White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/167897/

Version: Accepted Version

Article:
Ganian, R, Narayanaswamy, NS, Ordyniak, S et al. (2019) On the Complexity Landscape of Connected f-Factor Problems. Algorithmica, 81 (6). pp. 2606-2632. ISSN: 0178-4617

https://doi.org/10.1007/s00453-019-00546-z

© Springer Science+Business Media, LLC, part of Springer Nature 2019. This is an author produced version of an article published in Algorithmica. Uploaded in accordance with the publisher's self-archiving policy.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Noname manuscript No.
(will be inserted by the editor)

On the Complexity Landscape of Connected f-Factor Problems

R. Ganian[1] · **N. S. Narayanaswamy**[2] · **S. Ordyniak**[3] · **C. S. Rahul**[4] · **M. S. Ramanujan**[1]

the date of receipt and acceptance should be inserted later

Abstract Let G be an undirected simple graph having n vertices and let $f : V(G) \to \{0, \ldots, n-1\}$ be a function. An f-factor of G is a spanning subgraph H such that $d_H(v) = f(v)$ for every vertex $v \in V(G)$. The subgraph H is called a *connected f-factor* if, in addition, 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 we restrict $f(v)$ to be at least n^ϵ for each vertex v and constant $0 \leq \epsilon < 1$; on the other side of the spectrum of nontrivial lower bounds on f, the problem is 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 restrictions on the function f. In particular, we show that when $f(v)$ is restricted 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 $c \leq 1$. Furthermore, we show that when $c > 1$, the problem is NP-intermediate.

Keywords connected f-factors, quasi-polynomial time algorithms, randomized algorithms, NP-intermediate, exponential time hypothesis

[1]Algorithms and Complexity Group, TU Wien, Vienna, Austria
rganian@gmail.com, ramanujan@ac.tuwien.ac.at
[2]Indian Institute of Technology Madras, Chennai, India
swamy@cse.iitm.ac.in
[3]Department of Computer Science, University of Sheffield, Sheffield, UK
sordynian@gmail.com
[4]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland
rahulcs@mimuw.edu.pl

$f(v) \geq$	Complexity Class
$n^{\epsilon}, \forall \epsilon \in [0,1)$	NPC [6,5]
$\frac{n}{\mathsf{polylog}(n)}$	QP (Theorem 2)
$\frac{n}{\log n}$	RP (Theorem 3)
$\frac{n}{c}, \forall c \geq 3$	P (Theorem 1)

Table 1 The table depicting the known as well as new results on the complexity landscape of the CONNECTED f-FACTOR problem. Note that ϵ is an arbitrary constant in the given range.

1 Introduction

The concept of f-factors is fundamental in graph theory, dating back to the 19th century, specifically to the work of Petersen [16]. 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 [24]. 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 [21]. In addition, he developed a method for reducing the f-factor computation problem to the perfect matching [22] problem, which gives a straightforward polynomial time algorithm for the problem of deciding the existence of an f-factor. There are also several detailed surveys on f-factors of graphs, for instance by Chung and Graham [4], Akiyama and Kano [1], Lovász and Plummer [14].

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 [13] and Plummer [18]). Unlike the general f-factor problem, deciding the existence of a connected f-factor is NP-complete [8,3]. It is easy to see that the connected f-factor problem (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 connected f-factor problem [17], though there are such algorithms for special cases like the HAMILTONIAN CYCLE [10,2].

The NP-completeness of this problem has motivated several authors to study the CONNECTED f-FACTOR for various restrictions on the function f. Cornelissen et al. [6] showed that CONNECTED f-FACTOR remains NP-complete even when $f(v)$ is at least n^{ϵ} for each vertex v and any nonnegative constant ϵ less than 1. Similarly, it has been shown that the problem is polynomial time solvable when $f(v)$ is at least $\frac{n}{3}$ [15] for every vertex v. Aside from these two fairly extreme cases, the complexity landscape of CONNECTED f-FACTOR based on lower bounds on the function f, has largely been left uncharted.

Our results and techniques. In this paper, we provide new results (both positive and negative) on solving CONNECTED f-FACTOR based on lower bounds on the range of f. Since we 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 (CONNECTED g-BOUNDED f-FACTOR) below:

CONNECTED g-BOUNDED f-FACTOR
Instance: An n-vertex undirected simple graph G and a mapping $f : V(G) \to \mathbb{N}$ such that $f(v) \geq \frac{n}{g(n)}$.
Task: Find a connected f-factor H of G.

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 $c > 1$. This result generalizes 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 when $f(v)$ is at least $\frac{n}{\mathsf{polylog}(n)}$ for every vertex. To be precise, we prove the following two theorems (see Section 2 for an explanation of the function g in formal statements).

Theorem 1 *For every function $g(n) \in \mathcal{O}(1)$,* CONNECTED g-BOUNDED f-FACTOR *can be solved in polynomial time.*

Theorem 2 *For every $c > 0$ and function $g(n) \in \mathcal{O}((\log n)^c)$,* CONNECTED g-BOUNDED f-FACTOR *can be solved in time $n^{(\log n)^{\alpha(c)}}$ where $\alpha(c) \in \mathcal{O}(1)$.*

Second, we build upon these new techniques to obtain a *randomized polynomial time* algorithm which solves CONNECTED f-FACTOR in the more general case where $f(v)$ is lower-bounded by $\frac{n}{g(n)}$ for every vertex v and $g(n) \in \mathcal{O}(\log n)$. For this, we also require algebraic techniques that have found several applications in the design of fixed-parameter and exact algorithms for similar problems [7,23,9,17]. Precisely, we prove the following theorem.

Theorem 3 *For every function $g(n) \in \mathcal{O}(\log n)$,* 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 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 [11] holds. Formally speaking, we prove the following theorem.

Theorem 4 *For every $c > 1$ and for every $g(n) \in \Theta((\log n)^c)$,* 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 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 quasi-polynomial 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 possibilities for CONNECTED g-BOUNDED f-FACTOR for all polylogarithmic functions g in $\Theta((\log n)^c)$.

2 Preliminaries

2.1 Basic Definitions

We use standard definitions and notations from West [24]. The notation $d_G(v)$ denotes the **degree** of a vertex v in a graph G. Similarly, $N_G(v)$ represents the set of vertices adjacent to v in 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. A **cycle** is a circuit where all the $k-1$ vertices are distinct. An **Eulerian circuit** in a graph is a circuit in which each edge in the graph appears. Any graph having an Eulerian circuit is called an *Eulerian* graph.

Let V' be a subset of the vertices in the graph G. The **vertex-induced subgraph** $G[V']$ is the graph over vertex set V' containing all the edges in G whose endpoints are both in V'. Given $E' \subseteq E(G)$, $G[E']$ is the **edge-induced subgraph** of G whose edge set is E' and vertex set is the set of all vertices incident to edges in E'.

Given two subgraphs G_1 and G_2 of G, the graph $G_1 \triangle G_2$ is the subgraph $G[E(G_1) \triangle E(G_2)]$. The union of the graphs $G_1, G_2, \ldots, G_r$ is the graph $\bigcup_i^r G_i$ whose vertex set is $\bigcup_i^r V(G_i)$ and edge set is $\bigcup_i^r E(G_i)$.

Given a partition $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_r\}$ of the vertex set of G, the quotient 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, $i \neq j$, add an edge (Q_i, Q_j) to $G/\mathcal{Q}$. 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. Further, we address the graph G' to be a **partition connector**. A **refinement** $\mathcal{Q}'$ of a partition $\mathcal{Q}$ is a partition of $V(G)$ 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 [12]. A *spanning tree of the quotient graph* $G/\mathcal{Q}$ refers to a subgraph T of G with $|\mathcal{Q}|$-1 edges that connects $\mathcal{Q}$. The following lemma will later be used in the analysis of the error probability of our randomized algorithm.

