j be the connected components of the graph obtained by pre-assigning v.
11:            for i = 1, ..., j do
12:                (a_i, b_i) ← PAU-TREE(T_i)
13:            if Σ_i a_i meets τ(T) then
14:                k' ← 1 + Σ_i b_i ▷ "1" comes from |r| = 1
15:                if k' < k then
16:                    k ← k'
17:        Return (τ(T), k)
```

Lemma 4 ($\star$)**.** *Let T be a tree, which is not a star. Then, there exists a minimum feasible pre-assignment $\tilde{U}$ for T in the* INCLUDE *model, where $\tilde{U}$ does not contain a leaf vertex.*

The if-condition at line 13 is also changed to $\sum_i a_i = \tau(T) - 1$ so as v to be included in the vertex cover.

Now we give an upper bound on the number of subtrees of n vertices that may appear during the procedure in Algorithm 5. To make it easy to count subtrees, we fix a root; the number of subtrees is upper bounded by n times an upper bound of the number of rooted subtrees. Furthermore, we introduce the notion of isomorphism with a root to reduce multiple counting.

Definition 1. *Let $T^{(1)} = (V^{(1)}, E^{(1)}, r^{(1)})$ and $T^{(2)} = (V^{(2)}, E^{(2)}, r^{(2)})$ be trees rooted at $r^{(1)}$ and $r^{(2)}$, respectively. Then, $T^{(1)}$ and $T^{(2)}$ are called* isomorphic with respect to root *if for any pair of $u, v \in V^{(1)}$ there is a bijection $f : V^{(1)} \to V^{(2)}$ such that $\{u, v\} \in E^{(1)}$ if and only if $\{f(u), f(v)\} \in E^{(2)}$ and $f(r^{(1)}) = f(r^{(2)})$.*

For $T = (V, E)$ rooted at r, a connected subtree T' rooted at r is called a *rooted I-subtree* of T, if T' is T itself, or there exists a non-leaf v such that T' is the connected component with root r of $T - \{v\}$. Note that the graph consisting of only vertex r can be a rooted I-subtree.

Lemma 5 ($\star$)**.** *Any tree rooted at r has $O^*(2^{n/2})$ $(= O(1.4143^n))$ non-isomorphic rooted I-subtrees rooted at r, where n is the number of the vertices.*

Lemma 5 leads to the running time for INCLUDE model. By applying a different but similar argument, we achieve the same running time for EXCLUDE model

Theorem 9. PAU-VC *for trees under any model can be solved in time $O^*(2^{n/2}) = O(1.4143^n)$.*

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Numbers JP20H00595, JP21H05839, JP21K19765, JP22H00513, JP22H03549, and JP23H03344, and by MEXT KAKENHI Grant Numbers JP20H05964 and JP20H05967.

References

Asahiro, Y.; Iwama, K.; and Miyano, E. 1996. Random generation of test instances with controlled attributes. In Johnson, D. S.; and Trick, M. A., eds., *Cliques, Coloring, and Satisfiability, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, October 11-13, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 377–393. DIMACS/AMS.

Asuncion, A.; and Newman, D. 2007. UCI machine learning repository.

Blass, A.; and Gurevich, Y. 1982. On the unique satisfiability problem. *Information and Control*, 55(1): 80–88.

Bodlaender, H. L.; Kratsch, D.; and Timmer, S. T. 2015. Exact algorithms for Kayles. *Theor. Comput. Sci.*, 562: 165–176.

Cha, B.; and Iwama, K. 1995. Performance Test of Local Search Algorithms Using New Types of Random CNF Formulas. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, 304–311. Morgan Kaufmann.

Chen, J.; Kanj, I. A.; and Xia, G. 2010. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42): 3736–3756.

Corneil, D. G.; and Perl, Y. 1984. Clustering and domination in perfect graphs. *Discret. Appl. Math.*, 9(1): 27–39.

Demaine, E. D.; Demaine, M. L.; Fox-Epstein, E.; Hoang, D. A.; Ito, T.; Ono, H.; Otachi, Y.; Uehara, R.; and Yamada, T. 2015. Linear-time algorithm for sliding tokens on trees. *Theor. Comput. Sci.*, 600: 132–142.

Demaine, E. D.; Ma, F.; Schvartzman, A.; Waingarten, E.; and Aaronson, S. 2018. The fewest clues problem. *Theor. Comput. Sci.*, 748: 28–39.

Diestel, R. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer. ISBN 978-3-642-14278-9.

DIMACS. 2022. DIMACS Implementation Challenges. http://dimacs.rutgers.edu/programs/challenge/. Accessed: 2023-08-15.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer. ISBN 978-1-4471-5558-4.

