Editing to Connected f-Degree Graph*

Fedor V. Fomin[1], Petr Golovach[1], Fahad Panolan[3], and Saket Saurabh[4]

1 **Department of Informatics, University of Bergen, Norway**
`fomin@ii.uib.no`
1 **Department of Informatics, University of Bergen, Norway**
`petr.golovach@ii.uib.no`
3 **The Institute of Mathematical Sciences, India**
`fahad@imsc.res.in`
4 **Department of Informatics, University of Bergen, Norway; and The Institute of Mathematical Sciences, India**
`saket@imsc.res.in`

Abstract

In the Edge Editing to Connected f-Degree Graph problem we are given a graph G, an integer k and a function f assigning integers to vertices of G. The task is to decide whether there is a connected graph F on the same vertex set as G, such that for every vertex v, its degree in F is $f(v)$ and the number of edges in $E(G)\triangle E(F)$, the symmetric difference of $E(G)$ and $E(F)$, is at most k. We show that Edge Editing to Connected f-Degree Graph is fixed-parameter tractable (FPT) by providing an algorithm solving the problem on an n-vertex graph in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. Our FPT algorithm is based on a non-trivial combination of color-coding and fast computations of representative families over direct sum matroid of ℓ-elongation of co-graphic matroid associated with G and uniform matroid over $\overline{E(G)}$, the set of non-edges of G. We believe that this combination could be useful in designing parameterized algorithms for other edge editing problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Connected f-factor, FPT, Representative Family, Color Coding

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.36

1 Introduction

A subgraph F of a graph G is a factor of G, if F is a spanning subgraph of G. When a factor F is described in terms of its degrees, it is called a *degree-factor*. For example, one of the most fundamental notions in Graph Theory is 1-factor (or a perfect matching), the case when a factor F has all of its degrees equal to 1. Another example is r-factor, a regular spanning subgraph of degree r. More generally, for a function $f\colon V(G)\to\mathbb{N}$, subgraph F is a f-factor of G if for every $v\in V(G)$, $d_F(v)$, the degree of v in F is exactly equal to $f(v)$. The study of degree factors is one of the mainstays of combinatorics with a long history dating back to 1847 to the works of Kirkman [10], and Petersen [18]. We refer to surveys [1, 19], as well as the book of Lovász and Plummer [13] for an extensive overview of degree factors.

Another broad set of degree-factor problems is obtained by requesting the factor to be connected. The most famous examples are another old classical Graph Theory notion, the

* Supported by the European Research Council (ERC) via grants Rigorous Theory of Preprocessing, reference 267959 and PARAPPROX, reference 306992.

Hamiltonian cycle, which is a connected 2-factor, and the Eulerian subgraph, which is a connected even-degree factor. We refer to the survey of Kouider and Vestergaard [11] on connected factors, as well as to the book of Fleischner [6] for a thorough study of Eulerian graphs and related topics.

A natural algorithmic problem concerning (connected) f-factors is for a given graph G and a function f, to decide if G contains a (connected) f-factor. While deciding if a given graph contains an f-factor can be done in polynomial time for any function f [3], deciding the existence of even a connected 2-factor (Hamiltonian cycle) is NP-complete. In this work we study parameterized complexity of the following algorithmic generalization of the problem of finding a connected f-factor.

Edge Editing to Connected f-Degree Graph (EECG)
Input: An undirected graph G, a function $f : V(G) \to \{1, 2, \dots, d\}$ and $k \in \mathbb{N}$
Question: Does there exist a connected graph F such that for every vertex v, $d_F(v) = f(v)$, and the cardinality of the symmetric difference $|E(G) \triangle E(F)| \leq k$?

Apart from studying a classical problem algorithmically, one of the main motivations for our interest in the generalization of the classical f-factor problem comes from the recent developments in parameterized algorithms for graph modification problems. These recent algorithmic advances were important not only due to the problems they settled but also for the kind of techniques they brought to the area. For example, the work on cut (or edge-deletion) problems of Kawarabayashi and Thorup [9] and Chitnis et al. [4] brought recursive understanding and randomized contraction techniques. The work on Feedback Arc Set in Tournaments [2] led to the chromatic coding. Study of chordal graph completions from [7] triggered the usage of potential maximal cliques in subexponential parameterized algorithms. The main result of our paper is the following theorem.

▶ **Theorem 1.** EECG *is solvable in time* $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ *deterministically.*

The proof of our theorem is based on (i) color-coding and (ii) fast computation of representative family over a linear matroid. While using graphic matroids to resolve different types of connectivity issues has become a popular theme in algorithms, our proof requires the usage of some non-standard matroids. In particular, we use fast representative family computations over the direct sum matroid of ℓ-elongation of the co-graphic matroid associated with G and a uniform matroid over $\overline{E(G)}$, the set of non-edges of G. We believe that this combination could be useful for designing parameterized algorithms for other edge editing problems. To the best of our knowledge this is the first uses of elongation of matroids in designing of parameterized algorithms.

One of the nice properties of using (graphic) matroids is that often they allow us to lift solutions from unweighted problems to problems with costs. It would be natural to suggest that the nice properties of matroids would help us with the "weighted" version of EECG as well. However, in spite of our attempts, we could not extend Theorem 1 to the weighted version of EECG. We proved that the weighted version of EECG when parameterized by $k + d$ is W[1]-hard and its proof is deferred to the full version of the paper due to paucity of space. Also, proofs of lemmata marked with a $\star$ are deferred to the full version of the paper.

Previous work. It was shown by Mathieson and Szeider [15] that the problem of deleting k vertices to transform an input graph into an r-regular graph, where $r \geq 3$ is W[1]-hard parameterized by k. For edge-modification problems to a graph with certain degrees, Mathieson and Szeider have shown in [15] that Edge Editing to f-Degree Graph, the case when the requirement that the resulting graph F is connected is omitted, is solvable in polynomial time. As with the f-factor problem, the situation changes drastically when one adds the requirement that the resulting graph F is connected. A simple reduction from Hamiltonian cycle shows that in this case deciding if a graph can be edited into a connected 2-degree graph, i.e. a cycle, by changing at most k adjacencies, is NP-complete [8].

Golovach in [8] has shown that when parameterized by the maximum vertex degree d in the resulting graph plus the number of editing operations k, the problem Edge Editing to Connected f-Degree Graph is FPT. In the same paper, it was shown that the variant when the resulting graph F is regular, the problem is FPT parameterized by k. However, prior to our work the complexity status of Edge Editing to Connected f-Degree Graph parameterized by k remained open. Thus Theorem 1 resolves the problem in affirmative. However, we still do not know the kernelization status of this problem and leave it as an interesting open problem.

