





Degrees and Gaps: Tight Complexity Results of General Factor Problems Parameterized by Treewidth and Cutwidth

Dániel Marx 

CISPA Helmholtz Center for Information Security,
Saarland Informatics Campus, Saarbrücken, Germany

Govind S. Sankar  

Indian Institute of Technology Madras, Chennai, India

Philipp Schepper  

CISPA Helmholtz Center for Information Security,
Saarland Informatics Campus, Saarbrücken, Germany
Saarbrücken Graduate School of Computer Science,
Saarland Informatics Campus, Germany

Abstract

In the GENERAL FACTOR problem, we are given an undirected graph G and for each vertex $v \in V(G)$ a finite set B_v of non-negative integers. The task is to decide if there is a subset $S \subseteq E(G)$ such that $\deg_S(v) \in B_v$ for all vertices v of G . Define the max-gap of a finite integer set B to be the largest $d \geq 0$ such that there is an $a \geq 0$ with $[a, a + d + 1] \cap B = \{a, a + d + 1\}$. Cornuéjols showed in 1988 that if the max-gap of all sets B_v is at most 1, then the decision version of GENERAL FACTOR is polynomial-time solvable. This result was extended 2018 by Dudyecz and Paluch for the optimization (i.e. minimization and maximization) versions. We present a general algorithm counting the number of solutions of a certain size in time $(M + 1)^{\text{tw}} n^{\mathcal{O}(1)}$, given a tree decomposition of width tw , where M is the maximum integer over all B_v . By using convolution techniques from van Rooij (2020), we improve upon the previous $(M + 1)^{3\text{tw}} n^{\mathcal{O}(1)}$ time algorithm by Arulsevan et al. from 2018.

We prove that this algorithm is essentially optimal for all cases that are not trivial or polynomial time solvable for the decision, minimization or maximization versions. Our lower bounds show that such an improvement is not even possible for B -FACTOR, which is GENERAL FACTOR on graphs where all sets B_v agree with the fixed set B . We show that for every fixed B where the problem is NP-hard, our $(\max B + 1)^{\text{tw}} n^{\mathcal{O}(1)}$ algorithm cannot be significantly improved: assuming the STRONG EXPONENTIAL TIME HYPOTHESIS (SETH), no algorithm can solve B -FACTOR in time $(\max B + 1 - \epsilon)^{\text{tw}} n^{\mathcal{O}(1)}$ for any $\epsilon > 0$. We extend this bound to the counting version of B -FACTOR for arbitrary, non-trivial sets B , assuming #SETH.

We also investigate the parameterization of the problem by cutwidth. Unlike for treewidth, having a larger set B does not appear to make the problem harder: we give a $2^{\text{cutw}} n^{\mathcal{O}(1)}$ algorithm for any B and provide a matching lower bound that this is optimal for the NP-hard cases.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases General Factor, General Matching, Treewidth, Cutwidth

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.95

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://arxiv.org/abs/2105.08980>

Funding Research supported by the European Research Council (ERC) consolidator grant No. 725978 SYSTEMATICGRAPH.



© Dániel Marx, Govind S. Sankar, and Philipp Schepper;
licensed under Creative Commons License CC-BY 4.0

48th International Colloquium on Automata, Languages, and Programming (ICALP 2021).

Editors: Nikhil Bansal, Emanuela Merelli, and James Worrell; Article No. 95; pp. 95:1–95:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

Matching problems for graphs are widely studied in computer science [1, 7, 11, 18, 19, 24, 27, 31, 32, 34]. The most prominent ones are PERFECT MATCHING (PERFMATCH) and MAXIMUM-WEIGHT MATCHING. Both problems have long known polynomial-time algorithms [19, 32] and various generalizations were investigated in the graph-theory literature. These range from simple extensions such as the k -factor problem for a positive integer k (every vertex has to be incident to exactly k edges) [2, 27], to more complex ones, where the vertices are assigned intervals [34]. These problems are generally solved by a reduction to PERFMATCH by replacing the vertices of the original instance with suitable gadgets. Lovász introduced a general version of these problems which we call GENERAL FACTOR [31]:

► **Definition 1.1** (GENERAL FACTOR (GENFAC)). *Let $G = (V, E)$ be an undirected node labeled graph where the label of a vertex v is a set $B_v \subseteq \mathbb{N}$. We say $S \subseteq E$ is a solution if $\deg_S(v) \in B_v$ for all $v \in V$. GENFAC is the problem of deciding whether G has a solution.*

The minimization and maximization versions of GENFAC are the problems of finding the size of the solution with smallest and largest cardinality, respectively.

Polynomial-Time Solvable Cases. For several cases (e.g. k -factor, sets are intervals) reductions to PERFMATCH are known, leading directly to polynomial-time algorithms. Cornuéjols analyzed the complexity of the general problem to identify properties of the sets that make the problem easier to solve [7]. For this he introduced the *gap* of a set: A gap is a finite sequence of consecutive integers not contained in the set but whose boundaries are contained in the set (cf. Definition 2.2). For example, the set $\{1, 5, 6, 8\}$ has gaps of size 3 and 1. For a set S , $\text{max-gap } S$ denotes the size of its largest gap. Cornuéjols showed that if the max-gaps of all sets are at most 1, then the problem is polynomial-time solvable. Later this result was extended to the maximization and minimization (optimization) versions of GENFAC.

► **Theorem 1.2** ([7, 18]). *The decision, maximization, and minimization version of GENFAC can be solved in polynomial time on arbitrary graphs if for all nodes v , $\text{max-gap } B_v \leq 1$.*

On the other side Cornuéjols proved GENFAC to be NP-complete if there are nodes with a gap of size two, namely $\{1\}$ and $\{0, 3\}$, by a reduction from *exact 3-cover*. More generally, it can be deduced from the work of Feder [22] that GENFAC becomes NP-complete whenever every set B_v is restricted to be the same fixed set B having gap size at least two.

Treewidth. This paper is part of a long sequence of works studying problems parameterized by treewidth and related metrics like cutwidth or cliquewidth. Treewidth received significant attention as many NP-hard problems like COLOURING, INDEPENDENT SET, or DOMINATING SET (see [4] for a survey) are polynomial-time solvable on bounded-treewidth graphs. Courcelle’s Theorem [8, 9] shows that a large class of graph problems can be solved in linear time on graphs of bounded treewidth. Recent developments on the algorithmic side include various techniques such as Cut & Count [15, 33], rank-based dynamic programming [3, 14, 23] and fast subset convolution [37, 38]. On the negative side, there have been a large number of results showing lower bounds based on complexity assumptions such as the EXPONENTIAL TIME HYPOTHESIS (ETH) and the STRONG EXPONENTIAL TIME HYPOTHESIS (SETH) [5, 12, 29, 30]. For many such problems, their optimal algorithms utilize some form of dynamic programming, where a “state” is stored for every node in the tree decomposition. The number of such states determines the running time of the algorithms, seemingly suggesting

that this number is a natural barrier to the running time of any algorithm. Typically, the conditional lower bounds confirm this intuition by showing that no algorithm can break this barrier.

New Faster Algorithms. One of the first algorithmic results for GENFAC parameterized by treewidth was given by Arulselvan et al. [1]. They present an algorithm for a restricted version of the problem where the sets contain zero and an interval of integers. This algorithm can be easily extended to handle arbitrary instances while preserving the running time of $(M + 1)^{3\text{tw}}n^{\mathcal{O}(1)}$ where M is the maximum over all sets assigned to the vertices. Their algorithm is based on the standard dynamic programming approach when parameterizing by treewidth, i.e. it considers all possible states for each node of the tree decomposition. The number of states in the dynamic programming is about $(M + 1)^{\text{tw}+1}$: one needs to keep track of the degree of the partial solution at each of the at most $\text{tw} + 1$ vertices of a bag of the tree decomposition, and this degree can be between 0 and M . Therefore, a natural question is whether the algorithm can be improved to obtain an $(M + 1)^{\text{tw}}n^{\mathcal{O}(1)}$ running time, matching the number of states. Such improvements are known for other problems, for example for DOMINATING SET and #PERFMATCH in [38]. We base our algorithm on the same dynamic programming idea, but instead of processing all combination of states at join nodes, we make use of the technique of van Rooij [37] to compute fast convolutions, avoiding this bottle-neck of the computation. The algorithm can be easily generalized to the optimization and counting versions as well; to unify the results, we present the algorithm in a way that counts all solutions of a certain given size.

► **Theorem 1.3.** *Given a GENFAC instance G and a tree decomposition of width tw . Let $M = \max_{v \in V(G)} \max B_v$. Then for all s , we can count the solutions of size exactly s in time $(M + 1)^{\text{tw}}n^{\mathcal{O}(1)}$.*

As we shall see, this algorithm is essentially optimal for every fixed B where B -FACTOR is NP-hard. Note that in order to obtain this optimal running time, we have to use a well-known, but non-trivial technique; beyond that, our algorithm does not provide new insights into the problem. Due to space constraints, we omit the algorithm here and refer the reader instead to the full version.

