On the Complexity Landscape of Connected f-Factor Problems*

Robert Ganian[1], N. S. Narayanaswamy[2], Sebastian Ordyniak[3], C. S. Rahul[4], and M. S. Ramanujan[5]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
rganian@gmail.com
2 **Indian Institute of Technology Madras, Chennai, India**
swamy@cse.iitm.ac.in
3 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
sordyniak@gmail.com
4 **Indian Institute of Technology Madras, Chennai, India**
rahulcs@cse.iitm.ac.in
5 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
ramanujan@ac.tuwien.ac.at

Abstract

Given an n-vertex graph G and a function $f : V(G) \to \{0, \ldots, n-1\}$, an f-factor is a subgraph H of G such that $deg_H(v) = f(v)$ for every vertex $v \in V(G)$; we say that H is a *connected f-factor* if, in addition, the subgraph H is connected. A classical result of Tutte (1954) is the polynomial time algorithm to check whether a given graph has a specified f-factor. However, checking for the presence of a *connected* f-factor is easily seen to generalize Hamiltonian Cycle and hence is NP-complete. In fact, the Connected f-Factor problem remains NP-complete even when $f(v)$ is at least n^ϵ for each vertex v and $\epsilon < 1$; on the other side of the spectrum, the problem was known to be polynomial-time solvable when $f(v)$ is at least $\frac{n}{3}$ for every vertex v.

In this paper, we extend this line of work and obtain new complexity results based on restricting the function f. In particular, we show that when $f(v)$ is required to be at least $\frac{n}{(\log n)^c}$, the problem can be solved in quasi-polynomial time in general and in randomized polynomial time if $c \leq 1$. We also show that when $c > 1$, the problem is NP-intermediate.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, G.2.2 Graph Theory

Keywords and phrases f-factors, connected f-factors, quasi-polynomial time algorithms, randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.41

1 Introduction

f-factors are a fundamental concept of graph theory dating back to the 19th century, specifically to the work of Petersen [13]. In modern terminology, an f-factor is defined as a spanning subgraph which satisfies degree constraints (given in terms of the degree function f) placed on each vertex of the graph [22]. Some of the most fundamental results on f-factors were obtained by Tutte, who gave sufficient and necessary conditions for the existence of f-factors [19]. Tutte also developed a method for reducing the existence of an f-factor

* The authors acknowledge support by the Austrian Science Fund (FWF, project P26696). Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

Table 1 The table depicting the known as well as new results in the complexity landscape of theConnected f-Factor problem.

$f(v) \geq$	Complexity Class
$n^{\epsilon}, \forall \epsilon > 0$	NPC [4]
$\frac{n}{\text{polylog}n}$	QP (Theorem 2)
$\frac{n}{\log n}$	RP (Theorem 3)
$\frac{n}{c}, \forall c \geq 3$	P (Theorem 1)

to the existence of a perfect matching [18], which gives a straightforward polynomial time algorithm for the problem of deciding whether an f-factor exists. There are also several detailed surveys on f-factors of graphs, for instance by Chung and Graham [3], Akiyama and Kano [1].

Aside from work on general f-factors, substantial attention has been devoted to the variant of f-factors where we require the subgraph to be connected (see for instance the survey articles by Kouider and Vestergaard [20] and Plummer [16]). Unlike the general variant, deciding the existence of a connected f-factor (the Connected f-Factor problem) is an NP-complete problem [6, 2]. It is easy to see that Connected f-Factor generalizes Hamiltonian Cycle (set $f(v) = 2$ for every vertex v), and even the existence of a deterministic single-exponential (in the number of vertices) algorithm is open for the problem [15].

The NP-completeness of this problem has motivated several authors to study the Connected f-Factor problem for various restrictions of the function f. Cornelissen et al. [4] showed that Connected f-Factor remains NP-complete even when $f(v)$ is at least n^{ϵ} for each vertex v and any constant ϵ between 0 and 1. At the other end of the spectrum, it has been shown that the problem is polynomial-time solvable when $f(v)$ is at least $\frac{n}{3}$ [12] for every vertex v. Aside from these two fairly extreme cases, the complexity landscape of this problem based on lower bounds on the function f has largely been left uncharted.

Our results and techniques. In this paper, we provide new results for solving the Connected f-Factor problem based on lower bounds on the range of f, both positive and negative. Since we will study the complexity landscape of Connected f-Factor through the lens of the function f, it will be useful to formally capture bounds on the function f via an additional "bounding" function g. To this end, we introduce the Connected g-Bounded f-Factor problem below:

Connected g-Bounded f-Factor
Instance: An n-vertex graph G and a mapping $f : V \to \mathbb{N}$ such that $f(v) \geq \frac{n}{g(n)}$.
Task: Find a connected subgraph H of G such that each vertex $v \in V(G)$ satisfies $d_H(v) = f(v)$.

First, we obtain a polynomial time algorithm for Connected f-Factor when $f(v)$ is at least $\frac{n}{c}$ for every vertex v and any constant $c \geq 1$. This result improves upon the previously known polynomial-time algorithm for the case when $f(v)$ is at least $\frac{n}{3}$. This is achieved thanks to a novel approach for the problem, which introduces a natural way of converting one f-factor to another by exchanging a set of edges. Here we formalize this idea using the notion of *Alternating Circuits*. These allow us to focus on a simpler version of the problem,

where we merely need to ensure connectedness across a coarse partition of the vertex set. Furthermore, we extend this approach to obtain a quasi-polynomial time algorithm for the Connected f-Factor problem when $f(v)$ is at least $\frac{n}{polylog(n)}$. To be precise, we prove the following two theorems (see the next page for an explanation of the function g used in the formal statements).