Our Approach. Each solution to our problem is of the form $D \cup A$ where $D \subseteq E(G)$ and $A \subseteq \overline{E(G)}$. The sets D and A are called a *deletion set* and an *addition set*, respectively, corresponding to the solution $D \cup A$. We start by characterizing our solution in terms of deletion set D, called *nice deletion set*. Nice deletion sets satisfy certain properties and have an accompanying map ψ. This map together with other properties of nice deletion sets allows us to recover the addition set A in polynomial time. Thus, our whole effort is in finding a nice deletion set D with map ψ. This viewpoint allows us to concentrate on finding a nice deletion set with an accompanying map ψ. However, this is not useful for designing the dynamic programming (DP) algorithm. To achieve this we view the solution $D \cup A$ as a system of "alternating walks and alternating even closed walks". Alternating (closed) walks are essentially normal (closed) walks with edges from $D \cup A$ such that they do not have consecutive edges from either D or A, and no edge from $D \cup A$ is repeated in the walk. We take this view point to construct the dynamic programming algorithm as this allows us to proceed between the states of the program by using one edge (either of an addition set or of a deletion set). The number of states can be upper bounded by $2^{\mathcal{O}(k)}$. However, the number of sets $D' \cup A'$ that could satisfy the prerequisite of being in a particular table entry could be as large as $n^{\mathcal{O}(k)}$ and thus this would not lead to an FPT algorithm. However, we follow this template for our algorithm and use some more structural properties to prune the family of partial solutions stored at a DP table entry.

Our pruning is based on the proof that $D \cup A$ is an independent set in some linear matroid. The first observation towards an FPT algorithm is that after we delete the edges in D from the input graph G, the number of connected components can at most be $k - |D| + 1$. This allows us to show that in fact we can think of D being an independent set in the matroid $M_G(\ell)$. That is, ℓ-elongation of the co-graphic matroid, M_G, associated with G, where $\ell = |E(G)| - |V(G)| + k - |D| + 1$ (we refer to preliminaries for the definition). Next we show that for addition set A, all we need to store is some form of disjointness and that can be captured using uniform matroid over the universe $\overline{E(G)}$. Let $U_{m',k-k'}$ be a uniform matroid with ground set $\overline{E(G)}$, where $m' = |\overline{E(G)}|$ and $k' = |D|$. From the definition of $U_{m',k-k'}$, any set A of size at most $k - k'$ is independent in $U_{m',k-k'}$. We have already explained that we view the deletion set D as an independent set in $M_G(\ell)$ where $\ell = |E(G)| - |V(G)| + k - k' + 1$. Thus, to see the solution set $D \cup A$ as an independent set in a single matroid, we consider direct sum of $M_G(\ell)$ and $U_{m',k-k'}$. That is, let $M = M_G(\ell) \oplus U_{m',k-k'}$. In M, a set I is an independent set if and only if $I \cap E(G)$ is an independent set in $M_G(\ell)$ and $I \cap \overline{E(G)}$ is an independent set in $U_{m',k-k'}$. This ensures that any solution $D \cup A$ is an independent set in M. By viewing any solution of the problem as an independent set in the matroid M (which is linear), we can use fast computation of q-representative families to prune the table. However, we still need to take care of a technical requirement in the definition of a nice deletion set. Towards this, we show that for every deletion set D there exists a set of edges W_D, disjoint from D, of size at most $6k$ such that if these edges are not selected then we can satisfy that technical requirement. To achieve this we apply color coding technique and this can be derandomized using a standard application of universal sets.

2 Preliminaries

Throughout the paper we use ω to denote the exponent in the running time of matrix multiplication, the current best known bound for which $\omega < 2.373$ [20]. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a set U, $\binom{U}{2} = \{(u,v) \mid u \neq v \wedge u,v \in U\}$. For any multiset A and an element x, by $A(x)$ we denote the number occurrences of x in A. For any $W \subseteq U \times \mathbb{N}$, where U is a set, $W(u) = |\{(u,i) \in W \mid i \in \mathbb{N}\}|$. We use "graph" to denote simple graphs without self-loops, directions, or labels. We use standard terminology from the book of Diestel [5] for those graph-related terms which we do not explicitly define. In general we use G to denote a graph. We use $V(G)$ and $E(G)$, respectively, to denote the vertex and edge sets of a graph G. For a graph G, we use $\overline{E(G)}$ to denote the simple non-edge set $\binom{V(G)}{2} \setminus E(G)$. For a vertex $v \in V(G)$, we use $E_G(v)$ to denote the set of edges incident on v.

Now we give definitions related to matroids. A pair $M = (E, \mathcal{I})$, where E is a ground set and $\mathcal{I}$ is a family of subsets (called independent sets) of E, is a *matroid* if it satisfies the following conditions: (i) $\emptyset \in \mathcal{I}$, (ii) if $A' \subseteq A$ and $A \in \mathcal{I}$ then $A' \in \mathcal{I}$; and (iii) if $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is $e \in (B \setminus A)$ such that $A \cup \{e\} \in \mathcal{I}$. An inclusion wise maximal set of $\mathcal{I}$ is called a *basis* of the matroid. This size is called the *rank* of the matroid M, and is denoted by $\mathsf{rank}(M)$. The rank function of a matroid $M = (E, \mathcal{I})$ is a function $r_M : 2^E \to \mathbb{N}^+ \cup \{0\}$ which is defined as follows: $r_M(S)$ for any $S \subseteq E$ is the cardinality of the maximum sized independent set contained in S. For a broader overview on matroids, including linear representations of matroids, we refer to [17].

Direct Sum of Matroids. Let $M_1 = (E_1, \mathcal{I}_1)$, $\ldots$, $M_t = (E_t, \mathcal{I}_t)$ be t matroids with $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq t$. The direct sum $M_1 \oplus \cdots \oplus M_t$ is a matroid $M = (E, \mathcal{I})$ with $E := \bigcup_{i=1}^{t} E_i$ and $X \subseteq E$ is independent if and only if $X \cap E_i \in \mathcal{I}_i$ for all $i \in [t]$.

▶ **Proposition 2** ([14, Proposition 3.4]). *Given representations of matroids $M_1, \ldots, M_t$ over the same field $\mathbb{F}$, a representation of their direct sum can be found in polynomial time.*

Uniform Matroids. A pair $M = (E, \mathcal{I})$ over an n-element ground set E, is called a uniform matroid if the family of independent sets is given by $\mathcal{I} = \{A \subseteq E \mid |A| \leq k\}$, where k is some constant. This matroid is also denoted as $U_{n,k}$. Every uniform matroid is linear and can be represented over a finite field of size $\geq n+1$ by a $k \times n$ matrix A_M where $A_M[i,j] = e_j^{i-1}$, and $e_1, \ldots, e_n$ are distinct non-zero elements from the field.

Co-graphic Matroids. Given a graph G with r connected components, the co-graphic matroid associated with G, denoted by M_G, is defined as $(U, \mathcal{I})$, where $U = E(G)$ and $\mathcal{I} = \{S \subseteq E(G) : G - S$ has exactly r connected components$\}$. Co-graphic matroids are representable over any field of size at least 2 [17].

Elongation of a Matroid. The ℓ-*elongation* of a matroid M, where $\ell \geq \mathsf{rank}(M)$ is defined as a matriod $M' = (E, \mathcal{I}')$ such that $S \subseteq E$ is a basis in M' if and only if, $r_M(S) = r_M(E)$ and $|S| = \ell$. We use $M(\ell)$ to denote the ℓ-elongation of a matroid M.

