On the Optimality of Pseudo-polynomial Algorithms for Integer Programming

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway
fomin@ii.uib.no

Fahad Panolan
Department of Informatics, University of Bergen, Norway
fahad.panolan@ii.uib.no

M. S. Ramanujan
University of Warwick, United Kingdom
R.Maadapuzhi-Sridharan@warwick.ac.uk

Saket Saurabh
Institute of Mathematical Sciences, HBNI, Chennai, India and University of Bergen, Norway
saket@imsc.res.in

Abstract

In the classic *Integer Programming* (IP) problem, the objective is to decide whether, for a given $m \times n$ matrix A and an m-vector $b = (b_1, \ldots, b_m)$, there is a non-negative integer n-vector x such that $Ax = b$. Solving (IP) is an important step in numerous algorithms and it is important to obtain an understanding of the precise complexity of this problem as a function of natural parameters of the input.

The classic pseudo-polynomial time algorithm of Papadimitriou [J. ACM 1981] for instances of (IP) with a constant number of constraints was only recently improved upon by Eisenbrand and Weismantel [SODA 2018] and Jansen and Rohwedder [ArXiv 2018]. We continue this line of work and show that under the Exponential Time Hypothesis (ETH), the algorithm of Jansen and Rohwedder is nearly optimal. We also show that when the matrix A is assumed to be non-negative, a component of Papadimitriou's original algorithm is already nearly optimal under ETH.

This motivates us to pick up the line of research initiated by Cunningham and Geelen [IPCO 2007] who studied the complexity of solving (IP) with non-negative matrices in which the number of constraints may be unbounded, but the branch-width of the column-matroid corresponding to the constraint matrix is a constant. We prove a lower bound on the complexity of solving (IP) for such instances and obtain optimal results with respect to a closely related parameter, path-width. Specifically, we prove *matching* upper and lower bounds for (IP) when the *path-width* of the corresponding column-matroid is a constant.

2012 ACM Subject Classification Theory of computation → Integer programming

Keywords and phrases Integer Programming, Strong Exponential Time Hypothesis, Branch-width of a matrix, Fine-grained Complexity

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.31

Related Version A full version of the paper is available at `https://arxiv.org/abs/1607.05342`.

1 Introduction

In the classic *Integer Programming* problem, the input is an $m \times n$ integer matrix A, and an m-vector $b = (b_1, \ldots, b_m)$. We consider the feasibility version of the problem, where the objective is to find a non-negative integer n-vector x (if one exists) such that $Ax = b$. Solving this problem, denoted by (IP), is a fundamental step in numerous algorithms and it is important to obtain an understanding of the precise complexity of this problem as a function of natural parameters of the input. Throughout the paper we denote Δ for the largest absolute value of the entries of A.

(IP) is known to be NP-hard. However, there are two classic algorithms due to Lenstra [13] and Papadimitriou [16] solving (IP) in polynomial or pseudo-polynomial time for two important cases when the number of variables and the number of constraints are bounded. These algorithms in some sense complement each other.

The algorithm of Lenstra shows that (IP) is solvable in polynomial time when the number of variables is bounded. Actually, the result of Lenstra is even stronger: (IP) is *fixed-parameter tractable* parameterized by the number of variables. However, the running time of Lenstra's algorithm is doubly exponential in n. Later, Kannan [12] provided an algorithm for (IP) running in time $n^{\mathcal{O}(n)}$. Deciding whether the running time $n^{\mathcal{O}(n)}$ can be improved to $2^{\mathcal{O}(n)}$ is a long-standing open question.

Our work is motivated by the complexity analysis of the complementary case when the number of constraints is bounded. (IP) is NP-hard already for $m = 1$ (the Knapsack problem) but solvable in pseudo-polynomial time. In 1981, Papadimitriou [16] extended this result by showing that (IP) is solvable in pseudo-polynomial time on instances for which the number of constraints m is a constant. The algorithm of Papadimitriou consists of two steps. The first step is combinatorial, showing that if the entries of A and b are from $\{0, \pm 1, \ldots, \pm d\}$, and (IP) has a solution, then there is also a solution which is in $\{0, 1, \ldots, n(md)^{2m+1}\}^n$. The second, algorithmic step shows that if (IP) has a solution with the maximum entry at most B, then the problem is solvable in time $\mathcal{O}((nB)^{m+1})$. Thus the total running time of Papadimitriou's algorithm is $\mathcal{O}(n^{2m+2} \cdot (md)^{(m+1)(2m+1)})$, where $d = \max\{\Delta, \|b\|_\infty\}$. There was no algorithmic progress on this problem until the very recent breakthrough of Eisenbrand and Weismantel [6]. They proved the following result.

▶ **Proposition 1** (Theorem 2.2, Eisenbrand and Weismantel [6]). *(IP) with $m \times n$ matrix A is solvable in time $(m \cdot \Delta)^{\mathcal{O}(m)} \cdot \|b\|_\infty^2$.*

Then, Jansen and Rohwedder improved Proposition 1 and gave a matching lower bound very recently [10].

