0/1/all CSPs, Half-Integral A-path Packing, and Linear-Time FPT Algorithms

Yoichi Iwata  
National Institute of Informatics  
yiwata@nii.ac.jp

Yutaro Yamaguchi  
Osaka University  
yutaro_yamaguchi@ist.osaka-u.ac.jp

Yuichi Yoshida  
National Institute of Informatics  
yyoshida@nii.ac.jp

Abstract—A recent trend in the design of FPT algorithms is exploiting the half-integrality of LP relaxations. In other words, starting with a half-integral optimal solution to an LP relaxation, we assign integral values to variables one-by-one by branch and bound. This technique is general and the resulting time complexity has a low dependency on the parameter. However, the time complexity often becomes a large polynomial in the input size because we need to compute half-integral optimal LP solutions.

In this paper, we address this issue by providing an $O(km)$-time algorithm for solving the LPs arising from various FPT problems, where $k$ is the optimal value and $m$ is the number of edges/constraints. Our algorithm is based on interesting connections among 0/1/all constraints, which has been studied in the field of constraints satisfaction, A-path packing, which has been studied in the field of combinatorial optimization, and the LPs used in FPT algorithms. With the aid of this algorithm, we obtain linear-time FPT algorithms for various problems. The obtained running time for each problem is linear in the input size and has the current smallest dependency on the parameter. Most importantly, instead of using problem-specific approaches, we obtain all of these results by a unified approach, i.e., the branch-and-bound framework combined with the efficient computation of half-integral LPs, which demonstrates its generality.

Index Terms—FPT algorithms; Combinatorial Optimization; Half-integrality;

I. INTRODUCTION

A. FPT Algorithms using Half-Integral LP Relaxations

Parameterized complexity is the subject of studying the complexity of parameterized problems. A parameterized problem with a parameter $k$ is fixed parameter tractable (FPT) if we can solve the problem in $f(k)\text{poly}(n)$ time, where $n$ is the input size. Various parameterized problems are known to be FPT. See [8], [10] and references therein for a comprehensive list of FPT problems.

One of the motivations of studying parameterized complexity is to understand tractable subclasses of (NP-)hard problems; hence, the primary interest has been which parameterized problems admit FPT algorithms. However, from a practical point of view, both the factors in the running time with respect to the parameter ($f(k)$) and with respect to the input size ($\text{poly}(n)$) must be small. Indeed, linear-time FPT algorithms, which is the extreme goal of the improvement of the latter factor, have been proposed for several problems including TREEWIDTH [4], ALMOST 2-SAT [18], [29], FEEDBACK VERTEX SET (FVS) [3], SUBSET FVS [22], DIRECTED FVS [23], and NODE UNIQUE LABEL COVER [21]. On the other hand, because these works focused on reducing the running time with respect to the input size, dependency on the parameter is often suboptimal. For example, SUBSET FVS admits an FPT algorithm running in $O^*(4^k)$ time [19] whereas the current best linear-time FPT algorithm has time complexity $O(25.6^km)$ [22], where $m$ is the number of edges in the input graph.

One of the powerful techniques for obtaining FPT algorithms with small $f(k)$ factors is the branch-and-bound method using half-integral LP lower bound, or LP-branching. To see the idea, let us consider a minimization problem whose goal is to find a solution of size $k$, and suppose that it admits a half-integral LP relaxation\(^2\), that is, an LP relaxation with an optimal LP solution that only uses values in $\{0, \frac{1}{2}, 1\}$. First, we compute a half-integral (optimal) LP solution. We can stop if all the variables have integral values or the value of the LP solution exceeds $k$. Otherwise, we pick an arbitrary variable with value $\frac{1}{2}$, and branch into the case that its value is fixed to 0 and the case that its value is fixed to 1. By using LP solutions with an extremal condition, which we call the farthest condition herein (see Section II-B for details), we can ensure that the LP lower bound strictly increases for each branching, and thus the depth of the search tree is bounded. This approach has several big advantages. It can be applicable to a broad range of problems just by changing the LP [9], [14], [17], [19], [20], [39], [31], and has a small time complexity with respect to the parameter $k$. For many problems including ALMOST 2-SAT [20], NODE MULTIWAY CUT [9], and GROUP FVS [19], the current smallest dependency on the parameter is indeed achieved by this approach.

A drawback of the abovementioned approach is that it is not trivial how to efficiently compute half-integral LP solutions. Iwata, Wahlström, and Yoshida [19] tackled this issue by viewing half-integrality as discrete relaxation. They showed that one can compute half-integral LP solutions for ALMOST 2-SAT and EDGE UNIQUE LABEL COVER in time linear in

\(^1O^*(\cdot)\) hides a polynomial dependency on the input size. When focusing on reducing the $f(k)$ part, the $\text{poly}(n)$ part is often ignored using this notation.

\(^2\)Most of the LPs used in the FPT algorithms are not natural LP relaxations of the original problems, but are LP relaxations of rooted problems. For example, the rooted version of FVS is a problem of finding a minimum vertex subset $S$, such that the graph obtained by removing $S$ contains no cycles reachable from a prescribed vertex $s$. Note that the existence of a half-integral LP relaxation to the rooted problem does not imply a 2-approximability of the original problem.
The minimum $s$-$t$ cut approach unfortunately does not work well for other problems such as Group FVS, Subset FVS, Node Multiway Cut, and Node Unique Label Cover because it is essentially applicable only to edge-deletion problems\(^3\) and because the auxiliary network size is generally not linear in the input size (e.g., for Subset Feedback Edge Set, the size of the network becomes $2^{O(m)}$). Thus, we need to resort to solving linear programs for these problems, and hence the resulting FPT algorithms have large dependencies on the input size. We note that, among these problems, linear-time FPT algorithms have been obtained for Node Multiway Cut [6], Subset FVS [22], and Node Unique Label Cover [21]. Instead of using LP relaxations, these algorithms use problem-specific arguments that are not directly applicable to other problems, and moreover they have larger dependencies on $k$ (Table I).

One natural approach for obtaining linear-time FPT algorithms for these problems is to develop linear-time LP solvers; however, this direction was not well investigated possibly because it looks difficult. For example, the half-integral LP of Node Multiway Cut, which was developed to obtain a 2-approximation algorithm [13] and was used in the $O^*(2^k)$-time FPT algorithm [9], has been studied under the name of half-integral internally disjoint $A$-path packing in the field of combinatorial optimization; however, the current best algorithm requires $O(knm)$ time [2]. One exceptional result is a linear-time LP solver for FVS recently obtained by Iwata [17]. This algorithm exploits a special structure of the dual LP solution and iteratively augments the dual LP solution by searching and applying augmenting paths. Note that this algorithm does not find a farthest solution. For LPs that can be reduced to the minimum $s$-$t$ cut problem, we can compute a farthest solution by computing strongly connected components and using a structural property of all minimum $s$-$t$ cuts [28]. However, no such structural properties of all minimum solutions are known yet for other half-integral LPs.

The main contribution of this study is the development of an algorithm that computes farthest half-integral solutions to LPs for various problems in time linear in the input size. We find that, by carefully designing the definition of augmenting paths, the augmenting-path approach can be applied for a wide range of half-integral LPs, even for exponential-size LPs. Moreover, by exploiting the complementary slackness theorem, we can compute a farthest solution in linear time. Using the connection of computing farthest half-integral LP solutions and FPT algorithms, we obtain linear-time FPT algorithms for various problems, which are summarized in Table I. For Subset FVS, Node Multiway Cut, and Node Unique Label Cover, we substantially improve the dependency on the parameter. For the other problems, we obtain the first linear-time FPT algorithms. We note that, for every problem in the table, the $f(k)$ part in the running time of our algorithm matches or improves the smallest known. Most importantly, instead of using problem-specific approaches, we obtain all of these results by a unified approach, i.e., the branch-and-bound framework combined with the efficient computation of half-integral LPs, which demonstrates its generality.