▶ **Theorem 3** ([12]). *Given a representation of a matroid M and $\ell \geq \mathsf{rank}(M)$, there is a deterministic polynomial time algorithm to compute a representation of ℓ-elongation of M.*

▶ **Definition 4** (q-Representative Family [14]). Given a matroid $M = (E, \mathcal{I})$ and a family $\mathcal{S}$ of subsets of E, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is *q-representative* for $\mathcal{S}$ if the following holds: for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{S}$ disjoint from Y with $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y with $\widehat{X} \cup Y \in \mathcal{I}$. If $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is q-representative for $\mathcal{S}$, then we write $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$.

▶ **Theorem 5** ([12]). *Let $M = (E, \mathcal{I})$ be a linear matroid of rank n and let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of independent sets, each of size b. Let A be an $n \times |E|$ matrix representing M over a field $\mathbb{F}$, where $\mathbb{F} = \mathbb{F}_{p^\ell}$ or $\mathbb{F}$ is $\mathbb{Q}$. Then there is deterministic algorithm which computes a representative set $\widehat{\mathcal{S}} \subseteq^q_{rep} \mathcal{S}$ of size at most $nb\binom{b+q}{b}$, using $\mathcal{O}\left(\binom{b+q}{b} tb^3n^2 + t\binom{b+q}{b}^{\omega-1}(bn)^{\omega-1}\right) + (n + |E|)^{\mathcal{O}(1)}$ operations over the field $\mathbb{F}$.*

3 An FPT Algorithm for EECG

Let (G, f, k) be an instance of EECG. Our algorithm finds a solution of size *exactly* k (if one such solution exists) and outputs No otherwise. Any solution to our problem is of the form $D \cup A \in \binom{V(G)}{2}$ where $D \subseteq E(G)$ and $A \subseteq \overline{E(G)}$. The sets D and A are called a deletion set and an addition set, respectively, corresponding to the solution $D \cup A$.

Characterizing the solution. The starting point of our algorithm is a characterization of solution in terms of a deletion set D satisfying certain properties (such deletion sets are called *nice deletion sets*). This, allows us to focus on finding nice deletion sets. To describe the main steps and ideas involved in our algorithm we first give a definition of a nice deletion set. Towards this we need definitions of following two sets .

$$\begin{aligned} \mathsf{def}(G, f) &= \{v \mid v \in V(G), f(v) > d_G(v)\} - \text{ A set of deficient vertices.} \\ \mathsf{S}(G, f) &= \{(v, i) \mid v \in V(G), f(v) > d_G(v), i \in \{1, \ldots, f(v) - d_G(v)\}\} \end{aligned}$$

The second set specifies for every vertex $v \in \mathsf{def}(G, f)$ how many edges in addition set must be adjacent to v. Let $W \subseteq V(G) \times \mathbb{N}$. For convenience we denote a pair $(v, i) \in V(G) \times \mathbb{N}$ (or in W) by $v(i)$. Let $\psi : W \to W$ be a bijection. Given ψ, we define a multiset E_ψ as follows. For each $u(i) \in W$ we add (u, v) to E_ψ if $\psi(u(i)) = v(j)$ for some $j \geq i$. We say that ψ is a *proper deficiency map* if ψ is an involution, for any $u \in V(G)$ $(u, u) \notin E_\psi$, E_ψ is a set and not a multiset, and $E_\psi \cap E(G) = \emptyset$. In general we will have proper deficiency map over the domain $\mathsf{S}(G, f)$ or some set related to this. Finally, a set $D \subseteq E(G)$ is called a *nice deletion set* if

(i) For all $v \in V(G)$, $d_{G-D}(v) \leq f(v)$;
(ii) $|\mathsf{S}(G - D, f)| = 2(k - |D|)$;
(iii) The graph $G - D$ has at most $k - |D| + 1$ connected components;
(iv) Each connected component in $G - D$ contains a vertex v deficient in $G - D$, i.e, such that $d_{G-D}(v) < f(v)$.
(v) There exists a proper deficiency map $\psi : \mathsf{S}(G - D, f) \to \mathsf{S}(G - D, f)$.

Our main structural lemma is the following.

▶ **Lemma 6.** *Let (G, f, k) be an instance of* EECG *and let $D \subseteq E(G)$. Then there exists $A \subseteq \overline{E(G)}$, $|A| = k - |D|$ such that $A \cup D$ is a solution to* EECG *if and only if D is a nice deletion set. Moreover, given a nice deletion set $D \subseteq E(G)$ we can find $A \subseteq \overline{E(G)}$ such that $|A| = k - |D|$ and $D \cup A$ is a solution to* EECG *in polynomial time.*

Proof. $(\Rightarrow)$ Let $A \subseteq \overline{E(G)}, |A| = k - |D|$ such that $A \cup D$ is a solution to EECG. We need to show that $D \subseteq E(G)$ is a nice deletion set. Since $A \cup D$ is a solution to EECG, we have that $d_{G-D}(v) \leq f(v)$ for all $v \in V(G)$, satisfying condition (i). Furthermore, $A \cup D$ being a solution also implies that $\sum_{\{v\,:\,f(v)>d_{G-D}(v)\}} f(v) - d_{G-D}(v) = 2|A| = 2(k - |D|)$. Hence $|\mathsf{S}(G - D, f)| = 2(k - |D|)$, satisfying condition (ii). Since $G - D + A$ is a connected graph, $G - D$ can have at most $|A| + 1 = k - |D| + 1$ connected components, satisfying condition (iii) in the definition. The graph $G - D + A$ is connected and thus each connected component F in $G - D$ contains a vertex $v \in V(F)$ such that $(v, u) \in A$ for some $u \in V(G)$. Since $D \cup A$ is a solution to EECG, $d_{G-D+A}(v) = f(v)$ and hence $d_{G-D}(v) < f(v)$ (because $(v, u) \in A$), satisfying condition (iv). Finally, we show that D satisfies the last property. Let $A = \{e_1, e_2, \ldots, e_r\} \subseteq \overline{E(G)}$ where $r = k - |D|$. Since $D \cup A$ is a solution to EECG, we have that for any vertex v, there are exactly $f(v) - d_{G-D}(v)$ edges in A which are incident to v. Now we define a bijection $\psi : \mathsf{S}(G - D, f) \to \mathsf{S}(G - D, f)$ as follows: $\psi(u(i)) = v(j)$ if $(u, v) = e_\ell$ such that there are exactly $i - 1$ edges from $\{e_1, \ldots, e_{\ell-1}\}$ are incident on u and there are exactly $j - 1$ edges from $\{e_1, \ldots, e_{\ell-1}\}$ are incident on v. That is, we traverse the edges $e_1, \ldots, e_r$ from left to right and find the i^{th} edge incident to u, say $e_\ell = (u, v)$, and then we assign it $v(j)$, if there are exactly $j - 1$ edges incident to v present among $\{e_1, \ldots, e_{\ell-1}\}$.