▶ **Proposition 2** (Jansen and Rohwedder [10]). *(IP) with $m \times n$ matrix A is solvable in time $\mathcal{O}(m\Delta)^m \log(\Delta) \log(\Delta + \|b\|_\infty)$. Assuming the Strong Exponential Time Hypothesis (SETH), there is no algorithm for (IP) running in time $n^{\mathcal{O}(1)} \cdot \mathcal{O}(m(\Delta + \|b\|_\infty))^{m-\delta}$ for any $\delta > 0$.*

SETH is the hypothesis that CNF-SAT cannot be solved in time $(2 - \epsilon)^n m^{\mathcal{O}(1)}$ on n-variable m-clause formulas for any constant ϵ. ETH is the hypothesis that 3-SAT cannot be solved in time $2^{o(n)}$ on n-variable formulas. Both ETH and SETH were first introduced in the work of Impagliazzo and Paturi [8], which built upon earlier work of Impagliazzo, Paturi and Zane [9]. One of the natural question is whether the exponential dependence of $\|b\|_\infty$ can be improved significantly at the cost of super polynomial dependence on n. Our first theorem provides a conditional lower bound indicating that any significant improvements are unlikely.

▶ **Theorem 3.** *Unless the Exponential Time Hypothesis (ETH) fails, (IP) with $m \times n$ matrix A cannot be solved in time $n^{o(\frac{m}{\log m})} \cdot \|b\|_\infty^{o(m)}$ even when the constraint matrix A is non-negative and each entry in any feasible solution is at most* 2*.*

Let us note that since the bound in Theorem 3 holds for a non-negative matrix A, we can always reduce (in polynomial time) the original instance of the problem to an equivalent instance where the maximum value Δ in the constraint matrix A does not exceed $\|b\|_\infty$. Thus Theorem 3 also implies the conditional lower bound $n^{o(\frac{m}{\log m})} \cdot (\Delta \cdot \|b\|_\infty)^{o(m)}$. When $m = \mathcal{O}(n)$, our bound also implies the lower bound $(n \cdot m)^{o(\frac{m}{\log m})} \cdot (\Delta \cdot \|b\|_\infty)^{o(m)}$. We complement Theorem 3 by turning our focus to the dependence of algorithms solving (IP) on m alone, and obtaining the following theorem.

▶ **Theorem 4.** *Unless ETH fails, (IP) with $m \times n$ matrix A cannot be solved in time $f(m) \cdot (n \cdot \|b\|_\infty)^{o(\frac{m}{\log m})}$ for any computable function f. The result holds even when the constraint matrix A is non-negative and each entry in any feasible solution is at most* 1*.*

Although Theorem 3 provides a better dependence on $\|b\|_\infty$, Theorem 4 provides much more information on how the complexity of the problem depends on m. Since several parameters are involved in this running time estimation, a natural objective is to study the possible tradeoffs between them. For instance, consider the $\mathcal{O}(m\Delta)^m \log(\Delta) \log(\Delta + \|b\|_\infty)$ time algorithm (Proposition 2) for (IP). A natural follow up question is the following. Could it be that by allowing a significantly worse dependence (a superpolynomial dependence) on n and $\|b\|_\infty$ and an *arbitrary* dependence on m, one might be able to improve the dependence on Δ alone? Theorem 4 provides a strong argument against such an eventuality. Indeed, since the lower bound of Theorem 4 holds even for non-negative matrices, it rules out algorithms with running time $f(m) \cdot \Delta^{o(\frac{m}{\log m})} \cdot (n \cdot \|b\|_\infty)^{o(\frac{m}{\log m})}$. Therefore, obtaining a subexponential dependence of Δ on m even at the cost of a superpolynomial dependence of n and $\|b\|_\infty$ on m, and an arbitrarily bad dependence on m is as hard as obtaining a subexponential algorithm for 3-SAT.

We now motivate our remaining results. It is straightforward to see that when the matrix A happens to be non-negative, the algorithm of Papadimitriou [16] runs in time $\mathcal{O}((n \cdot \|b\|_\infty)^{m+1})$. Due to Theorems 3 and 4, the dynamic programming step of the algorithm of Papadimitriou for (IP) when the maximum entry in a solution as well as in the constraint matrix is bounded, is already close to optimal. Consequently, any quest for "faster" algorithms for (IP) must be built around the use of additional structural properties of the matrix A. Cunningham and Geelen [1] introduced such an approach by considering the *branch decomposition* of the matrix A. They were motivated by the fact that the result of Papadimitriou can be interpreted as a result for matrices of constant *rank* and branch-width is a parameter which is upper bounded by rank plus one. For a matrix A, the *column-matroid* of A denotes the matroid whose elements are the columns of A and whose independent sets are precisely the linearly independent sets of columns of A. We postpone the formal definitions of branch decomposition and branch-width till the next section. For (IP) with a *non-negative* matrix A, Cunningham and Geelen [1] showed that when the branch-width of the column-matroid of A is constant, (IP) is solvable in pseudo-polynomial time.