### Table I: Summary of our FPT results.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Smallest $f(k)$</th>
<th>Existing linear-time FPT</th>
<th>Our result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/1/ALL DELETION</td>
<td>$O^*(d^{2k})$</td>
<td>$-$</td>
<td>$O(d^{2k})$</td>
</tr>
<tr>
<td>NODE UNIQUE LABEL COVER</td>
<td>$O^*(</td>
<td>I</td>
<td>^{2k})$</td>
</tr>
<tr>
<td>TWO-FAN DELETION</td>
<td>$O^*(2^k)$</td>
<td>$-$</td>
<td>$O(2^k)$</td>
</tr>
<tr>
<td>MONOCHROMATICALLY ORIENTABLE DELETION</td>
<td>$-$</td>
<td>$-$</td>
<td>$O(4^k)$</td>
</tr>
<tr>
<td>SUBSET PSEUDOFOREST DELETION</td>
<td>$-$</td>
<td>$-$</td>
<td>$O(4^k)$</td>
</tr>
<tr>
<td>NODE MULTIWAY CUT</td>
<td>$O^*(2^k)$</td>
<td>$O(4^k)$</td>
<td>$O(2^k)$</td>
</tr>
<tr>
<td>GROUP FVS</td>
<td>$O^*(4^k T_{T'})$</td>
<td>$-$</td>
<td>$O(4^k T_{T'})$</td>
</tr>
<tr>
<td>SUBSET FVS</td>
<td>$O^*(4^k)$</td>
<td>$O(25.6^m)$ (randomized)</td>
<td>$O(4^k)$</td>
</tr>
<tr>
<td>NON-MONOCROMATIC CYCLE TRANSVERSAL</td>
<td>$O^*(4^k)$</td>
<td>$-$</td>
<td>$O(4^k)$</td>
</tr>
</tbody>
</table>

---

3A precise running time is $O^*(4^k - \mu)$, where $\mu$ is the LP lower bound. Because $k \leq 2\mu$ holds, we have $4^k - \mu \leq 4^k$. When parameterized by the gap $k - \mu$, our algorithm also runs in $O(4^k - \mu)$ time, which improves the previous best. However, this is not a linear-time FPT algorithm because $k$ can be $\Omega(n)$ even when $k - \mu$ is a constant.

4Note that edge-deletion problems are easily reducible to the corresponding vertex-deletion problems in most cases by subdividing the edges and creating $k$ copies of the original vertices.
first parameterized algorithms and computational experiments challenge\textsuperscript{3} by implementing an LP-branching algorithm for FVS [19] combined with the linear-time LP solver [17]. These results suggest that our algorithm would also be practical.

**B. 0/1/ALL DELETION and A-path Packing**

We consider the following problem, called 0/1/ALL DELETION, to establish a unified framework and provide linear-time FPT algorithms for the abovementioned problems using half-integral LP relaxations. Let \( V \) be a set of variables. Each variable \( v \in V \) has an individual domain \( D(v) \). A function \( \varphi \) on \( V \) that maps each variable \( v \in V \) to a value \( \varphi(v) \in D(v) \) is called an assignment for \( V \). We consider the following two types of binary constraints on \((u,v) \in V \times V\), called 0/1/ALL constraints [7]:

1. Permutation constraint \( \pi(\varphi(u)) = \varphi(v) \), where \( \pi : D(u) \rightarrow D(v) \) is a bijection.
2. Two-fan constraint \( (\varphi(u) = a) \vee (\varphi(v) = b) \), where \( a \in D(u) \) and \( b \in D(v) \).

Let \( C \) be a set of 0/1/ALL constraints on \( V \). We assume that \( C \) contains at most one constraint for each pair of distinct variables \( u,v \in V \). A constraint on \((u,v) \in V\) in \( C \) is denoted by \( C_{uv} \). For a subset \( U \subseteq V \), we denote the set \( \{ C_{uv} \in C \mid u,v \in U \} \) by \( C[U] \).

![0/1/ALL DELETION Parameter: k, d](image)

**Input:** A set of variables \( V \) each of which has a domain of size at most \( d \), a set \( C \) of 0/1/ALL constraints each of which is given as a table of size \( O(d) \), a partial assignment \( \varphi_A \) for a subset \( A \subseteq V \), and an integer \( k \).

**Question:** Is there a pair of a set \( X \subseteq V \) of at most \( k \) variables and a partial assignment \( \varphi \) for \( V \setminus X \) such that (1) \( \varphi(v) = \varphi_A(v) \) holds for every \( v \in A \setminus X \) and (2) \( \varphi \) satisfies all of \( C[V \setminus X]? \)

The set \( X \) in the question is called a deletion set. The task of the optimization version of this problem is to compute the size of a minimum deletion set. Various FPT problems can be expressed as 0/1/ALL DELETION (by using a large \( d \)). Note that for several problems, we need exponential-size domains, and hence a linear-time FPT algorithm for 0/1/ALL DELETION does not directly imply linear-time FPT algorithms for such problems. We show that, by giving constraints not as a table but as an oracle and by using a specialized branching strategy, we can obtain linear-time FPT algorithms even for such problems in a unified way. Our goal is to prove the following main theorem from which we obtain all the results in Table 1. See Theorem 5 for a formal statement of the theorem.

**Theorem 1** (Informal version of Theorem 5). Let \( I = (C, \varphi_A) \) be a pair of a set \( C \) of 0/1/ALL constraints on a variable set \( V \) and a partial assignment \( \varphi_A \) for a subset \( A \subseteq V \). Given \( I \) as a certain oracle, we can correctly answer whether \( I \) admits a deletion set of size at most \( k \) or not in \( O(c^k knT) \) time, where \( m \) is the number of constraints, \( c \) is an integer depending on a branching strategy we use, and \( T \) is the running time of the oracle.

We now introduce a relation between 0/1/ALL DELETION and A-path packing. The primal graph for \( C \) is a simple undirected graph \( G = (V,E) \) such that an edge \( uv \in E \) exists if and only if a constraint \( C_{uv} \) on \((u,v) \) exists. An important property of the 0/1/ALL constraints is that, when fixing the value of a variable \( u \in V \) to \( p \in D(u) \), the set of values of \( v \in V \) satisfying the constraint \( C_{uv} \) is either \( D(v) \) (when the constraint is a two-fan with \( a = p \)) or a singleton \( \{ q \} \) (when the constraint is a permutation with \( q = \pi(p) \) or a two-fan with \( a \neq p \) and \( b = q \)). The latter-type implication is called a unit propagation.

When we are given a partial assignment \( \varphi_A \) for a subset \( A \subseteq V \), unit propagations occur along walks in the primal graph \( G \) starting at the vertices in \( A \). If the unit propagations along two different walks starting at (possibly the same) vertices in \( A \) lead to a contradiction (i.e., implicate distinct singletons for the same variable), then at least one variable on the two walks must be contained in the deletion set. The concatenation of such two walks is a walk between the vertices in \( A \) (called an A-walk) that is said to be conflicting (see Section II-B for a formal definition).

This observation provides a lower bound on the minimum size of a deletion set as follows. Suppose that a deletion set \( X \subseteq V \) with \( |X| = k \) exists, and let \( F = F_{C,\varphi_A} \) be the set of all conflicting A-walks in the primal graph \( G \) for \( C \) with respect to \( \varphi_A \). Then, the remaining graph \( G - X \) cannot have a walk in \( F \) (i.e., \( X \) is a cover (or a hitting set) of \( F \)). Hence, the minimum size of such a cover is at most \( k \).

We now consider an LP relaxation of finding a minimum cover of the set \( F \) of all conflicting A-walks in \( G \), called the F-covering problem: we are required to find a function \( x : V \rightarrow \mathbb{R}_{\geq 0} \) minimizing the total value \( |x| := \sum_{v \in V} x(v) \) under the constraint that \( \sum_{v \in W} x(v) \geq 1 \) for every \( W \in F \), where "\( \sum_{v \in W} \)" means the summation over the occurrences of vertices \( v \) in \( W \) considering the multiplicity (e.g., \( x(v) \) is summed twice if \( W \) intersects \( v \) twice). The dual of this LP is written down as follows, as the F-packing problem: we are required to find a function \( y : F \rightarrow \mathbb{R}_{\geq 0} \) maximizing the total value \( |y| := \sum_{W \in F} y(W) \) subject to \( \sum_{W \in F, v \in W} y(W) \leq 1 \) for every \( v \in V \), where we also consider the multiplicity of the occurrences of vertices in a walk in the summation "\( \sum_{W \in F, v \in W} \)" (e.g., \( y(W) \) is summed twice if \( W \) intersects \( v \) twice).

We observe that the half-integral LPs used in the existing FPT algorithms can be seen as the F-covering problem. Instead of directly solving the F-covering problem, we propose an \( O(mT) \)-time algorithm to augment a given F-packing with a certain special structure, where \( m = |C| \) and \( T \) is the running time of a certain oracle for simulating unit propagations (see Section II-C for the detail). Using this augmenting-path

\textsuperscript{3}https://pacechallenge.wordpress.com/pace-2016/

\textsuperscript{4}Precisely speaking, the permutation and two-fan constraints together with empty and complete constraints are obtained by enforcing arc-consistency on the 0/1/ALL constraints introduced in [7].
algorithm, we obtain an $O(kmT)$-time algorithm for finding a pair of a half-integral $F$-cover $x$ and a half-integral $F$-packing $y$ with $|x| = |y| \leq \frac{1}{2}$ (if exists, and otherwise correctly concluding it). Note that, by LP duality, these $x$ and $y$ are both optimal solutions to the $F$-covering and $F$-packing problems, respectively.

**C. Related Work on Half-Integral A-path Packing**

When $C$ contains only the permutation constraints\(^7\), the integral version of the $F$-packing problem has been studied under the name of non-returning $A$-path packing \([24, 26, 32]\). For a further special case, called internally disjoint $A$-path packing, the integral version of the dual covering problem coincides with NODE MULTIWAY CUT, and its LP relaxation is used in a branch-and-bound FPT algorithm \([9]\) and a 2-approximation algorithm \([13]\) for this problem.