▶ **Claim 7.** $\psi : \mathsf{S}(G - D, f) \to \mathsf{S}(G - D, f)$ *is a proper deficiency map.*

Proof. By the definition of ψ, if $\psi(u(i)) = v(j)$ then $\psi(v(j)) = u(i)$, and so ψ is an involution. Since $G - D + A$ is a simple graph, for any $u \in V(G)$, $(u, u) \notin E_\psi$. Now we need to show that E_ψ is not a multiset. Suppose not, then there exists $u, v \in V(G)$ such that $\psi(u(i_1)) = v(j_1)$ and $\psi(u(i_2)) = v(j_2)$ for some $i_1 \neq i_2$ and $j_1 \neq j_2$. This implies that there exist $e_i, e_j \in A$, $i \neq j$ such that $e_i = e_j = (u, v)$. This contradicts the fact that $G - D + A$ is a simple graph. Since A is disjoint from $E(G)$, $E_\psi \cap E(G) = \emptyset$. ◀

$(\Leftarrow)$ Let $D \subseteq E(G)$ be a nice deletion set. We need to show that we can find $A \subseteq \overline{E(G)}$ such that $|A| = k - |D|$ and $G - D + A$ is a solution to EECG. Properties (i) and (ii) imply that $d_{G-D}(v) \leq f(v)$ for all $v \in V(G)$ and $|\mathsf{S}(G - D, f)| = 2(k - |D|)$. Due to the property (v) in the definition of nice deletion set, we know that there exists a proper deficiency map $\psi : \mathsf{S}(G - D, f) \to \mathsf{S}(G - D, f)$. Define $A_1 = E_\psi$. By the definition of E_ψ, $|A_1| = |E_\psi| = |\mathsf{S}(G - D, f)|/2 = k - |D|$. Also note that $A_1 \cap E(G) = \emptyset$ because ψ is a proper deficiency map. Now consider the graph $G_1 = G - D + A_1$. Note that G_1 is simple graph because ψ is a proper deficiency map. Also by the definition of E_ψ, $d_{G-D+A_1}(v) = f(v)$ for all $v \in V(G)$. So G_1 satisfies the degree constraints. If G_1 is connected then $D \cup A_1$ is a solution to EECG. Thus, we assume that G_1 is not connected. In what follows we give an iterative procedure that finds the desired A. Suppose we are in i^{th} iteration and we have a set A_i such that $G - D + A_i$ satisfies all degree constraints but $G - D + A_i$ is not connected. Then in the next iteration we find another addition set of size $k - |D|$, say A_{i+1}, such that $G - D + A_{i+1}$ satisfies all degree constraints and $G - D + A_{i+1}$ has strictly less number of connected components than in $G - D + A_i$. The procedure is started with $A_1 = E_\psi$. Let $i \geq 1$ and we have A_i such that $G - D + A_i$ satisfies all degree constraints but $G - D + A_i$ is not connected. Since $|A_i| = k - |D|$ and $G - D$ has at most $k - |D| + 1$ connected components, $G_i = G - D + A_i$ has a component F such that there is an edge

$(u_1, v_1) \in A_i$ with the property that $u_1, v_1 \in V(F)$ and (u_1, v_1) is not a bridge in F. Let F' be another connected component in G_i. Since each connected component in $G - D$ contains a vertex $w \in V(G)$ such that $d_{G-D}(w) < f(w)$, there exists an edge $(u_2, v_2) \in A_i$ such that $u_2, v_2 \in V(F')$. Now let $A_{i+1} = (A_i \setminus \{(u_1, v_1), (u_2, v_2)\}) \cup \{(u_1, u_2), (v_1, v_2)\}$. Observe that $G_{i+1} = G - D + A_{i+1}$ is a simple graph with strictly less connected component than in G_i and $d_{G_{i+1}}(v) = f(v)$ for all $v \in V(G)$. Observe that when the procedure stops we would have found the desired A.

Given a nice deletion set D, we can find the desired A using the iterative procedure described in the reverse direction of the proof. Clearly, this procedure can be implemented in polynomial time. This completes the proof of the lemma. ◀

Thus, our problem reduces to finding a nice deletion set $D \subseteq E(G)$ and an accompanying proper deficiency map ψ on $\mathsf{S}(G - D, f)$, if it exists, using dynamic programming (DP).

Towards the states of dynamic programming algorithm. So how to find a nice deletion set D? Throughout this section we will work with a *hypothetical* deletion set D. We partition the vertices of G into Green and Red in the following way. We color $v \in V(G)$ green if $d_G(v) > f(v)$, and red otherwise. Let $E_r = E(G[\mathsf{Red}])$ and $E_g = E(G) \setminus E_r$. We need a quick sanity check. That is, if $\sum_{\{v : d_G(v) \neq f(v)\}} |d_G(v) - f(v)| > 2k$ then we output NO, because in this case any solution to EECG requires more than k edge edits (addition/deletion operations). Now we guess the size $k' \leq k$ of D such that $2k' \geq \sum_{v : d_G(v) > f(v)} d_G(v) - f(v)$. Since D is our hypothetical deletion set, we have that for any $v \in \mathsf{Green}$, the number of edges in D which are incident with v is at least $d_G(v) - f(v)$. Now we guess the number k_1 of edges in D which are incident with only green vertices and the number k_2 of edges in D which are incident with at least one vertex in Red. Note that $k_1 + k_2 = k'$. Also note that the number of ways we can guess (k', k_1, k_2) is at most k^2. Now for every $v \in \mathsf{Green}$, we guess the number of edges in D which are incident with v. In particular, we guess a function $\Phi : \mathsf{Green} \to \mathbb{N}$ such that for all $v \in \mathsf{Green}$ we have that $\Phi(v) \geq d_G(v) - f(v)$ and $\sum_{v \in \mathsf{Green}} \Phi(v) \leq 2k_1 + k_2$. The number of possible such functions Φ is upper bounded by $\mathcal{O}(4^k k)$. From now onwards we will assume that we are given function Φ. In other words we have guessed the function Φ corresponding to the hypothetical solution D.

We start with an intuitive explanation of the structure of the solution that is helpful in designing partial solution for the DP algorithm. Given $D \cup A$, we first define a notion of an alternating walk. An *alternating walk* is a sequence of vertices $u_1, u_2, \ldots, u_\ell$ such that consecutive pairs of vertices $((u_i, u_{i+1}), (u_{i+1}, u_{i+2}))$ either belong to $D \times A$ or $A \times D$ and $\{(u_i, u_{i+1}) \mid 1 \leq i < \ell\}$ is a set (not a multiset). That is, an edge from D is followed by an edge from A or vice-versa. In an *alternating even length closed walk*, $u_1 = u_\ell$ and ℓ is even. One might wonder about the definition of alternating odd length closed walk. For our purposes we will think of them as alternating walks that start and end at the same vertex. Essentially, these will be alternating walks that start and end with the same vertex and the first and the last edge either both belong to D or both belong to A. From now onwards whenever we say an alternating closed walk, we mean an alternating even length closed walk. For every intermediate, i.e. not the endpoint, vertex in an alternating walk or in an alternating closed walk, one of the edges incident with it belongs to D and while the second edge belongs to A. Thus the degree of any vertex is not disturbed by an alternating walk where this vertex is intermediate. We define alternating walks for the following reason. Let $D \cup A$ be a solution of EECG that satisfies Φ. We can think of edges in $D \cup A$ forming *a family $\mathcal{P}$ of edge disjoint alternating walks and alternating closed walks* with the following properties.