▶ **Theorem 1.** *For every function $g(n) = \mathcal{O}(1)$,* Connected g-Bounded f-Factor *can be solved in polynomial time.*

▶ **Theorem 2.** *For every $c > 0$ and function $g(n) = \mathcal{O}((\log n)^c)$,* Connected g-Bounded f-Factor *can be solved in time $n^{(\log n)^{\mathcal{O}(1)}}$.*

Second, we build upon these new techniques to obtain a *randomized polynomial-time* algorithm which solves Connected f-Factor in the more general case when $f(v)$ is lower-bounded by $\frac{n}{\mathcal{O}(\log n)}$ for every vertex v. For this, we also require algebraic techniques that have found several applications in the design of fixed-parameter and exact algorithms for similar problems [5, 21, 7, 14]. Precisely, we prove the following theorem.

▶ **Theorem 3.** *For every function $g(n) = \mathcal{O}(\log n)$,* Connected g-Bounded f-Factor *can be solved in polynomial time with constant error probability.*

We remark that the randomized algorithm in the above theorem has one-sided error with 'Yes' answers always being correct. Finally, we also obtain a lower bound result for Connected f-Factor when $f(n)$ is at least $\frac{n}{(\log n)^c}$ for $c > 1$. Specifically, in this case we show that the problem is in fact NP-intermediate, assuming the Exponential Time Hypothesis [8] holds. Formally speaking, we prove the following theorem.

▶ **Theorem 4.** *For every $c > 1$, for every $g(n) = \Theta((\log n)^c)$,* Connected g-Bounded f-Factor *is neither in* P *nor* NP*-hard unless the Exponential Time Hypothesis fails.*

We detail the known as well as new results on the complexity landscape of Connected f-Factor in Table 1.

Organization of the paper. After presenting required definitions and preliminaries in Section 2, we proceed to the key technique and framework used for our algorithmic results, which forms the main part of Section 3. In the next Section 3.2, we obtain both of our deterministic algorithms, which are formally given as Theorem 1 (for the polynomial-time algorithm) and Theorem 2 (for the quasipolynomial-time algorithm). Section 4 then concentrates on our randomized polynomial-time algorithm, presented in Theorem 3. Finally, Section 5 focuses on ruling out (under established complexity assumptions) both NP-completeness and inclusion in P for all polylogarithmic functions g where we do not already have a polynomial-time algorithm.

2 Preliminaries

2.1 Basic Definitions and Graphs

We use standard definitions and notations from West [22]. $d_G(v)$ denotes the **degree** of a vertex v in a graph G. A **component** in a graph is a maximal subgraph that is connected. Note that the set of components in a graph uniquely determines a partition of the vertex set. A **circuit** in a graph is a cyclic sequence $v_0, e_1, v_1, \cdots, e_k, v_k = v_0$ where each e_i is of

the form $\{v_{i-1}, v_i\}$ and occurs at most once in the sequence. An **Eulerian circuit** in a graph is a circuit in which each edge in the graph appears exactly once.

Let V' be a subset of the vertices in the graph G. The **induced subgraph** $G[V']$ is the graph over vertex set V' containing all the edges in G whose endpoints are both in V'.

Given a partition $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_r\}$ of the vertex set of G, the graph $G/\mathcal{Q}$ is constructed as follows: The vertex set of $G/\mathcal{Q}$ is $\mathcal{Q}$. Corresponding to each edge (u, v) in G where u in Q_i, v in Q_j, $i \neq j$, there exists an edge (Q_i, Q_j) in $G/\mathcal{Q}$. Thus, $G/\mathcal{Q}$ is a multigraph without loops. For a subgraph G' of G, we say G' **connects** a partition $\mathcal{Q}$ if $G'/\mathcal{Q}$ is connected. A **refinement** $\mathcal{Q}'$ of a partition $\mathcal{Q}$ is a partition of V where each part Q' in $\mathcal{Q}'$ is a subset of some part Q in $\mathcal{Q}$. This notion of partition refinement was used, e.g., by Kaiser [9]. A **spanning tree of** $G/\mathcal{Q}$ refers to a subgraph T of G with $|\mathcal{Q}|$-1 edges that connects $\mathcal{Q}$.

2.2 Colored Graphs, (Minimal) Alternating Circuits, and f-Factors

Recall the definition of the Connected g-Bounded f-Factor problem. Given an instance (G, f) of Connected g-Bounded f-Factor, a subgraph H of G is an *f-factor* if $d_H(v) = f(v)$ for each $v \in V(G)$; sometimes we also use the term f-factor to refer to $E(H)$.

A colored graph G is one in which each edge is assigned a color from the set $\{red, blue\}$. In a colored graph G, we use R and B to denote subgraphs of G whose edges are the set of red edges ($E(R)$) and blue edges ($E(B)$) of G, respectively, and $V(R) = V(B) = V(G)$. We will use this coloring in our algorithm to distinguish between edge sets of two distinct f-factors of the same graph G. A crucial computational step in our algorithms is to consider the symmetric difference between edge sets of two distinct f-factors and perform a sequence of edge exchanges preserving the degree of each vertex. The following definition is used extensively in our algorithms.

▶ **Definition 5.** A colored graph A is called an **alternating circuit** if there exists an Eulerian circuit in A where every pair of consecutive edges are of different colors.

Clearly, an alternating circuit has an even number of edges and is connected. Furthermore, $d_R(v) = d_B(v)$ for each v in A, where $d_R(v)$ and $d_B(v)$ denote the number of red and blue edges incident to v, respectively. A **minimal alternating circuit** M is an alternating circuit where each vertex v in M has at most two red edges incident to v. By definition, an empty graph is an alternating circuit.

We will also use the following lemma.