▶ **Proposition 5** (Cunningham and Geelen [1])**.** *(IP) with a non-negative $m \times n$ matrix A given together with a branch decomposition of its column matroid of width k, is solvable in time $\mathcal{O}((\|b\|_\infty + 1)^{2k} mn + m^2 n)$.*

We analyze the complexity of (IP) parameterized by the branch-width of A, by making use of SETH.

▶ **Theorem 6.** *Unless SETH fails, (IP) with a non-negative $m \times n$ constraint matrix A cannot be solved in time $f(\mathsf{bw})(\|b\|_\infty+1)^{(1-\epsilon)\mathsf{bw}}(mn)^{\mathcal{O}(1)}$ or $f(\|b\|_\infty)(\|b\|_\infty+1)^{(1-\epsilon)\mathsf{bw}}(mn)^{\mathcal{O}(1)}$, for any computable function f. Here bw is the branchwidth of the column matroid of A.*

In recent years, SETH has been used to obtain several tight conditional bounds on the running time of algorithms for various optimization problems on graphs of bounded treewidth [14]. In fact, Theorem 6 follows from stronger lower bounds we prove using the path-width of A as our parameter of interest instead of the branch-width. The parameter *path-width* is closely related to the notion of *trellis-width* of a linear code, which is a parameter commonly used in coding theory [7]. For a matrix $A \in \mathbb{R}^{m\times n}$, computing the path-width of the column matroid of A is equivalent to computing the trellis-width of the linear code generated by A. Roughly speaking, the path-width of the column matroid of A is at most k, if there is a permutation of the columns of A such that in the matrix A' obtained from A by applying this column-permutation, for every $1 \leq i \leq n-1$, the *dimension* of the subspace of $\mathbb{R}^m$ obtained by taking the intersection of the subspace of $\mathbb{R}^m$ spanned by the first i columns with the subspace of $\mathbb{R}^m$ spanned by the remaining columns, is at most $k-1$.

The value of the parameter path-width is always at least the value of branch-width and thus Theorem 6 follows from the following theorems.

▶ **Theorem 7.** *Unless SETH fails, (IP) with even a non-negative $m \times n$ constraint matrix A cannot be solved in time $f(k)(\|b\|_\infty+1)^{(1-\epsilon)k}(mn)^{\mathcal{O}(1)}$ for any computable function f and $\epsilon > 0$, where k is the path-width of the column matroid of A.*

▶ **Theorem 8.** *Unless SETH fails, (IP) with even a non-negative $m \times n$ constraint matrix A cannot be solved in time $f(\|b\|_\infty)(\|b\|_\infty+1)^{(1-\epsilon)k}(mn)^{\mathcal{O}(1)}$ for any computable function f and $\epsilon > 0$, where k is the path-width of the column matroid of A.*

Although the proofs of both lower bounds have a similar structure, we believe that there are sufficiently many differences in the proofs to warrant stating and proving them separately.

Note that although there is still a gap between the upper bound of Cunningham and Geelen from Proposition 5 and the lower bound provided by Theorem 6, the lower bounds given in Theorems 8 and 7 are asymptotically tight in the following sense. The proof of Cunningham and Geelen in [1] actually implies the upper bound stated in Theorem 9. We provide a self-contained proof in the appended full version of the paper for the reader's convenience.

▶ **Theorem 9.** *(IP) with non-negative $m\times n$ matrix A given together with a path decomposition of its column matroid of width k is solvable in time $\mathcal{O}((\|b\|_\infty+1)^{k+1}mn + m^2n)$.*

Then by Theorem 7, we cannot relax the $(\|b\|_\infty+1)^k$ factor in Theorem 9 even if we allow in the running time an arbitrary function depending on k, while Theorem 8 shows a similar lower bound in terms of $\|b\|_\infty$ instead of k. Put together the results imply that no matter how much one is allowed to compromise on either the path-width or the bound on $\|b\|_\infty$, it is unlikely that the algorithm of Theorem 9 can be improved.

The path-width of matrix A does not exceed its rank and thus the number of constraints in (IP). Hence, similar to Proposition 5, Theorem 9 generalizes the result of Papadimitriou when restricted to non-negative matrices. Also we note that the assumption of non-negativity is unavoidable (without any further assumptions such as a bounded domain for the variables) in this setting because (IP) is NP-hard when the constraint matrix A is allowed to have negative values (in fact even when restricted to $\{-1,0,1\}$) and the branchwidth of the column matroid of A is at most 3. A close inspection of the instances they construct in their NP-hardness reduction shows that the column matroids of the resulting constraint matrices are in fact direct sums of circuits, implying that even their *path-width* is bounded by 3.