- For every vertex $v \in V(G)$ and a set $Z \in \{D, A\}$, we define $\mathsf{apdeg}(\mathcal{P}, Z, v)$ as the number of edges from Z that are incident with v and appear as (i) the first edge in alternating walks from $\mathcal{P}$ that start with v; and (ii) the last edge in alternating walks from $\mathcal{P}$ that end in v. Note that, if there is an alternating walk that both starts and ends in v and the start edge as well as the last edge belong to Z then this walk contributes two to $\mathsf{apdeg}(\mathcal{P}, Z, v)$. For every vertex $v \in \mathsf{Green}$, $\mathsf{apdeg}(\mathcal{P}, D, v) = d_G(v) - f(v)$ and $\mathsf{apdeg}(\mathcal{P}, A, v) = 0$. Furthermore, for every vertex $v \in \mathsf{Red}$, $\mathsf{apdeg}(\mathcal{P}, D, v) = 0$ and $\mathsf{apdeg}(\mathcal{P}, A, v) = f(v) - d_G(v)$. When the number of edges in D which are incident with $v \in \mathsf{Green}$ is greater than $d_G(v) - f(v)$, then the number of times v appear as intermediate vertices in alternating (closed) walks is exactly equal to the number of *excess* edges and these excess edges will not contribute to $\mathsf{apdeg}(\mathcal{P}, D, v)$.
- Every vertex $v \in \mathsf{Green}$, appears as an intermediate vertex in an alternating (closed) walk of $\mathcal{P}$ exactly $\Phi(v) - (d_G(v) - f(v))$ times.

For any solution $\mathcal{P} = \{P_1, P_2, \ldots, P_\alpha\}$, without loss of generality we assume that there is η such that $P_1, \ldots, P_\eta$ are alternating walks and $P_{\eta+1}, \ldots, P_\alpha$ are alternating closed walks. This walk system view allows us to make a dynamic programming algorithm where we can move from one state to another using one edge addition or deletion. In particular, the algorithm works by constructing all alternating walks $P_1, \ldots, P_\eta$ first and then construct alternating closed walks $P_{\eta+1}, \ldots, P_\alpha$. Given a partially constructed walk system we try to append an edge to the current walk we are constructing by adding an edge from $\binom{V(G)}{2}$ to it; or declaring that we are finished with the current walk and move to construct a new walk. During this process we also keep a partial proper deficiency map ψ' such that $E_{\psi'}$ are addition edges in the current partial solution. Thus, a state in the DP algorithm is given by our current guesses and a subset of domain of partial proper deficiency map. For a formal description of our DP table we need the following sets: (a) a multiset set T_m containing $d_G(v) - f(v)$ many copies of v for all $v \in \mathsf{Green}$; (b) a set $T_g = \{v(i) \mid v \in \mathsf{Green}, \Phi(v) > d_G(v) - f(v), i \in [\Phi(v) - (d_G(v) - f(v))]\}$; (c) $T_r = \{v(i) \mid v \in \mathsf{Red}, f(v) > d_G(v), i \in [f(v) - d_G(v)]\}$. We also need to keep two sets X - a multiset and a set Y to take care of local deficiencies that occur while constructing our walk system. Let X be a multiset of size 2. Then for all $u \in X$ and $B \subseteq E(G)$ such that $f(u) > d_{G-B}(u) + X(u) - 1$, define $X_{B,f} = \{u(i) \mid u \in X, f(u) - d_{G-B}(u) - X(u) + 1 \leq i \leq f(u) - d_{G-B}(u), i \in \mathbb{N}\}$.

Now we formally define a notion of partial solutions. Given an instance (G, f, k) we define T_m, T_g and T_r as described earlier. Also, recall that we have k_1, k_2 and Φ. For any $T'_m \subseteq T_m, T'_g \subseteq T_g, T'_r \subseteq T_r, k'_1 \leq k_1, k'_2 \leq k_2, i \leq k$, a multiset X containing elements from $V(G)$ and $Y \subseteq V(G)$ such that $|X| \leq 2, |Y| \leq 1, |X \cup Y| \leq 2$ and $X \cap Y = \emptyset$, we define a family $\mathcal{Q}(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ of subsets of $\binom{V(G)}{2}$ as follows. For any $B \subseteq E(G)$ and $A \subseteq \overline{E(G)}$, $B \cup A \in \mathcal{Q}(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ if the following conditions are met:

(a) $|E(G[\mathsf{Green}]) \cap B| = k'_1$, $|B \setminus E(G[\mathsf{Green}])| = k'_2$ and $|B \cup A| = i$.
(b) For any $v \in \mathsf{Green}$, the number of edges in B which are incident with v is exactly equal to $T'_m(v) + T'_g(v) + X(v)$. That is, for all $v \in \mathsf{Green}$, $|B \cap E_G(v)| = T'_m(v) + T'_g(v) + X(v)$.
(c) $|X_{B,f}| = |X|$ and $X_{B,f} \subseteq S(G - B, f) \setminus T_r$.
(d) $G - B$ has at most $k - k' + 1$ connected components.
(e) There is a proper deficiency map $\psi' : (S(G - B, f) \cup T'_g \cup Y) \setminus (T'_r \cup X_{B,f}) \to (S(G - B, f) \cup T'_g \cup Y) \setminus (T'_r \cup X_{B,f})$ such that $A = E_\psi$. Furthermore, for $w \in Y$, if $\psi'(w) = u(j)$ for some j then $T_m(u) = T'_m(u)$.

For $T'_m \subseteq T_m, T'_g \subseteq T_g, T'_r \subseteq T_r, k'_1 \leq k_1, k'_2 \leq k_2, i \leq k$, a multiset X containing elements from $V(G)$ and $Y \subseteq V(G)$, we say that the tuple $(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ is a *valid* tuple if the following happens.

1. $|X| \leq 2, |Y| \leq 1, |X \cup Y| \leq 2$ and $X \cap Y = \emptyset$
2. For $w \in Y$, $w(j) \notin T'_r$ for all j.
3. If $u(j) \in T'_g$ then for all $0 < j' < j$, $u(j') \in T'_g$.
4. For any $v \in X$, $T_m(v) = T'_m(v)$.

For the correctness of the algorithm, it is enough to focus on partial solutions defined over valid tuples. Now we prove that in fact $\mathcal{Q}(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ is "a correct notion of partial solution".