Tight Lower Bounds for the Decision Version. To investigate how the properties of the sets B_v influence the complexity of the problem, we give conditional lower bounds based on SETH for the restrictive B -FACTOR problem, where all sets have to be the same fixed set B . By a careful design our lower bounds also hold for a parameterization by *pathwidth*. Note that if the set is not fixed, Arulselvan et al. showed that GENFAC is W[1]-hard when parameterizing only by treewidth [1]. Thus, it is reasonable to focus only on the cases with fixed sets to prove tight lower bounds.

► **Theorem 1.4 (Lower Bound for Decision Version).** *Let $B \subseteq \mathbb{N}$ be a fixed, finite set with $0 \notin B$ and $\max\text{-gap } B > 1$. If, given a path decomposition of width pw , B -FACTOR can be solved in time $(\max B + 1 - \epsilon)^{\text{pw}}n^{\mathcal{O}(1)}$ for some $\epsilon > 0$ even on graphs with degree at most $2 \max B$, then SETH is false.*

The same result immediately follows for treewidth as for all graphs the pathwidth forms an upper-bound for the treewidth [13]. Hence, our algorithm is optimal not only for GENFAC but also for B -FACTOR parameterized by treewidth and does not allow major improvements.

Tight Lower Bounds for the Optimization Version. It suffices to consider the maximization version with max-gap $B > 1$ and $0 \in B$ for the optimization version. The other cases are either polynomial-time solvable (max-gap $B \leq 1$ or $0 \in B$ for MIN- B -FACTOR) [18] or the hardness directly follows from the lower bound for the decision version. Observe that the assumption $0 \in B$ does not make the problem trivially solvable. For these cases, we give essentially the same lower bound as for the decision version. Again the bound rules out that we can improve the given algorithm substantially; the running time is essentially optimal.

► **Theorem 1.5** (Lower Bound for Maximization Version). *Let $B \subseteq \mathbb{N}$ be a fixed, finite set with max-gap $B > 1$. If, given a path decomposition of width pw , MAX- B -FACTOR can be solved in time $(\max B + 1 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ for some $\epsilon > 0$ even on graphs with degree at most $2 \max B$, then SETH is false.*

Counting. It is well known that PERFMATCH can be solved in polynomial time [19]. Surprisingly, Valiant showed in [35] that *counting* the number of perfect matchings of a graph is as hard as counting satisfying assignment of a boolean formula. This is curious as (presumably) no polynomial-time algorithm for the decision version of the latter problem exists. The observation then led to the definition of the complexity class #P containing the counting problems whose corresponding decision version lies in NP. Indeed, this feature that some structures are easy to find but hard to count appears in our work as well. Apart from #PERFMATCH, which itself is #\{1\}-FACTOR, our results imply that # B -FACTOR is #P-hard for any finite, fixed B . This contrasts with the decision version, where the problem is easy when max-gap $B \leq 1$. Over and above showing #P-hardness, we show a tight lower bound for # B -FACTOR, assuming #SETH, the counting version of SETH. There have been several results [10, 11, 17] based on #SETH and #ETH. Some of our constructions were inspired by one such work by Curticapean and Marx [11], where they show a lower bound of $(2 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ for #PERFMATCH on graphs assuming #SETH. We prove a wide generalization of this result by providing a tight lower bound for every # B -FACTOR problem. As for the optimization and decision version, our algorithm shows the tightness of this lower bound.

► **Theorem 1.6** (Lower Bound for Counting Version). *Let $B \subseteq \mathbb{N}$ be a nonempty, fixed, and finite set such that $B \neq \{0\}$. If, given a path decomposition of width pw , # B -FACTOR can be solved in time $(\max B + 1 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ for some $\epsilon > 0$ even on graphs with degree at most $2 \max B + 6$, then #SETH is false.*

We also investigate #MAX- B -FACTOR, the problem of counting maximum-sized solutions. The following argument shows that #\{\max B\}-FACTOR can be reduced to #MAX- B -FACTOR without increasing pathwidth, hence Theorem 1.6 gives a lower bound of $(\max B + 1 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$. Consider an instance of #\{\max B\}-FACTOR on a graph G of pathwidth pw . In polynomial time, check if G has some \{\max B\}-FACTOR [7]. If it does not, then output 0. If it does, then solve #MAX- B -FACTOR on G . As now every maximum-sized B -factor is actually a \{\max B\}-factor, this indeed solves the #\{\max B\}-FACTOR problem.

► **Corollary 1.7.** *Let $B \subseteq \mathbb{N}$ be a fixed, finite set such that $B \neq \{0\}$. If, given a path decomposition of width pw , #MAX- B -FACTOR can be solved in time $(\max B + 1 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ for some $\epsilon > 0$ even on graphs with degree at most $2 \max B + 6$, then SETH is false.*

We leave open the question of a tight lower bound for the minimization version.

Parameterizing by Cutwidth. As previously mentioned, pathwidth and treewidth are not the only parameters used in parameterized complexity. Cutwidth, cliquewidth, genus, and crossing number are only a few more examples of a vast class of possible parameters. For cutwidth, we consider linear layouts of graphs with n vertices, which are just enumerations v_1, \dots, v_n of all graph vertices. Then the cut after vertex v_i consists of all edges in G with one end in $\{v_1, \dots, v_i\}$ and the other end in $\{v_{i+1}, \dots, v_n\}$. The *cutwidth* of a linear layout is the maximum over the size of the cut after every v_i . The cutwidth cutw of a graph is the minimum over the cutwidths of all possible linear layouts. As $\text{tw} \leq \text{pw} \leq \text{cutw}$, it is not completely surprising that we get different upper bounds for cutwidth. But now a simple dynamic program suffices to prove the upper bound for this case. Further, the running time of the algorithm is independent from the maximum of the set B .

► **Theorem 1.8.** *Given a linear layout of a GENFAC instance G with width cutw , for all s we can count the number of solutions of size exactly s in time $2^{\text{cutw}} n^{\mathcal{O}(1)}$.*

This again matches the number of states for each cut of the linear layout. Like before, we omit the algorithm here and refer the reader to the full version. Note that the running times appearing in Theorems 1.3 and 1.8 cannot be directly compared: the base is lower when parameterized by cutwidth, but cutwidth can be larger than treewidth.

By a modified high-level construction, we show matching lower bounds based on SETH for the decision and optimization versions, and, based on #SETH, for the counting version.

► **Theorem 1.9 (Lower Bounds for Cutwidth).** *Let $B \subseteq \mathbb{N}$ be a fixed, nonempty set of finite size. If, given a linear layout of width cutw , the following problems can be solved in time $(2 - \epsilon)^{\text{cutw}} n^{\mathcal{O}(1)}$ for any $\epsilon > 0$ even on graphs with degree at most $2 \max B + 6$, then SETH (resp. #SETH) fails: (1) B -FACTOR and MIN- B -FACTOR if $0 \notin B$ and $\max\text{-gap } B > 1$, (2) MAX- B -FACTOR if $\max\text{-gap } B > 1$, and (3) # B -FACTOR if $B \neq \{0\}$.*

2 Preliminaries

We introduce homogeneous graphs to formally define B -FACTOR.

► **Definition 2.1 (Homogeneous Graphs and B -FACTOR).** *Let $B \subseteq \mathbb{N}$ be some fixed, finite set. We say a node-labeled graph is B -homogeneous if for each node $v \in V$ it holds that $B_v = B$. Then B -FACTOR is the restriction of GENFAC to B -homogeneous graphs.*

This definition directly transfers to the optimization and counting version. We now formally introduce the max-gap of integer sets along with some other properties.

► **Definition 2.2.** *Let $B \subseteq \mathbb{N}$ be finite. We define $\max\text{-gap } B$ as the largest non-negative integer d such that there is an $a \in B$ with $[a, a + d + 1] \cap B = \{a, a + d + 1\}$.*

In this paper we regularly insert graphs into other graphs. To make this operation formal, we make use of *dangling edges*: these are edges that have only one endpoint. We denote a dangling edge with endpoint v by $(?, v)$. For the sake of completeness we now formally define this procedure of replacing the relations, i.e. the insertion of a graph into another graph.