2 Preliminaries

We use $\mathbb{Z}_{\geq 0}$ and $\mathbb{R}$ to denote the set of non negative integers and real numbers, respectively. For any positive integer n, we use $[n]$ and $\mathbb{Z}_n$ to denote the sets $\{1,\ldots,n\}$ and $\{0,1,\ldots,n-1\}$, respectively. For convenience, we say that $[0]=\emptyset$. For any two vectors $b,b'\in\mathbb{R}^m$ and $i\in[m]$, we use $b[i]$ to denote the i^{th} coordinate of b and we write $b'\leq b$, if $b'[i]\leq b[i]$ for all $i\in[m]$. We often use 0 to denote the zero-vector whose length will be clear from the context. For a matrix $A\in\mathbb{R}^{m\times n}$, $I\subseteq[m]$ and $J\subseteq[n]$, $A[I,J]$ denote the submatrix of A obtained by the restriction of A to the rows indexed by I and columns indexed by J. The notion of the branch-width of graphs, and implicitly of matroids, was introduced by Robertson and Seymour in [17]. Let $M=(U,\mathcal{F})$ be a matroid with universe set U and family $\mathcal{F}$ of independent sets over U. We use r_M to denote the rank function of M. That is, for any $S\subseteq U$, $r_M(S)=\max_{S'\subseteq S,S'\in\mathcal{F}}|S'|$. For $X\subseteq U$, the *connectivity function* of M is defined as $\lambda_M(X)=r_M(X)+r_M(U\setminus X)-r_M(U)+1$.

For matrix $A\in\mathbb{R}^{m\times n}$, we use $M(A)$ to denote the column-matroid of A. In this case the connectivity function $\lambda_{M(A)}$ has the following interpretation. For $E=\{1,\ldots,n\}$ and $X\subseteq E$, we define $S(A,X)=\text{span}(A|X)\cap\text{span}(A|E\setminus X)$, where $A|X$ is the set of columns of A restricted to X and $\text{span}(A|X)$ is the subspace of $\mathbb{R}^m$ spanned by the columns $A|X$. It is easy to see that the dimension of $S(A,X)$ is equal to $\lambda_{M(A)}(X)-1$.

A tree is *cubic* if its internal vertices all have degree 3. A *branch decomposition* of matroid M with universe set U is a cubic tree T and mapping μ which maps elements of U to leaves of T. Let e be an edge of T. Then the forest $T-e$ consists of two connected components T_1 and T_2. Thus every edge e of T corresponds to the partitioning of U into two sets X_e and $U\setminus X_e$ such that $\mu(X_e)$ are the leaves of T_1 and $\mu(U\setminus X_e)$ are the leaves of T_2. The *width* of edge e is $\lambda_M(X_e)$ and the width of branch decomposition (T,μ) is the maximum edge width, where maximum is taken over all edges of T. Finally, the *branch-width* of M is the minimum width taken over all possible branch decompositions of M.

The *path-width* of a matroid is defined as follows. Recall that a *caterpillar* is a tree which is obtained from a path by attaching leaves to some vertices of the path. Then the path-width of a matroid is the minimum width of a branch decomposition (T,μ), where T is a cubic caterpillar. Let us note that every mapping of elements of a matroid to the leaves of a cubic caterpillar corresponds to an ordering of these elements. Jeong, Kim, and Oum [11] gave a constructive fixed-parameter tractable algorithm to construct a path decomposition of width at most k for a column matroid of a given matrix.

For $q\geq 3$, let δ_q be the infimum of the set of constants c for which there exists an algorithm solving q-SAT with n variables and m clauses in time $2^{cn}\cdot m^{\mathcal{O}(1)}$. The *Exponential-Time Hypothesis (ETH)* and *Strong Exponential-Time Hypothesis (SETH)* are then formally defined as follows. ETH conjectures that $\delta_3>0$ and SETH that $\lim_{q\to\infty}\delta_q=1$.

3 ETH lower bounds on pseudopolynomial solvability of (IP)

In this section we prove Theorem 4. Here, we give a brief overview of the reduction and the intuition behind it. We use the ETH based lower bound result of Marx [15] for Partitioned Subgraph Isomorphism. For two graphs G and H, a map $\phi\colon V(G)\mapsto V(H)$ is called a *subgraph isomorphism* from G to H, if ϕ is injective and for any $\{u,v\}\in E(G)$, $\{\phi(u),\phi(v)\}\in E(H)$. In the Partitioned Subgraph Isomorphism problem, the input consists of two graphs G,H, a bijection $c_G\colon V(G)\mapsto[\ell]$ and a function $c_H\colon V(H)\mapsto[\ell]$, where $\ell=|V(G)|$ and the objective is to decide whether there a subgraph isomorphism ϕ from G to H such that for any $v\in V(G)$, $c_G(v)=c_H(\phi(v))$.

▶ **Lemma 10** ([15]). *If* Partitioned Subgraph Isomorphism *can be solved in time* $f(G)n^{o(\frac{k}{\log k})}$*, where* f *is an arbitrary function,* $n = |V(H)|$ *and* $k = |E(G)|$*, then ETH fails.*

To prove Theorem 4 we give a polynomial time reduction from Partitioned Subgraph Isomorphism to (IP) such that for every instance (G, H, c_G, c_H) of Partitioned Subgraph Isomorphism the reduction outputs an instance of (IP) where the constraint matrix has dimension $\mathcal{O}(|E(G)|) \times \mathcal{O}(|E(H)|)$ and the largest value in the target vector is $\max\{|E(H)|, |V(H)|\}$.