▶ **Lemma 8** (⋆). *Let (G, f, k) be an* Yes *instance of* EECG *with a solution $D \cup A$ such that $D \subseteq E(G), A \subseteq \overline{E(G)}$, $|D \cap E(G[\mathsf{Green}])| = k_1, |D \setminus E(G[\mathsf{Green}])| = k_2$, $k_1 + k_2 = k'$ and $|D \cap E_G(v)| = \Phi(v)$ for all $v \in \mathsf{Green}$. Let ψ be a proper deficiency map over $S(G - D, f)$ such that $E_\psi = A$. Then for each $i \leq k$, there exists $D' \cup A' \subseteq D \cup A$ and a valid tuple $(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ such that $D' \cup A' \in \mathcal{Q}(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ and there is a proper deficiency map ψ' over $R = (S(G - D', f) \cup T'_g \cup Y) \setminus (T'_r \cup X_{D',f})$ with $E_{\psi'} = A'$.*

Our DP algorithm keeps a table entry $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ for each valid tuple $(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$. The idea is to store a subset of $\mathcal{Q}(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ in the DP table entry $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ which is sufficient to maintain the correctness of the algorithm. The algorithm computes $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ for all valid tuple $(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ and if there exists $D \cup A \in \mathcal{D}[T_m, T_g, \emptyset, k_1, k_2, k, \emptyset, \emptyset]$ such that $D \cup A$ is a solution to EECG, then outputs Yes, otherwise outputs No. In fact one can show the following simple lemma which bounds the number of DP table entries:

▶ **Lemma 9** (⋆). *The number of valid tuples $(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ is at most $2^{\mathcal{O}(k)} n^3$.*

However, the size of family stored at (one table entry) $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ can potentially be $n^{\mathcal{O}(i)}$, thus this algorithm takes time $n^{\mathcal{O}(k)}$. Next we observe that we can prune each table entry to $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ and thus this leads to an FPT algorithm.

Pruning the DP table entry and an FPT algorithm. We need to prune the DP table in a way that we do not change the answer to the given instance (G, f, k). Towards this we show that if some subset we have stored in a DP table entry could lead to a nice deletion set then we do have at least one such set after the pruning operation. Our guessing of k_1, k_2 and Φ allows us to satisfy the properties (i) and (ii) of a nice deletion set. Property (iii) of nice deletion sets implies that D is an independent set in the matroid $M_G(\ell)$, the ℓ-elongation of the co-graphic matroid M_G associated with G, where $\ell = |E(G)| - |V(G)| + k - |D| + 1$. Thus by only considering those D which are independent sets in $M_G(\ell)$ we ensure that property (iii) of nice deletion sets is satisfied. Now consider the property (v) of the nice deletion set, i.e, there exists a proper deficiency map $\psi : S(G - D, f) \to S(G - D, f)$. Our objective is to get a set $D \cup A$ such that there is a proper deficiency map ψ over $S(G - D, f)$ with the property that $E_\psi = A$, along with other properties as well. Let $D_1 \cup A_1, D_2 \cup A_2$ be two partial solutions belonging to the same equivalence class where $D_1, D_2 \subseteq E(G)$ and $A_1, A_2 \subseteq \overline{E(G)}$. Suppose $D' \subseteq E(G)$, $A' \subseteq \overline{E(G)}$, $(D_1 \cup D') \bigcup (A_1 \cup A')$ is a solution and $A_2 \cap A' = \emptyset$. Since $D_1 \cup A_1$, $D_2 \cup A_2$ belongs to same equivalence class and $A_2 \cap A'$ is disjoint, there is a proper deficiency map ψ' over $S(G - (D_2 \cup D'), f)$ such that $E_{\psi'} = A_2 \cup A'$. To take care of the disjointness property between the current addition set and the future addition set while doing the DP, we view the addition set A of the solution as an independent set in a uniform matroid over the universe $\overline{E(G)}$. Let $U_{m',k-k'}$ be a uniform matroid with ground set $\overline{E(G)}$, where $m' = |\overline{E(G)}|$. From the definition of $U_{m',k-k'}$, every set A of size at most $k - k'$ is independent in $U_{m',k-k'}$. We have already explained that we view the deletion set D as an independent set in $M_G(\ell)$ where $\ell = |E(G)| - |V(G)| + k - k' + 1$. To view the solution set $D \cup A$ as an independent set in a matroid, we consider the direct sum $M = M_G(\ell) \oplus U_{m',k-k'}$ of two matroids. In M, a set I is an independent set if and only if $I \cap E(G)$ is an independent set in $M_G(\ell)$ and $I \cap \overline{E(G)}$ is an independent set in $U_{m',k-k'}$. This ensures that any solution $D \cup A$ is an independent set in M. By viewing any solution of the problem as an independent set in a matroid M (which is linear by Proposition 2), we can use the q-representative families, Theorem 5, to prune the table size to $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$.

However, we still need to ensure that property (iv) of nice deletion set is satisfied. In what follows we explain how we achieve this. The following lemma helps us to satisfy property (iv) partially.

▶ **Lemma 10** ($\star$). *Let F be a connected component in the graph $G[\mathsf{Red}]$ and $D' \subseteq E(G)$. If at least one edge in D' is incident to a vertex in $V(F)$, then for any connected component C in $G - D'$ such that $V(C) \cap V(F) \neq \emptyset$ there is a vertex $v \in V(C) \cap \mathsf{def}(G - D', f)$.*

Now we explain how Lemma 10 is useful in satisfying property (iv) partially. Let $\mathcal{C}$ be the set of connected components in G such that for each vertex v in the component, $d_G(v) = f(v)$.

$$\mathcal{C} = \{C \mid C \text{ is a connected component in } G \wedge \forall v \in V(C), d_G(v) = f(v)\}.$$

Let D_1 and D_2 be deletion sets corresponding to two partial solutions such that for all $C \in \mathcal{C}$, $D_1 \cap E(C) \neq \emptyset$ if and only if $D_2 \cap E(C) \neq \emptyset$. Suppose there is a set $D' \subseteq E(G)$ such that $D_1 \cup D'$ is a nice deletion set. Then we can show that any connected component F in $G - (D_2 \cup D')$ containing only red vertices will have a deficient vertex. Essentially due to Lemma 10, if we partition our partial solutions based on how these partial solutions hit the edges from $\mathcal{C}$ and keep at least one from each equivalence class property (iv) of nice deletion set will be satisfied partially. But this only allows us to take care of connected components containing only red vertices. Now we explain how we can ensure property (iv) for the connected components containing vertices from Green as well.

To achieve this we will prove that corresponding to every deletion set D of a solution, there is a "witness" of $\mathcal{O}(k)$ sized subset of edges whose disjointness from D will ensure property (iv) of nice deletion sets. That is, these witnesses depend on solutions; the witness for solution D could be different from the witness for solution D^*. Even then, these witnesses allow us to satisfy property (iv). In order to avoid this witness being picked in a deletion set D, that is to keep this witness non deletable, we use color coding in our algorithm on top of representative family based pruning of table entries. Towards that we define a weight function w on $E(G)$ as follows.

$$\mathrm{w}((u,v)) = \begin{cases} 0 & \text{if } u, v \in \mathsf{Red} \\ 1 & \text{otherwise} \end{cases}$$

For any subset $S \subseteq E(G)$, $\mathrm{w}(S) = \sum_{e \in S} \mathrm{w}(e)$. The next lemma is crucial for our approach as this not only defines the witness but also gives an upper bound on its size.