Lemma 5 *The following holds for every $n, c \in \mathbb{N}$ with $n > c$:*

$$1 - \frac{(c\lceil \log n \rceil)^2}{n^2} \leq \left(1 - \frac{1}{n^2}\right)^{c \log n}$$

Proof. Using simple term manipulations, we obtain

$$\left(1-\frac{1}{n^2}\right)^{c\log n} = \left(\frac{n^2-1}{n^2}\right)^{c\log n} = \frac{(n^2-1)^{c\log n}}{(n^2)^{c\log n}}. \tag{1}$$

Since $\frac{n^2-1}{n^2} < 1$, it follows that

$$\frac{(n^2-1)^{c\lceil\log n\rceil}}{(n^2)^{c\lceil\log n\rceil}} \leq \frac{(n^2-1)^{c\log n}}{(n^2)^{c\log n}}. \tag{2}$$

By using the binomial formula, we obtain

$$\begin{aligned}
(n^2-1)^{c\lceil\log n\rceil} &= \sum_{i=0}^{c\lceil\log n\rceil}\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i \\
&= n^{2c\lceil\log n\rceil} + \sum_{i=1}^{c\lceil\log n\rceil}\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i \\
&= n^{2c\lceil\log n\rceil} - \sum_{i=1}^{c\lceil\log n\rceil} -\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i.
\end{aligned}$$

To obtain an upper bound on $\sum_{i=1}^{c\lceil\log n\rceil} -\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i$, we show next that the absolute values of the terms in the sum are decreasing with increasing i.

$$\begin{aligned}
\left|\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i\right| &= \binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i} \\
&\geq \frac{c\lceil\log n\rceil}{n^2}\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i} \\
&\geq c\lceil\log n\rceil\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-(i+1)} \\
&\geq \binom{c\lceil\log n\rceil}{i+1}(n^2)^{c\lceil\log n\rceil-(i+1)}
\end{aligned}$$

The first inequality above holds because $n > c$. Hence, we obtain

$$\begin{aligned}
\sum_{i=1}^{c\lceil\log n\rceil} -\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i &\leq \sum_{i=1}^{c\lceil\log n\rceil}\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i} \\
&\leq (c\lceil\log n\rceil)\cdot(c\lceil\log n\rceil)(n^2)^{c\lceil\log n\rceil-1} \\
&= (c\lceil\log n\rceil)^2(n^2)^{c\lceil\log n\rceil-1}.
\end{aligned}$$

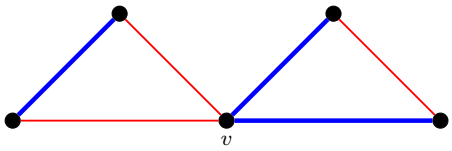

Fig. 1 The alternating circuit A_{11} illustrating that not every alternating circuit can be decomposed into edge-disjoint alternating circuits that are cycles. Here the blue edges are marked with thick lines and the red edges are thin.

Putting the above back into Equation (1) and using Inequality (2) together with the expressions above, we obtain

$$\begin{aligned}\left(1-\frac{1}{n^2}\right)^{c\log n} &\geq \frac{(n^2-1)^{c\lceil\log n\rceil}}{(n^2)^{c\lceil\log n\rceil}}\\ &= \frac{n^{2c\lceil\log n\rceil}-\sum_{i=1}^{c\lceil\log n\rceil}-\binom{c\lceil\log n\rceil}{i}(n^2)^{c\lceil\log n\rceil-i}(-1)^i}{(n^2)^{c\lceil\log n\rceil}}\\ &\geq \frac{n^{2c\lceil\log n\rceil}-(c\lceil\log n\rceil)^2(n^2)^{c\lceil\log n\rceil-1}}{(n^2)^{c\lceil\log n\rceil}}\\ &\geq 1-\frac{(c\lceil\log n\rceil)^2}{n^2}.\end{aligned}$$

The above concludes the proof of the lemma. □

The function g we deal with is always a positive real-valued function, defined on the set of positive integers. For the cases we consider, the function always takes a value greater than 1. Unless otherwise mentioned, $g(n)$ is in $\mathcal{O}(\mathsf{polylog}(n))$. Whenever g is part of the problem definition, the target set of the function f is the set of integers $\{\lceil n/g(n)\rceil,\ldots,n-1\}$. Consequently, we have the following fact.

Fact 6 *Let G be a graph and let $f(v)\geq n/g(n)$ for each v in $V(G)$. If H is an f-factor of G, then the number of components in H is at most $\lceil g(n)\rceil-1$.*

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

A graph G is colored if each edge in G is assigned a color from the set $\{red, blue\}$. In a colored graph G, we use R and B to denote spanning subgraphs of G whose edge sets are the set of red edges ($E(R)$) and blue edges ($E(B)$) respectively. We 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 7 A colored graph A is an **alternating circuit** if there exists an Eulerian circuit in A, where each pair of consecutive edges are of different colors.

Clearly, an alternating circuit has an even number of edges and is connected. Further, $d_R(v) = d_B(v)$ for each v in A. A **minimal alternating circuit** A is an alternating circuit where each vertex v in A has at most two red edges incident to v. Note that alternating circuits, as opposed to Eulerian circuits, cannot always be decomposed into edge-disjoint alternating circuits that are cycles. As an example, consider for each $r_1, r_2 \geq 1$ the alternating circuit $A_{r_1 r_2}$ which consists of two (edge-disjoint) cycles of length $2r_1 + 1$ and length $2r_2 + 1$, respectively, that share one common vertex v. Let the coloring of the edges of $A_{r_1 r_2}$ be as illustrated in Figure 2.2. Informally, the edges of both cycles are colored in an alternating manner along each cycle so that the edges of the first cycle incident to v have the same color, which is distinct from the color given to the edges incident with v in the second cycle. Every alternating circuit in $A_{r_1 r_2}$ contains all edges of $A_{r_1 r_2}$ and cannot be decomposed further into smaller alternating circuits.

Fact 8 *Let S be a subset of $E(G)$. An f-factor H of G containing all the edges in S, if one exists, can be computed in polynomial time.*

The fact follows from the observation that a candidate for $H \setminus S$ can be computed from an f'-factor H' of the spanning subgraph $G \setminus S$, where $f'(v) = f(v) - d_{G[S]}(v)$ for each $v \in V(G)$. Note that given a partition $\mathcal{Q} = \{X, V \setminus X\}$ of $V(G)$, one can check for the existence of an f-factor connecting $\mathcal{Q}$ in polynomial time by iterating over each edge e in the cut $[X, V \setminus X]_G$ and applying Fact 8 by setting $S = \{e\}$. Further, this can be extended for any arbitrary partition $\mathcal{Q}$ of constant size, or when we are provided with a spanning tree S of $G/\mathcal{Q}$ that is guaranteed to be contained in some f-factor of G.

Definition 9 Let M and H be two subgraphs of G where each component in M is Eulerian. Let $\mathtt{c} : E(M) \to \{red, blue\}$ be the unique coloring function which colors the edges in $E(M) \cap E(H)$ with color red and those in $E(M) \setminus E(H)$ with color blue. The subgraph M is called a **switch** on H if every component of the colored graph obtained by applying $\mathtt{c}$ on M, is an alternating circuit.

Definition 10 For a subgraph M which is a switch on another subgraph H of G, we define **Switching(**H**,**M**)** to be the subgraph $M \triangle H$ of G.

We use switching as an operator where the role of the second operand is to bring in specific edges to the first, retaining the degrees of vertices by omitting some less significant edges. One can easily infer that if the result of applying the coloring function $\mathtt{c}$ to M is a minimal alternating circuit, then the switching operation replaces at most two edges incident to each vertex v in H.

Fact 11 *Let A be an alternating circuit and S be a subset of edges in A. There is a polynomial time algorithm that outputs a set $\mathcal{M}$ of edge disjoint minimal alternating circuits in A, each of which has at least one edge from S and such that every edge in S is contained in some minimal alternating circuit in $\mathcal{M}$.*

It is not difficult to see the proof of Fact 11. A skeptical reader can refer [15, Lemma 6]. Note that given S and A, M is not unique.

Fact 12 *Let H be an f-factor of G and let $\mathcal{Q}$ be a partitioning of the vertex set of G. If $H/\mathcal{Q}$ is connected and $H[Q]$ is connected for each Q in $\mathcal{Q}$, then H is a connected f-factor.*

Fact 12 implies that if H is not a connected f-factor and $H/\mathcal{Q}$ is connected then there exists some $Q \in \mathcal{Q}$ such that $H[Q]$ is not connected.