Let (G, H, c_G, c_H) be an instance of Partitioned Subgraph Isomorphism. Let $k = |E(G)|$ and $n = |V(H)|$. We construct an instance $Ax = b$ of (IP) from (G, H, c_G, c_H) in polynomial time. Without loss of generality we assume that $[n] = V(H)$ and that there are no isolated vertices in G. Hence, the number of vertices in G is at most $2k$. Let $m = |E(H)|$. For each $e \in E(H)$ we assign a unique integer from $[m]$. Let $\alpha\colon E(H) \mapsto [m]$ be the bijection which represents the assignment mentioned above. For any $i, j \in [\ell]$, we use $E_H(i, j)$ as a shorthand for the set of edges of H between $c_H^{-1}(i)$ and $c_H^{-1}(j)$. Finally, for ease of presentation we let $\{v_1, \ldots, v_\ell\} = V(G)$ and $c_G(v_i) = i$ for all $i \in [\ell]$, where $\ell = |V(G)|$.

We now formally define the (IP) instance output by our reduction. The set of indeterminants x of the (IP) instance is $\{x(\{a, b\}, c_H(a), c_H(b))\colon \{a, b\} \in E(H)\}$. Notice that for any $\{a, b\} \in E(H)$, there are two indeterminants $x(\{a, b\}, c_H(a), c_H(b))$ and $x(\{a, b\}, c_H(b), c_H(a))$ associated with it. Thus the cardinality of x is upper bounded by $2|E(H)| = 2m$. Recall that $\{v_1, \ldots, v_\ell\} = V(G)$ and $c_G(v_i) = i$ for all $i \in [\ell]$, where $\ell = |V(G)|$. For each $v_i \in V(G)$ we define $2d_G(v_i) - 1$ many constraints as explained below. Let $r = d_G(v_i)$ and $N_G(v_i) = \{v_{j_1}, \ldots, v_{j_r}\}$. The constraints for $v_i \in V(G)$ are the following. For all $q \in [r]$,

$$\sum_{e \in E_H(i, j_q)} x(e, i, j_q) = 1 \tag{1}$$

The constraints of the form above encode the "selection" constraint in Partitioned Subgraph Isomorphism, which says that for every edge $\{i, j\}$ in G, we must pick an edge in H which has one endpoint in color class i and the other in color class j. For all $q \in [r-1]$,

$$\sum_{\substack{\{a,b\} \in E_H(i, j_q) \\ a \in c_H^{-1}(i)}} a \cdot x(\{a, b\}, i, j_q) + \sum_{\substack{\{a,b'\} \in E_H(i, j_{q+1}) \\ a \in c_H^{-1}(i)}} (n - a) \cdot x(\{a, b'\}, i, j_{q+1}) = n \tag{2}$$

For each $\{v_i, v_j\} \in E(G)$ with $i < j$, we define the following constraint in the (IP) instance.

$$\sum_{\substack{\{a,b\} \in E_H(i, j) \\ a \in c_H^{-1}(i)}} \alpha(\{a, b\}) \cdot x(\{a, b\}, i, j) + \sum_{\substack{\{a,b\} \in E_H(i, j) \\ b \in c_H^{-1}(j)}} (m - \alpha(\{a, b\})) \cdot x(\{a, b\}, j, i) = m \tag{3}$$

The two sets of constraints above enforce the property that for any color class i in H, the set of edges that we have selected in the solution among those with exactly one endpoint in i, in fact have the same endpoint in the color class i. Together these constraints allow one to reconstruct the solution to the Partitioned Subgraph Isomorphism instance from a feasible solution for the resulting (IP) instance. Clearly, the number of rows in A is $|E(G)| + \sum_{v \in V(G)} 2d_G(v) - 1 \leq 5k$ and number of columns in A is $2m$. In order to prove Theorem 4, we first show that (G, H, c_G, c_H) is a Yes instance of Partitioned Subgraph Isomorphism if and only if $Ax = b, x \geq 0$ is feasible and if $Ax = b, x \geq 0$ is feasible, then for any solution x^*, each entry of x^* belongs to $\{0, 1\}$.

4 Path-width parameterization: SETH bounds

We prove Theorems 7 and 8 by giving reductions from CNF-SAT. At this point, one might be tempted to start the reduction from k-CNF SAT as seen in [2]. However, the fact that in our case we also need to control the path-width of the reduced instance poses serious technical difficulties if one were to take this route. Therefore, we take a different route and reduce from CNF-SAT which allows us to construct appropriate gadgets for propagation of consistency in our instance while simultaneously controlling the path-width. Moreover, the parameters in the reduced instances are required to obey certain strict conditions. For example, the reduction we give to prove Theorem 7 must output an instance of (IP), where the path-width of the column matroid $M(A)$ of the constraint matrix A is a constant and the upper bound on the largest entry in b depends on the path-width. Similarly, in the reduction used to prove Theorem 8, we need to construct an instance of (IP) where the largest entry in the target vector is upper bounded by a constant. These stringent requirements on the parameters make the SETH-based reductions quite challenging. However, reductions under SETH are allowed to take super polynomial time – they can even take $2^{(1-\epsilon)n}$ time for some $\epsilon > 0$, where n is the number of variables in the instance of CNF-SAT. This freedom to avail exponential time in SETH-based reductions is used crucially in the proofs of Theorems 7 and 8.