The half-integral version of internally disjoint $A$-path packing (and NODE MULTIWAY CUT in the dual sense) has been studied in the field of combinatorial optimization. Garg et al. \([13]\) and Pap \([25, 27]\) found that an LP relaxation of internally disjoint $A$-path packing and its dual always enjoy half-integral optimal solutions even if each non-terminal vertex has an individual integral capacity other than 1. Hirai \([15]\) and Pap \([25, 27]\) developed algorithms for finding such half-integral optimal solutions, which both run in strongly polynomial time (i.e., the number of elementary operations performed through each algorithm does not depend on the capacity values). One makes use of a sophisticated algorithm for the submodular flow problem \([12]\) whereas the other relies on the ellipsoid method to solve LPs whose coefficient matrices only have $0, \pm 1$ entries \([11]\). Specializing on the uncapped case, Babenko \([2]\) provided an $O(knm)$-time algorithm for finding a maximum half-integral packing, where $n$ and $m$ are the numbers of vertices and edges, respectively, and $k$ denotes the optimal value, which is at most $O(n)$.

Our algorithm for the $F$-packing/covering problems improves the previous best running time even against this (internally disjoint) special case.

Another special case of the $F$-packing problem is the LP for FVS, for which Iwata \([17]\) obtained an $O(knm)$-time augmenting-path algorithm. We can see that this LP is the easiest case for augmenting-path algorithms because every non-trivial $A$-walk is in the set $F\$. For solving general $F$-packing problems, we need to ensure that the augmentation increases only a walk in the set $F$ which consists of a restricted class of $A$-walks. We find that this is always possible for the set of conflicting $A$-walks of 0/1/all constraints.

**D. Organization**

We introduce the notions used throughout the paper in Section II. Section III describes a fast algorithm that computes a maximum half-integral $F$-packing and transforms it into a minimum half-integral $F$-cover. Section IV presents a fast algorithm for computing a farthest minimum half-integral $F$-cover. We prove Theorem I in Section V using this algorithm. Finally, in Section VI, we obtain the results in Table I by applying Theorem I. Due to space limitations, we only provide proof sketches. For detailed proofs, see the full version available at arXiv:1704.02700.

**II. DEFINITIONS**

**A. Basic Notations**

The multiplicity function $1_S : U \rightarrow \mathbb{Z}_{\geq 0}$ for a multiset $S$ on the ground set $U$ is defined such that $1_S(a)$ is the number of times that $a \in U$ appears in $S$. For two multisets $A$ and $B$ on the same ground set $U$, we denote by $A \setminus B$ the multiset such that $1_{A \setminus B}(a) = \max\{1_A(a) - 1_B(a), 0\}$ holds for any element $a \in U$. For a function $f : U \rightarrow \mathbb{R}$ and a multiset $S$ on the ground set $U$, we define $f(S) := \sum_{a \in U} 1_S(a)f(a)$. For a value $i \in \mathbb{R}$, we define $f^{-1}(i) := \{a \in U \mid f(a) = i\}$.

All the graphs in this study are undirected. However, we sometimes need to take care of the direction of edges. For an undirected graph $G = (V, E)$, we use the symbol $\bar{E}$ when we take care of the direction of the edges, i.e., $uv = vu$ for $uv \in E$ but $uv \neq vu$ for $uv \in \bar{E}$. For simplicity, we assume that the graphs are simple; if a graph contains multiple edges or self-loops, we can easily obtain an equivalent simple graph by subdividing the edges. For vertex $v \in V$, we denote the set of incident edges by $\delta(v)$. For a subset $U \subseteq V$, we denote the induced subgraph by $G[U] = (U, E[U])$.

For an undirected graph $G = (V, \bar{E})$, we define a walk in $G$ as an ordered list $W = (v_0, v_1, ..., v_t)$ of vertices such that $v_{i-1}v_i \in E$ for all $i = 1, ..., t$. The integer $t$ is called the length of the walk. We denote the first and last vertices of $W$ by $s(W) = v_0$ and by $t(W) = v_t$, respectively, and we say that $W$ starts from $s(W)$ and ends at $t(W)$.

We denote the multisets of vertices, inner vertices, and (undi-
rected) edges appeared in $W$ by $V(W) = \{v_0, ..., v_t\}$, by $V_i(W) = \{v_1, ..., v_{i-1}\}$, and by $E(W) = \{v_0v_1, ..., v_{t-1}v_t\}$, respectively. For an edge $e = uv \in \bar{E}$, we simply use the same symbol $e$ to denote the walk $(u, v)$. A walk $W$ is a (simple) path if $1_{V(W)}(v) \leq 1$ for every $v \in V$. A walk $W$ is called a closed walk if $s(W) = t(W)$. A closed walk $W$ is called a (simple) cycle if $t \geq 3$ and $1_{V(W) \setminus \{s(W)\}}(v) \leq 1$ for every $v \in V$ (which implies $1_{V(W)}(s(W)) = 2$). We may regard a walk $W$ as a subpath by ignoring the direction and the multiplicity. We say that a walk $W$ is internally disjoint from a subgraph $G' = (V', E')$ if none of the inner vertices of $W$ are in $V'$ and none of the edges of $W$ are in $E'$.

For a walk $W = (v_0, ..., v_t)$, we define the reversed walk as
B. 0/1/ALL DELETION and Half-Integral Relaxation

We first recall 0/1/ALL DELETION defined in Section I-B. We are given a set \( V \) of variables \( v \) with individual domains \( D(v) \), a set \( C \) of 0/1/all constraints (permutation and two-fan constraints) that can be represented by a simple undirected graph \( G \), called the primal graph, and a partial assignment \( \varphi_A \) for a subset \( A \subseteq V \). The task is to determine whether a deletion set \( X \subseteq V \) of at most \( k \) variables exists, for which there exists an assignment \( \varphi \) for \( V \setminus X \) such that (1) \( \varphi(v) = \varphi_A(v) \) holds for every \( v \in A \setminus X \), and (2) \( \varphi \) satisfies every constraint \( C_{uv} \in C[V \setminus X] \).

As described in Section I-B, an important property of the 0/1/all constraints is that, when fixing the value of a variable \( u \in V \) to \( p \in D(u) \), the set of values of \( v \in V \) satisfying the constraint \( C_{uv} \) is either \( D(v) \) or a singleton \( \{ q \} \). We define \( C_{uv}(p) := \text{all} \) in the former case and define \( C_{uv}(p) := \{ q \} \) in the latter case. We extend this definition to walks in the primal graph as follows. For a walk \( s \) of length zero and an element \( p \in D(s) \), we define \( C_{s}(p) := p \). For a walk \( W = W' \circ e \) starting from \( s \in V \) and an element \( p \in D(s) \), we define \( C_{W}(p) := \text{all} \) when \( C_{W'}(p) = \text{all} \) and \( C_{W}(p) := C_{e}(C_{W'}(p)) \) when \( C_{W'}(p) \neq \text{all} \). Suppose that an assignment \( \varphi \) satisfying all the constraints exists. Then, either \( C_{W}(\varphi(s(W))) = \text{all} \) or \( \varphi(t(W)) = C_{W}(\varphi(s(W))) \) holds for any walk \( W = W' \circ e \) in the primal graph.

Let \( \varphi_A \) be a partial assignment for a subset \( A \subseteq V \). For a walk \( W \) with \( s(W) \in A \), we define \( \text{imp}_{\varphi_A}(W) := C_{W}(\varphi_A(s(W))) \), which represents the set of assignments for \( t(W) \) induced by \( W \). A walk \( W \) is called a \( \varphi_A \)-implicational walk if \( s(W) \in A \) and \( \text{imp}_{\varphi_A}(W) \neq \text{all} \). A \( \varphi_A \)-implicational walk \( W \) is called a \( \varphi_A \)-conflicting walk if \( t(W) \in A \) and \( \text{imp}_{\varphi_A}(W) \neq \varphi_A(t(W)) \) hold. We omit the prefix/subscript \( \varphi_A \) if it is clear from the context. We use the following lemma, whose proof is given in the full version.

**Lemma 1.** For any two \( \varphi_A \)-implicational walks \( P \) and \( Q \) ending at the same vertex, \( P \circ Q^{-1} \) is \( \varphi_A \)-conflicting if and only if \( \text{imp}_{\varphi_A}(P) \neq \text{imp}_{\varphi_A}(Q) \).

Corollary 1. If a walk \( W \) is conflicting, then \( W^{-1} \) is also conflicting.

Corollary 2. For three implicational walks \( P, Q, \text{ and } R \) ending at the same vertex, if \( P \circ Q^{-1} \) is conflicting, then at least one of \( P \circ R^{-1} \text{ and } R \circ Q^{-1} \) is conflicting.