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 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, $f : V(G) \to \mathbb{N}$, 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 are presented in the latter parts of this article. Specifically, a deterministic algorithm that runs in quasi-polynomial time whenever $g(n) = \mathcal{O}(\mathsf{polylog}(n))$ (Section 3.2) and a randomized polynomial time algorithm for the case when $g(n) = \mathcal{O}(\log n)$ (Section 4) are given.

The majority of this section is devoted to proving the key Theorem 13 stated below, which establishes the link between PARTITION CONNECTOR and CONNECTED g-BOUNDED f-FACTOR.

Theorem 13 (a) *Let $g(n) \in \mathcal{O}(\mathsf{polylog}(n))$. If there is a deterministic algorithm running in time $\mathcal{O}^*(n^{2(|\mathcal{Q}|-1)})$[1] for* PARTITION CONNECTOR*, then there is a deterministic quasi-polynomial time algorithm for* CONNECTED g-BOUNDED f-FACTOR *with running time $\mathcal{O}^*(n^{2g(n)})$.*

(b) *Let $g(n) \in \mathcal{O}(\log n)$. If there exists a randomized algorithm running in time $\mathcal{O}^*(2^{|\mathcal{Q}|})$ with error probability $\mathcal{O}(|\mathcal{Q}|^2/n^2)$ for* PARTITION CONNECTOR*, then there exists a randomized polynomial time algorithm for* CONNECTED g-BOUNDED f-FACTOR *that has a constant one-sided 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 14 *Let G be an undirected graph and f be a function $f : V(G) \to \mathbb{N}$. The graph G has a connected f-factor if and only if for each partition $\mathcal{Q}$ of the vertex set $V(G)$, there exists an f-factor H of G that connects $\mathcal{Q}$.*

[1] We use $\mathcal{O}^*(f(n))$ to denote $\mathcal{O}(f(n) \cdot n^{\mathcal{O}(1)})$, i.e., $\mathcal{O}^*$ omits polynomial factors, for any function $f(n)$.

We remark that for the running time analysis for our generic algorithm we 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 $\mathcal{O}(\mathsf{polylog}(n))$, 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 and not the instance given, 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 sequence $(H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ of pairs which is *maximal* (cannot be extended further) satisfying the following properties:

(M1) For $0 \leq i \leq k$, each $\mathcal{Q}_i$ is a partition of the vertex set $V(G)$, and $\mathcal{Q}_0 = \{V(G)\}$.
(M2) For $0 \leq i \leq k$, each H_i 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 15 *Let (G, f) be an instance of* CONNECTED g-BOUNDED f-FACTOR *and let $\mathcal{S} = (H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ be a maximal sequence satisfying (M1)–(M3). Then, G has a connected f-factor if and only if H_k is a connected f-factor of G.*

Proof. Towards showing the forward direction of the claim, let us suppose for a contradiction that H_k is not a connected f-factor of G and that G does contain a connected f-factor. Because H_k connects $\mathcal{Q}_k$ (Property (M2)), it follows from Fact 12 that there is some part $Q' \in \mathcal{Q}_k$ such that $H_k[Q']$ is not connected. Consider the refinement $\mathcal{Q}_{k+1}$ of $\mathcal{Q}_k$ ($\mathcal{Q}_{k+1} \neq \mathcal{Q}_k$) that splits each part Q in $\mathcal{Q}_k$ into the parts corresponding to the components of $H_k[Q]$. Further, because G has a connected f-factor and Observation 14, we obtain that there exists a connected f-factor H_{k+1} that connects any partition $\mathcal{Q}_{k+1}$. Now, the sequence $\mathcal{S}$ could be extended by appending the pair $(H_{k+1}, \mathcal{Q}_{k+1})$ to its end, a contradiction to our assumption that $\mathcal{S}$ was a maximal sequence. The reverse direction is trivial. □

We deploy an algorithm that computes a maximal sequence $\mathcal{S}$ satisfying (M1)–(M3) and thereby use the above lemma to solve the connected f-factor problem by testing whether the last f-factor in the sequence is connected. This involves computing $\mathcal{Q}_{i+1}$ from H_i and $\mathcal{Q}_i$ followed by the computation of H_{i+1} connecting $\mathcal{Q}_{i+1}$. However, if the number of parts in the last partition $\mathcal{Q}_k$ is allowed to grow to n, then such an algorithm would eventually have to solve the connected f-factor problem to compute an H_k satisfying (M2). Our algorithm does the incremental computation of the sequence $\mathcal{S}$ in such a way that lets us to establish a lower bound on the size of any part $Q \in \mathcal{Q}_i$ as a function of g. This implies we have an upper bound on the number of parts in any partition $\mathcal{Q}_i$ in $\mathcal{S}$ which in turn bounds the length of the sequence $\mathcal{S}$ as a function of g.

The following lemma shows that given the recently computed pair $(H, \mathcal{Q}) = (H_i, \mathcal{Q}_i)$ in the sequence, the partition $\mathcal{Q}' = \mathcal{Q}_{i+1}$ and a candidate H'' for H_{i+1}, one can compute a *better* candidate H' for H_{i+1} which is *closer to* H_i in the sense that *most of* the neighbors of a vertex v in H_i are retained as it is in H'. The properties of H' then allow us to lower-bound the size of each part in $\mathcal{Q}_{i+2}$ as a function of the size of the smallest part in $\mathcal{Q}_{i+1}$.

Lemma 16 *Let $(H, \mathcal{Q})$, $(H'', \mathcal{Q}')$ be two consecutive pairs occurring 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.*

Proof. From the premise that H'' is an f-factor connecting $\mathcal{Q}'$, we know that there exists a spanning tree T of $H''/\mathcal{Q}'$. Color the edges in H with color red and those in H'' with color blue. Let $\mathcal{A}$ be the graph $H \triangle H''$. Notice that each component in $\mathcal{A}$ is an alternating circuit. Furthermore, note that the set $S = E(T \setminus H)$ of blue edges is a subset of $\mathcal{A}$ as $E(T)$ is a subset of $E(H'')$. Let S_i be the set $A_i \cap S$ where A_i is the i^{th} component in $\mathcal{A}$. We compute the set $\mathcal{M}_i$ of edge disjoint minimal alternating circuits using Fact 11 for each (A_i, S_i) pair. The size of the set $\mathcal{M}_i$ is at most $|S_i|$ and hence there are at most $|S|$ minimal alternating circuits in $\mathcal{M} = \bigcup_i \mathcal{M}_i$. Let $\mathcal{M}_S = \bigcup_{M \in \mathcal{M}} M$ and H' be the f-factor defined as Switching(H,$\mathcal{M}_S$). We argue that this switching operation removes at most $2(|\mathcal{Q}'| - 1)$ edges incident to any vertex v in $H[Q']$ for every $Q' \in \mathcal{Q}'$.

Considering the fact that the minimal alternating circuits in $\mathcal{M}$ are edge disjoint, we visualize switching with $\mathcal{M}_S$ as a sequence of switching operations on H each with a distinct minimal alternating circuit M in $\mathcal{M}$. In each such M, the number of red edges incident to a vertex v that leaves H during switching is at most two and the operation Switching(H,$\mathcal{M}_S$) retains at least $N_H(v) - 2|\mathcal{M}|$ neighbors of each vertex. Thus, for any subset Q' of $V(G)$ if we consider the subgraph $H[Q']$ alone, it must be the case that $|N_{H[Q']}(v) \cap N_{H'[Q']}(v)| \geq |N_{H[Q']}(v)| - 2|\mathcal{M}|$ for each v in Q'. Furthermore, $|\mathcal{M}|$ is at most $|S| = |\mathcal{Q}'| - 1$. Since the set $E(T)$ is a subset of $E(H')$, H' connects $\mathcal{Q}'$. From Fact 11, the computation of $\mathcal{M}$ and hence of H' takes polynomial time. This completes the proof of the lemma. □

By employing the above lemma, our algorithm ensures that the maximal sequence $(H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ thus constructed 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 plays a key role in the analysis of our algorithm as it allows us to bound the number of parts in each partition $\mathcal{Q}_i$. Towards this aim we require the following auxiliary lemma.