Now we give an overview of the reduction used to prove Theorem 7. Let ψ be an instance of CNF-SAT with n variables and m clauses. Given ψ and a fixed constant $c \geq 2$, we construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ of (IP) satisfying certain properties. Since for every $c \geq 2$, we have a different $A_{(\psi,c)}$ and $b_{(\psi,c)}$, this can be viewed as a family of instances of (IP). In particular our main technical lemma is the following.

▶ **Lemma 11.** *Let ψ be an instance of* CNF-SAT *with n variables and m clauses. Let $c \geq 2$ be a fixed integer. Then, in time $\mathcal{O}(m^2 2^{\frac{n}{c}})$, we can construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP) with the following properties.*

(a.) *ψ is satisfiable if and only if $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ is feasible.*

(b.) *The matrix $A_{(\psi,c)}$ is non-negative and has dimension $\mathcal{O}(m) \times \mathcal{O}(m2^{\frac{n}{c}})$.*

(c.) *The path-width of the column matroid of $A_{(\psi,c)}$ is at most $c + 4$.*

(d.) *The largest entry in $b_{(\psi,c)}$ is at most $2^{\lceil \frac{n}{c} \rceil} - 1$.*

Once we have Lemma 11, we prove Theorem 7 using the fact that if we have an algorithm $\mathcal{A}$ solving (IP) in time $f(k)(\|b\|_\infty + 1)^{(1-\epsilon)k}(mn)^a$ for some $\epsilon, a > 0$, then we can use this algorithm to refute SETH. In particular, given an instance ψ of CNF-SAT, we choose an appropriate c depending only on ϵ and a, construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP), and run $\mathcal{A}$ on it. Our careful choice of c will imply a faster algorithm for CNF-SAT, refuting SETH. More formally, we choose c to be an integer such that $(1-\epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c} < 1$. Then the total running time to test whether ψ is satisfiable, is the time require to construct $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ plus the time required by $\mathcal{A}$ to solve the constructed instance of (IP). That is, the time required to test whether ψ is satisfiable is

$$\mathcal{O}(m^2 2^{\frac{n}{c}}) + f(c+4)2^{\frac{n}{c}(1-\epsilon)(c+4)}2^{\frac{a \cdot n}{c}}m^{\mathcal{O}(1)} = 2^{\left((1-\epsilon)+\frac{4(1-\epsilon)}{c}+\frac{a}{c}\right)n}m^{\mathcal{O}(1)} = 2^{\epsilon' n}m^{\mathcal{O}(1)},$$

where $\epsilon' < 1$ is a constant depending on the choice of c. It is important to note that the utility of the reduction described in Lemma 11 is extremely sensitive to the value of the numerical parameters involved. In particular, even when the path-width blows up slightly, say up to δc, or when the largest entry in $b_{(\psi,c)}$ blows up slightly, say up to $2^{\delta \frac{n}{c}}$, for some $\delta > 1$, then the calculation above will *not* give us the desired refutation of SETH. Thus, the

challenging part of the reduction described in Lemma 11 is making it work under these strict restrictions on the relevant parameters and we focus on this part in the extended abstract.

As stated in Lemma 11, in our reduction, we need to obtain a constraint matrix with small path-width. An important first step towards this is understanding what a matrix of small path-width looks like. We first give an intuitive description of the structure of such matrices. Let A be a $m \times n$ matrix of small path-width and let $M(A)$ be the column matroid of A. For any $i \in \{1, \ldots, n-1\}$, let $A|\{1, \ldots i\}$ denote the set of columns (or vectors) in A whose index is at most i (that is, the first i columns) and let $A|\{i+1, \ldots n\}$ denote the set of columns with index strictly greater than i. The path-width of $M(A)$ is at most $\max_i \dim\langle \text{span}(A|\{1, \ldots, i\}) \cap \text{span}(A|\{i+1, \ldots, n\})\rangle + 1$. Consequently, in order to obtain a bound on the pathwidth, it is sufficient to bound $\dim\langle \text{span}(A|\{1, \ldots, i\}) \cap \text{span}(A|\{i+1, \ldots, n\})\rangle$ for every $i \in [n]$.

The construction used in Lemma 11 takes as input an instance ψ of CNF-SAT with n variables and a fixed integer $c \geq 2$, and outputs an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP), that satisfies all four properties of the lemma. Let X denote the set of variables in the input CNF-formula $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$. For the purposes of the present discussion we assume that c divides n. We partition the variable set X into c blocks $X_0, \ldots, X_{c-1}$, each of size $\frac{n}{c}$. Let $\mathcal{X}_i$, $i \in \{0, \ldots, c-1\}$, denote the set of assignments of variables corresponding to X_i. Set $\ell = \frac{n}{c}$ and $L = 2^\ell$. Clearly, the size of $\mathcal{X}_i$ is upper bounded by $2^{\frac{n}{c}} = 2^\ell = L$. We denote the assignments in $\mathcal{X}_i$ by $\phi_0(X_i), \phi_1(X_i), \ldots, \phi_{L-1}(X_i)$. To construct the matrix $A_{(\psi,c)}$, we view "each of these assignments as a different assignment for each clause". In other words we have separate sets of variables in the constraints corresponding to different pairs (C_r, X_i), where C_r is a clause and X_i is a block in the partition of X. That is for each clause C_r and block X_i, we have variables $\{y_{C_r,i,a}\ a \in \mathbb{Z}_{2L}\ \}$. In other words for each C_r and assignment $\phi_a(X_i)$, $a \in \mathbb{Z}_L$, we have two variables $y_{C_r,i,2a}$ and $y_{C_r,i,2a+1}$. For any clause C_r, $i \in \mathbb{Z}_c$ and $a \in \mathbb{Z}_{2L}$, assigning value 1 to $y_{C,i,a}$ corresponds to choosing an assignment $\phi_{\lfloor \frac{a}{2} \rfloor}(X_i)$ for X_i. In our reduction we will create the following set of constraints.