We now introduce a half-integral relaxation\(^{10}\) of 0/1/ALL DELETION. Let \( F_{C,\varphi_A} \) denote the set of all \( \varphi_A \)-conflicting walks whose internal vertices do not intersect with \( A \). When \( C \) and \( \varphi_A \) are clear from the context, we simply write \( F \) to refer to \( F_{C,\varphi_A} \). Note that from Corollary 1, we can ignore the direction of walks in \( F \). A function \( x : V \to \{ 0, \frac{1}{2}, 1 \} \) is called a half-integral \( F \)-cover if \( x(V(W)) \geq 1 \) for every \( W \in F \). The size of \( x \) is defined as \( |x| = x(V) \). A function \( y : F \to \{ 0, \frac{1}{2}, 1 \} \) is called a half-integral \( F \)-packing if for every vertex \( v \in V \), it holds that \( \sum_{W \in F} 1_{V(W)}(v)y(W) \leq 1 \). The size of \( y \) is defined as \( |y| = y(F) \). From the LP-duality, we have \( |x| \geq |y| \) for any pair of half-integral \( F \)-cover \( x \) and half-integral \( F \)-packing \( y \). Any deletion set \( X \) for 0/1/ALL DELETION must intersect every \( \varphi_A \)-conflicting walk; hence \( 1_X \) is an integral \( F \)-cover\(^{11}\). Therefore, the size of the minimum half-integral \( F \)-cover \( x \) is conflicting and \( \ell \) provides a lower bound on the size of the minimum deletion set. For a half-integral \( F \)-cover \( x \), let \( R(x) \) denote the set of vertices \( t \) such that an implicational walk \( W \) with \( x(V(W)) = 0 \) ending at \( t \) exists.

We can prove the following property called persistence\(^{12}\) by a careful consideration of the results in [19] (see the full version for a detailed discussion).

**Theorem 2.** Let \( C \) be a set of 0/1/all constraints on a variable set \( V \) and \( \varphi_A \) be a partial assignment for a subset \( A \subseteq V \). For any minimum half-integral \( F_{C,\varphi_A} \)-cover \( x \), there exists a minimum deletion set \( X \) containing every vertex \( u \) with \( x(u) = 1 \) but avoiding every vertex in \( R(x) \).

We say that a minimum half-integral \( F \)-cover \( x' \) dominates a minimum half-integral \( F \)-cover \( x \) if \( R(x) \subset R(x') \) holds. A minimum half-integral \( F \)-cover \( x \) is called farthest if there exists no \( F \)-cover dominating \( x \). Suppose that we have an \( O(T) \)-time algorithm for computing a farthest minimum half-integral \( F \)-cover. From Theorem 2, it is not difficult to obtain an \( O(d^kT) \)-time FPT algorithm for 0/1/ALL DELETION. Moreover, the base \( d \) of the exponent can be improved to a constant for several special cases as discussed in Section V.

C. Single-Branching Pair and Incremental-Test Oracle

As shown in Section VI, various important NP-hard problems can be expressed as a special case of 0/1/ALL DELETION. However, the domain size \( d \) is often \( \omega(1) \) (or

---

\(^{9}\)When at least one of \( P \) or \( Q \) is not implicational, neither \( P \equiv Q \) nor \( P \not\equiv Q \) holds.

\(^{10}\)Originally we should define this as an LP relaxation, but we discuss it by assuming its half-integrality, which is proved via our algorithm and LP duality.

\(^{11}\)This constraint is only for simplicity. From Corollary 2, if \( P \circ Q^{-1} \) is conflicting for some walks \( P \) and \( Q \) ending in \( A \), at least one of \( P \) and \( Q \) is conflicting. Therefore, a maximum packing of conflicting walks that uses none of such walks always exists.

\(^{12}\)Note that the converse may not hold. For example, we have \( F = \emptyset \) when \( A = \emptyset \). Therefore, \( X = \emptyset \) is the minimum integral \( F \)-cover. In contrast, \( X = \emptyset \) may not be a deletion set. Thus, the half-integral relaxation does not always lead to a 2-approximation algorithm.
even \( \exp(m) \) for several cases, where \( m \) is the number of edges/constraints). Hence, if every constraint is given as the table of size \( d \), the total size of these tables already becomes super-linear. For obtaining linear-time FPT algorithms, we use oracles instead of the explicit expression of the constraints to efficiently check whether a given walk is implicational/conflicting or not.

A pair \((P, Q)\) of implicational walks ending at the same vertex is called single-branching if either \( P \circ Q^{-1} \) forms a simple path or they can be written as \( P = R \circ P' \) and \( Q = R \circ Q' \) for a (possibly zero-length) path \( R \) and two walks \( P' \) and \( Q' \) for which \( P' \circ Q'^{-1} \) forms a simple cycle that is internally disjoint from \( R \). In our algorithm, we need to test whether a given walk is implicational and whether \( P \circ Q^{-1} \) is conflicting for a given single-branching pair \((P, Q)\). To efficiently answer these queries, we use a tuple \((U, I, A, T)\), called an incremental-test oracle, of a set \( U \) and functions \( I, A, \) and \( T \) satisfying the following.

- **Init** \( I : A \to U \).
- **Append** \( A : U \times E \to U \cup \{\text{all}\} \). Let \( A^* \) be a function such that

\[
\begin{align*}
A^*((s)) &= \begin{cases} I(s) & (s \in A) \\
\text{all} & (s \notin A) \end{cases} \\
A^*(W \circ e) &= \begin{cases} A(A^*(W), e) & (A^*(W) \neq \text{all}) \\
\text{all} & (A^*(W) = \text{all}) \end{cases}
\end{align*}
\]

Then, for any walk \( W \), \( A^*(W) \neq \text{all} \) if and only if \( W \) is implicational.

- **Test** \( T : U \times U \to \{\text{true}, \text{false}\} \). For any single-branching pair \((P, Q)\), \( T(A^*(P), A^*(Q)) = \text{true} \) if and only if \( P \circ Q^{-1} \) is conflicting.

The running time of the incremental-test oracle is defined as the maximum running time of the three functions. In general, we can naively implement the incremental-test oracle by setting \( U := \bigcup_{v \in V} D(v) \), \( I(s) := \varphi_A(s) \), \( A(x, e) = C_e(x) \), and \( T(x, y) := \text{true} \) iff \( x \neq y \). We can obtain a constant-time oracle by this naive implementation for several cases including NODE MULTIWAY CUT. Meanwhile, the naive implementation takes \( O(m) \) time for several other cases including SUBSET FEEDBACK VERTEX SET. We will see in Section \( \text{VI} \) that we can implement a constant-time oracle for these cases by exploiting the constraint that the inputs to the test function are restricted to single-branching pairs.

### III. Half-Integral Packing and Covering

**Theorem 3.** Let \( C \) be a set of \( 0/1 \) all constraints on variables \( V \) and \( \varphi_A \) be a partial assignment for a subset \( A \subseteq V \). Given the primal graph of \( C \), the set \( A \), an incremental-test oracle \( (C, \varphi_A) \), and an integer \( k \), we can compute a pair of minimum half-integral \( F_{C, \varphi_A} \)-cover \( x \) and maximum half-integral \( F_{C, \varphi_A} \)-packing \( y \) with \( |x| = |y| \leq \frac{d}{2} \) or correctly conclude that the size of the minimum half-integral \( F_{C, \varphi_A} \)-cover is at least \( \frac{d}{2} + \frac{1}{2} \) in \( O(kmT) \) time, where \( m \) is the number of constraints and \( T \) is the running time of the incremental-test oracle.

**Fig. 1: Wheel of degree 5.**

Our algorithm is based on a simple augmentation strategy summarized as follows. Starting with \( y(W) = 0 \) for every \( W \in F \), we repeatedly update a half-integral \( F \)-packing \( y \) that always consists of only two types of conflicting walks defined in Section \( \text{III-A} \). In each iteration, we search an augmenting path/pair (see Section \( \text{III-B} \)) in \( O(nmT) \) time. If one is found, we can improve the current \( F \)-packing \( y \) in linear time by Lemma \( 2 \) (Augmentation); otherwise, we can naturally construct a half-integral \( F \)-cover of size \( |y| \), which guarantees the optimality of \( y \) with the aid of the LP-duality. Since each augmentation increases \( |y| \) by at least \( \frac{1}{2} \) on the number of conflicting walks of weight \( \frac{1}{2} \).

#### A. Basic \( F \)-Packing

In what follows, we focus on \( F \)-packings that consist of only two types of conflicting walks. One is a simple path \( I \in F \) of weight \( 1 \), which is called an integral path. The other is a wheel defined as follows and consists of an odd number of conflicting walks of weight \( \frac{1}{2} \).

**Definition 1.** A pair of a simple cycle \( C = H_1 \circ \ldots \circ H_d \) of weight \( \frac{1}{2} \) and (possibly zero-length) paths \( \{S_1, \ldots, S_d\} \) of weight 1 is called a wheel if it satisfies the following conditions (Figure 1).

1. \( d \) is an odd positive integer.
2. For any \( i \), \( S_i \) is a path from \( A \) to \( s(H_i) = t(H_{i-1}) \) (where \( H_0 = H_d \)) that is internally disjoint from \( C \).
3. For any distinct \( i \) and \( j \), \( S_i \) and \( S_j \) share no vertices.
4. For any \( i \), \( S_i \circ H_i \circ S_i^{-1} \in F \), where \( S_{d+1} = S_1 \).