Lemma 17 *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 $v \in Q$.*

Proof. We show the claim by induction on i starting from $i = 1$. Let $Q \in \mathcal{Q}_1$ and $v \in Q$. Because H_0 is an f-factor of G and Q is a component of H_0, we obtain

that $|N_{H_0}(v) \cap Q| = f(v)$. Using Property (M4) for $i = 1$, we obtain $|N_{H_1}(v) \cap Q| \geq |N_{H_0}(v) \cap Q| - 2(|\mathcal{Q}_1| - 1) = f(v) - 2(|\mathcal{Q}_1| - 1)$, as required. Hence assume that the claim holds for $i - 1$ and we want to show the claim for i. Let $Q_i \in \mathcal{Q}_i$ and $v \in Q_i$ and let Q_{i-1} be the part in $\mathcal{Q}_{i-1}$ containing Q_i. Note that $v \in Q_{i-1}$. From the induction hypothesis we obtain that $|N_{H_{i-1}}(v) \cap Q_{i-1}| \geq f(v) - \sum_{1 \leq j \leq i-1} 2(|\mathcal{Q}_j| - 1)$. Because Q_i is a component of $H_{i-1}[Q_{i-1}]$, it holds that $|N_{H_{i-1}}(v) \cap Q_{i-1}| = |N_{H_{i-1}}(v) \cap Q_i|$. Hence together with Property (M4), we obtain

$$\begin{aligned}
|N_{H_i}(v) \cap Q_i| &\geq |N_{H_{i-1}}(v) \cap Q_i| - 2(|\mathcal{Q}_i| - 1) \\
&= |N_{H_{i-1}}(v) \cap Q_{i-1}| - 2(|\mathcal{Q}_i| - 1) \\
&\geq f(v) - (\sum_{1 \leq j \leq i-1} 2(|\mathcal{Q}_j| - 1)) - 2(|\mathcal{Q}_i| - 1) \\
&= f(v) - (\sum_{1 \leq j \leq i} 2(|\mathcal{Q}_j| - 1))
\end{aligned}$$

as required. This completes the proof of the lemma. □

Recall that $f(v)$ is at least $n/g(n)$ for each $v \in V(G)$. Our next step is to show that the length of the maximal sequence constructed by our algorithm does not exceed $g(n) + 1$.

Lemma 18 *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 $g(n) + 1$.*

Proof. The claim clearly holds for $\mathcal{Q}_0$. It also holds for $\mathcal{Q}_1$ because the parts in $\mathcal{Q}_1$ correspond to the components of H_0, which are at most $g(n)$ due to Fact 6. Assume for a contradiction that the claim does not hold and let $\mathcal{S} = (H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ be a maximal sequence satisfying (M1)–(M4) witnessing this and let ℓ be the smallest integer such that $|\mathcal{Q}_\ell| > g(n) + 1$. Then, $\ell > 1$ and $|\mathcal{Q}_{\ell-1}| \leq g(n) + 1$. Because $|\mathcal{Q}_0| = 1$ and for every i, $|\mathcal{Q}_i|$ is larger than $|\mathcal{Q}_{i-1}|$, we obtain that $i \leq |\mathcal{Q}_i| - 1$ for every i. Hence, $\ell - 1 \leq |\mathcal{Q}_{\ell-1}| - 1 \leq g(n) + 1 - 1 = g(n)$ or in other words $\ell \leq g(n) + 1$.

From Lemma 17, we obtain that

$$\begin{aligned}
|N_{H_{\ell-1}}(v) \cap Q| &\geq f(v) - \sum_{1 \leq j < \ell} 2(|\mathcal{Q}_j| - 1) \\
&\geq \frac{n}{g(n)} - \sum_{1 \leq j < \ell} 2(g(n)) \\
&\geq \frac{n}{g(n)} - 2(\ell - 1)g(n) \\
&\geq \frac{n}{g(n)} - 2(g(n))^2
\end{aligned}$$

for every $Q \in \mathcal{Q}_{\ell-1}$ and $v \in Q$. This implies that every component of $H_{\ell-1}[Q]$ for some $Q \in \mathcal{Q}_{\ell-1}$, and hence also every part of $\mathcal{Q}_\ell$ has size at least $n/g(n) - 2g(n)^2 + 1$. Since $|\mathcal{Q}_\ell| > g(n) + 1$, we conclude that the number n of vertices of G

is greater than $(n/g(n) - 2g(n)^2 + 1)(g(n) + 1)$. Rearranging for n we obtain that $n < 2g(n)^4 + 2g(n)^3 + g(n)^2 + g(n) < 6g(n)^4$ which contradicts our assumption that $n \geq 6g(n)^4$. Since $\mathcal{Q}_i$ is a proper refinement of $\mathcal{Q}_{i+1}$ for every i with $1 \leq i < k$ and $|\mathcal{Q}_0| = 1$, we infer that the length of the sequence $\mathcal{S}$ is at most $g(n) + 1$. This completes the proof of the lemma. □

We are now ready to prove the main theorem of this section which outlines how the running time of CONNECTED g-BOUNDED f-FACTOR is dominated by the PARTITION CONNECTOR module.

Proof of Theorem 13. We present an algorithm for CONNECTED g-BOUNDED f-FACTOR that employs an algorithm for PARTITION CONNECTOR as a subroutine. All parts of the algorithm apart from the subroutine PARTITION CONNECTOR will be deterministic and run in polynomial time. The main idea is to construct a maximal sequence $\mathcal{S} = (H_0, \mathcal{Q}_0), \ldots, (H_k, \mathcal{Q}_k)$ satisfying properties (M1)–(M4). Recall our assumption that $n \geq 6g(n)^4$. Let (G, f) be an instance of CONNECTED g-BOUNDED f-FACTOR. The algorithm starts by computing an arbitrary f-factor H_0. If no f-factor exists, then clearly the algorithm reports failure. If on the other hand the computed f-factor H_0 is already connected, then the algorithm returns H_0 and exits.

Observe that $(H_0, \mathcal{Q}_0)$, where $\mathcal{Q}_0 = \{V(G)\}$, is a valid starting pair for a sequence $\mathcal{S}$ satisfying properties (M1)–(M4). Furthermore, the algorithm extends the sequence $\mathcal{S}$ by adding successors as long as one exists. The sequence is extended by invoking a recursive subroutine **Restricted-f-factor** with parameters (G, f) and the most recently added pair $(H, \mathcal{Q})$ to compute a new pair $(H', \mathcal{Q}')$ that can be appended to the sequence, if one exists. Otherwise, the procedure concludes that $\mathcal{S}$ can no longer be extended, in which case it either returns a connected f-factor of G or reports nonexistence of one. The subroutine **Restricted-f-factor** works as follows.

The procedure starts by computing a refinement $\mathcal{Q}'$ of $\mathcal{Q}$ containing one part $V(C)$ for every component C in $H[Q]$ where Q is a part in $\mathcal{Q}$. If $\mathcal{Q}' = \mathcal{Q}$ then because of Fact 12, H already constitutes a connected f-factor of G and the procedure correctly returns H. Otherwise, the procedure calls the provided algorithm for PARTITION CONNECTOR on G, f, and $\mathcal{Q}'$ to obtain an f-factor H'' connecting $\mathcal{Q}'$. Note that if there does not exist an f-factor connecting $\mathcal{Q}'$, then Observation 14 implies that there does not exist a connected f-factor for G. Thus, if the provided algorithm for PARTITION CONNECTOR returns failure, then our procedure also returns failure, relying on Observation 14. Otherwise, observe that the pair $(H'', \mathcal{Q}')$ already constitutes a valid successor of the pair $(H, \mathcal{Q})$ in any sequence satisfying properties (M1)–(M3). To ensure Property (M4), the procedure now calls a polynomial time subroutine on the pairs $(H, \mathcal{Q})$ and $(H'', \mathcal{Q}')$ to obtain the desired f-factor H' connecting $\mathcal{Q}'$ and such that the pairs $(H, \mathcal{Q})$ and $(H', \mathcal{Q}')$ satisfy Property (M4). The existence of such a polynomial time subroutine is from Lemma 16. The procedure now calls itself on the pair $(H', \mathcal{Q}')$. This completes the description of the algorithm.