► **Definition 2.3 (Insertion).** *Let $G = (V, E)$ be a graph and $v \in V$ be of degree k , with incident edges $e_1 = (v_1, v), \dots, e_k = (v_k, v)$ that are ordered in some fixed way. Let $H = (W, F)$ be a graph with dangling edges $d_1 = (?, u_1), \dots, d_k = (?, u_k)$ where the u_i are not necessarily pairwise distinct. Inserting H in G at v gives us a new graph $G' = (V', E')$ where:*

$$V' = (V \cup W) \setminus \{v\} \quad \text{and} \quad E' = (E \cup F) \setminus \{e_1, \dots, e_k, d_1, \dots, d_k\} \cup \{(v_1, u_1), \dots, (v_k, u_k)\}$$

All lower bounds we prove in this paper are based on the STRONG EXPONENTIAL TIME HYPOTHESIS. But instead of using the original statement we use a formulation which is more useful to work with.

► **Conjecture 2.4** (STRONG EXPONENTIAL TIME HYPOTHESIS (SETH) [6, 25]). *For all $\delta > 0$, there is a $k \geq 3$ such that satisfiability of k -CNF formulas on n variables requires more than $(2 - \delta)^n$ time.*

About Relations. A relation $R : \{0, 1\}^k \rightarrow \{0, 1\}$ can also be seen as a set $R' \subseteq \{0, 1\}^k$ such that $x \in R'$ iff $R(x) = 1$. We can also identify R with a set $R'' \subseteq 2^{[k]}$, where each element of R'' contains the positions of the 1s of an accepted input. Precisely, $x' = \{i \mid x[i] = 1\} \in R''$ iff $R(x) = 1$. We switch between these definitions depending on the context. Recall, a relation is *symmetric* if its output only depends on the Hamming weight of its input.

To simplify notation, we introduce the following generic classes of symmetric relations.

► **Definition 2.5.** *For a vector $x \in \{0, 1\}^k$, we define $\text{hw}(x)$ as the number of 1s in x , i.e. the Hamming weight of x . We define the following for $S \subseteq \mathbb{N}$, and $i, j \in \mathbb{N}$:*

$$\text{HW}_{\in S}^{(j)} := \{x \in \{0, 1\}^j \mid \text{hw}(x) \in S\} \quad \text{EQ}_j := \text{HW}_{\in \{0, j\}}^{(j)} \quad \text{HW}_{=i}^{(j)} := \text{HW}_{\in \{i\}}^{(j)}$$

EQ_j is the equality relation on j inputs. We use $\text{HW}_{\in S}$ to denote $\text{HW}_{\in S}^{(j)}$ when the arity j of the relation is implicit. We also use this as the set of the relations $\text{HW}_{\in S}^{(j)}$ for all $j \in \mathbb{N}$. We transfer this abuse of notation to $\text{HW}_{=i}$.

Note that assigning the relations $\text{HW}_{\in B}$ to a vertex corresponds to assigning the set B to the vertex. Which notation is used depends on the context we are in.

3 Lower Bound when Parameterizing by Pathwidth

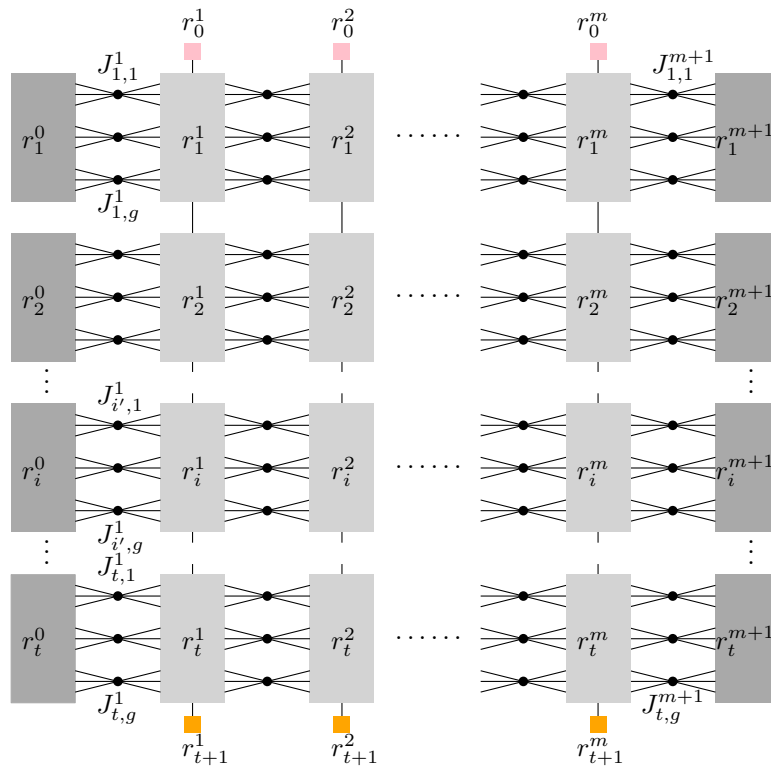
We show the lower bounds in two steps. The first step is a reduction from CNF-SAT to the intermediate problem B -FACTOR WITH RELATIONS. In the second step, we reduce to the actual version of B -FACTOR for which we want to show the lower bound. As the lower bounds are for pathwidth they immediately hold for treewidth as it can be upper-bounded by pathwidth for all graphs.

► **Definition 3.1** (B -FACTOR WITH RELATIONS (B -FACTOR $^{\mathcal{R}}$)). *Let $B \subseteq \mathbb{N}$ be fixed of finite size. $G = (V_S \cup V_C, E)$ is an instance of B -FACTOR WITH RELATIONS if all nodes in V_S are labeled with set B and all nodes $v \in V_C$ are labeled with a relation R_v that is given as a truth table such that the following holds:*

1. *Let $I(v)$ be the set of edges incident to v in G . Then $R_v \subseteq 2^{I(v)}$.*
2. *There is an even $c_v > 0$ such that for all $x \in R_v$ we have $\text{hw}(x) = c_v$.*

A set $\hat{E} \subseteq E$ is a solution for G if (1) for $v \in V_S$: $\deg_{\hat{E}}(v) \in B$ and (2) for $v \in V_C$: $I(v) \cap \hat{E} \in R_v$. B -FACTOR $^{\mathcal{R}}$ is the problem of deciding if such an instance has a solution. We call V_S the set of simple nodes and V_C the set of complex nodes.

Using this intermediate problem, we can formally state the first part of the reduction. The lower bound needs a careful formulation: when we reduce B -FACTOR $^{\mathcal{R}}$ to B -FACTOR by inserting gadgets realizing the relations at the complex nodes, the size and pathwidth of the graph can increase significantly. Therefore, we state a stronger lower bound that can tolerate additional terms to take care of such increases. The key point is that this increase is mainly influenced by the total degree of the complex nodes in a bag of the path decomposition.



■ **Figure 1** An example illustrating the construction from the proof of Theorem 3.2. The simple nodes are represented by circles while the complex nodes are represented by boxes.

► **Theorem 3.2.** *Let $B \subseteq \mathbb{N}$ be a fixed set of finite size with $B \neq \{0\}$. Given a B -FACTOR^R instance along with a path decomposition of width pw such that $\Delta^* = \max_{bag} X \sum_{v \in X \cap V_C} \deg(v)$. Assume B -FACTOR^R can be solved in $(\max B + 1 - \epsilon)^{pw + f_B(\Delta^*)} n^{O(1)}$ time on graphs with n vertices for some $\epsilon > 0$ and some function $f_B : \mathbb{N} \rightarrow \mathbb{R}^+$ that may depend on the set B . Then SETH fails.*

High Level Idea. We follow the ideas of previous lower bound reductions from [30] and combine them with the concept of using relations from [11]. From now on let $M := \max B$. Let ϕ be the given CNF formula with n variables and clauses C_1, \dots, C_m . Instead of encoding each variable separately, we group g variables together and encode (partial) assignments to these groups. For each partial assignment, we define a vector in $[0, M]^g$, where g is chosen such that $2^g \leq (M + 1)^g$. For each group we define a *layer* with g parallel rows, where each row corresponds to one dimension of the vector. The layers consist of an alternation of g parallel simple nodes and a complex node that is related to a clause. All simple nodes are connected to their neighboring complex nodes by M parallel edges. The vector from above then corresponds to the number of selected edges from a simple node to the following shared complex node. The complex nodes check whether the assignment represented by the selected edges of a layer satisfies the related clause. For each clause we connect the related complex nodes by a path. This path is used to propagate the information whether the clause is already satisfied by some partial assignment or whether it still needs to be satisfied. We ensure that each clause is initially not satisfied and eventually all clauses must be satisfied.