▶ **Lemma 6** ([12, Lemma 5]). *Let $S \subseteq E(G)$. An f-factor H containing all the edges in S, if one exists, can be computed in polynomial time.*

3 A Generic Algorithm for Finding Connected g-Bounded f-Factors

Our goal in this section is to present a generic algorithm for Connected g-Bounded f-Factor. In particular, we will in a certain sense reduce the question of solving Connected g-Bounded f-Factor to solving a related problem which we call Partition Connector. This can be viewed as a relaxed version of the original problem, since instead of a connected f-factor it merely asks for an f-factor which connects a specified partitioning of the vertex set. A formal definition is provided below.

Partition Connector
Instance: An n-vertex graph G, $f : V \to \mathbb{N}$, and a partition $\mathcal{Q}$ of $V(G)$.
Task: Find an f-factor of G that connects $\mathcal{Q}$.

The algorithms for solving Partition Connector will then be presented in the later parts of this article—specifically, a deterministic algorithm that runs in quasipolynomial time whenever $g(n) = \mathcal{O}(\texttt{polylog}(n))$ (Section 3.2) and a randomized polynomial-time algorithm for the case when $g(n) = \mathcal{O}(\log n)$ (Section 4).

The majority of this section is devoted to proving the key Theorem 7 stated below, which establishes the link between Partition Connector and Connected g-Bounded f-Factor.

▶ **Theorem 7.** **(a)** *Let $g(n) = \mathcal{O}(\texttt{polylog}(n))$. If there is a deterministic algorithm running in time $\mathcal{O}(n^{2(|\mathcal{Q}|-1)})$ for* Partition Connector*, then there is a deterministic quasi-polynomial time algorithm for* Connected g-Bounded f-factor *with running time $\mathcal{O}(g(n) \cdot n^{2g(n)})$.*

(b) *Let $g(n) = \mathcal{O}(\log n)$. If there is a randomized algorithm running in time $\mathcal{O}(2^{|\mathcal{Q}|}n^{\mathcal{O}(1)})$ with error probability $\mathcal{O}(|\mathcal{Q}|^2/n^2)$ for* Partition Connector*, then there is a randomized polynomial-time algorithm for* Connected g-Bounded f-factor *that has a constant error probability.*

3.1 A generic algorithm for Connected g-Bounded f-Factor

The starting point of our generic algorithm is the following observation.

▶ **Observation 8.** Let G be an undirected graph and f be a function $f : V \to \mathbb{N}$. The graph G has a connected f-factor if and only if for each partition $\mathcal{Q}$ of the vertex set V, there exists an f-factor H of G that connects $\mathcal{Q}$.

We remark that for the running time analysis for our generic algorithm we will assume that we are only dealing with instances of Connected g-Bounded f-Factor, where the number of vertices exceeds $6g(n)^4$. As $g(n)$ is in $\mathcal{O}(\texttt{polylog}(n))$, this does not reduce the applicability of our algorithms, since there is a constant n_0 such that $n \geq 6g(n)^4$ for every $n \geq n_0$; because $g(n)$ is part of the problem description, n_0 does not depend on the input instance. Consequently, we can solve instances of Connected g-Bounded f-Factor where $n < n_0$ by brute-force in constant time. We will therefore assume without loss of generality in the following that $n \geq n_0$ and hence $n \geq 6g(n)^4$.

Our algorithm constructs a *maximal* sequence of pairs $(H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ satisfying the following properties:

(M1) Each $\mathcal{Q}_i, 0 \leq i \leq k$ is a partition of the vertex set V, and $\mathcal{Q}_0 = \{V(G)\}$.
(M2) Each $H_i, 0 \leq i \leq k$ is an f-factor of G, and H_i connects $\mathcal{Q}_i$.
(M3) For each $1 \leq i \leq k$, $\mathcal{Q}_i$ is a refinement of $\mathcal{Q}_{i-1}$ satisfying the following:
- **(a)** Each part Y in $\mathcal{Q}_i$ induces a component $H_{i-1}[Y]$ in $H_{i-1}[Q]$, for some Q in $\mathcal{Q}_{i-1}$.
- **(b)** $\mathcal{Q}_i \neq \mathcal{Q}_{i-1}$.

The following lemma links the existence of a connected f-factor to the properties of maximal sequences satisfying (M1)–(M3).

▶ **Lemma 9.** *Let (G, f) be an instance of* Connected g-Bounded f-Factor *and let $(H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ be any maximal sequence satisfying Properties (M1)–(M3). Then G has a connected f-factor if and only if H_k is a connected f-factor of G.*

The above lemma shows that if we had an algorithm for constructing a maximal sequence satisfying (M1)–(M3), then we could solve the f-factor problem by testing whether the last f-factor in that sequence is connected. However, if the number of parts of the partitions $\mathcal{Q}_i$ is allowed to grow to n, then such an algorithm would eventually have to solve the connected f-factor problem. Hence, to employ the idea for an efficient algorithm, we first need to establish an upper bound on the number of parts in any partition $\mathcal{Q}_i$ of a maximal sequence. The following lemma, which shows a lower bound on the minimum degree and hence a lower bound on the size of any part in $\mathcal{Q}_i$, is crucial for estabilishing such an upper bound.

▶ **Lemma 10.** *Let $(H, \mathcal{Q})$, $(H', \mathcal{Q}')$ be two consequitive pairs occuring in a sequence satisfying properties (M1)–(M3). Then there is an f-factor H'' of G connecting $\mathcal{Q}'$ such that $|N_{H''}(v) \cap Q'| \geq |N_H(v) \cap Q'| - 2(|\mathcal{Q}'| - 1)$ for every $Q' \in \mathcal{Q}'$ and $v \in Q'$. Moreover, H'' can be computed from $\mathcal{Q}'$, H, and H' in polynomial time.*

Our algorithm will employ the above lemma to construct a maximal sequence $(H_0, \mathcal{Q}_0)$, $\ldots, (H_k, \mathcal{Q}_k)$ that satisfies the following additional property:

(M4) For every $1 \leq i \leq k$, every $Q \in \mathcal{Q}_i$ and $v \in Q$ it holds that $|N_{H_i}(v) \cap Q| \geq |N_{H_{i-1}}(v) \cap Q| - 2(|\mathcal{Q}_i| - 1)$.