▶ **Lemma 11.** *Let $\mathsf{Green} = \{v_1, v_2, \ldots, v_\eta\}, \eta \leq 2k$. Let $D \subseteq E(G)$ such that for any connected component F in $G - D$, $V(F) \cap \mathsf{def}(G - D, f) \neq \emptyset$. Then there exist paths $P_1, \ldots, P_\eta$ such that for all i, P_i is a path in $G - D$ from v_i to a vertex in $\mathsf{def}(G - D, f)$, and $\mathrm{w}(\bigcup_i E(P_i)) \leq 6k$ where $\bigcup_i E(P_i)$ is the set of edges in the paths $P_1, \ldots P_\eta$.*

Proof. We construct $P_1, \ldots, P_\eta$ with the required property. Pick an arbitrary vertex $u_1 \in \mathsf{def}(G - D, f)$ such that v_1 and u_1 are in the same connected component in $G - D$. Let P_1 be a smallest weight path according to weight function w, from v_1 to u_1 in $G - D$. Now we explain how to construct P_i, given that we have already constructed paths $P_1, \ldots, P_{i-1}$. Pick an arbitrary vertex $u_i \in \mathsf{def}(G - D, f)$ such that v_i and u_i are in the same connected component in $G - D$. Let P be a smallest weight path from v_i to u_i in $G - D$. If P is vertex disjoint from $P_1, \ldots, P_{i-1}$, then we set $P_i = P$. Otherwise, let x be the first vertex in P such that $x \in V(P_1) \cup \ldots \cup V(P_{i-1})$. Let $x \in P_j$ where $j < i$. Let $P = P'P''$ such that P' ends in x and P'' starts at x. Let $P_j = P'_jP''_j$ such that P'_j ends in x and P''_j starts at x. Now we set $P_i = P'P''_j$. Note that P_i is a path in $G - D$ from v_i to a vertex in $\mathsf{def}(G - D, f)$.

Now we claim that $\mathrm{w}(\bigcup_{i=1}^{\eta} E(P_i)) \leq 6k$. Towards the proof we need to count that $|(\bigcup_{i=1}^{\eta} E(P_i)) \cap \mathrm{w}^{-1}(1)| \leq 6k$. We assign each vertex v in $\bigcup_{i=1}^{\eta} V(P_i)$ to the smallest indexed path P_j such that $v \in V(P_j)$. That is, v is assigned to P_j, if $v \in V(P_j)$ and $v \notin (\bigcup_{i=1}^{j-1} V(P_i))$. Note that each vertex in $\bigcup_{i=1}^{\eta} V(P_i)$ is assigned to a unique path. Consider the edge set $A^* \subseteq (\bigcup_{i=1}^{\eta} E(P_i)) \cap \mathrm{w}^{-1}(1)$ as follows. An edge $e = (u, v)$ belongs to A^* if $\mathrm{w}(e) = 1, e \in E(P_j)$, and vertices u and v are assigned to path P_j for some j. Observe that each edge $e \in A^*$ has at least one end point in Green. Since each vertex is assigned to exactly one path, each vertex in a path has degree at most 2 and $|\mathsf{Green}| \leq 2k$, we have that $|A^*| \leq 4k$.

Now we show that there exists sets $\emptyset = B_1 \subseteq B_2 \subseteq \ldots B_\eta$ such that $(\bigcup_{i=1}^{j} E(P_i)) \cap \mathrm{w}^{-1}(1) \subseteq A^* \cup B_j$ and $|B_j| \leq j$. We prove the statement using induction on j. For $j = 1$, we know that $(\bigcup_{i=1}^{j} E(P_i)) \cap \mathrm{w}^{-1}(1) \subseteq A^*$. Thus the statement is true. Now suppose the statement is true for $j - 1$. Consider any path P_j. If the vertices in P_j are disjoint from $\bigcup_{i}^{j-1} V(P_i)$, then all the weight one edges in $E(P_j)$ are counted in A^*. So we can set $B_j = B_{j-1}$ and the statement is true. Otherwise by the construction of P_j, we have that $P_j = P'_jP''_j$ and there exists $r < j$ such that $P_r = P'_rP''_j$. Let (u_1, u_2) be the last edge in P'_j. Note that all the weight one edges in $E(P''_j)$ are counted in $A^* \cup B_{j-1}$ and all the weight one edges in $E(P'_j) \setminus \{(u_1, u_2)\}$ are counted in A^*. In this case we set $B_j = B_{j-1}$ if $\mathrm{w}((u_1, u_2) = 0)$ and $B_j = B_{j-1} \cup \{(u_1, u_2)\}$ otherwise. This implies that $|(\bigcup_{i=1}^{\eta} E(P_i)) \cap \mathrm{w}^{-1}(1)| \leq 6k$. This concludes the proof. ◀

Recall that $E_r = E(G[\mathsf{Red}])$ and $E_g = E(G) \setminus E_r$. Note that in Lemma 11, the weight of each edge in E_g is 1 and the weight of each edge in E_r is 0. By Lemma 11, we have that if D is a nice deletion set, then there exists $E' \subseteq E_g$ of cardinality at most $6k$ such that E' witnesses that each connected component of $G - D$ containing at least one vertex from Green, will also contain a vertex from $\mathsf{def}(G - D, f)$. We call such an edge set E' as ***certificate*** of D. Now we explain how Lemma 11 helps us to satisfy property (iv) of nice deletion sets for components containing at least one vertex from Green. Let $\mathsf{Green} = \{v_1, \ldots, v_\eta\}$ and $D_1 \cup D'$ be a deletion set corresponding to a solution. By Lemma 11 we know that there are paths $P_1, \ldots, P_\eta$ such that the total number of edges from E_g among these paths is bounded by $6k$, and each path P_i is from v_i to a vertex in $\mathsf{def}(G - (D_1 \cup D'), f)$. Suppose we color the edges in E_g with black and orange such that the coloring guarantees that all the edges in $E_g \cap (\bigcup_{i=1}^{\eta} E(P_i))$ are colored black and all the edges in $E_g \cap (D_1 \cup D')$ are colored orange. Assume that we are going to find a nice deletion set which does not contains black color edges. Let D_2 be a deletion set corresponding to a partial solution. Also for a vertex $v_i \in \mathsf{Green}$, there is path from v_i to a vertex in $\mathsf{def}(G - D_1, f)$ in the graph $G - D_1$ which does not contain any orange colored edge if and only if there is path from v_i to a vertex in $\mathsf{def}(G - D_2, f)$ in the graph $G - D_2$ which does not contain any orange colored edge. We can show that any connected component in $G - (D_2 \cup D')$ containing a vertex from Green will contain a vertex from $\mathsf{def}(G - (D_2 \cup D'), f)$. Essentially by Lemma 11 we get the following. Suppose we take all partial solutions corresponding to a DP table entry (or a subset of it) and now we partition these partial solutions based on which all green vertices have found their deficient vertex currently (there are $2^{|\mathsf{Green}|}$ such partitions), then it is enough to keep a partial solution from each class. Furthermore, suppose $\mathcal{A}$ corresponds to partial solutions with respect to one particular subset of Green and we have kept a set D_1 in $\mathcal{A}$ and deleted rest of the partial solutions from $\mathcal{A}$ (say one of the partial solution we threw out was D_2). Then, if there is D' such that $D_2 \cup D'$ is a solution, then all the connected components in $G-(D_1 \cup D')$ containing at least one green vertex will have a deficient vertex. Just a word of caution that in our actual algorithm in fact we keep a subset of $\mathcal{A}$ of size $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ so that we can also take care of all other properties of a nice deletion set.