Constructing the B -FACTOR \mathcal{R} Instance. See Figure 1 for an example of the following construction. Split the variables of ϕ into $t := \lceil n/q \rceil$ groups F_1, \dots, F_t of size at most q , where q is chosen later. For each of the t groups we encode the 2^q partial assignments by vectors from $[0, M]^g$ for some g chosen later. Instead of using all $(M+1)^g$ possible encodings we only use those vectors where the total weight of the coordinates is equal to $gM/2$ (we will choose g as a multiple of 4, hence $gM/2$ is an even number). It can easily be shown that there are more than $(M+1)^g/(gM+1)$ vectors with exactly this weight. Thus, after setting $q = \lfloor \log((M+1)^g) - \log(gM+1) \rfloor$, we can map each of the 2^q assignments of a group F_i to a distinct vector $[0, M]^g$ with weight exactly $gM/2$. We say that an partial assignment τ to a group F_i satisfies a clause C_j if at least one literal in the clause is satisfied under the assignment τ . Note, that a group F_i does not have to cover all variables of C_j to satisfy the clause.

We define the graph now as follows:

1. For all $i \in [t]$, $\ell \in [g]$, and $j \in [m]$: create a simple node $J_{i,\ell}^j$.
2. For all $i \in [t]$ and $j \in [m]$: create complex nodes r_i^j with relation R_i^j to be defined later.
3. For all $i \in [t]$: create complex nodes r_i^0 and r_i^{m+1} with relation R^0 .
4. For all $j \in [m]$: create complex nodes r_0^j (resp. r_{t+1}^j) with relation $\text{HW}_{=0}$ (resp. $\text{HW}_{=1}$).
5. For all $i \in [t]$, $\ell \in [g]$, and $j \in [m]$: make $J_{i,\ell}^j$ adjacent to r_{i-1}^j and r_i^j by M parallel edges each. We call these edges backwards and forwards edges, respectively.
6. For all $i \in [t]$ and $j \in [m]$, make r_i^j additionally adjacent to r_{i-1}^j and r_{i+1}^j by one edge each. The degree of the nodes is now $2gM+2$.

We call the set of nodes $\{r_i^j, J_{i,\ell}^j\}_{j,\ell}$ the i th layer. The set $\{J_{i,\ell}^j\}_j$ forms the ℓ th row of the i th layer. For a fixed $j \in [m]$, the set of nodes $\{r_i^j\}_i$ is called the j th column.

The idea is now the following: For each partial assignment τ to a group F_i , we define a vector $v_\tau \in [0, M]^g$ of weight $gM/2$ as its *encoding*.¹ Then $v_\tau[\ell]$ corresponds to the number of selected forward edges of the simple nodes in the ℓ th row of the i th layer. The vertical edges encode whether a clause was already satisfied. That is, if the edge between r_i^j and r_{i+1}^j is selected, then there is some group F_k with $k \leq i$ where the corresponding assignment satisfies the clause C_j . By the relation of the nodes r_0^j every clause is initially not satisfied. But the relation of the r_{t+1}^j nodes ensures that every clause is eventually satisfied.

Defining the Relations. R^0 accepts exactly those inputs of Hamming weight exactly $gM/2$, an even number by assumption, where the selected edges for each row must precede the unselected edges, i.e. the first k edges are selected, the next $M-k$ are not selected.

The relation $R_i^j \subseteq \{0, 1\}^{2Mg+2}$ of node r_i^j is defined as follows:

- For $\ell \in [g]$, let x_ℓ (resp. y_ℓ) be the number of selected incident edges to $J_{i,\ell}^j$ (resp. $J_{i,\ell}^{j+1}$).
- $\sum_{\ell \in [g]} x_\ell = gM/2 = \sum_{\ell \in [g]} y_\ell$.
- $\langle x_1, \dots, x_g \rangle$ describes a valid encoding, i.e. it corresponds to a partial assignment for F_i .
- $x_\ell + y_\ell = M$. Further, the x_ℓ (resp. y_ℓ) selected edges precede the $M-x_\ell$ (resp. $M-y_\ell$) unselected edges of the M parallel edges going to a simple node.
- If the ingoing top edge is selected, then the outgoing bottom edge is also selected.
- If the ingoing top edge is not selected:
 - If C_j does not contain a variable of F_i , then the outgoing bottom edge is not selected.
 - If C_j contains at least one variable of F_i , then the outgoing bottom edge is selected if and only if the selected edges correspond to a valid partial assignment satisfying C_j .

¹ Note that for different groups the encoding of the same partial assignment do not need to be the same.

Final Modifications. Due to parity issues, we can only realize relations where the Hamming weight of the accepted inputs is an *even* constant for each relation. Thus, we slightly have to modify this construction. We leave the exact details to the full version.

► **Lemma 3.3.** ϕ is satisfiable if and only if there is a solution to the B -FACTOR $^{\mathcal{R}}$ instance.

To obtain a tight lower bound, we need to analyze the pathwidth of our construction and have to bound the degree of the complex nodes.

► **Lemma 3.4.** The graph has $\mathcal{O}(tgm)$ simple and $\mathcal{O}(tm)$ complex nodes. The degree of the complex nodes is bounded by $2gM + 4$. The degree of the simple nodes is bounded by $2M$. We can efficiently construct a path decomposition of width $tg + \mathcal{O}(1)$ where at most three complex nodes are simultaneously in one bag.

Now we have everything ready to prove the lower bound for the intermediate problem B -FACTOR $^{\mathcal{R}}$ based on the previous construction. Recall, we defined Δ^* as the maximum total degree of the complex nodes appearing in one bag, that is $\Delta^* = \max_{\text{bag } X} \sum_{v \in X \cap V_C} \deg(v)$.

Proof of Theorem 3.2 (Sketch). As $(M + 1)^g \approx 2^g$ and $t \approx n/q$, we intuitively get

$$(M + 1 - \epsilon)^{tg} n^{\mathcal{O}(1)} = ((M + 1 - \epsilon)^g)^{\frac{n}{q}} n^{\mathcal{O}(1)} \ll ((M + 1)^g)^{\frac{n}{q}} n^{\mathcal{O}(1)} = (2^g)^{\frac{n}{q}} n^{\mathcal{O}(1)} = 2^n n^{\mathcal{O}(1)},$$

showing that the reduction and the assumed algorithm would solve the SAT instance too fast. Note that due to rounding and other issues the calculation has to be done way more carefully and is thus deferred to the full version. ◀

4 Decision Version

In this section we prove the lower bound for the decision version of B -FACTOR by a reduction from the intermediate B -FACTOR $^{\mathcal{R}}$ problem. For this we formally define the concept of realizations and show that we can realize all relations of a B -FACTOR $^{\mathcal{R}}$ instance. Replacing the nodes and their relations by these realizations yields the final lower bound.

► **Definition 4.1 (Realization).** Let $R \subseteq \{0, 1\}^k$ be a relation. Let G be a node-labeled graph with dangling edges $D = \{d_1, \dots, d_k\} \subseteq E(G)$. We say that graph G realizes R if for all $D' \subseteq D$: $D' \in R$ if and only if there is a solution $S \subseteq E(G)$ with $S \cap D = D'$. We say that G B -realizes R if G is B -homogeneous. The endpoints of the dangling edges are called portals.

The crucial part of the reduction is the proof of the following theorem. We postpone its proof and first show the lower bound.

► **Theorem 4.2.** Let $B \subseteq \mathbb{N}$ be a fixed set of finite size with $\max\text{-gap } B > 1$ and $0 \notin B$. There is a $f : \mathbb{N} \rightarrow \mathbb{N}$ such that the following holds. Let $R \subseteq \{0, 1\}^e$ be an even relation (i.e. $\text{hw}(x)$ is even for all $x \in R$). Then we can B -realize R by a simple graph with $f(e)$ vertices of degree at most $\max B + 2$, the portal nodes are pairwise distinct.

Now we can prove the lower bound under SETH. We assume that $B \subseteq \mathbb{N}$ is a fixed, finite set such that $0 \notin B$ and $\max\text{-gap } B > 1$.

Proof of Theorem 1.4 (Sketch). Replace every complex node v and its relation R_v in the B -FACTOR $^{\mathcal{R}}$ instance H by its B -realization of size at most $f(\deg(v))$ according to Theorem 4.2. This increases the size of the graph at most by a factor of $f(\Delta^*)$. As we can bound the pathwidth of the inserted graphs by their size, we modify each bag of the path decomposition

95:10 Tight Complexity Results of General Factor Problems

of H by replacing all complex nodes with the nodes of their realization. Thus, the pathwidth of the new graph is bounded by $\text{pw}_H + \Delta^* f(\Delta^*)$. Assuming the faster algorithm, this already contradicts SETH by Theorem 3.2. ◀

From now on let $B \subseteq \mathbb{N}$ be our fixed, finite set with $\min B \geq 1$ and $\text{max-gap } B = d > 1$ such that $[a, a + d + 1] \cap B = \{a, a + d + 1\}$ for some $a \geq 1$. We first realize three quite basic relations which we use later to realize the more complex relations.