Note that given the correctness of the algorithm for PARTITION CONNECTOR the correctness of the algorithm follows from Lemma 15. Let us now analysis the running time of the algorithm. Apart from the calls to the provided subroutine for PARTITION CONNECTOR, all parts of the algorithm run in polynomial time. Because the algorithm calls the provided algorithm for PARTITION CONNECTOR at most once for

every pair $(H, \mathcal{Q})$ in a maximal sequence satisfying properties (M1)–(M4), we obtain from Lemma 18 that the number of those calls is bounded by $g(n)+1$. Moreover, from the same lemma, we obtain that the size of a partition $\mathcal{Q}$ given as an input to the algorithm for PARTITION CONNECTOR is at most $g(n) + 1$. Hence, if PARTITION CONNECTOR can be solved in time $\mathcal{O}^*(n^{2(|\mathcal{Q}|-1)})$, then the algorithm runs in time $\mathcal{O}^*(g(n)n^{2(g(n))})$ showing the first statement of the theorem. Similarly, if PARTITION CONNECTOR can be solved in time $\mathcal{O}^*(2^{|\mathcal{Q}|})$, then the algorithm runs in time $\mathcal{O}^*(g(n)2^{g(n)+1})$. Thus given $g(n) \in \mathcal{O}(\log n)$, we have the polynomial time algorithm claimed in the second statement of the theorem. The following lemma (proved in Section 4) completes the proof of the second part of the theorem.

Lemma 19 *The* PARTITION CONNECTOR *can be solved by a randomized algorithm with running time* $\mathcal{O}^*(2^{|\mathcal{Q}|})$ *and error probability* $\mathcal{O}(1 - (1 - \frac{1}{n^2})^{|\mathcal{Q}|})$.

It remains to show that the randomized algorithm has the stated error probability. Towards this aim we calculate a lower bound on the success probability of the algorithm, i.e., the probability that the algorithm returns a connected f-factor of G if such an f-factor exists. Hence, let us suppose that G has a connected f-factor. It follows from Observation 14 that G contains an f-factor connecting $\mathcal{Q}$ for every partition $\mathcal{Q}$ of its vertex set. Hence every call to the subroutine **Partition Connector** is made for a "Yes"-instance, which together with Lemma 19 implies that every such call succeeds with probability at least $(1 - \frac{1}{n^2})^{|\mathcal{Q}|}$. Because $|\mathcal{Q}| \leq g(n) + 1 \in \mathcal{O}(\log n)$, we obtain from Lemma 5 that this probability is at least $(1 - \frac{c\lceil \log n \rceil^2}{n^2})$ for some constant c. Since there are at most $g(n) + 1 = c \log n$ such calls, the probability that the algorithm succeeds for all of these calls is hence at least $(1 - \frac{c\lceil \log n \rceil^2}{n^2})^{c \log n} > 0$, as required. This completes the proof of the theorem. □

3.2 A Quasi-polynomial 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 20 *For every* $c > 0$ *and function* $g(n) \in \mathcal{O}((\log n)^c)$*, the* CONNECTED g-BOUNDED f-FACTOR *problem can be solved in* $\mathcal{O}^*(n^{2g(n)})$ *time.*

We make use of the following simple lemma.

Lemma 21 *Let* G *be a graph having a connected* f*-factor. Let* $\mathcal{Q}$ *be a partition of the vertex set* $V(G)$*. 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 H' be a connected f-factor of G. For any partition $\mathcal{Q}$ of the vertex set, it follows from Observation 14 that $H'/\mathcal{Q}$ is connected. Consider a spanning tree T of $H'/\mathcal{Q}$. 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 Fact 8. □

In light of Theorem 13, it now suffices to prove the following Lemma 22, from which Theorem 20 immediately follows.

Lemma 22 PARTITION CONNECTOR *can be solved in time* $\mathcal{O}^*(n^{2(|\mathcal{Q}|-1)})$.

Proof. It follows from Lemma 21 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 13, it is sufficient for us to provide a randomized algorithm for PARTITION CONNECTOR with running time $\mathcal{O}^*(2^{|\mathcal{Q}|})$ and error probability $\mathcal{O}(g(n)^2/n^2)$. This is precisely what we do in the rest of this section (Lemma 19). As a first step, we 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, $f : V(G) \to \mathbb{N}$, and a partition $\mathcal{Q}$ of $V(G)$.
Question: Is there an f-factor of G that connects $\mathcal{Q}$?

We 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 $g(n) \in \mathcal{O}(\log n)$.

Lemma 23 *There exists an algorithm that, given the graph* G*, a function* $f : V(G) \to \mathbb{N}$*, and a partition* $\mathcal{Q}$ *of* $V(G)$*, runs in time* $\mathcal{O}^*(2^{|\mathcal{Q}|})$ *and outputs*

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

We design this algorithm by starting from the exact-exponential algorithm in [17] and making appropriate modifications. During the description, we point out the main differences between our algorithm and that in [17]. 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 [17], but since they are required in the description and proof of correctness of our algorithm, we state them here.

Definition 24 (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 constructed 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 25 (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 from H. 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 2).

Lemma 26 ([22]) *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 [23,9,17].

Definition 27 (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

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

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

Following terminology in [17], 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. Let $\det \mathcal{T}(G)$ denote the determinant of the Tutte matrix of the graph G.

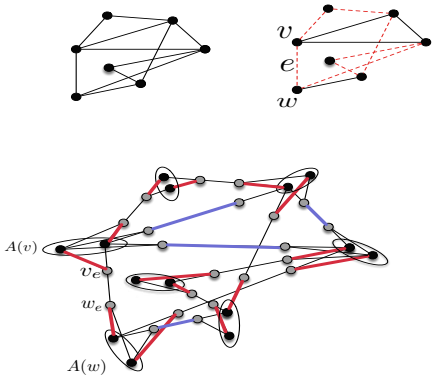

Fig. 2 An illustration of a graph G with a 2-factor H (the red dashed edges) and one possible corresponding perfect matching in $\mathcal{B}(G)$ (thick edges). It is important to note that an edge $e = (v, w)$ is *not* in H if and only if the edge (v_e, w_e) is present in the corresponding perfect matching.

Proposition 28 ([20]) $\det \mathcal{T}(G)$ *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.*

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,

$$\det \mathcal{T}(G) = \text{perm}(\mathcal{T}(G)) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} \mathcal{T}(G)(i, \sigma(i)), \tag{3}$$

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. We formally state and give a proof of the latter fact for the sake of completeness and because we intend to use this particular formulation of it.

Lemma 29 *Let* G *be a graph and let* $\mathcal{T}(G)$ *be the Tutte Matrix of* G *as in Definition 27. Let* $\det \mathcal{T}(G)$ *denote the determinant of* $\mathcal{T}(G)$*. Then the following statements hold.*

1. *If* $M = \{(i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell)\}$ *is a perfect matching of* G *then the product* $\prod_{(i_k, j_k) \in M} x_{i_k j_k}^2$ *appears exactly once in the naive expansion of* $\det \mathcal{T}(G)$

as in Equation 3 and hence as a surviving monomial in the sum on the right-hand side when expanded and simplified over any field of characteristic two.

2. *Conversely, if* $\det \mathcal{T}(G)$ *is expanded as in Equation 3 and if* $\det \mathcal{T}(G)$ *is not identically zero when simplified over any field of characteristic two, then each surviving monomial in the simplified expansion of* $\det \mathcal{T}(G)$ *must be of the form* $\prod_{(i_k,j_k)\in M} x_{i_k j_k}^2$ *where* $M = \{(i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell)\}$ *is a perfect matching of* G*.*

Proof. For the first statement, consider the permutation $\sigma \in S_n$ comprising precisely the 2-cycles $\{(i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell)\}$. The corresponding monomial given by the definition of $\mathcal{T}(G)$ over a field of characteristic two is precisely $\prod_{(i_k,j_k)\in M} x_{i_k j_k}^2$. For every other permutation $\sigma' \in S_n$, the corresponding monomial given by the definition of $\mathcal{T}(G)$ contains at least one variable $x_{i_r j_r}$ where i_r is not mapped to j_r in σ. This implies that no other monomial in the naive expansion of Equation 3 is equal to $\prod_{(i_k,j_k)\in M} x_{i_k j_k}^2$ even when considered over a field of characteristic two. This completes the argument for the first statement.