The integer \( d \) is called the degree of the wheel, the cycle \( C \) is called the half-integral cycle of the wheel, and the paths \( \{S_1, \ldots, S_d\} \) are called the spokes of the wheel.

Note that a wheel of degree 1 is a closed walk \( S_1 \circ C \circ S_1^{-1} \in F \) of weight \( \frac{1}{2} \), and a wheel of degree \( d \geq 3 \) is a sum of \( d \) simple paths \( \{S_1 \circ H_1 \circ S_2^{-1}, \ldots, S_d \circ H_d \circ S_1^{-1}\} \subseteq F \) of weight \( \frac{1}{2} \).

A half-integral \( F \)-packing \( y \) is called a basic \( F \)-packing if it is a sum of integral paths and wheels such that each vertex is contained in at most one of the integral paths and the wheels. In our algorithm, a basic \( F \)-packing \( y \) is dealt
with as a weighted graph\(^{13}\) so that we can efficiently update integral paths and wheels in \(y\). We denote by \(V(y)\) and \(E(y)\) the sets of vertices and (undirected) edges, respectively, that are contained in some integral path or wheel in \(y\) (i.e., of positive weights), and particularly by \(V_1(y)\) and \(E_1(y)\), the sets of those contained in some integral path or spoke in \(y\) (i.e., of weight 1). A walk is called internally disjoint from \(y\) if it is internally disjoint from the subgraph \((V(y), E(y))\).

For a basic \(F\)-packing \(y\), we define two functions \(F_y\) (Forward) and \(B_y\) (Backward) as follows. Let \(P\) be a positive-length path contained in an integral path \(I\) in \(y\). From Corollary 1, we can assume that \(I\) has the same direction as \(P\). We then define paths \(F_y(P)\) and \(B_y(P)\) such that \(I = F_y(P) \circ P \circ B_y(P)^{-1}\) holds. For a vertex \(v\) contained in a spoke \(S\) in \(y\), we denote by \(F_y(v)\) the path from \(t(S)\) to \(v\) along \(S\) and by \(B_y(v)\) the path from \(s(S)\) to \(v\) along \(S\) (i.e., \(F_y(v) \circ B_y(v)^{-1} = S^{-1}\)). For a path \(P\) contained in a spoke \(S\) in \(y\) in the opposite direction to \(S\), we define \(F_y(P) := F_y(s(P))\) and \(B_y(P) := B_y(t(P))\) (i.e., \(F_y(P) \circ P \circ B_y(P)^{-1} = S^{-1}\)). We omit the subscript \(y\) if it is clear from the context.

### B. Augmenting Path/Pair

We first define an alternating path to define our augmenting path/pair.

**Definition 2.** For a basic \(F\)-packing \(y\), a concatenation of paths \(P = P_1 \circ \cdots \circ P_p\) is called a y-alternating path if it satisfies all the following conditions.

1. The edges in \(E(P)\) are distinct (i.e., \(1_E(P)(e) \leq 1\) for every \(e \in E\)).
2. Every vertex in \(P\) that is not contained in any integral path or spoke in \(y\) appears in \(P\) at most once\(^{14}\) (i.e., \(1_V(P)(v) \leq 1\) for every \(v \in V \setminus V_1(y)\)).
3. \(s(P) \in A \setminus V(y)\).
4. Each \(P_i\) is a path of positive length satisfying the following conditions.
   a) For any odd \(i\), \(P_i\) is internally disjoint from \(y\), and no internal vertex of \(P_i\) is in \(A\).
   b) For any even \(i\), \(P_i\) is contained in an integral path or a spoke in \(y\). In the latter case, \(P_i\) has the opposite direction to the spoke.
5. Let us define \(B(P_0) := (s(P))\). The following conditions are satisfied for any \(i\)\(^{15}\) (see Figure 2).
   a) If \(i\) is odd, \(B(P_{i-1}) \circ P_i\) is implicative.
   b) If \(i\) is even and \(P_i\) is contained in an integral path, then \(B(P_{i-2}) \circ P_{i-1} \neq F(P_i)\), and none of the

\(^{13}\)The weight is naturally defined for each vertex \(v\) and edge \(e\) by \(\sum_{E \in \mathcal{F}} 1_{E(W)}(v)g(W)\) and \(\sum_{E \in \mathcal{F}} 1_{E(W)}(e)g(W)\), respectively.

\(^{14}\)From the other properties of the \(y\)-alternating paths, a vertex contained in an integral path or a spoke can appear twice in \(P\). Hence, \(P\) may not be a path in the precise sense.

\(^{15}\)Technically, the conditions “none of the \(P_j\)’s are contained in \(\ldots\)” mean that \(P\) does not admit a shortcut (e.g., if there is some \(P_i\) with \(j > i\) contained in \(B(P_i)\), we can obtain another \(y\)-alternating path \(P_1 \circ \cdots \circ P_{i-1} \circ W \circ P_{j+1} \circ \cdots \circ P_p\), where \(W\) is the path from \(s(P_i)\) to \(t(P_j)\) along the integral path or spoke). As in the case of matroid intersection, we need a shortcut-less alternating path.

![Fig. 2: Conditions for alternating paths.](image)

\(P_j\)’s are contained in \(B(P_i)\) for \(j < i\). Moreover, if \(B(P_{i-2}) \circ P_{i-1} \neq B(P_i) \circ P_i^{-1}\), none of the \(P_j\)’s are contained in \(F(P_i)\) for \(j < i\).

b) If \(i\) is even and \(P_i\) is contained in a spoke, then \(B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}\), and none of the \(P_j\)’s are contained in \(B(P_i)\) for \(j < i\).

c) If \(i\) is even and \(P_i\) is contained in a spoke, \(B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}\), and none of the \(P_j\)’s are contained in \(B(P_i)\) for \(j < i\).

d) If \(i\) is odd and \(P_i\) is contained in a spoke, \(B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}\), and none of the \(P_j\)’s are contained in \(B(P_i)\) for \(j < i\).

We define \(T_y(P)\) (Tail) for a \(y\)-alternating path \(P = P_1 \circ \cdots \circ P_p\) as follows: \(T_y(P) := B_y(P_{p-1}) \circ P_p\) if \(p\) is odd, and \(T_y(P) := B_y(P_p)\) if \(p\) is even. Note that \(T_y(P)\) is always implicative. We omit the subscript \(y\) if it is clear from the context.

**Definition 3.** A \(y\)-alternating path \(P = P_1 \circ \cdots \circ P_p\) is called a \(y\)-augmenting path if \(p\) is odd and one of the following conditions is satisfied.

1. \(t(P) \in A \setminus V(y)\), and \(T(P)\) is conflicting.
2. \(t(P)\) is contained in a half-integral cycle, but in no spokes (i.e., \(t(P) \in V(y) \setminus V_1(y)\)).
3. \(t(P)\) is contained in a spoke \(S\), and the following two conditions are satisfied.
   a) \(T(P) \neq B(t(P))\).
   b) For any \(P_j\) contained in the spoke \(S\), \(t(P)\) is contained in \(F(P_j)\).

**Definition 4.** Let \(P = P_1 \circ \cdots \circ P_p\) and \(Q = Q_1 \circ \cdots \circ Q_q\) be a pair of \(y\)-alternating paths ending at the same vertex. \((P, Q)\) is called a \(y\)-augmenting pair if it satisfies all the following conditions.

1. \(P\) and \(Q\) can be written as \(P = R \circ P'\) and \(Q = R \circ Q'\), respectively, for some walk \(R\) such that \(P'\) and \(Q'\) share no edges.
2. \(T(P) \neq T(Q)\).
3. At least one of \(p\) and \(q\) is odd, and if both of \(p\) and \(q\) are odd, \(t(P) \notin V(y)\).
4. For any \(P_i\) contained in an integral path, none of the \(Q_j\)’s are contained in \(B(P_i)\) in the opposite direction. Moreover, none of the \(Q_j\)’s are contained in \(F(P_i)\) in the same direction if \(B(P_{i-2}) \circ P_{i-1} \neq B(P_i) \circ P_i^{-1}\). The symmetric condition holds for any \(Q_i\) contained in an integral path.

**Lemma 2.** Given a basic \(F\)-packing \(y\) and a \(y\)-augmenting path/pair, a basic \(F\)-packing of size at least \(|y| + \frac{1}{2}\) can be constructed in linear time.
vertices which is initialized as follows. For a vertex in \( A \setminus V(y) \), we augment \( y \) by 1 by taking the symmetric difference of \( E(y) \) and \( E(W) \). For a \( y \)-augmenting path ending at a vertex in a wheel of degree 2d+1, we augment \( y \) by \( \frac{1}{2} \) by taking the symmetric difference of \( E(y) \) and \( E(W) \) and then decomposing the wheel into \( d+1 \) integral paths (see Figure 3). For a \( y \)-augmenting pair \((R \circ P, R \circ Q)\) with a common prefix \( R \), we augment \( y \) by \( \frac{1}{2} \) by taking the symmetric difference of \( E(y) \) and \( E(R) \) and then introducing a new wheel whose half-integral cycle is \( P \circ Q^{-1} \) (see Figure 4). In addition to this basic strategy, we need several additional operations in ensuring that the obtained packing is a basic \( F \)-packing. See the full version for more detail.