► **Lemma 4.3.** *We can B -realize each of the relations $\text{HW}_{=2}^{(2)}$, EQ_{d+1} , and EQ_2 by a simple graph with $\mathcal{O}(\text{poly}(\max B))$ vertices of degree at most $\max B$.*

Proof.

1. Define a $\min B + 1$ -clique with new vertices. Split an arbitrary edge (u, v) into two dangling edges $(?, u)$ and $(?, v)$. The construction of the clique and the fact that we chose $\min B$ as degree forces the two dangling edges to be selected in any solution.
2. We start with two new vertices u, v and connect each to a many common $\text{HW}_{=2}^{(2)}$ nodes. We add $d + 1$ dangling edges to u and zero to v . Finally the nodes are replaced by their realization. Observe that u has a forced edges and $d + 1$ dangling edges. Thus we must select none or all of the dangling edges since $[a, a + d + 1] \cap B = \{a, a + d + 1\}$.
3. Define a $d + 2$ -clique with EQ_{d+1} nodes. Split an arbitrary edge (u, v) into two dangling edges $(?, u)$ and $(?, v)$. Replace the nodes by their realization. Either both dangling edges are selected in which case all nodes have $d + 1$ incident edges in the solution, or neither is selected in which case every node has zero incident edges in the solution. ◀

The following lemma helps us to keep the later constructions simple. Instead of constructing the relations for arbitrary degree, only the very low degree cases are necessary.

► **Lemma 4.4.** *If we can realize $\text{HW}_{=1}^{(a)}$ for $a \in \{1, 2, 3\}$ by a simple graph with at most N vertices of degree at most D , then we can realize $\text{HW}_{=1}^{(k)}$ for all $k \geq 1$ by a simple graph using $\mathcal{O}(kN)$ nodes of degree at most D .*

Proof. We construct the graph for the realization inductively starting with the basis for $k = 1, 2, 3$. See Figure 2 for an example.

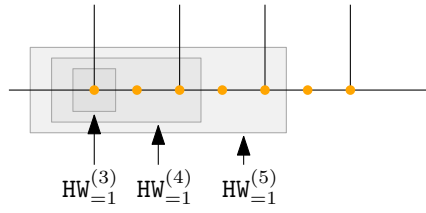
For the inductive step from k to $k + 1$ we start with a node u with relation $\text{HW}_{=1}^{(k)}$. Connect one dangling edge of u to a new node v with $\text{HW}_{=1}^{(2)}$. Connect the other dangling edge of v to a node w with relation $\text{HW}_{=1}^{(3)}$. Observe that the final graph has $k + 1$ dangling edges.

Assume one dangling edge of u is selected, then the edge between u and v is not selected but the edge from v to w is. Hence, no dangling edge of w can be selected. The analogue holds if one of the dangling edges of w is selected. It cannot be the case that more than one or zero dangling edges are selected, as then the relation of one of the three nodes u, v, w would not be satisfied. ◀

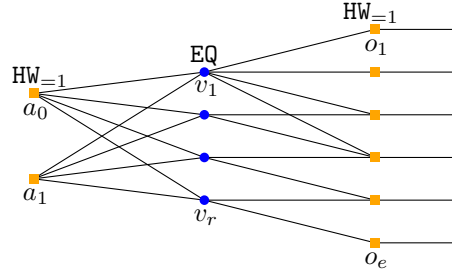
Due to parity issues, the construction of the realizations depends on the set B . Each of the possible cases (B contains only even numbers, only odd number, or even and odd numbers) is treated separately. We focus on the all even case and refer the reader to the full version for the other cases.

► **Lemma 4.5.** *If B contains only even numbers, we can B -realize the following relations by simple graphs:*

1. EQ_k for even $k \geq 2$ using $\mathcal{O}(k \text{ poly}(\max B))$ vertices of degree at most $\max B$.
2. $\text{HW}_{=1}^{(k)}$ together with $\text{HW}_{=1}^{(\ell)}$ for all $k, \ell \geq 1$ using $\mathcal{O}((k + \ell) \text{ poly}(\max B))$ vertices of degree at most $\max B + 2$.



■ **Figure 2** Example of the inductive construction from Lemma 4.4 for $\text{HW}_{=1}^{(6)}$ using $\text{HW}_{=1}^{(3)}$ and $\text{HW}_{=1}^{(2)}$.



■ **Figure 3** An example illustrating the construction from the proof of Theorem 4.2 for the relation R with $R(000011) = R(110011) = R(111001) = R(111100) = 1$ and zero otherwise.

Proof.

1. For $k = 2$ we can use the construction of Lemma 4.3. For the other case we first realize EQ_4 . Then we use a chain of these relations to realize EQ_k for even $k \geq 6$.
 Start with a EQ_{d+1} node u and make it adjacent to $\frac{d+1-4}{2}$ many EQ_2 nodes (note that an even B can have only gaps of odd size, hence d is odd). Then we add four dangling edges to u . hence the construction actually works. The graph is simple as the dangling edges in the realization of EQ_2 are different.
2. To use Lemma 4.4 for the general construction, observe that the number of $\text{HW}_{=1}$ nodes used in the construction is odd. Hence, we will always realize two nodes. For this we show how to realize $\text{HW}_{=1}^{(k)}$ together with $\text{HW}_{=1}^{(\ell)}$ for all $k, \ell \in \{1, 2, 3\}$.
 Start with two vertices u, v . Make u and v adjacent to $\max B - 1$ common $\text{HW}_{=2}^{(2)}$ nodes. We add k dangling edges to u and ℓ dangling edges to v . As B does not contain $\max B - 1$, the correctness follows. ◀

Now we have everything ready to prove that even relations can be realized.

Proof of Theorem 4.2. See Figure 3 for an example of the following construction. We use essentially the construction from Lemma 3.3 in [11]. Let $R = \{x_1, \dots, x_r\} \subseteq \{0, 1\}^e$ be the even relation for some r . Let further $P = \{(1 + e \bmod 2), 0\}$.

1. Create nodes o_1, \dots, o_e with relation $\text{HW}_{=1}$.
2. Create vertices a_j for all $j \in P$ with relation $\text{HW}_{=1}$.
3. For all $i \in [r]$:
 - a. Let $O_i = \{n_1^{(i)}, \dots, n_{h_i}^{(i)}\} = \{k \in [e] \mid x_i[k] = 0\}$ for $h_i = e - \text{hw}(x_i)$.
 - b. Create the node v_i with relation EQ and connect it to $o_{n_j^{(i)}}$ for all $j \in [h_i]$.
 - c. Connect v_i to all a_j .
4. Replace all nodes by their realization.

There are $|P| + e$ many $\text{HW}_{=1}$ nodes. Since $|P| = 1 + (1 + e \bmod 2)$, we can replace pairs of these nodes by their realization. Every v_i is connected to $|P| + |O_i|$ nodes, where $|O_i| = e - \text{hw}(x_i)$. Thus, v_i has even degree as the relation R is even, i.e. $\text{hw}(x)$ is even. Hence, we can replace these nodes by their realization according to the previous lemmas.

95:12 Tight Complexity Results of General Factor Problems

To show that the construction actually realizes the relation, assume the selected dangling edges corresponds to some element $x \in R$, let it w.l.o.g. be x_1 . Then we can select all edges incident to x_1 , the dangling edges, and the extension of this to all nodes as a solution. As x_1 is adjacent to all a_j they are in a valid state. Further x_1 is adjacent to those o_k where $x_1[k] = 0$ and hence every o_k is incident to exactly one edge in the solution.

Now assume we are given a solution. As the nodes a_j have exactly one incident edge in the solution, there is exactly one node v_i where all incident edges are in the solution. Let O be the set of nodes o_k to which v_i is adjacent. By construction v_i corresponds to some $x \in R$ with $x[k] = 0$ iff $k \in O$. As all selected dangling edges must be in the solution, let O' be the set of nodes incident to the selected dangling edges. But as we are given a solution we get $O \cup O' = \{o_1, \dots, o_e\}$. Hence, the dangling edges correspond to x . ◀

5 Optimization Version

In the previous section we have seen the realization of the relations for the decision version. As we are interested in the largest solution for MAX- B -FACTOR, we also allow $0 \in B$ since this does not make the problem trivially solvable. This makes it necessary to change the definition of a realization, as the pure existence of a solution is not sufficient anymore. We change it such that if the relation is satisfied (i.e. the dangling edges are selected in a good way), then there is a *large* solution. Otherwise, there must be a gap by which any solution is *smaller* compared to the solutions in the good cases. We call this gap the penalty (of the realization).