Fomin, F. V.; and Kratsch, D. 2010. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer. ISBN 978-3-642-16532-0.

Goergen, K.; Fernau, H.; Oest, E.; and Wolf, P. 2022. All Paths Lead to Rome. *CoRR*, abs/2207.09439.

Higuchi, Y.; and Kimura, K. 2019. NP-Completeness of Fill-a-Pix and Σ_2^P-Completeness of Its Fewest Clues Problem. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 102-A(11): 1490–1496.

Hoos, H. H.; and Stützle, T. 2000. SATLIB: An online resource for research on SAT. *Sat*, 2000: 283–292.

Hopcroft, J. E.; and Karp, R. M. 1973. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 2(4): 225–231.

Horie, S.; and Watanabe, O. 1997. Hard Instance Generation for SAT (Extended Abstract). In Leong, H. W.; Imai, H.; and Jain, S., eds., *Algorithms and Computation, 8th International Symposium, ISAAC '97, Singapore, December 17-19, 1997, Proceedings*, volume 1350 of *Lecture Notes in Computer Science*, 22–31. Springer.

Horiyama, T.; Kobayashi, Y.; Ono, H.; Seto, K.; and Suzuki, R. 2023. Theoretical Aspects of Generating Instances with Unique Solutions: Pre-assignment Models for Unique Vertex Cover. *CoRR*, abs/2312.10599.

Hudry, O.; and Lobstein, A. 2019a. Complexity of unique (optimal) solutions in graphs: Vertex Cover and Domination. *J. Comb. Math. Comb. Comput.*, 110: 217–240.

Hudry, O.; and Lobstein, A. 2019b. Unique (optimal) solutions: Complexity results for identifying and locating-dominating codes. *Theor. Comput. Sci.*, 767: 83–102.

Juban, L. 1999. Dichotomy Theorem for the Generalized Unique Satisfiability Problem. In Ciobanu, G.; and Paun, G., eds., *Fundamentals of Computation Theory, 12th International Symposium, FCT '99, Iasi, Romania, August 30 - September 3, 1999, Proceedings*, volume 1684 of *Lecture Notes in Computer Science*, 327–337. Springer.

Krivelevich, M.; and Vilenchik, D. 2006. Solving random satisfiable 3CNF formulas in expected polynomial time. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, 454–463. ACM Press.

Matsuyama, Y.; and Miyazaki, S. 2021. Hardness of Instance Generation with Optimal Solutions for the Stable Marriage Problem. *J. Inf. Process.*, 29: 166–173.

McCreesh, C.; Prosser, P.; Solnon, C.; and Trimble, J. 2018. When Subgraph Isomorphism is Really Hard, and Why This Matters for Graph Databases. *J. Artif. Intell. Res.*, 61: 723–759.

Mulmuley, K.; Vazirani, U. V.; and Vazirani, V. V. 1987. Matching is as easy as matrix inversion. *Comb.*, 7(1): 105–113.

Neuen, D.; and Schweitzer, P. 2017. Benchmark Graphs for Practical Graph Isomorphism. In Pruhs, K.; and Sohler, C., eds., *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Papadimitriou, C. H. 1984. On the complexity of unique solutions. *J. ACM*, 31(2): 392–400.

Reinelt, G. 1991. TSPLIB - A Traveling Salesman Problem Library. *INFORMS J. Comput.*, 3(4): 376–384.

Sanchis, L. A. 1990. On the Complexity of Test Case Generation for NP-Hard Problems. *Inf. Process. Lett.*, 36(3): 135–140.

Sanchis, L. A. 1995. Generating Hard and Diverse Test Sets for NP-hard Graph Problems. *Discret. Appl. Math.*, 58(1): 35–66.

Scheder, D. 2022. PPSZ is better than you think. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, 205–216. IEEE.

Ullrich, M.; Weise, T.; Awasthi, A.; and Lässig, J. 2018. A generic problem instance generator for discrete optimization problems. In Aguirre, H. E.; and Takadama, K., eds., *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, 1761–1768. ACM.

Valiant, L. G.; and Vazirani, V. V. 1986. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47(3): 85–93.

Xiao, M.; and Nagamochi, H. 2017. Exact algorithms for maximum independent set. *Inf. Comput.*, 255: 126–146.

Yoshiwatari, K.; Kiya, H.; Hanaka, T.; and Ono, H. 2022. Winner Determination Algorithms for Graph Games with Matching Structures. In *Combinatorial Algorithms: 33rd International Workshop, IWOCA 2022, Trier, Germany, June 7–9, 2022, Proceedings*, 509–522. Springer.