This property will be key for the analysis of our algorithm because it allows us to bound the number of parts in each partition $\mathcal{Q}_i$. Towards this aim we require the following auxirilliary lemma.

▶ **Lemma 11.** *Let $\mathcal{S} = (H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ be a sequence satisfying Properties (M1)–(M4). Then $|N_{H_i}(v) \cap Q| \geq f(v) - \sum_{1 \leq j \leq i} 2(|\mathcal{Q}_j| - 1)$ for every i with $1 \leq i \leq k$, $Q \in \mathcal{Q}_i$ and $v \in Q$.*

We are now ready to show that the number of parts in any partition $\mathcal{Q}_i$ in a maximal sequence does not exceed $g(n) + 1$.

▶ **Lemma 12.** *Let $\mathcal{S} = (H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ be a maximal sequence satisfying Properties (M1)–(M4). Then $|\mathcal{Q}_i| \leq g(n) + 1$ for every i with $0 \leq i \leq k$. Moreover, the length of $\mathcal{S}$ is at most $g(n) + 1$.*

We are now ready to prove the main theorem of this section.

Sketch of Proof of Theorem 7. Let (G, f) be an instance of Connected g-Bounded f-factor. The algorithm constructs a maximal sequence satisfying Properties (M1)–(M4). The first pair $(H_0, \mathcal{Q}_0)$ is obtained by computing an arbitrary f-factor of G and setting $\mathcal{Q}_0 = \{V(G)\}$. The crucial ingredient of the algorithm is then a procedure that given the i-th pair $(H_i, \mathcal{Q}_i)$ of a maximal sequence computes a $(i+1)$-th pair $(H_{i+1}, \mathcal{Q}_{i+1})$ that can be appended to the sequence without violating any of the Properties (M1)–(M4). Informally, this is achieved by first setting $\mathcal{Q}_{i+1}$ to be the refinement of $\mathcal{Q}_i$ containing one part for every part $Q \in \mathcal{Q}_i$ and every component of $H_i[Q]$. Then an f-factor connecting $\mathcal{Q}_{i+1}$ is computed using the given algorithm for Partition Connector. If no such f-factor exists, the algorithm returns failure (which is correct due to Observation 8). If such an f-factor exists, the procedure uses the given f-factor and Lemma 10 to compute an f-factor H_{i+1} connecting $\mathcal{Q}_{i+1}$ such that the pair $(H_{i+1}, \mathcal{Q}_{i+1})$ satisfies Property (M4). Clearly, if any of the computed f-factors are connected f-factors of G the procedure returns the corresponding f-factor as a solution. Otherwise, the procedure now tries to find a successor of $(H_{i+1}, \mathcal{Q}_{i+1})$ in the already computed sequence satisfying (M1)–(M4). ◀

3.2 A Quasipolynomial Time Algorithm for Polylogarithmic Bounds

In this section, we prove Theorem 1 and Theorem 2. In fact, we prove a more general result, from which both theorems directly follow.

▶ **Theorem 13.** *For every $c > 0$ and function $g(n) = \mathcal{O}((\log n)^c)$, the* Connected g-Bounded f-factor *problem can be solved in $\tilde{\mathcal{O}}(n^{g(n)})$ time.*

We will make use of the following simple lemma.

▶ **Lemma 14.** *Let G be a graph having a connected f-factor. Let $\mathcal{Q}$ be a partition of the vertex set V. There exists a spanning tree T of $G/\mathcal{Q}$ such that for some f-factor H of G, $E(T) \subseteq E(H)$. Furthermore, H can be computed from T in polynomial time.*

Proof. Let G' be a connected f-factor of G. For any partition $\mathcal{Q}$ of the vertex set, it follows from Theorem 8 that $G'/\mathcal{Q}$ is connected. Consider a spanning tree T of $G'/\mathcal{Q}$. Clearly, there exists at least one f-factor H containing $E(T)$ and hence $H/\mathcal{Q}$ is connected. Once we have $E(T)$, H can be computed in polynomial time using Lemma 6. ◀

In light of Theorem 7, it now suffices to prove the following Lemma 15, from which Theorem 13 immediately follows.

▶ **Lemma 15.** Partition Connector *can be solved in time $\mathcal{O}(n^{2(|\mathcal{Q}|-1)})$.*

Proof. It follows from Lemma 14 that we can solve Partition Connector by going over all spanning trees T of $G/\mathcal{Q}$ and checking for each of them whether there is an f-factor of G containing the edges of T. The lemma now follows because the number of spanning trees of $G/\mathcal{Q}$ is at most $\binom{|E(G)|}{|\mathcal{Q}|-1}$, which is upper bounded by $\mathcal{O}(n^{2(|\mathcal{Q}|-1)})$, and for every such tree T we can check the existence of an f-factor containing T in polynomial time. ◀

4 A Randomized Polynomial Time Algorithm for Logarithmic Bounds

In this section we prove Theorem 3. Due to Theorem 7, it is sufficient for us to provide a randomized algorithm for Partition Connector with running time $\mathcal{O}(2^{|\mathcal{Q}|}n^{\mathcal{O}(1)})$ and error probability $\mathcal{O}(g(n)^2/n^2)$. This is precisely what we do in the rest of this section (Lemma 28). As a first step, we will design an algorithm for the "existential version" of the problem which we call ∃-Partition Connector and define as follows.

∃-Partition Connector
Input: A graph G with n vertices, $f : V \to \mathbb{N}$, and a partition $\mathcal{Q}$ of $V(G)$.
Question: Is there an f-factor of G that connects $\mathcal{Q}$?