► **Definition 5.1 (Realization).** Let $R \subseteq \{0, 1\}^k$ be a relation. Let G be a node labeled graph, with dangling edges $D = \{d_1, \dots, d_k\}$. We say that graph G realizes R with penalty β if we can efficiently construct/find a target value $\alpha > 0$ such that for all $D' \subseteq D$:

- If $D' \in R$, then there is a solution $S \subseteq E(G)$ with $S \cap D = D'$ and $|S| = \alpha$.
- If $D' \notin R$, then for all solutions $S \subseteq E(G)$ with $S \cap D = D'$ we have $|S| \leq \alpha - \beta$.

We say that G B -realizes R if G is additionally B -homogeneous. We call the endpoints of the dangling edges portal nodes.

In the main part of this section we show how to realize the relations of B -FACTOR $^{\mathcal{R}}$. The following theorem corresponds to Theorem 4.2 for the decision version.

► **Theorem 5.2 (Realization of Relations).** Let $B \subseteq \mathbb{N}$ be a fixed, finite set with max-gap $B > 1$ and $0 \in B$. There is a $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that the following holds. Let $R \subseteq \{0, 1\}^e$ be a relation with a constant $c_R \in 2\mathbb{N}$ such that for all $x \in R$ we have $\text{hw}(x) = c_R$.

We can B -realize the relation R with arbitrary penalty $\beta > 0$ by a simple graph with $f(e, \beta)$ vertices of degree at most $\max B + 2$.

It remains to compute the target value by which we decide if the B -FACTOR $^{\mathcal{R}}$ instance has a solution or not.

► **Lemma 5.3.** Let G be a B -FACTOR $^{\mathcal{R}}$ instance from Section 3. Let G' be a B -FACTOR instance resulting from G by replacing every complex nodes with degree δ by its realization with penalty 2δ . Then, there is an efficiently computable constant α such that G has a solution if and only if the largest solution for G' has size α .

Proof (Sketch). The target value α is essentially the sum of the target values α_v for the realizations of the complex nodes $v \in V_C$. But we have to take care that the edges between complex nodes are not counted twice. ◀

Now we are ready to prove the conditional lower bound for MAX- B -FACTOR when $0 \in B$.

Proof of Theorem 1.5. Use Lemma 5.3 to construct the final graph and the target value. Then the proof goes analogous to the proof for the decision version (cf. Theorem 1.4). ◀

High Girth Graphs. We know that there is a gap of size at least two between a and $a + d + 1$ in B . This allows us to define relatively simple conditions of the form “if one incident edge of a vertex with degree $a + d + 1$ is not selected, then another edge is also not selected”. In other words, this propagates the penalty to a neighboring vertex. We combine this with high girth graphs to introduce an arbitrary large penalty for not selecting an edge.

The construction of r regular graphs with girth g is a long studied problem in graph theory. Erdős and Sachs proved the existence of such graphs for all combinations of r and g .

► **Lemma 5.4** (Theorem 1 in [20]). *For all $r \geq 2$ and $g \geq 3$, there is a r -regular graph $G_{r,g}$ of girth g with at most $4gr^g$ vertices.*

Finding the smallest graph for each r, g is a non-trivial task and known as the (r, g) -cage problem. For several cases (e.g. r is a prime power) constructions are known reducing the number of vertices in the graph. See [16, 21, 26, 28] for more results.

Realizing Relations. From now on let $d := \max\text{-gap } B > 1$ such that $[a, a + d + 1] \cap B = \{a, a + d + 1\}$ for some $a \geq 0$. As we allow $0 \in B$, we can always find a trivial solution. Thus, we cannot force edges as we did for the decision version. Instead we construct a gadget where we can select many edges when the “forced” edges are selected. Otherwise we ensure that the solution is small. We use the graphs with high girth for this.

► **Lemma 5.5.** *There is a $f : \mathbb{N} \rightarrow \mathbb{N}$ such that the following holds. We can B -realize $\text{HW}_{=2}^{(2)}$ (with distinct portal vertices) with arbitrary penalty β by simple graphs using at most $f(\beta)$ vertices of degree at most $\max B$.*

Proof. We use Lemma 5.4 to get an $a + d + 1$ -regular graph $G_{a+d+1,\beta}$ of girth at least β . Split an arbitrary edge (u, v) into two dangling edges for u and v each and assign the set B to every vertex.

The graph has the claimed properties: If both dangling edges are selected, then we can use the set of all edges in the graph as a solution since $a + d + 1 \in B$.

It remains to check the case when at least one dangling edge is not selected, let it w.l.o.g. be the one incident to u . Assume S is the optimal solution. We show that this solution does not contain more than $|E_{a+d+1,\beta}| - \beta$ edges.

By assumption $\deg_S(u) \leq a$. Hence, there must be at least one other incident edge to u that is not in the solution, because $a + d - 1 \geq a + 1 \notin B$. Then we can apply this argument always to the next vertex. Observe that this sequence can only stop if we reach another vertex w we have already visited because for this vertex we already know that two incident edges were not selected in the solution. The length of this path, i.e. the number of not selected edges from w to w , is at least the girth of the graph. Hence the number of edges that are not selected in the solution is at least the girth of the graph which is at least β . ◀

The remaining part follows mainly the constructions from the decision version. However, as we care about the size of the solution a more careful construction and analysis is needed. The detailed construction of the realization is given in the full paper.

6 Counting Version

From a certain perspective the optimization version can be seen as a relaxation of the decision version: The assumption $\min B > 0$ is dropped while still assuming max-gap $B > 1$. For the counting version we now even drop this last assumption such that there might be no gap at all in B . Thus the only polynomial-time solvable cases for the counting version are $B = \{0\}$ and $B = \emptyset$ with one and zero solutions, respectively. This implies that we additionally must realize equality relations. Surprisingly this also reduces to realizing $\text{HW}_{=1}^{(1)}$ nodes in the end, i.e. forcing edges.

We use the Holant framework and lemmas and definitions analogous to those from [11]. A signature graph Ω is a graph with weights w_e for all edges e and all vertices are labeled by signatures $f_v : \{0, 1\}^{I(v)} \rightarrow \mathbb{Q}$, which are rational functions on the incidence vector $I(v)$ of the edges incident to v . We define $\text{Holan}(\Omega)$ to be the quantity

$$\sum_{x \in \{0,1\}^{E(\Omega)}} \prod_{e \in E} w_e \prod_{v \in V(\Omega)} f_v(x|_{I(v)}).$$

The Holant framework can be seen as a natural generalization of GENFAC . If each signature f_v is a symmetric Boolean function and each edge weight is 1, then it is exactly $\#\text{GENFAC}$. If additionally each vertex has signature $\text{HW}_{\in B}$, this corresponds to $\#B\text{-FACTOR}$.

► **Definition 6.1** ($\text{Holan}(F)$). *If F is a set of rational functions, we say that $\text{Holan}(F)$ is the set of all Holant problems where the signature graph has signatures only from F .*

► **Definition 6.2** (Gate). *A gate is a signature graph Γ , possibly containing a set $D \subseteq E(\Gamma)$ of dangling edges, all of which have edge weight 1. The signature realized by Γ is the function $\text{SIG}(\Gamma) : \{0, 1\}^D \rightarrow \mathbb{Q}$ that maps an assignment of dangling edges $x \in \{0, 1\}^D$ to*

$$\text{SIG}(\Gamma, x) = \sum_{y \in \{0,1\}^{E(\Gamma) \setminus D}} \left(\prod_{e \in E(\Gamma)} w(e) \prod_{v \in V(\Gamma)} f_v((x \cup y)|_{I(v)}) \right)$$

Note that unless mentioned otherwise, we restrict ourselves to signature graphs with unit edge weights and hence they are usually omitted.

In essence, gates in the Holant framework play the role of realizations in the previous sections. Given these definitions, we are now ready to state our main theorem, which can then be used to prove Theorem 1.6. Observe for this that the reduction in Theorem 3.2 is parsimonious.

► **Theorem 6.3.** *For all fixed, finite $B \subseteq \mathbb{N}$ with $B \neq \{0\}$ there is a $f : \mathbb{N} \rightarrow \mathbb{N}$ such that the following holds. Let $G = (V_S \dot{\cup} V_C, E)$ be an instance of $\#B\text{-FACTOR}^{\mathcal{R}}$ with a path decomposition of width pw such that $\Delta^* = \max_{\text{bag}} x \sum_{v \in X \cap V_C} \deg(v)$. Then there is a $f(\Delta^*)n^{O(1)}$ time Turing reduction from $\#B\text{-FACTOR}^{\mathcal{R}}$ to $\#B\text{-FACTOR}$ such that for every constructed instance of $\#B\text{-FACTOR}$ pathwidth and cutwidth increase at most by $f(\Delta^*)$.*

We can think of $\#B\text{-FACTOR}^{\mathcal{R}}$ as a Holant problem where the allowed signatures are either $\text{HW}_{\in B}$ or restricted even relations. We first use a lemma from [11] to realize these relations through nodes with signature $\text{HW}_{=1}$. Since their constructions are in the perfect matching setting, they can equivalently be seen as gates that use vertices with signature $\text{HW}_{=1}$. After using this lemma to reduce from $\#B\text{-FACTOR}^{\mathcal{R}}$ to a Holant problem, we give a chain of reductions (see Figure 4) that ends at $\#B\text{-FACTOR}$ and preserves the pathwidth up to an additive constant.