Due to space limitations, we only describe an outline of our \( O(mnT) \)-time algorithm for searching an augmenting path/pair. For more detail, see the full version. In our algorithm, we iteratively expand a set of visited vertices which is initialized as \( A \). For each visited vertex \( v \), we know that there is an alternating path \( P(v) \) ending at \( v \) and we store the value \( \text{tail}(v) := A^*(T(P(v))) \) using the incremental-test oracle. In each step, we pick an unprocessed edge \( uv \in E \) such that \( v \) is visited and \( P(u) \circ uv \) forms an alternating path. If \( P(u) \circ uv \) satisfies the condition for augmenting paths, we return it, and if \( v \) is not visited yet, we mark \( v \) visited and set \( \text{tail}(v) = A(\text{tail}(u), uv) \). If \( v \) is already visited, we test \( T(P(u)) \neq T(P(u) \circ uv) \) by querying \( T(\text{tail}(v), A(\text{tail}(u), uv)) \). If it returns false, we do nothing and forget the obtained alternating path \( P(u) \circ uv \), and otherwise, \((P(v), P(u) \circ uv)\) forms an augmenting pair. Because we only keep at most one alternating path for each vertex and each step takes \( O(T) \) time, the total running time is \( O(mT) \).

When the search fails to find an augmenting path/pair, we can construct a minimum half-integral \( F \)-cover as follows. In the case of the maximum \( s-t \) flow, when we failed to find an augmenting path, we can construct a minimum \( s-t \) cut by taking the set of edges on the flow that separates the vertices visited by the failed search from the unvisited vertices. Similarly, we can construct a minimum half-integral \( F \)-cover by taking the set of vertices on the half-integral \( F \)-packing that separates visited vertices from the unvisited vertices.

IV. FARDEST COVER

**Theorem 4.** Given the same input as in Theorem 3, we can compute a pair of a farthest minimum half-integral \( F_{C,F,A} \)-cover \( x \) and a maximum half-integral \( F_{C,F,A} \)-packing \( y \) with \(|x| = |y| \leq \frac{k+1}{2} \) or correctly conclude that the size of the minimum half-integral \( F_{C,F,A} \)-cover is at least \( \frac{k+1}{2} \) in \( O(kmT) \) time.

**Proof sketch.** In the existing work using the \( s-t \) cut approach\cite{19}, a farthest cover is obtained by exploiting a structural property of all the minimum \( s-t \) cuts\cite{28}. We use a different approach because we do not have the corresponding structural property for all the minimum half-integral \( F \)-covers. In our algorithm, we exploit the following complementary slackness condition, whose proof is given in the full version.

**Lemma 3.** The following holds for any maximum basic \( F \)-packing \( y \) and any minimum half-integral \( F \)-cover \( x \).

1) \( x(V(I)) = 1 \) for any integral path \( I \) of \( y \).
2) \( x(V(S)) = \frac{1}{2} \) for any spoke \( S \) of \( y \).

Fix an arbitrary maximum basic \( F \)-packing. From the above lemma, for every minimum half-integral \( F \)-cover \( x \), we can construct unique indices \( a_x \) and \( b_x \) satisfying the following.

1) For every integral path \( I = (v_0, \ldots, v_t) \), \( x(v_{a_x(I)}) \geq \frac{1}{2} \) and \( a_x(I) \leq b_x(I) \) hold.
2) For every spoke \( S = (v_0, \ldots, v_t) \), \( x(v_{a_x(S)}) = \frac{1}{2} \) holds.

Therefore, we obtain the following corollary.

**Corollary 3.** A minimum half-integral \( F \)-cover \( x' \) dominates a minimum half-integral \( F \)-cover \( x \) if and only if the following conditions hold.

1) For every integral path \( I \), \( a_{x'}(I) \leq a_x(I) \leq b_x(I) \) holds.
2) For every spoke \( S \), \( a_{x'}(S) \leq a_x(S) \) holds.

Let \( I_1, I_2, \ldots \) and \( S_1, S_2, \ldots \) be the integral paths and spokes in \( y \). By exploiting the above corollary, we can obtain a farthest minimum half-integral \( F \)-cover by maximizing...
Let \( I = (C, \varphi_A) \) be a pair with the empty partial assignment (i.e., \( A = \emptyset \)). A set of pairs \( B \subseteq \{(u, a) \mid u \in V, a \in D(u)\} \) is called a branching set for \( I \) if it has the following property: any deletion set for \( I \) is a deletion set for at least one of \((C, \varphi_B(u))\) with \( \{(u, a) \mid u \in B \} \) and \( \varphi_B(u) = a \). The running time of our FPT algorithm depends on the choice of branching sets. In general, we can use the following standard choice: pick a vertex \( u \in V \) and set \( B := \{(u, a) \mid a \in D(u)\} \). We can choose different branching sets for problem-specific improvements.

**Theorem 5.** Let \( I = (C, \varphi_A) \) be a pair of a set \( C \) of 0/1 all constraints on a variable set \( V \) and a partial assignment \( \varphi_A \) for a subset \( A \subseteq V \). We are given the primal graph of \( C \), the set \( A \), an \( O(T) \)-time incremental-test oracle for \((C, \varphi_A)\), and an integer \( k \). Under the following assumptions, we can correctly answer whether \((C, \varphi_A)\) admits a deletion set of size at most \( k \) or not in \( O(\max(n, 2^k)k^2n2^{(k-c(I))k\cdot T}) \) time, where \( c \) is the integer in the assumption, \( n \) is the number of constraints, and \( c(I) \) is the size of the minimum half-integral \( F_{C, \varphi_A} \)-cover.

1. For any \( V' \subseteq V \), we can choose a branching set for \((C[V'], \varphi_B)\) of size at most \( b \).
2. For any \((u, a) \in B\) for a possible branching set \( B \) for \((C[V'], \varphi_B)\), we have an \( O(T) \)-time incremental-test oracle for \((C[V'], \varphi_{A})\) with \( \varphi_{A}(u) = a \).

**Proof sketch.** We use the following branch-and-bound method. First, we compute a farthest minimum half-integral \( F \)-cover \( x \) in \( O(knT) \) time. Because the size of \( x \) is a lower bound on the size of the minimum deletion set, if \( |x| > k \) holds, there exists no deletion set of size at most \( k \). Otherwise, we have \( k - c(I) \geq 0 \). Using Theorem 2, we can safely include the set \( X := \{v \in V \mid x(v) = 1\} \) into the deletion set and delete \( X \cup R(x) \) from \( C \). This does not change the gap \( k - c(I) \). Because \( x \) is a farthest minimum half-integral \( F \)-cover, we can prove that the obtained instance has a unique minimum half-integral \( F \)-cover that assigns \( \frac{1}{2} \) to each \( u \in A \) and 0 to the others.

If \( A = \emptyset \), we pick a vertex \( u \in A \) and branch into two cases: either include \( u \) to the deletion set or not. Because of the uniqueness of the minimum half-integral \( F \)-cover, the gap \( k - c(I) \) strictly decreases for each case. From the half-integrality, the decrease is at least \( \frac{1}{2} \).

If \( A = \emptyset \), we choose a branching set \( B \) of size at most \( b \) and branch into \(|B| \) cases: \( (C, \varphi_B(u)) \) with \( \varphi_B(u) = a \) for each \( (u, a) \in B \). For obtaining a linear-time FPT algorithm, we need to ensure that the gap strictly decreases for each case. A big obstacle here is the existence of two-fan constraints. When \( C \) contains no two-fan constraints, if some of \((C, \varphi_{(u)})\) has a half-integral \( F \)-cover of size zero, then it also has a deletion set of size zero. On the other hand, when \( C \) contains two-fan constraints, some of \((C, \varphi_{(u)})\) can have a half-integral \( F \)-cover of size zero even if it does not have a deletion set of size zero, and thus the gap may not decrease. We overcome this obstacle as follows.

Because 2-SAT can be expressed as 0/1 ALL DELETION with \( k = 0 \) and \( d = 2 \), any linear-time FPT algorithm for 0/1 ALL DELETION must be able to solve 2-SAT in linear time. The standard linear-time algorithm for 2-SAT uses the strongly connected component decomposition of the implication graph, and the existing linear-time FPT algorithms for ALMOST 2-SAT, a parameterized version of 2-SAT, also use the strongly connected component decomposition of an auxiliary network [18], [19], [29]. We cannot use this approach because the size of the auxiliary network becomes super-linear for our problems. In our algorithm, we do not use the strongly connected component decomposition, but instead use a parallel unit-propagation, which is an alternative linear-time algorithm for 2-SAT. With the aid of the parallel unit-propagation, we can find a branching set for which every case strictly decreases the gap in \( O(nT) \) time.