We will then describe how to use our algorithm for this problem as a subroutine in our algorithm to solve Partition Connector.

4.1 Solving ∃-Partition Connector in Randomized Polynomial Time

The objective of this subsection is to prove the following lemma which implies a randomized polynomial time algorithm for ∃-Partition Connector when $g(n) = \mathcal{O}(\log n)$.

▶ **Lemma 16.** *There exists an algorithm that, given a graph G, a function $f : V \to \mathbb{N}$, and a partition $\mathcal{Q}$ of $V(G)$, runs in time $\mathcal{O}(2^{|\mathcal{Q}|}|V(G)|^{\mathcal{O}(1)})$ and outputs*

- NO *if G has no f-factor connecting $\mathcal{P}$*
- YES *with probability at least $1 - \frac{1}{n^2}$ otherwise.*

We design this algorithm by starting from the exact-exponential algorithm in [14] and making appropriate modifications. During the description, we will point out the main differences between our algorithm and that in [14]. We now proceed to the details of the algorithm. We begin by recalling a few important definitions and known results on f-factors. These are mostly standard and are also present in [14], but since they are required in the description and proof of correctness of our algorithm, we will state them here.

▶ **Definition 17** (f-Blowup)**.** Let G be a graph and let $f : V(G) \to \mathbb{N}$ be such that $f(v) \leq deg(v)$ for each $v \in V(G)$. Let H be the graph defined as follows

1. For each vertex v of G, we add a vertex set $A(v)$ of size $f(v)$ to H.
2. For each edge $e = \{v, w\}$ of G we add to H vertices v_e and w_e and edges (u, v_e) for every $u \in A(v)$ and (w_e, u) for every $u \in A(w)$. Finally, we add the edge (v_e, w_e).

This completes the construction. The graph H is called the *f-blowup* of graph G. We use $\mathcal{B}_f(G)$ to denote the f-blowup of G. We omit the subscript when there is no scope for ambiguity.

▶ **Definition 18** (Induced f-blowup)**.** For a subset $S \subseteq V(G)$, we define the f-blowup of G *induced* by S as follows. Let the f-blowup of G be H. Begin with the graph H and for every edge $e = (v, w) \in E(G)$ such that $v \in S$ and $w \notin S$, delete the vertices v_e and w_e. Let the graph H' be the union of those connected components of the resulting graph which contain the vertex sets $A(v)$ for vertices $v \in S$. Then, the graph H' is called the f-blowup of G *induced* by the set S and is denoted by $\mathcal{B}_f(G)[S]$.

We now recall the relation between perfect matchings in the f-blowup and f-factors (see Figure 1).

▶ **Lemma 19** ([10])**.** *A graph G has an f-factor if and only if the f-blowup of G has a perfect matching.*

The relationship between the Tutte matrix and perfect matchings is well-known and this has already been exploited in the design of fixed-parameter and exact algorithms [21, 7].

▶ **Definition 20** (Tutte matrix)**.** The *Tutte matrix* of a graph G with n vertices is an $n \times n$ skew-symmetric matrix T over the set $\{x_{ij} | 1 \leq i < j \leq |V(G)|\}$ of indeterminates whose $(i, j)^{th}$ element is defined to be

$$T(i,j) = \begin{cases} x_{ij} & \text{if } \{i,j\} \in E(G) \text{ and } i < j \\ -x_{ij} & \text{if } \{i,j\} \in E(G) \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

We use $\mathcal{T}(G)$ to denote the Tutte matrix of the graph G.

Following terminology in [14], when we refer to expanded forms of *succinct* representations (such as summations and determinants) of polynomials, we use the term *naive expansion* (or summation) to denote that expanded form of the polynomial which is obtained by merely writing out the operations indicated by the succinct representation. We use the term *simplified expansion* to denote the expanded form of the polynomial which results after we apply all possible simplifications (such as cancellations) to a naive expansion. We call a monomial m which has a non-zero coefficient in a simplified expansion of a polynomial P, a *surviving* monomial of P in the simplified expansion.

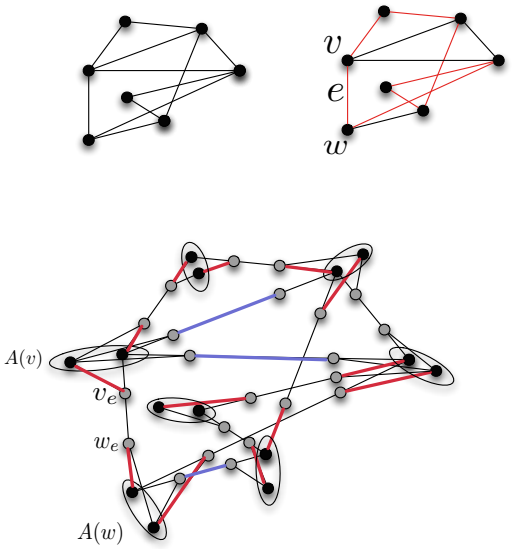

Figure 1 An illustration of a graph G with a 2-factor H (the red edges) and one possible corresponding perfect matching in $\mathcal{B}(G)$. It is important to note that an edge $e = (v, w)$ is *not* in H if and only if the edge (v_e, w_e) is present in the corresponding perfect matching.

The following basic facts about the Tutte matrix $\mathcal{T}(G)$ of a graph G are well-known. When evaluated over any field of characteristic two, the determinant and the permanent of the matrix $\mathcal{T}(G)$ (indeed, of any matrix) coincide. That is,

$$\det \mathcal{T}(G) = \text{perm}(\mathcal{T}(G)) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} \mathcal{T}(G)(i, \sigma(i)), \tag{1}$$

where S_n is the set of all *permutations* of $[n]$. Furthermore, there is a one-to-one correspondence between the set of all *perfect matchings* of the graph G and the *surviving monomials* in the above expression for $\det \mathcal{T}(G)$ when its simplified expansion is computed over any field of characteristic two.