We now consider the second statement. First of all, since we only consider simple graphs, we have that for any permutation $\sigma \in S_n$ with a fixed point, the corresponding monomial is 0 since $x_{ii} = 0$ for every $i \in |V(G)|$. Let $S_n^{\geq 3}$ denote the set of all permutations in S_n with a cycle of length at least 3. We now argue that for any permutation $\sigma \in S_n^{\geq 3}$, the corresponding monomial vanishes in the simplified expansion of Equation 3. In order to do so, we give a bijection $\beta : S_n \to S_n$ such that (a) for every $\sigma \in S_n \setminus S_n^{\geq 3}$, $\beta(\sigma) = \sigma$, (b) for every $\sigma \in S_n$, $\beta(\beta(\sigma)) = \sigma$, and (c) for every $\sigma \in S_n$, the monomials corresponding to σ and $\beta(\sigma)$ are equal over any field of characteristic two.

We first define $\beta(\sigma)$ for a $\sigma \in S_n^{\geq 3}$ as follows. Note that we have already fixed an ordering of the vertices of G. Let v be the first vertex of G in this ordering which appears in a cycle of length at least 3 in σ and let C denote this cycle. We now define $\beta(\sigma)$ to be the permutation obtained from σ by inverting C and leaving every other cycle unchanged. Finally, for every $\sigma \in S_n \setminus S_n^{\geq 3}$, simply set $\beta(\sigma) = \sigma$.

It is straightforward to see that the resulting mapping β is indeed a bijection and moreover, $\beta(\beta(\sigma)) = \sigma$ for every $\sigma \in S_n$ as required. Finally, it follows from the definition of $\det \mathcal{T}(G)$ that over a field of characteristic two, the factor of the monomial corresponding to σ contributed by any cycle C is the same as that contributed by the inverse of this cycle to the monomial corresponding to $\beta(\sigma)$. Hence we have the third property and conclude that for any permutation $\sigma \in S_n^{\geq 3}$, the corresponding monomial vanishes in the simplified expansion of Equation 3. This implies that the only surviving monomials are those corresponding to permutations in $S_n \setminus S_n^{\geq 3}$ without a fixed point, implying that these permutations comprise only 2-cycles. This in turn implies that any such surviving monomial must correspond to a perfect matching of G as required. This completes the proof of the lemma. □

Lemma 30 (Schwartz-Zippel Lemma, [19,25]) *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 31 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_{ij}^2 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 24 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:

$$P_{\mathcal{Q}}(\bar{x}) = \sum_{\{1\} \subseteq I \subseteq [\ell]} (\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])) \cdot (\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])) \cdot m_I, \tag{4}$$

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

Remark 32 The definition of the polynomial $P_{\mathcal{Q}}(\bar{x})$ is the main difference between our algorithm and the algorithm in [17]. 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 33 *Given values for the variables x_{ij} in 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 $\mathcal{O}^*(2^\ell)$.*

Proof. The algorithm to evaluate $P_{\mathcal{Q}}(\bar{x})$ over the field $\mathbb{F}$ proceeds as follows. Given the values for the variables x_{ij} in the matrix $\mathcal{T}(\mathcal{B})$, we go over all $\{1\} \subseteq I \subseteq [\ell]$ and for each I, we evaluate $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$ and $\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$ in polynomial time via standard polynomial time determinant computation. Once this value is computed, we multiply their product with the evaluation of the monomial m_I. Since we go over 2^ℓ possible sets I and for each I the computation takes polynomial time, the claimed running time follows. □

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 34 We say that an f-factor H of G *contributes* a monomial $x^2_{i_1j_1} \dots x^2_{i_rj_r}$ to the naive expansion of the right-hand side of Equation 4 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 $m_1 \cdot m_2 \cdot m_I = x^2_{i_1j_1} \dots x^2_{i_rj_r}$.

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 4.

Lemma 35 *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 4 is contributed by an f-factor of G to the naive expansion of the right-hand size of Equation 4. 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 4.*
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 4.*

Proof. For the first statement, let m be a monomial which survives in the simplified expansion of the right-hand side of Equation 4. Then it must be of the form $x^2_{i_1j_1} \dots x^2_{i_rj_r}$ and must correspond to a perfect matching of $\mathcal{T}(\mathcal{B})$. This is a direct consequence of Lemma 29 (2). Let M be this perfect matching. We now define an f-factor H based on M and argue that H indeed contributes this monomial m to the naive expansion of the right-hand size of Equation 4 as per Definition 34. The f-factor H is defined as follows. An edge $(v, w) \in E(G)$ is in H if and only if the edge (v_e, w_e) is not in M. We now argue that H contributes m.

Consider the first condition in Definition 34. Since $e = (v, w) \in E(H)$, it must be the case that $(v_e, w_e) \notin M$. Since M is a perfect matching and the vertices v_e and w_e each have exactly one neighbor other than each other, it must be the case that M contains edges e_1 and e_2 where $e_1 = (u, v_e)$ for some $u \in A(v)$ and $e_2 = (u', w_e)$ for some $u' \in A(w)$. The fact that the second condition is satisfied follows directly from the definition of H. For the third condition, suppose that for some $1 \leq p, q \leq r$, and $e = (u, v) \in E(G)$, it holds that $\{u, v_e\} = \{i_p, j_p\}$ and $\{u', w_e\} = \{i_q, j_q\}$. The fact that M corresponds to m implies that the edges (u, v_e) and (u', w_e) are in M, which in turn implies that the edge (v_e, w_e) is *not* in M. Hence, by definition of H,

we conclude that $e \in E(H)$. An analogous argument implies that the fourth condition is satisfied as well. We now come to the final condition. Suppose that $1 \in I \subseteq [\ell]$ such that H has no edge crossing the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$. Now, observe that for every $(v, w) \in E(G)$ which crosses the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$ the edge $e \notin E(H)$, which by definition implies that $(v_e, w_e) \in M$. We define $\hat{M}$ to be the subset of edges $(v_e, w_e) \in M$ which cross the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$. Hence, for every edge (v_e, w_e) in $M \setminus \hat{M}$, the vertices v and w lie on the same side of the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$. We now define a partition $M' \uplus M''$ of $M \setminus \hat{M}$ as follows. For $v \in V(G)$ and $u \in V(\mathcal{B})$, an edge $(u, v_e) \in M$ is in M' if and only if $v \in \mathcal{Q}(I)$. Clearly, $M' \uplus M'' \uplus \hat{M}$ is now a partition of M. Furthermore, it is easy to see that M' is a perfect matching of $\mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$, M'' is a perfect matching of $\mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$.

Due to Proposition 28, we know that M' corresponds to a surviving monomial m' in the simplified expansion of $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$ and M'' corresponds to a surviving monomial m'' in the simplified expansion of $\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$. Finally, let $\hat{m}$ denote the monomial $\prod_{(i_k, j_k) \in \hat{M}} x^2_{i_k j_k}$. It is easy to see that $m = m' \cdot m'' \cdot \hat{m}$. Furthermore, $\hat{m} = m_I$. Hence we conclude that m is indeed contributed by H and proceed to the remaining two statements of the lemma. However, before we prove the remaining statements, we need the following claim.

Claim Let $1 \in I \subseteq [\ell]$.

1. If there is no edge of H crossing the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$, then each monomial contributed by H to the naive expansion of the polynomial $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)]) \cdot \det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}]) \cdot m_I$ is contributed exactly once.
2. If there is an edge of H crossing the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$ then H does not contribute a monomial to the naive expansion of the polynomial $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)]) \cdot \det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}]) \cdot m_I$.

Proof. We begin with the proof of the first statement. By Definition 34 it holds that every monomial contributed by H contains m_I. Let H' be the subgraph of H induced on $\mathcal{Q}(I)$ and let H'' be the subgraph of H induced on $\overline{\mathcal{Q}(I)}$. Observe that H' is an f-factor of $G[\mathcal{Q}(I)]$ and H'' is an f-factor of $G[\overline{\mathcal{Q}(I)}]$. By Proposition 28 and Lemma 26, we know that every f-factor of $G[\mathcal{Q}(I)]$ ($G[\overline{\mathcal{Q}(I)}]$) appears exactly once in the naive expansion of $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$ ($\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$) (since it is nothing but a perfect matching of the f-blowup induced by $\mathcal{Q}(I)$ or $\overline{\mathcal{Q}(I)}$).