► **Lemma 6.4** (Informal, Lemma 3.3 from [11]). *Every even relation can be realized through a graph whose vertices have signature $\text{HW}_{=1}$ and whose edges have weights in $\{-1, \frac{1}{2}, 1\}$.*

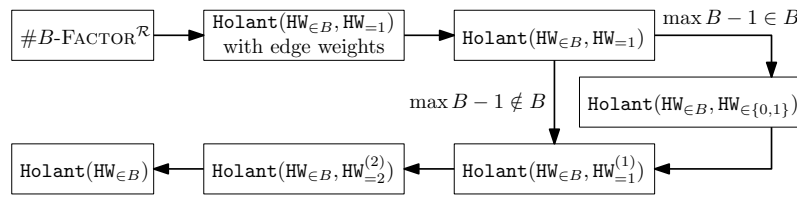


Figure 4 The chain of reductions that starts with $B\text{-FACTOR}^{\mathcal{R}}$ and ends at $\#B\text{-FACTOR}$ (i.e. $\text{Holant}(\text{HW}_{\in B})$). Arrows show the direction of Turing or many-one reductions.

Main Ideas. The next step is to remove the edge weights. We do this through polynomial interpolation, which was first used by Valiant [36]. The idea is that we can recover a polynomial $P(\cdot)$ if we know the value of $P(x)$ for sufficiently many x . We represent the solution of one problem as the value of a polynomial $P(\cdot)$ and the second problem as a function $f(P)$ of the polynomial itself. Then, we recover the value of the second problem by using an oracle of the first problem, giving a Turing reduction from the second problem to the first.

For the removal of the edge weight it suffices by the polynomial interpolation to consider edge weights that are a power of two. Assume for simplicity, we just have edge weight 2. Replace such edges by two parallel edges of unit weight. This leaves the output unchanged, as we duplicated the number of solutions, which compensates for the unit edge weight.

The main difficulty is to realize $\text{HW}_{=1}$ nodes using $\text{HW}_{\in B}$ nodes. To replace the $\text{HW}_{=1}$ nodes, we distinguish between the case where $\max B - 1$ is in B or not. In the latter case, the construction from the decision version works. But in the former case we use an argument similar to the procedure for the final step. The last step replaces $\text{HW}_{=2}^{(2)}$ nodes by $\text{HW}_{\in B}$ nodes. For this we “separate” the case where the forced edges are selected and where they are not. We define a pathlike gadget with many solutions if the dangling edges are not selected and significantly fewer otherwise. Each vertex on this long path is connected to many fresh vertices. We choose their number to be higher than the maximum element of B . Then, if an incident edge of the path is already selected, there are fewer solutions as if the edge is not selected. Combining this with the interpolation we arrive at a point, where all nodes have relation $\text{HW}_{\in B}$.

We now describe one case in the final step in the chain of reductions, where we realize $\text{HW}_{=2}^{(2)}$ nodes by $\text{HW}_{\in B}$ nodes. For the remaining cases and steps, we refer the reader to the full version of this paper.

For the interpolation we make use of the following result which is proven in the full version.

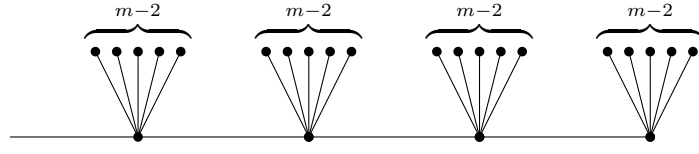
► **Proposition 6.5.** *Suppose we have two non-zero sequences $\{A_n\}_{n \in \mathbb{N}}, \{B_n\}_{n \in \mathbb{N}}$ that are related as*

$$\begin{bmatrix} A_n \\ B_n \end{bmatrix} = M \begin{bmatrix} A_{n-1} \\ B_{n-1} \end{bmatrix} = M^n U \quad , \text{ where } U = \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}$$

and M is a symmetric and invertible 2×2 matrix such that U is not an eigenvector of M . Then $\{\frac{B_n}{A_n}\}_{n \in \mathbb{N}}$ is a sequence which does not contain any repetitions.

► **Lemma 6.6.** *Let $B \subseteq \mathbb{N}$ be a fixed finite set. There is a polynomial-time Turing reduction from $\text{Holant}(\text{HW}_{\in B}, \text{HW}_{=2}^{(2)})$ to $\text{Holant}(\text{HW}_{\in B})$ increasing pathwidth and cutwidth only by a fixed constant and leaving the max degree unaffected, or increasing it to $2 \max B + 6$.*

95:16 Tight Complexity Results of General Factor Problems



■ **Figure 5** The gadget for case 1: Black nodes are $\text{HW}_{\in B}$ nodes.

Proof. If $0 \notin B$, we can use the construction from Lemma 4.3 to get a $\text{HW}_{=2}^{(2)}$ node. For the case when $0 \in B$, we do a case-by-case analysis depending on B . In either case, we attach a subgraph with a constant pathwidth and cutwidth to vertices. This does not affect either of them by more than $2 \max B + 6$, a fixed constant.

Case 1: B contains 1. Define $m \geq 2$ to be the smallest integer not in B . Consider the gadget in Figure 5. Suppose there are d such vertices with $m - 2$ pendant nodes each. Let all of them have the relation $\text{HW}_{\in B}$. Let $P_1(d)$ be the number of solutions of the gadget where the dangling edge is selected in the solution. Similarly define $P_0(d)$ when the dangling edge is not selected. We claim that the gadget described can be effectively used to force edges, i.e. a $\text{HW}_{=1}^{(1)}$ node. Two such gadgets will give us a $\text{HW}_{=2}^{(2)}$ node. Suppose any graph G contains t such gadgets. We have

$$\text{Holant}(G) = \sum_{i=0}^t A_i (P_0(d))^{t-i} (P_1(d))^i = (P_0(d))^t \sum_{i=0}^t A_i \left(\frac{P_1(d)}{P_0(d)} \right)^i$$

where A_i is the number of ways of extending the solution in G when i of the gadgets choose to match their dangling edge. Through standard interpolation techniques, we can recover the A_i s, and thus A_t will give us the solution where each gadget behaves like a $\text{HW}_{=1}^{(1)}$. Now, we can replace $\text{HW}_{=2}^{(2)}$ nodes in the $\text{Holant}(\text{HW}_{\in B}, \text{HW}_{=2}^{(2)})$ instance with pairs of $\text{HW}_{=1}^{(1)}$ nodes.

To argue that we can do the interpolation, we need to show that $\frac{P_1(d)}{P_0(d)}$ will take at least t unique values, and that these are computable in polynomial time. Since we can define such a gadget for any integer d we have

$$P_0(d) = kP_0(d-1) + kP_1(d-1) \quad \text{and} \quad P_1(d) = kP_0(d-1) + (k-1)P_1(d-1)$$

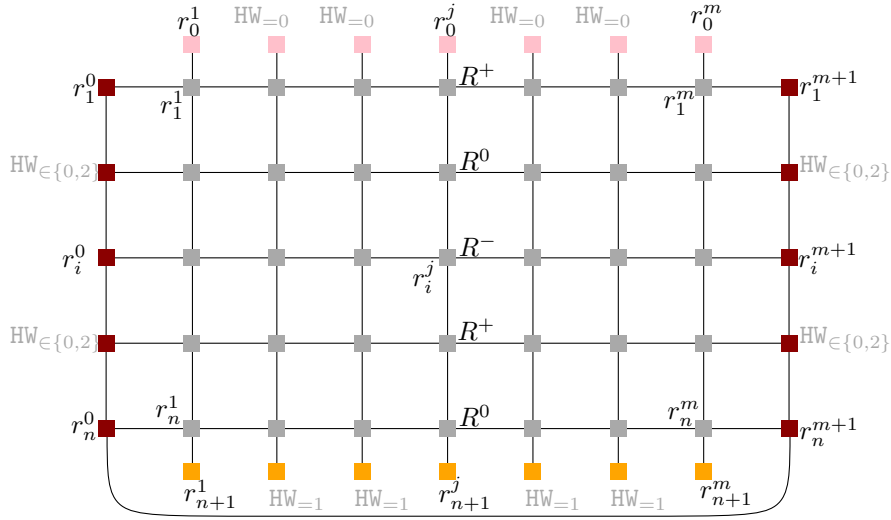
for $k = 2^{m-2}$. We now apply Proposition 6.5 with $M = \begin{bmatrix} k & k \\ k & k-1 \end{bmatrix}$ and $U = \begin{bmatrix} k \\ k \end{bmatrix}$.