▶ **Proposition 21** ([11]). If $M = \{(i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell)\}$ is a perfect matching of a graph G, then the product $\prod_{(i_k, j_k) \in M} x^2_{i_k j_k}$ appears exactly once in the naive expansion and hence as a surviving monomial in the sum on the right-hand side of Equation 1 when this sum is expanded and simplified over any field of characteristic two. Conversely, each surviving monomial in a simplified expansion of this sum over a field of characteristic two is of the form $\prod_{(i_k, j_k) \in M} x^2_{i_k j_k}$ where $M = \{(i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell)\}$ is a perfect matching of G. In particular, $\det \mathcal{T}(G)$ is identically zero when expanded and simplified over a field of characteristic two if and only if the graph G does not have a perfect matching.

▶ **Lemma 22** (Schwartz-Zippel Lemma, [17, 23]). *Let $P(x_1, \ldots, x_n)$ be a multivariate polynomial of degree at most d over a field $\mathbb{F}$ such that P is not identically zero. Furthermore, let $r_1, \ldots, r_n$ be chosen uniformly at random from $\mathbb{F}$. Then, $Prob[P(r_1, \ldots, r_n) = 0] \leq \frac{d}{|\mathbb{F}|}$.*

▶ **Definition 23.** For a partition of $V(G)$, $\mathcal{Q} = \{Q_1, \ldots, Q_\ell\}$ and a subset $I \subseteq [\ell]$, we denote by $\mathcal{Q}(I)$ the set $\bigcup_{i \in I} Q_i$. Furthermore, with every set $\emptyset \neq I \subset [\ell]$, we associate a specific monomial m_I which is defined to be the product of the terms x^2_{ij} where $i < j$

and $\{i,j\} = \{v_e, w_e\}$, $e = (v,w) \in E(G)$ crosses the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$ and v_e, w_e, are as in Definition 17 of the f-blowup $\mathcal{B}(G)$ of G. For $I = [\ell]$, we define $m_I = 1$.

From now on, for a set $X \subseteq V(G)$, we denote by $\overline{X}$ the set $V(G) \setminus X$. Also, since we always deal with a fixed graph G and function f, for the sake of notational convenience, we refer to the graph $\mathcal{B}_f(G)$ simply as $\mathcal{B}$. We now define a polynomial $P_{\mathcal{Q}}(\bar{x})$ over the indeterminates from the Tutte matrix $\mathcal{T}(\mathcal{B})$ of the f-blowup of G, as follows:

$$P_{\mathcal{Q}}(\bar{x}) = \sum_{\{1\} \subseteq I \subseteq [\ell]} (\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])) \cdot (\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])) \cdot m_I, \tag{2}$$

where if a graph H has no vertices or edges then we set $\det \mathcal{T}(H) = 1$. In future, we will always deal with a fixed partition $\mathcal{Q} = \{Q_1, \ldots, Q_\ell\}$ of $V(G)$.

▶ Remark. The definition of the polynomial $P_{\mathcal{Q}}(\bar{x})$ is the main difference between our algorithm and the algorithm in [14]. The rest of the details are identical. The main algorithmic consequence of this difference is the time it takes to evaluate this polynomial at a given set of points. This is captured in the following lemma whose proof follows from the fact that determinant computation is a polynomial time solvable problem.

▶ **Lemma 24.** *Given values for the variables x_{ij} in the matrix $\mathcal{T}(\mathcal{B})$, the polynomial $P_{\mathcal{Q}}(\bar{x})$ can be evaluated over a field $\mathbb{F}$ of character 2 and size $\Omega(n^6)$ in time $\mathcal{O}(2^\ell n^{\mathcal{O}(1)})$.*

Having shown that this polynomial can be efficiently evaluated, we will now turn to the way we use it in our algorithm. Our algorithm for ∃-Partition Connector takes as input $G, f, \mathcal{Q}$, evaluates the polynomial $P_{\mathcal{Q}}(\bar{x})$ at points chosen independently and uniformly at random from a field $\mathbb{F}$ of size $\Omega(n^6)$ and characteristic 2 and returns Yes if and only if the polynomial does not vanish at the chosen points. In what follows we will prove certain properties of this polynomial which will be used in the formal proof of correctness of this algorithm. We need another definition before we can state the main lemma capturing the properties of the polynomial. Recall that for every $v \in V(G)$, the set $A(v)$ is the set of 'copies' of v in the f-blowup of G. Furthermore, for a set $X \subseteq V(G)$, we say that an edge $e \in E(G)$ *crosses* the cut $(X, \overline{X})$ if e has exactly one endpoint in X.

▶ **Definition 25.** We say that an f-factor H of G *contributes* a monomial $x_{i_1 j_1}^2 \ldots x_{i_r j_r}^2$ to the naive expansion of the right-hand side of Equation 2 if and only if the following conditions hold.

1. For every $e = (v,w) \in E(H)$, there is a $u \in A(v)$, $u' \in A(w)$ and $1 \leq p, q \leq r$ such that $\{u, v_e\} = \{i_p, j_p\}$ and $\{u', w_e\} = \{i_q, j_q\}$.
2. For every $e = (v,w) \in E(G) \setminus E(H)$, there is a $1 \leq p \leq r$ such that $\{v_e, w_e\} = \{i_p, j_p\}$.
3. For every $1 \leq p, q \leq r$, if $\{u, v_e\} = \{i_p, j_p\}$ and $\{u', w_e\} = \{i_q, j_q\}$ for some $e \in E(G)$, then $e \in E(H)$.
4. For every $1 \leq p \leq r$, if $\{i_p, j_p\} = \{v_e, w_e\}$ for some $e \in E(G)$, then $e \notin E(H)$.
5. For every $1 \in I \subseteq [\ell]$ such that H has no edge crossing the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$, there is a pair of monomials m_1 and m_2 such that m_1 is a surviving monomial in the simplified expansion of $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$, m_2 is a surviving monomial in the simplified expansion of $\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$, and $m_1 \cdot m_2 \cdot m_I = x_{i_1 j_1}^2 \ldots x_{i_r j_r}^2$.