Therefore, each monomial corresponding to a perfect matching of $\mathcal{B}[\mathcal{Q}(I)]$ which is equivalent to H' appears exactly once in the naive expansion of the polynomial $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$; similarly, each monomial corresponding to a perfect matching of $\mathcal{B}[\overline{\mathcal{Q}(I)}]$ which is equivalent to H'' appears exactly once in the naive expansion of $\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$. Since every monomial contributed by H to the naive expansion of $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)]) \cdot \det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}]) \cdot m_I$ is a product of m_I and a monomial each from $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$ and $\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$, and these monomials themselves occur exactly once in the naive expansion of $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)])$ and $\det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}])$ respectively, the first statement follows.

We now prove the second statement of the claim. Here, there must be vertices $v, w \in V(G)$ such that $v \in \mathcal{Q}(I)$, $w \in \overline{\mathcal{Q}(I)}$ and $(v, w) \in H$. Therefore, by Definition 34, we have that no monomial contributed by H has the term x^2_{jk} where

$\{j,k\} = \{v_e, w_e\}$. However, m_I contains the term x_{jk}^2 by definition. Therefore, H does not contribute a monomial to $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)]) \cdot \det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}]) \cdot m_I$. This completes the proof of the claim. □

Let α be the number of connected components of the graph $H/\mathcal{Q}$. If H is an f-factor of G that does not connect $\mathcal{Q}$ it must be the case that $\alpha > 1$. Due to the above claim, observe that there are exactly 2^α sets I such that H contributes each of its monomials exactly once to the simplified expansion of the right hand side of Equation 4 and H does not contributes any monomials to any other sets I. Since 2^α is even for $\alpha \geq 1$, we conclude that Statement 1 holds.

We now move on to Statement 2. That is, we assume that H is an f-factor that connects $\mathcal{Q}$. Due to the above claim, we know that H does not contribute a monomial to any polynomial $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)]) \cdot \det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}]) \cdot m_I$ where $1 \in I \subset [\ell]$ is such that H has an edge which crosses the cut $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$. However, since H connects $\mathcal{Q}$, it crosses *every* $(\mathcal{Q}(I), \overline{\mathcal{Q}(I)})$ cut where $1 \in I \subset [\ell]$. But observe that since H is an f-factor of G it will contribute a monomial to the polynomial $\det \mathcal{T}(\mathcal{B}[\mathcal{Q}(I)]) \cdot \det \mathcal{T}(\mathcal{B}[\overline{\mathcal{Q}(I)}]) \cdot m_I$ when $I = [\ell]$. Hence, we conclude that any monomial contributed by H occurs exactly once in the naive expansion of the right-hand side of Equation 4, completing the proof of the lemma. □

This implies the following result, which is the last ingredient we need to prove Lemma 23.

Lemma 36 *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 23. 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 $\exists$-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 36, 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 $\exists$-PARTITION CONNECTOR, we now return to the algorithm for the computational version, PARTITION CONNECTOR.

4.2 Solving PARTITION CONNECTOR in Randomized Polynomial Time

Proof of Lemma 19. 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 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 23 as $\mathcal{A}'$. If $\ell > 1$,

then $\mathcal{A}$ first calls $\mathcal{A}'$ and outputs NO if $\mathcal{A}'$ 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 $e = (v, w)$ between Q_1 and some Q_i ($i \neq 1$), it will compute a subinstance $(G^e, f^e, \mathcal{Q}^e)$ defined by setting:

- $G^e = G - e$, and
- $f^e(v) = f(v) - 1$, $f^e(w) = f(w) - 1$ and $f^e = f$ for all the remaining vertices of G, and
- $\mathcal{Q}^e$ is obtained from $\mathcal{Q}$ by merging Q_1 and Q_i into a new set; formally (assuming $i < \ell$), $\mathcal{Q}^e = \{Q_1^e = Q_1 \cup Q_i, Q_2, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_\ell\}$.

Intuitively, each such new instance corresponds to forcing the f-factor to choose the edge e. $\mathcal{A}$ then queries $\mathcal{A}'$ on $(G^e, f^e, \mathcal{Q}^e)$. If $\mathcal{A}'$ answers NO for each such tuple $(G^e, f^e, \mathcal{Q}^e)$ obtained from each edge e in E_1, then $\mathcal{A}$ immediately terminates and answers NO. Otherwise, let e be the first edge where $\mathcal{A}'$ answered YES; then $\mathcal{A}$ adds 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 $|\mathcal{Q}^e| > 1$ then $\mathcal{A}$ restarts the recursive procedure with $(G, f, \mathcal{Q}) := (G^e, f^e, \mathcal{Q}^e)$; observe that $|\mathcal{Q}^e| \leq |\mathcal{Q}| - 1$.

Before arguing correctness, we show that the algorithm runs in the required time. Since each edge in the partitioning is processed at most ℓ times, the runtime of $\mathcal{A}$ is asymptotically upper-bounded by its at most $\ell \cdot n^2 \leq n^3$ many calls to $\mathcal{A}'$. From Lemma 23, we then conclude that the total runtime of $\mathcal{A}(G, f, \mathcal{Q})$ is upper-bounded by $2^\ell \cdot n^{\mathcal{O}(1)}$.

For correctness, let us first consider the hypothetical situation where $\mathcal{A}'$ always answers correctly. If no partition connector exists, then $\mathcal{A}$ correctly outputs NO after the first call to $\mathcal{A}'$. Otherwise, there exists a partition connector, and such a partition connector must contain at least one edge in E_1 at every recursion of the algorithm. This implies that $\mathcal{A}'$ would output YES for at least one edge e of E_1. Moreover, it is easily seen that for any partition connector T containing e, $T \setminus \{e\}$ is also a partition connector in $(G^e, f^e, \mathcal{Q}^e)$, and so by the same argument $\mathcal{A}'$ would also output YES for at least one edge in the individual sets E_1 constructed in the recursive calls of $\mathcal{A}$. In particular, if $\mathcal{A}'$ would always answer correctly, then $\mathcal{A}$ would correctly output a partition connector $H \cup F$ at the end of its run. For further considerations, let us fix the set F which would be computed by $\mathcal{A}$ under the assumption that $\mathcal{A}'$ always answers correctly; in other words, F is the lexicographically first tuple of edges in E_1 which intersects a partition connector.

We are now ready to argue that $\mathcal{A}$ succeeds with the desired probability; recall that $\mathcal{A}'$ only allows one-sided errors. So, if the input is a no-instance, then $\mathcal{A}$ is guaranteed to correctly output NO after the first query to $\mathcal{A}'$. Furthermore, by the definition of F, for each edge $e \notin F$ processed by $\mathcal{A}$, the algorithm $\mathcal{A}'$ must also answer NO on $(G^e, f^e, \mathcal{Q}^e)$. So, assuming $\mathcal{A}'$ always answers correctly, in total $\mathcal{A}'$ would only be called at most times on yes-instances, and in all remaining calls it receives a no-instance. Given that $\mathcal{A}'$ has a success probability of at least $1 - \frac{1}{n^2}$, the probability that $\mathcal{A}'$ is called at most $|F| + 1 = \ell$ times on YES-instances (not counting the initial call on G), and that it succeeds in all these calls, is at least $(1 - \frac{1}{n^2})^\ell$. Hence the error

probability of the algorithm is at most $1-(1-\frac{1}{n^2})^{\ell}$. This completes the proof of the lemma. □

5 Classification Results

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

Theorem 4 *For every $c > 1$ and for every $g(n) \in \Theta((\log n)^c)$,* 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.

Lemma 37 (Exponential Time Hypothesis (ETH), [11]) *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 38 *For every $c > 1$ and for every $g(n) \in \Theta((\log n)^c)$,* CONNECTED g-BOUNDED f-FACTOR *is not* NP*-hard unless the Exponential Time Hypothesis fails.*

Proof. Due to Theorem 2, we know that when $g(n) \in \Theta((\log n)^c)$, 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. □

Following lemma uses a reduction from HAMILTONIAN CYCLE to come up with a hardness result.