Now, we have proved that, in each step, the algorithm branches into at most \( \max(2, b) \) cases, and for each case, the gap \( k - c(I) \) decreases by at least \( \frac{1}{2} \). Thus, the total running time is \( O(\max(2, b)2^{(k-c(I))k\cdot T}) \).

**VI. APPLICATIONS**

Finally, we show that various NP-hard problems can be expressed as a special case of 0/1 ALL DELETION. Note that we use \( A = \emptyset \) for every problem other than NODE MULTIWAY CUT. We obtain a linear-time FPT algorithm for each problem by giving an incremental-test oracle and a specialized choice of branching sets.

**NODE UNIQUE LABEL COVER**

**Input:** A finite alphabet \( \Sigma \), a graph \( G = (V, E) \), a permutation \( \pi \) of \( \Sigma \) for every edge \( e \in E \) such that \( \pi_{uv} = \pi_{vu}^{-1} \) given as a table of size \( |\Sigma|^2 \), and an integer \( k \).

**Question:** Is there a pair of set \( X \subseteq V \) of at most \( k \) vertices and assignment \( \varphi : V \setminus X \to \Sigma \) such that \( \pi_{uv}(\varphi(u)) = \varphi(v) \) for every \( uv \in E \setminus V \)?

**TWO-FAN DELETION**

**Input:** A set of variables \( V \), a set of two-fan constraints on \( V \) of the form \( (\varphi(u) = u) \lor (\varphi(v) = v) \) given as a pair \( (a, b) \), and an integer \( k \).

**Question:** Is there a pair of set \( X \subseteq V \) of at most \( k \) variables and assignment \( \varphi \) satisfying every constraint \( C_{uv} \in E \setminus V \setminus X \)?

---

16For example, consider a 2-CNF \( C = (u \lor v) \land (v \lor w) \land (v \lor w) \land (v \lor w) \land (v \lor w) \) and a branching set \( B = \{(u, true), (u, false)\} \). The size of the minimum deletion set is 1 but the instance \( (C, \varphi_{(u)}) \) with \( \varphi_{(u)}(u) = true \) has an \( F \)-cover of size zero.
These two problems are special cases of 0/1 ALL DELETION such that the set of constraints is limited to permutation or two-fan constraints. Note that the size of the domain is not a parameter for TWO-FAN DELETION. The naive implementation of the incremental-test oracle for these problems runs in a constant time. Thus, we can solve NODE UNIQUE LABEL COVER in \(O(\Sigma^{2km})\) time. We can use the following choice of a branching set for TWO-FAN DELETION: pick a two-fan constraint \(\langle \phi(u) = a \rangle \vee \langle \phi(v) = b \rangle\) and set \(B := \{(u, a), (v, b)\}\). Thus, we can solve TWO-FAN DELETION in \(O(2^{2km}) = O(4^{km})\) time.

**Lemma 4.** \(B := \{(u, a), (v, b)\}\) for a two-fan constraint \(\langle \phi(u) = a \rangle \vee \langle \phi(v) = b \rangle\) is a branching set.

**Proof.** Let \(X\) be a deletion set for \(I\) and let \(\phi\) be an assignment for \(V \setminus X\) satisfying \(C[V \setminus X]\). If \(u \in X\), \(X\) is a deletion set for \(I[u \leftarrow a]\), and if \(v \in X\), \(X\) is a deletion set for \(I[v \leftarrow b]\). Otherwise, at least one of \(\phi(u) = a\) or \(\phi(v) = b\) holds. \(X\) is a deletion set for \(I[u \leftarrow a]\) in the former case, while \(X\) is a deletion set for \(I[v \leftarrow b]\) in the latter case.

The next two problems generalize PSEUDOFOREST DELETION [5] in different directions, where a pseudoforest is a graph in which the number of edges is at most the number of vertices for every connected component.

A graph is called monochromatically orientable if there exists an edge orientation such that, for every vertex, all the incoming edges are monochromatic. It is known that a graph is a pseudoforest if and only if there exists an edge orientation such that, for every vertex, the number of incoming edges is at most one. Therefore, when every edge has a distinct color, a graph is monochromatically orientable if and only if it is a pseudoforest. Thus, the following problem is a generalization of PSEUDOFOREST DELETION.

**PSEUDOFOREST DELETION**

**Parameter:** \(k\)

**Input:** An edge-colored graph \(G = (V, E)\) and an integer \(k\).

**Question:** Is there a set \(X \subseteq V\) of at most \(k\) vertices such that \(G - X\) is monochromatically orientable?

We can solve MONOCHROMATICALLY ORIENTABLE DELETION in \(O(4^{k}km)\) time by the following reduction to TWO-FAN DELETION. Let \(L\) be the set of colors. For each vertex \(v \in V\), we create a variable \(v\) with domain \(D(v) = L\), which represents the color of the incoming edges. We create a two-fan constraint \(\langle \phi(u) = c \rangle \vee \langle \phi(v) = c \rangle\) for each edge \(uv \in E\) of color \(c \in L\).

**Lemma 5.** A graph \(G = (V, E)\) is monochromatically orientable if and only if the corresponding set of two-fan constraints is satisfiable.

**Proof.** From a monochromatic orientation of \(G\), we can construct a satisfying assignment \(\phi\) as follows: we set \(\phi(v) := c\) for each vertex \(v \in V\), where \(c\) is the unique color of the incoming edges or an arbitrary color if \(v\) has no incoming edges. From a satisfying assignment \(\phi\), we can construct a monochromatic orientation by orienting each edge \(uv \in E\) of color \(c\) so that the edge is directed toward \(u\) if \(\phi(u) = c\), and toward \(v\) otherwise.

Another generalization of PSEUDOFOREST DELETION is presented as follows. For a graph \(G = (V, E)\) and an edge \(e = uv \in E\), contracting \(e\) is an operation deleting the edge \(e\) and merging \(u\) and \(v\) into a new vertex \(e\). Note that this operation may create parallel edges (edges \(uw\) and \(vw\) become parallel edges) and self-loops. If \(G\) has parallel edges connecting \(u\) and \(v\), the operation only removes one of them, and the rest becomes self-loops.

Let \(S\) be a subset of edges. A subset \(X \subseteq V\) is called a subset feedback vertex set if \(G - X\) has no simple cycle passing through an edge of \(S\), or equivalently, the graph obtained from \(G - X\) by contracting every edge \(e \in E[V \setminus X]\) \(S\) is a forest. Similarly, we call \(X\) a subset pseudoforest deletion set if the graph obtained from \(G - X\) by contracting every edge \(e \in E[V \setminus X]\) \(S\) is a pseudoforest.

**PSEUDOFOREST DELETION**

**Parameter:** \(k\)

**Input:** A graph \(G = (V, E)\), a set \(S \subseteq E\), and an integer \(k\).

**Question:** Is there a set \(X \subseteq V\) of at most \(k\) vertices such that the graph obtained from \(G - X\) by contracting every edge \(e \in E[V \setminus X]\) \(S\) is a pseudoforest?

This problem can be expressed as 0/1 ALL DELETION as follows. Every vertex \(v\) has the same domain \(D(v) = S\). We introduce a two-fan constraint \(\langle \phi(u) = e \rangle \vee \langle \phi(v) = e \rangle\) for every edge \(e = uv \in S\). We also introduce an equality (identity permutation) constraint \(\phi(u) = \phi(v)\) for every edge \(uv \notin S\). Let \(C\) be the obtained set of constraints. If \(C[V']\) has no two-fan constraints for some \(V' \subseteq V\), \((C[V'], \phi)\) has a deletion set of size zero. Therefore, we can use the same choice of branching sets as for TWO-FAN DELETION. Thus, the algorithm runs in \(O(4^{k}km)\) time. The correctness of the expression follows from the following lemma.

**Lemma 6.** Let \(G = (V, E)\) be a graph with a subset \(S \subseteq E\) and let \(C\) be the corresponding set of 0/1/all constraints. Then, the graph obtained by contracting every edge \(e \in E[S]\) is a pseudoforest if and only if \(C\) is satisfiable.

**Proof.** We modify \(C\) for each contraction of an edge \(e = uv\) by removing the constraint \(C_{uv}\) and replacing every occurrence of \(u\) and \(v\) by \(e\). When we obtain the graph \(G'\) by contracting every edge \(e \in E[S]\), we also obtain the set \(C'\) of constraints corresponding to \(G'\). Because every contracted edge \(e = uv\) has the equality constraint, \(C\) is satisfiable if and only if \(C'\) is satisfiable. From Lemma 5, \(C'\) is satisfiable if and only if \(G'\) is a pseudoforest. Therefore, \(C\) is satisfiable if and only if \(G'\) is a pseudoforest.