Having set up the required notation, we now state the main lemma which allows us to show that monomials contributed by f-factors that do not connect $\mathcal{Q}$, do not survive in the simplified expansion of the right hand side of Equation 2.

▶ **Lemma 26.** *Every monomial in the polynomial $P_{\mathcal{Q}}(\bar{x})$ which is a surviving monomial in the simplified expansion of the right-hand side of Equation 2 is contributed by an f-factor of G to the naive expansion of the right-hand size of Equation 2. Furthermore, for any f-factor of G, say H, the following statements hold.*

1. *If H does not connect $\mathcal{Q}$ then every monomial contributed by H occurs an* even *number of times in the polynomial $P_{\mathcal{Q}}(\bar{x})$ in the naive expansion of the right-hand side of Equation 2.*
2. *If H connects $\mathcal{Q}$, then every monomial contributed by H occurs exactly once in the polynomial $P_{\mathcal{Q}}(\bar{x})$ in the naive expansion of the right-hand side of Equation 2.*

This implies the following result, which is the last ingredient we need to prove Lemma 16.

▶ **Lemma 27.** *The polynomial $P_{\mathcal{Q}}(\bar{x})$ is not identically zero over $\mathbb{F}$ if and only if G has an f-factor connecting $\mathcal{Q}$.*

Proof of Lemma 16. It follows from the definition of $P(\bar{x})$ that its degree is $\mathcal{O}(n^4)$ since the number of vertices in the f-blowup of G is $\mathcal{O}(n^2)$. As mentioned earlier, our algorithm for ∃-Partition Connector takes as input $G, f, \mathcal{Q}$, evaluates the polynomial $P_{\mathcal{Q}}(\bar{x})$ at points chosen independently and uniformly at random from a field $\mathbb{F}$ of size $\Omega(n^6)$ and characteristic 2 and returns Yes if and only if the polynomial does not vanish at the chosen points. Due to Lemma 27, we know that the polynomial $P_{\mathcal{Q}}(\bar{x})$ is identically zero if and only if G has an f-factor containing $\mathcal{Q}$ and by the Schwartz-Zippel Lemma, the probability that the polynomial is not identically zero and still vanishes upon evaluation is at most $\frac{1}{n^2}$. This completes the proof of the lemma. ◀

Having obtained the algorithm for ∃-Partition Connector, we now return to the algorithm for the computational version, Partition Connector.

4.2 Solving Partition Connector in Randomized Polynomial Time

▶ **Lemma 28.** *The* Partition Connector *problem can be solved by a randomized algorithm with running time $\mathcal{O}(2^{|\mathcal{Q}|}n^{\mathcal{O}(1)})$ and error probability $\mathcal{O}(1-(1-\frac{1}{n^2})^{|\mathcal{Q}|})$.*

Sketch of Proof. Consider the following algorithm $\mathcal{A}$. Algorithm $\mathcal{A}$ takes as input an n-vertex instance of Partition Connector with the partition $\mathcal{Q} = \{Q_1, \ldots, Q_\ell\}$, along with a separate set of edges F which will store the edges that have been previously selected to be included in the partition connector. Let F be initialized as $\emptyset$. As its first step, Algorithm $\mathcal{A}$ checks if $\ell = 1$; if this is the case, then it computes an arbitrary f-factor H, and outputs $H \cup F$. To proceed, let us denote the algorithm of Lemma 16 as $\mathcal{B}$. If $\ell > 1$, then $\mathcal{A}$ first calls $\mathcal{B}$ and outputs NO if $\mathcal{B}$ outputs NO. Otherwise, it fixes an arbitrary ordering $E^{\leq}$ of the edge set E and recursively proceeds as follows.

$\mathcal{A}$ constructs the set E_1 of all edges with precisely one endpoint in Q_1, and loops over all edges in E_1 (in the ordering given by $E^{\leq}$). For each processed edge e between Q_1 and some Q_i with endpoints c and d, it will compute a subinstance (G^e, f^e, Q^e) defined by setting:

- $G^e = G - e$, and
- $f^e(c) = f(c) - 1$, $f^e(d) - f(d) - 1$ and $f^e = f$ for all other vertices of G, and
- $\mathcal{Q}^e$ is obtained from $\mathcal{Q}$ by merging Q_1 and Q_i into a new set; formally, $\mathcal{Q}^e = (\mathcal{Q} \setminus \{Q_1, Q_i\}) \cup \{Q_1 \cup Q_i\}$.

Intuitively, each such new instance corresponds to us forcing the f-factor to choose the edge e. $\mathcal{A}$ then queries $\mathcal{B}$ on $(G^e, f^e, \mathcal{Q}^e)$. If $\mathcal{B}$ answers NO for each such tuple $(G^e, f^e, \mathcal{Q}^e)$ obtained from each edge in E_1, then $\mathcal{A}$ immediately terminates and answers NO. Otherwise let e be the first edge where $\mathcal{B}$ answered YES; then $\mathcal{A}$ will add e into F. If $|\mathcal{Q}^e| = 1$ then

the algorithm computes an arbitrary f-factor H of (G^e, f^e) and outputs $H \cup F$. On the other hand, if $|\mathcal{Q}| > 1$ then $\mathcal{A}$ restarts the recursive procedure with $(G, f, \mathcal{Q}) := (G^e, f^e, \mathcal{Q}^e)$; observe that $|\mathcal{Q}^e| \leq |\mathcal{Q}| - 1$. To complete the proof, it suffices to verify the correctness and the running time. ◀

5 Classification Results

In this section, we prove Theorem 4 which we restate for the sake of completeness.