This completes the realization for the case when B contains 1. The remaining cases when B does not contain 1 but some odd number and when B only consists of even numbers can be found in the full version. ◀

Proof of Theorem 6.3 (Sketch). Given any instance of $\#B\text{-FACTOR}^{\mathcal{R}}$, we can sequentially apply the reductions from Figure 4 to get a polynomial number of instances of $\#B\text{-FACTOR}$ such that the pathwidth is affected only by some function of Δ^* . ◀

7 Lower Bound when Parameterizing by Cutwidth

The algorithmic result from Theorem 1.8 shows that the pathwidth lower bound breaks when parameterizing by cutwidth. Nevertheless, we can show that this “improved” running time is the best we can hope for assuming SETH and $\#SETH$. For this we use the same high level ideas Curticapean and Marx presented in Figure 6 of [11] where they reduce from $\#SAT$ to computing the Holant and then reduce to counting perfect matchings. But the construction



■ **Figure 6** The example graph for a formula containing the clause $(x_1 \vee \bar{x}_3 \vee x_4)$.

can also be seen as a modification of our reduction for the pathwidth lower bound. We again first reduce to the intermediate problem $B\text{-FACTOR}^{\mathcal{R}}$ and then to $B\text{-FACTOR}$. By this we can reuse the results of realizing relations that we have seen in the previous sections.

► **Theorem 7.1.** *Let $B \subseteq \mathbb{N}$ be a fixed set of finite size. Given a CNF-formula ϕ with n variables and m clauses. We can construct a (simple) $B\text{-FACTOR}^{\mathcal{R}}$ instance G with $\mathcal{O}(nm)$ vertices, bounded degree and a linear layout of width $\text{cutw} \leq n + \mathcal{O}(1)$ in time linear in the output size. Further, the number of solutions for ϕ is equal to the number of solutions for G .*

Recall, that for the pathwidth lower bound we grouped variables together. This was needed to keep the pathwidth of the construction low. But this increased the cutwidth of the graph. Now, we do not group variables together but encode each variable on its own. See Figure 6 for an example of the construction we describe formally in the following.

Let x_1, \dots, x_n be the variables and C_1, \dots, C_m the clauses of ϕ . For each $i \in [n]$ and every $j \in [m]$ we create a vertex r_i^j . We assign the relation R^+ to r_i^j if x_i appears positively in C_j , R^- if it appears negatively, and otherwise R^0 , where R^0 , R^+ , and R^- are defined later. Additionally add vertices r_i^0 and r_i^{m+1} with relation $\text{HW}_{\in\{0,2\}}$ for all $i \in [n]$. We say that the vertices r_i^0, \dots, r_i^{m+1} form the i th *row*, i.e. the row of variable x_i . Create new nodes r_0^j and r_{n+1}^j and assign the relations $\text{HW}_{=0}$ and $\text{HW}_{=1}$ to them for all $j \in [m]$, respectively. We say the vertices $\{r_i^j\}_i$ form the j th *column*. We connect two nodes r_i^j and $r_{i'}^{j'}$ by an edge if $|i - i'| \leq 1$ and $|j - j'| \leq 1$ for all $i, i' \in [0, n + 1]$ and $j, j' \in [0, m + 1]$, i.e. if they are neighbors in the grid.

The idea is the same as for the pathwidth construction, except that selecting the edges of the i th row corresponds to setting the variable x_i to true. The edges between the nodes of a column represent if a clause is already satisfied. The relation $\text{HW}_{=0}$ ensures that we start with an initially unsatisfied clause. At each node r_i^j we check whether the assignment to this variable x_i satisfies the clause C_j and then force the output edge (i.e. the bottom edge) to be selected. Otherwise we propagate the current state (i.e. the selection of edges). Eventually we reach r_{n+1}^j with relation $\text{HW}_{=1}$ where the edge has to be selected and thus the clause must be satisfied.

The relations R^0 , R^+ , and R^- accept exactly those inputs that satisfy all of the following conditions:

1. The left edge is selected if and only if the right edge is selected.
2. If the top edge is selected, the bottom edge is selected.
3. Only for R^+ : If the top edge is unselected and the left edge is selected, then the bottom edge is selected.
4. Only for R^- : If the top edge is unselected and the left edge is not selected, then the bottom edge is selected.

For the proofs of the lower bounds, i.e. Theorem 1.9, we follow the ideas from the pathwidth lower bounds in Section 3. Thus we also have to modify the graph a bit such that we obtain a $B\text{-FACTOR}^{\mathcal{R}}$ instance and can replace all relations by their realizations. The details can be found in the full version of the paper.

References

- 1 Ashwin Arulselman, Ágnes Cseh, Martin Groß, David F. Manlove, and Jannik Matuschke. Matchings with lower quotas: Algorithms and complexity. *Algorithmica*, 80(1):185–208, 2018. doi:10.1007/s00453-016-0252-6.
- 2 Claude Berge. *Graphs and Hypergraphs*. North-Holland mathematical library, Amsterdam, 1973.
- 3 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 2013. doi:10.1007/978-3-642-39206-1_17.
- 4 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.
- 5 Liming Cai and David W. Juedes. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci.*, 67(4):789–807, 2003. doi:10.1016/S0022-0000(03)00074-6.
- 6 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0_6.
- 7 Gérard Cornuéjols. General factors of graphs. *J. Comb. Theory, Ser. B*, 45(2):185–198, 1988. doi:10.1016/0095-8956(88)90068-8.
- 8 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
- 9 Bruno Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *RAIRO Theor. Informatics Appl.*, 26:257–286, 1992. doi:10.1051/ita/1992260302571.
- 10 Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.47.
- 11 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.ch113.

- 12 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. doi:10.1145/2925416.
- 13 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 14 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.
- 15 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. doi:10.1109/FOCS.2011.23.
- 16 Xavier Dahan. Regular graphs of large girth and arbitrary degree. *Comb.*, 34(4):407–426, 2014. doi:10.1007/s00493-014-2897-6.
- 17 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslamán, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.
- 18 Szymon Dudycz and Katarzyna Paluch. Optimal general matchings. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2018. Full version: arXiv:1706.07418. doi:10.1007/978-3-030-00256-5_15.
- 19 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.
- 20 Paul Erdős and Horst Sachs. Reguläre Graphen gegebener Tailenweite mit minimaler Knotenzahl. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 12(251-257):22, 1963.
- 21 Geoffrey Exoo and Robert Jajcay. Recursive constructions of small regular graphs of given degree and girth. *Discret. Math.*, 312(17):2612–2619, 2012. doi:10.1016/j.disc.2011.10.021.
- 22 Tomás Feder. Fanout limitations on constraint systems. *Theor. Comput. Sci.*, 255(1-2):281–293, 2001. doi:10.1016/S0304-3975(99)00288-1.
- 23 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 142–151. SIAM, 2014. doi:10.1137/1.9781611973402.10.
- 24 Arne Hoffmann and Lutz Volkmann. On unique k -factors and unique $[1, k]$ -factors in graphs. *Discret. Math.*, 278(1-3):127–138, 2004. doi:10.1016/S0012-365X(03)00248-6.
- 25 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k -SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 26 Wilfried Imrich. Explicit construction of regular graphs without small cycles. *Comb.*, 4(1):53–59, 1984. doi:10.1007/BF02579157.
- 27 Sanjana Kolisetty, Linh Le, Ilya Volkovich, and Mihalis Yannakakis. The complexity of finding S -factors in regular graphs. *Electron. Colloquium Comput. Complex.*, 26:40, 2019. URL: <https://eccc.weizmann.ac.il/report/2019/040>.
- 28 Felix Lazebnik, Vasily A Ustimenko, and Andrew J Woldar. A new series of dense graphs of high girth. *Bulletin of the American mathematical society*, 32(1):73–79, 1995.
- 29 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011. URL: <http://eatcs.org/beatcs/index.php/beatcs/article/view/92>.

- 30 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.
- 31 László Lovász. The factorization of graphs. II. *Acta Mathematica Hungarica*, 23(1-2):223–246, 1972.
- 32 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|v|} |E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 17–27. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.12.
- 33 Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.
- 34 Yossi Shiloach. Another look at the degree constrained subgraph problem. *Inf. Process. Lett.*, 12(2):89–92, 1981. doi:10.1016/0020-0190(81)90009-0.
- 35 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- 36 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi:10.1137/0208032.
- 37 Johan M. M. van Rooij. Fast algorithms for join operations on tree decompositions. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 262–297. Springer, 2020. doi:10.1007/978-3-030-42071-0_18.
- 38 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.