Lemma 39 *For every $c > 1$ and for every $g(n) \in \Theta((\log n)^c)$,* CONNECTED g-BOUNDED f-FACTOR *is not in* P *unless the Exponential Time Hypothesis fails.*

Proof. Assume for a contradiction that CONNECTED g-BOUNDED f-FACTOR is in P for some $g(n) \in \Theta((\log n)^c)$ and $c > 1$. Let us fix this function g for the remainder of the proof. In particular, there exists constants c_1 and $\epsilon > 0$ such that $g(n) \geq c_1(\log n)^{1+\epsilon}$ for sufficiently large n. The proof is structured as follows. First, we present a *subexponential time* reduction from HAMILTONIAN CYCLE to CONNECTED g-BOUNDED f-FACTOR. We then show that such a reduction would imply a subexponential time algorithm for HAMILTONIAN CYCLE, which is known to violate ETH [11].

The reduction algorithm R_ϵ takes a graph G on z vertices as input, computes $s = \frac{2^{(\frac{z}{c_1})^{1/(1+\epsilon)}}}{z}$, and outputs an n-vertex instance (G', f) of CONNECTED g-BOUNDED f-FACTOR which satisfies the following conditions:

1. $f(v) \geq \frac{n}{c_1(\log n)^{1+\epsilon}}$ for every v in G'.
2. n is upper-bounded by a subexponential function of z.
3. G has a Hamiltonian cycle if and only if (G', f) contains a connected f-factor.

Crucially, observe that for sufficiently large z, we have $s > z$. The algorithm R_ϵ works as follows. Given a graph G, for each vertex v it constructs a clique C_v of size $\lceil s \rceil - 1$ and makes each vertex in C_v adjacent to v. This construction is very similar in spirit to the hardness reduction in [5, Theorem 5.2]. For each vertex $x \in C_v$, it sets $f(x) = \lceil s \rceil - 1$, while for v it sets $f(v) = \lceil s \rceil + 1$.

Next, we argue that (G', f) satisfies conditions (1), (2) and (3). For Condition (1), we need to ensure that the bound on f holds for vertices in each C_v, meaning that we need to verify that $\lceil s \rceil - 1 \geq \frac{n}{c_1(\log n)^{1+\epsilon}} = \frac{\lceil s \rceil \cdot z}{c_1(\log(\lceil s \rceil \cdot z))^{1+\epsilon}}$ holds. By plugging in the definition of s, we obtain

$$\frac{\lceil s \rceil \cdot z}{c_1(\log(\lceil s \rceil \cdot z))^{1+\epsilon}} = \frac{\lceil s \rceil \cdot z}{c_1(\log \lceil \frac{2^{(\frac{z}{c_1})^{1/(1+\epsilon)}}}{z} \rceil \cdot z)^{1+\epsilon}}.$$

Because $s > z$, we obtain

$$\frac{\lceil s \rceil \cdot z}{c_1(\log(\frac{2^{(\frac{z}{c_1})^{1/(1+\epsilon)}}}{z} \cdot z))^{1+\epsilon}} \geq \frac{\lceil s \rceil \cdot z}{c_1(\log(\lceil \frac{2^{(\frac{z}{c_1})^{1/(1+\epsilon)}}}{z} \rceil \cdot z))^{1+\epsilon}}.$$

The following equation shows that $\lceil s \rceil - 1$ is at least equal to the left expression.

$$\frac{\lceil s \rceil \cdot z}{c_1(\log(\frac{2^{(\frac{z}{c_1})^{1/(1+\epsilon)}}}{z} \cdot z))^{1+\epsilon}} = \frac{\lceil s \rceil \cdot z}{z \cdot (\log 2)^{1+\epsilon}} = \frac{\lceil s \rceil}{(\log 2)^{1+\epsilon}}$$

Since $\lceil s \rceil - 1 \geq \frac{\lceil s \rceil}{(\log 2)^{1+\epsilon}}$, Condition (1) holds. For Condition (2), it suffices to note that $n = \lceil s \rceil \cdot z = z \cdot \lceil \frac{2^{(\frac{z}{c_1})^{1/(1+\epsilon)}}}{z} \rceil$, which is clearly a subexponential function.

Finally, for Condition (3), observe that every edge in each clique C_v must be used in every connected f-factor of (G', f). Furthermore, all the other edges in every such connected f-factor must induce a connected subgraph of G with degree 2, which is a Hamiltonian cycle. Hence there is a one-to-one correspondence between Hamiltonian cycles in G and connected f-factors of (G', f), and Condition (3) also holds.

To complete the proof, recall that we assumed that there exists a polynomial time algorithm for CONNECTED g-BOUNDED f-FACTOR for our choice of g. Then, given an instance G of HAMILTONIAN CYCLE, we can apply R_ϵ on G followed by the hypothetical polynomial time algorithm on the resulting instance (G', f) (whose size is subexponential in $|V(G)|$) to solve G in subexponential time. As was mentioned earlier in the proof, such an algorithm would violate ETH. □

Lemmas 38 and 39 together give us Theorem 4.

6 Concluding remarks

We have come up with new complexity results for CONNECTED f-FACTOR with respect to lower bounds on the function f. As our main results, we have shown 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, 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 have obtained 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 when $g(n) = \mathcal{O}(\log n)$.

Acknowledgments. The authors wish to thank the anonymous reviewers for their helpful comments. The authors acknowledge support by the Austrian Science Fund (FWF, project P26696), and project TOTAL funded by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 677651). Robert Ganian is also affiliated with FI MU, Brno, Czech Republic.

References

1. Jin Akiyama and Mikio Kano. Factors and factorizations of graphsa survey. *Journal of Graph Theory*, 9(1):1–42, 1985.
2. R. E. Bellman. Dynamic programming treatment of the traveling salesman problem. *J. ACM*, 9:61–63, 1962.
3. F. Cheah and Derek G. Corneil. The complexity of regular subgraph recognition. *Discrete Applied Mathematics*, 27(1-2):59–68, 1990.
4. F. R. K. Chung and R. L. Graham. Recent results in graph decompositions. *London Mathematical Society, Lecture Note Series*, 52:103–123, 1981.
5. Kamiel Cornelissen, Ruben Hoeksma, Bodo Manthey, N. S. Narayanaswamy, C. S. Rahul, and Marten Waanders. Approximation algorithms for connected graph factors of minimum weight. *Theory of Computing Systems*, 62(2):441–464, Feb 2018.
6. 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.
7. 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 Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011.
8. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
9. Gregory Z. Gutin, Magnus Wahlström, and Anders Yeo. Rural postman parameterized by the number of components of required edges. *J. Comput. Syst. Sci.*, 83(1):121–131, 2017.
10. M. Held and R. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.
11. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001.

12. Tomáš Kaiser. A short proof of the tree-packing theorem. *Discrete Mathematics*, 312(10):1689–1691, 2012.
13. Mekkia Kouider and Preben D. Vestergaard. Connected factors in graphs - a survey. *Graphs and Combinatorics*, 21(1):1–26, 2005.
14. László Lovász and Michael D. Plummer. *Matching Theory*, volume 121 of *North-Holland Mathematics Studies*. Elsevier, 1986.
15. N. S. Narayanaswamy and C. S. Rahul. Approximation and exact algorithms for special cases of connected f-factors. In Lev D. Beklemishev and Daniil V. Musatov, editors, *Computer Science - Theory and Applications - 10th International Computer Science Symposium in Russia, CSR 2015, Listvyanka, Russia, July 13-17, 2015, Proceedings*, volume 9139 of *Lecture Notes in Computer Science*, pages 350–363. Springer, 2015.
16. Julius Petersen. Die Theorie der regulären graphs. *Acta Mathematica*, 15(1):193–220, 1891.
17. Geevarghese Philip and M. S. Ramanujan. Vertex exponential algorithms for connected f-factors. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 61–71. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.
18. Michael D. Plummer. Graph factors and factorization: 1985-2003: A survey. *Discrete Mathematics*, 307(7-8):791–821, 2007.
19. Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
20. William Thomas Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947.
21. William Thomas Tutte. The factors of graphs. *Canadadian Journal of Mathematics*, 4(3):314–328, 1952.
22. William Thomas Tutte. A short proof of the factor theorem for finite graphs. *Canadian Journal of Mathematics*, 6(1954):347–352, 1954.
23. Magnus Wahlström. Abusing the tutte matrix: An algebraic instance compression for the k-set-cycle problem. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, volume 20 of *LIPIcs*, pages 341–352. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
24. Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.
25. Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.