▶ **Theorem 4.** *For every $c > 1$, for every $g(n) = \Theta((\log n)^c)$,* Connected g-Bounded f-Factor *is neither in* P *nor* NP*-hard unless the Exponential Time Hypothesis fails.*

The result relies on the established Exponential Time Hypothesis, which we recall below.

▶ **Definition 29** (Exponential Time Hypothesis (ETH), [8])**.** There exists a constant $s > 0$ such that 3-SAT with n variables and m clauses cannot be solved in time $2^{sn}(n+m)^{\mathcal{O}(1)}$.

We first show that the problem is not NP-hard unless the ETH fails. We remark that we can actually prove a stronger statement here by weakening the premise to "NP is not contained in Quasi-Polynomial Time". However, since we are only able to show the other part of Theorem 4 under the ETH, we phrase the statement in this way.

▶ **Lemma 30.** *For every $c > 1$, for every $g(n) = \Theta((\log n)^c)$,* Connected g-Bounded f-Factor *is not* NP*-hard unless the Exponential Time Hypothesis fails.*

Proof. Due to Theorem 2, we know that when $g(n) = \Theta((\log n)^c)$, Connected g-Bounded f-Factor can be solved in quasi-polynomial time. Hence, this problem cannot be NP-hard unless NP is contained in the complexity-class Quasi-Polynomial Time, QP. Furthermore, observe that $NP \subseteq QP$ implies that the ETH is false. Hence, we conclude that Connected g-Bounded f-Factor is not NP-hard unless the Exponential Time Hypothesis fails. ◀

Next, we use a reduction from Hamiltonian Cycle to obtain:

▶ **Lemma 31.** *For every $c > 1$, for every $g(n) = \Theta((\log n)^c)$,* Connected g-Bounded f-Factor *is not in* P *unless the Exponential Time Hypothesis fails.*

Lemmas 30 and 31 together give us Theorem 4.

6 Concluding remarks

We obtained new complexity results for Connected f-Factor with respect to lower bounds on the function f. As our main results, we showed that when $f(v)$ is required to be at least $\frac{n}{(\log n)^c}$, the problem can be solved in quasi-polynomial time in general and in randomized polynomial time if $c \leq 1$. Consequently, we show that the problem can be solved in polynomial-time when $f(v)$ is at least $\frac{n}{c}$ for any constant c. We complement the picture with matching classification results.

As a by-product we obtain a generic approach reducing Connected f-Factor to the "simpler" Partition Connector problem. Hence future algorithmic improvements of Partition Connector carry over to the Connected f-Factor problem. Finally, it would be interesting to investigate the possibility of derandomizing the polynomial-time algorithm for the case that $g(n) = \mathcal{O}(\log n)$.

Acknowledgments. The authors wish to thank the anonymous reviewers for their helpful comments.

References

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

2 F. Cheah and D. G. Corneil. The complexity of regular subgraph recognition. *Discrete Applied Mathematics*, 27(1-2):59–68, 1990.

3 FRK Chung and RL Graham. Recent results in graph decompositions. *London Mathematical Society, Lecture Note Series*, 52:103–123, 1981.

4 Kamiel Cornelissen, Ruben Hoeksma, Bodo Manthey, N.S. Narayanaswamy, and C.S. Rahul. Approximability of connected factors. In Christos Kaklamanis and Kirk Pruhs, editors, *Approximation and Online Algorithms*, volume 8447 of *Lecture Notes in Computer Science*, pages 120–131. Springer International Publishing, 2014. doi:10.1007/978-3-319-08001-7_11.

5 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS*, pages 150–159, 2011.

6 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

7 Gregory Gutin, Magnus Wahlström, and Anders Yeo. Parameterized rural postman and conjoining bipartite matching problems. *CoRR*, abs/1308.2599, 2013. URL: http://arxiv.org/abs/1308.2599.

8 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001. doi:10.1006/jcss.2001.1774.

9 Tomáš Kaiser. A short proof of the tree-packing theorem. *Discrete Mathematics*, 312(10):1689–1691, 2012.

10 László Lovász. The factorization of graphs. ii. *Acta Mathematica Academiae Scientiarum Hungarica*, 23(1-2):223–246, 1972.

11 László Lovász. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computation Theory FCT '79*, pages 565–574, Berlin, 1979. Akademie-Verlag.

12 NS Narayanaswamy and CS Rahul. Approximation and exact algorithms for special cases of connected f-factors. In *Computer Science–Theory and Applications*, pages 350–363. Springer, 2015.

13 Julius Petersen. Die theorie der regulären graphs. *Acta Mathematica*, 15(1):193–220, 1891.

14 Geevarghese Philip and M. S. Ramanujan. Vertex exponential algorithms for connected f-factors. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 61–71, 2014.

15 Geevarghese Philip and MS Ramanujan. Vertex exponential algorithms for connected f-factors. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

16 M. D. Plummer. Graph factors and factorization: 1985–2003: a survey. *Discrete Mathematics*, 307(7):791–821, 2007.

17 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

18 W. T. Tutte. A short proof of the factor theorem for finite graphs. *Canadian Journal of Mathematics*, 6(1954):347–352, 1954. doi:10.4153/CJM-1954-033-3.

19 WT Tutte. The factors of graphs. *Canad. J. Math*, 4(3):314–328, 1952.

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

21 Magnus Wahlström. Abusing the tutte matrix: An algebraic instance compression for the k-set-cycle problem. In *STACS*, pages 341–352, 2013.

22 D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.

23 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation*, volume 72, pages 216–226. 1979.