As mentioned in Section 1-C, NODE MULTIWAY CUT is also a special case of our problem.
**Node Multiway Cut**

**Parameter:** $k$

**Input:** A graph $G = (V, E)$, a set of terminals $T \subseteq V$, and an integer $k$.

**Question:** Is there a set $X \subseteq V \setminus T$ of size at most $k$ such that every terminal in $T$ lies in a different connected component of $G - X$?

This problem can be expressed as $0/1$ KELLEY DELETION as follows. First, we split each terminal $s \in T$ as follows to make $s$ undeletable: for each edge $sv \in \delta(s)$, we create a new vertex $s_v$ and replace the edge $sv$ with $s_vv$. Let $G' = (V', E')$ be the obtained graph. We introduce a variable $v$ with $D(v) = T$ for each vertex $v \in V'$ and an equality (identity permutation) constraint $\varphi(u) = \varphi(v)$ for each edge $uv \in E'$. Finally, we set $A = \{s_v \mid s \in T, sv \in \delta(s)\}$ and $\varphi_A(s_v) := s$. We can observe that a minimum deletion set $X$ avoiding every $s_v$ always exists because each vertex $s_v$ has degree one, and such $X$ is actually a minimum multiway cut.

Let $I = (C, \varphi_A)$ be the obtained instance. We can construct a multiway cut of size at most $|T|$ by rounding up a half-integral $F_I$-cover $x$. Therefore, we have $k - c(I) \leq \frac{1}{2}k$. Because the domain size is $|T| = O(n)$, the naive implementation of the incremental-test oracle runs in a constant time. Because any $I = (C[V'], \varphi_0)$ has a deletion set of size zero, we do not need branching sets. Thus, we can solve NODE MULTIWAY CUT in $O(2^k + km) = O(2^km)$ time.

We finally show GROUP FEEDBACK VERTEX SET and its applications.

**Group Feedback Vertex Set**

**Parameter:** $k$

**Input:** A group $\Gamma = (D, \cdot)$ given as an $O(T_{\Gamma})$-time oracle performing the group operation $\cdot$, a $\Gamma$-labeled graph $G = (V, E)$ with labeling $\lambda : E \to D$ with $\lambda(uv) \cdot \lambda(vx) = 1_{\Gamma}$ for every $uv \in E$, where $1_{\Gamma}$ is the unity of $\Gamma$, and an integer $k$.

**Question:** Is there a set $X \subseteq V$ of at most $k$ vertices such that $G - X$ has a consistent labeling? That is, is there a labeling $\varphi : V \setminus X \to D$ such that $\varphi(u) \cdot \lambda(uv) = \varphi(v)$ for every $uv \in E[V \setminus X]$?

This problem can be expressed as $0/1$ DELETION because a function $\pi_{\ell}(a) := a \cdot \lambda(e)$ is a permutation. Note that $G - X$ has a consistent labeling if and only if it admits no non-zero cycles (i.e., it admits no cycle $(v_0, v_1, \ldots, v_l)$ with $\lambda(v_0v_1) \cdot \lambda(v_1v_2) \cdots \lambda(v_{l-1}v_l) \neq 1_{\Gamma}$). In contrast to NODE UNIQUE LABEL COVER, the domain size is not a parameter, and each permutation is given not as a table of size $|D|$, but as an $O(T_{\Gamma})$-time oracle answering $a \cdot b$ for given $a, b \in D$. Therefore, the naive implementation of the incremental-test oracle runs in $O(T_{\Gamma})$ time. We can use the following choice of a branching set: pick a vertex $s$ and set $B := \{(s, 1)\}$. Thus, we can solve GROUP FEEDBACK VERTEX SET in $O(2^km) = O(4^km)$ time.

**Lemma 7.** $B := \{(s, 1)\}$ is a branching set for GROUP FEEDBACK VERTEX SET.

**Proof.** Let $X$ be a deletion set for $I$, and let $\varphi$ be an assignment for $V \setminus X$ satisfying $C[V \setminus X]$ if $s \in X$, $X$ is a deletion set for $I[s \leftarrow 1]$; otherwise, $\varphi'$ such that $\varphi'(v) := \varphi(v) \cdot \varphi(s)^{-1}$ is a satisfying assignment for $C[V \setminus X]$ with $\varphi'(s) = 1_{\Gamma}$.

**Subset Feedback Vertex Set**

**Parameter:** $k$

**Input:** A graph $G = (V, E)$, a set $S \subseteq E$, and an integer $k$.

**Question:** Is there a set $X \subseteq V$ of at most $k$ vertices such that no cycle passes through an edge of $S$ in $G - X$?

This problem can be expressed as GROUP FEEDBACK VERTEX SET as follows. We use a group $\Gamma = (2^S, \oplus)$, where $\oplus$ is the XOR operator $(X \oplus Y = (X \setminus Y) \cup (Y \setminus X))$. We set $\lambda(e) = \{1\}$ for each edge $e \in S$ and $\lambda(e) = \emptyset$ for each edge $e \in E \setminus S$. Then, a cycle is non-zero if and only if it contains an edge in $S$.

The group operation takes $O(|S|) = O(m)$ time. Therefore, the naive implementation of the incremental-test oracle takes $O(m)$ time. We now provide a constant-time incremental-test oracle for $(C[V'], \varphi_A)$ with $\varphi_A(s) = \emptyset$. We use the following implementation.

**Proof.** Let $X$ be a deletion set for $I$, and let $\varphi$ be an assignment for $V \setminus X$ satisfying $C[V \setminus X]$ if $s \in X$, $X$ is a deletion set for $I[s \leftarrow 1]$; otherwise, $\varphi'$ such that $\varphi'(v) := \varphi(v) \cdot \varphi(s)^{-1}$ is a satisfying assignment for $C[V \setminus X]$ with $\varphi'(s) = 1_{\Gamma}$.

**Non-monochromatic Cycle Transversal**

**Parameter:** $k$

**Input:** An edge-colored graph $G = (V, E)$ and an integer $k$.

**Question:** Is there a set $X \subseteq V$ of at most $k$ vertices such that $G - X$ contains no non-monochromatic cycles?

This problem can be expressed as GROUP FEEDBACK VERTEX SET as follows. Let $L$ be the set of colors, and let $c(e) \in L$ denote the color of an edge $e$. We use the group $\Gamma = (2^V \times L, \oplus)$. We set $\lambda(e) = \{(u, c(e)) | (u, c(e))\}$ for each edge $e = uv \in E$. Then, a cycle is non-zero if and only if it is non-monochromatic.

The naive implementation of the incremental-test oracle takes $O(n |L|) = O(nm)$ time. We now provide a constant-time incremental-test oracle for $(C[V'], \varphi_A)$ with $\varphi_A(s) = \emptyset$. We use the following implementation.

**Proof.** Let $X$ be a deletion set for $I$, and let $\varphi$ be an assignment for $V \setminus X$ satisfying $C[V \setminus X]$ if $s \in X$, $X$ is a deletion set for $I[s \leftarrow 1]$; otherwise, $\varphi'$ such that $\varphi'(v) := \varphi(v) \cdot \varphi(s)^{-1}$ is a satisfying assignment for $C[V \setminus X]$ with $\varphi'(s) = 1_{\Gamma}$.
\[ A(c, s) = (s, c(sv)). \]

\[ A((w, c), uv) = \begin{cases} 
  \star & (w = v \land c = c(uw)) \\
  (w, c) & (w \neq v \land c = c(uw)) \\
  (u, c(uv)) & (c \neq c(uv)). 
\end{cases} \]

Let \( W = (v_0, \ldots, v_\ell) \) be a walk with \( \ell > 0 \), and let \( c = c(v_{i-1}v_i) \) be the longest monochromatic suffix of \( W \) if \( c(v_{i-1}v_{i+1}) = c \) for every \( j \geq i \) and \( c(v_{i-1}v_i) \neq c \) or \( i = 0 \) holds. We can observe that the longest monochromatic suffix of \( W \) starts from \( v_i \) and has the color \( c \) if \( A^*(W) = (v_i, c) \), and the longest monochromatic suffix of \( W \) forms a monochromatic cycle if \( A^*(W) = \star \). Then, for a single-branching pair \((P, Q)\), we have \( P \neq Q \iff \text{the simple cycle contained in } P \circ Q^{-1} \text{ is non-monochromatic} \iff \text{none of } P \text{ and } Q \text{ induces a monochromatic cycle and the longest monochromatic suffixes of } P \text{ and } Q \text{ start from different vertices or have different colors} \iff \star \neq A^*(P) \neq A^*(Q) \neq \star \). Therefore, the abovementioned implementation is correct. Thus, we can solve NON-MONOCROMATIC CYCLE TRANSVERSAL in \( O(4^k km) \) time.

ACKNOWLEDGMENT

We would like to thank Yusuke Kobayashi for pointing out to us the simulation of two-fan constraints by permutation constraints. Yoichi Iwata is supported by JSPS KAKENHI Grant Number JP17K12643. Yutaro Yamaguchi is supported by JSPS KAKENHI Grant Number JP16H06931 and JST ACT-I Grant Number JPMJER1305 and JSPS KAKENHI Grant Number JP17H04676.

REFERENCES