We have explained how we will ensure each of the individual properties of a nice deletion set. Now we design a randomized FPT algorithm for the problem. Later we derandomize the algorithm. The algorithm is a DP algorithm in which we have DP table entries indexed exactly in the same way as in the case of the XP algorithm. But instead of keeping $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$, we store a small representative family of $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ which is enough to maintain the correctness of the algorithm. The algorithm uses both color coding and representative family techniques. We have explained that we use color coding to separate the proposed deletion set from its certificate mentioned in Lemma 11. Now our algorithm works with the edge colored graph (edges in E_g are colored black or orange) and output a nice deletion set D, with the property that $D \cap E_g \subseteq E_o$, if there exists such a deletion set. Note that the edges in E_r is uncolored.

Recall that $\mathcal{C}$ is the set of connected components in G such that for each vertex v in the component, $d_G(v) = f(v)$. Now we define a family $\mathcal{J}$ of functions as,

$$\mathcal{J} = \{g : \mathsf{Green} \cup \mathcal{C} \to \{0,1\}\}.$$

Now we explain how to reduce the size of $\mathcal{D}[T'_g, T'_r, k'_1, k'_2, i, X, Y]$. We say a partial solution $B \in \mathcal{D}[T'_g, T'_r, k'_1, k'_2, i, X, Y]$ is *properly colored*, if $B \cap E_b = \emptyset$. Since our objective is to find out a nice deletion set disjoint from E_b, we delete all partial solutions which contains an edge from E_b. That is, if $B \in \mathcal{D}[T'_g, T'_r, k'_1, k'_2, i, X, Y]$ and $B \cap E_b \neq \emptyset$, then we delete B from $\mathcal{D}[T'_g, T'_r, k'_1, k'_2, i, X, Y]$. So now onwards we assume that for each $B \in \mathcal{D}[T'_g, T'_r, k'_1, k'_2, i, X, Y]$, $B \cap E_b = \emptyset$. Further pruning of the DP table entry $\mathcal{D}[T'_g, T'_r, k'_1, k'_2, i, X, Y]$ is discussed below.

For each valid tuple $(T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y)$ we compute a representative partial solutions for $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ in the increasing order of i and store it instead of $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$. Now we explain how to compute it. First we compute a subfamily $\mathcal{S}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ of $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ using the DP table entries computed for value $i-1$ and deleting all partial solutions which contain edges from E_b. Now we partition $\mathcal{S}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ according to the refinement of each function in $\mathcal{J}$. That is $\mathcal{S}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y] = \bigcup_{g \in \mathcal{J}} \mathcal{A}_g$ where $\mathcal{A}_g$ is defined as follows. For each $g \in \mathcal{J}$ and $S \cup R \in \mathcal{S}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ where $S \in E(G)$ and $R \in \overline{E(G)}$, $S \cup R \in \mathcal{A}_g$ if the following happens.

(i) For any $v \in \mathsf{Green}$, $g(v) = 1$ if and only if there exists a path from v to a vertex in $\mathsf{def}(G-S, f)$ in $G[E_b \cup (E_r \setminus S)]$ (checking whether there is a witness path that do not use edges in E_o).

(ii) For any $C \in \mathcal{C}$, $g(C) = 1$ if and only if $S \cap E(C) \neq \emptyset$.

Recall that any set $S \cup R \in \mathcal{S}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$ is an independent set of size i in M. Now we compute $\widehat{\mathcal{A}}_g \subseteq^{k-i}_{rep} \mathcal{A}_g$ using Theorem 5. Then we set

$$\widehat{\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]} = \bigcup_{g \in \mathcal{J}} \widehat{\mathcal{A}}_g$$

and store it instead of $\mathcal{D}[T'_m, T'_g, T'_r, k'_1, k'_2, i, X, Y]$. We can show the correctness of algorithm even after pruning the DP table entries and our algorithm runs in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ with success probability at least $(1 - \frac{1}{e})$. Our algorithm can be derandomized using $(n, 7k)$-universal sets [16].

References

1 Jin Akiyama and Mikio Kano. Factors and factorizations of graphs – a survey. *Journal of Graph Theory*, 9(1):1–42, 1985.

2 Noga Alon, Daniel Lokshtanov, and Saket Saurabh. Fast FAST. In *Proceedings of the 36th International Colloquium of Automata, Languages and Programming (ICALP)*, volume 5555 of *Lecture Notes in Comput. Sci.*, pages 49–58. Springer, 2009.

3 R. P. Anstee. An algorithmic proof of Tutte's f-factor theorem. *J. Algorithms*, 6(1):112–131, 1985.

4 Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 460–469, 2012.

5 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.

6 Herbert Fleischner. *Eulerian Graphs and Related Topics, Part 1, Volume 1.* Annals of Discrete Mathematics 45. Elsevier, Amsterdam, 1990.

7 Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM J. Computing*, 42(6):2197–2216, 2013.

8 Petr A. Golovach. Editing to a connected graph of given degrees. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 8635 of *Lecture Notes in Comput. Sci.*, pages 324–335. Springer, 2014.

9 Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–169. IEEE, 2011.

10 T Kirkman. On a problem in combinatorics. *Cambridge and Dublin Math. J .*, 2:191–204, 1847.

11 Mekkia Kouider and Preben D. Vestergaard. Connected factors in graphs – a survey. *Graphs and Combinatorics*, 21(1):1–26, 2005.

12 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 922–934. Springer, 2015.

13 László Lovász and Michael D. Plummer. *Matching theory.* AMS Chelsea Publishing, Providence, RI, 2009.

14 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computers Science*, 410(44):4471–4479, 2009.

15 Luke Mathieson and Stefan Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *J. Comput. Syst. Sci.*, 78(1):179–191, 2012.

16 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 182–191. IEEE, 1995.

17 James G. Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford university press, 2nd edition, 2010.

18 Julius Petersen. Die Theorie der regulären graphs. *Acta Math.*, 15(1):193–220, 1891. doi:10.1007/BF02392606.

19 Michael D. Plummer. Graph factors and factorization: 1985-2003: A survey. *Discrete Mathematics*, 307(7-8):791–821, 2007.

20 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 887–898. ACM, 2012.