$$\sum_{\substack{i \in [c], a \in \mathbb{Z}_{2L} \text{ such that} \\ a \text{ is even and} \\ \phi_{\lfloor \frac{a}{2} \rfloor}(X_i) \text{ satisfies } C}} y_{C,i,a} = 1 \qquad \text{for all } C \in \mathcal{C} \tag{4}$$

$$\sum_{a \in \mathbb{Z}_{2L}} y_{C,i,a} = 1 \qquad \text{for all } C \in \mathcal{C} \text{ and } i \in \mathbb{Z}_c \tag{5}$$

Equation (4) takes care of satisfiability of clauses, while Equation (5) allows us to pick only one assignment from $\{\phi_0(X_i), \phi_1(X_i), \ldots, \phi_{L-1}(X_i)\}$ per clause C and block X_i. Note that this implies that we will choose an assignment in $\mathcal{X}_i$ for each clause C_r. That way we might choose m assignments from $\mathcal{X}_i$ corresponding to m different clauses. However, for the backward direction of the proof, it is important that we choose the *same* assignment from $\mathcal{X}_i$ for each clause. This will ensure that we have selected an assignment to the variables in X_i. Towards this we will have a third set of constraints as follows.

$$\sum_{a \in \mathbb{Z}_{2L}} \left(\lfloor \frac{a}{2} \rfloor \cdot y_{C_r,i,a} \right) + \left((L - 1 - \lfloor \frac{a}{2} \rfloor) y_{C_{r+1},i,a} \right) = L - 1 \ \forall r \in [m-1]\ ,\ i \in \mathbb{Z}_c \tag{6}$$

Equation (6) enforce consistencies of assignments of blocks across clauses in a *sequential* manner. That is, for any block X_i, we make sure that the two variables set to 1 corresponding to (C_r, X_i) and (C_{r+1}, X_i) are consistent for any $r \in \{1, \ldots, m-1\}$, as opposed to checking the consistency for every pair (C_r, X_i) and $(C_{r'}, X_i)$ for $r \neq r'$. Thus in some sense these

consistencies *propagate*. Furthermore, the idea of making consistency in a sequential manner also allows us to bound the path-width of column matroid of $A_{(\psi,c)}$ by $c+4$.

The proof technique for Theorem 8 is similar to that for Theorem 7. This is achieved by modifying the matrix $A_{(\psi,c)}$ constructed in the reduction described for Lemma 11. The largest entry in $A_{(\psi,c)}$ is $2^{\frac{n}{c}}-1$ (see Equation (6)). So each of these values can be represented by a binary string of length at most $\ell = \frac{n}{c}$. We remove each row, say row indexed by γ, with entries greater than 1 and replace it with $\frac{n}{c}$ rows, $\gamma_1, \ldots, \gamma_\ell$. Where, for any j, if the value $A_{(\psi,c)}[\gamma, j] = W$ then $A_{(\psi,c)}[\gamma_k, j] = \eta_k$, where η_k is the k^{th} bit in the ℓ-sized binary representation of W. This modification reduces the largest entry in $A_{(\psi,c)}$ to 1 and increases the path-width from constant to approximately n. Finally, we set all the entries in $b_{(\psi,c)}$ to be 1. This concludes the overview of our reductions.

4.1 Proof of Theorem 7

In this section we give a more detailed sketch of the proof of Theorem 7. Towards this, we first present the main details in the proof of our most technical lemma (Lemma 11).

Let $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an instance of CNF-SAT with variable set $X = \{x_1, x_2, \ldots, x_n\}$ and let $c \geq 2$ be a fixed constant given in the statement of Lemma 11. We construct the instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ of (IP) as follows.

Construction. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$. Without loss of generality, we assume that n is divisible by c, otherwise we add at most c dummy variables to X such that $|X|$ is divisible by c. We divide X into c blocks $X_0, X_1, \ldots, X_{c-1}$. That is $X_i = \{x_{\frac{i\cdot n}{c}+1}, x_{\frac{i\cdot n}{c}+2}, \ldots, x_{\frac{(i+1)\cdot n}{c}}\}$ for each $i \in \mathbb{Z}_c$. Let $\ell = \frac{n}{c}$ and $L = 2^\ell$. For each block X_i, there are exactly 2^ℓ assignments. We denote these assignments by $\phi_0(X_i), \phi_1(X_i), \ldots, \phi_{L-1}(X_i)$.

Now, we create $m \cdot c \cdot 2^{\ell+1}$ variables; they are named $y_{C,i,a}$, where $C \in \mathcal{C}$, $i \in \mathbb{Z}_c$ and $a \in \mathbb{Z}_{2L} = \mathbb{Z}_{2^{\ell+1}}$. In other words, for a clause C, a block X_i and an assignment $\phi_a(X_i)$, we create two variables; they are $y_{C,i,2a}$ and $y_{C,i,2a+1}$. Then, we create the (IP) constraints given by Equations (4), (5), and (6).

This completes the construction of (IP) instance. Let $A_{(\psi,c)}y = b_{(\psi,c)}$ be the (IP) instance defined using Equations (4), (5), and (6). The purpose of Equation (4) is to ensure satisfiability of all the clauses. Because of Equation (5), for each clause C and for each block X_i, we select only one assignment. Notice, that, so far it is allowed to choose many assignments from a block X_i, for different clauses. To ensure the consistency of assignments in each block across clauses, we added a system of constraints (Equation (6)). Equation (6) ensures the consistency of assignments in the adjacent clauses (in the order $C_1, \ldots, C_m$). Thus, the consistency of assignments propagates in a sequential manner. Notice that number constraints defined by Equations (4), (5), and (6) are m, $m \cdot c$ and $(m-1) \cdot c$, respectively. The number of variables is $m \cdot c \cdot 2^{\ell+1}$. Also notice that all the coefficients in Equations (4), (5) and (6) are non-negative. This implies that $A_{(\psi,c)}$ is non-negative and has dimension $\mathcal{O}(m) \times \mathcal{O}(m2^{\frac{n}{c}})$. Thus, the property (*b.*) of Lemma 11 is satisfied. The largest entry in $b_{(\psi,c)}$ is $L-1 = 2^{\lceil \frac{n}{c} \rceil} - 1$ (see Equation (6)) and hence the property (*d.*) of Lemma 11 is satisfied. The complete details for the proof of property (*a.*) can be found in the appended full version. Moving forward, we simplify the notation by using A instead of $A_{(\psi,c)}$ and b instead of $b_{(\psi,c)}$.

Now we need to prove property (*c.*) of Lemma 11. That is the path-width of A is at most $c+4$. Towards that we need to understand the structure of matrix A. We decompose the matrix A into m disjoint submatrices $B_1, \ldots B_m$ which are disjoint and cover all the non-zero

entries in the matrix A. First we define some notations and fix the column indices of A corresponding the the variables in the constraints. Let Y denote the set $\{y_{C,i,a} \mid C \in \mathcal{C}, i \in \mathbb{Z}_c, a \in \mathbb{Z}_{2L}\}$ of variables in the constraints defined by Equations (4), (5) and (6). These variables can be partitioned into $\biguplus_{C\in\mathcal{C}} Y_C$, where $Y_C = \{y_{C,i,a} \mid i \in \mathbb{Z}_c, a \in \mathbb{Z}_{2L}\}$. Further for each $C \in \mathcal{C}$, Y_C can be partitioned into $\bigcup_{i\in\mathbb{Z}_c} Y_{C,i}$, where $Y_{C,i} = \{y_{C,i,a} \mid a \in \mathbb{Z}_{2L}\}$. The set of columns indexed by $[r \cdot c\dot{2}^{\ell+1}] \setminus [(r-1)\cdot c \cdot 2^{\ell+1}]$, for any $r \in [m]$, corresponds to the set of variables in Y_{C_r}. Among the set of columns corresponding to Y_C, the first $2^{\ell+1}$ columns corresponds to the variables in $Y_{C,1}$, second $2^{\ell+1}$ columns corresponds to the variables in $Y_{C,2}$, and so on. Among the set of columns corresponds to $Y_{C,i}$ for any $C \in \mathcal{C}$ and $i \in \mathbb{Z}_c$, the first two columns corresponds to the variable $y_{C,i,0}$ and $y_{C,i,1}$, and second two columns corresponds to the variables $y_{C,i,2}$ and $y_{C,i,3}$, and so on.

Now we move to the description of B_j, $j \in [m]$. The matrix B_j will cover the coefficients of Y_{C_j} in Equations (4), (5) and (6). In other words B_j covers the non-zero entries in the columns corresponding to Y_{C_j}, i.e, in the columns of A indexed by $[j \cdot c \cdot 2^{\ell+1}] \setminus [(j-1)\cdot c \cdot 2^{\ell+1}]$. Now we explain these submatrices. Each matrix B_j has $c \cdot 2^{\ell+1}$ columns; each of them corresponds to a variable in Y_{C_j}. Each row in A corresponds to a constraint in the system of equations defined by Equations (4), (5) and (6). So we use notations $f(C_1), \ldots f(C_m)$ to represents the constraints defined by Equations (4). Similarly we use notations $\{s(C,i) \mid \ C \in \mathcal{C}, i \in \mathbb{Z}_c\}$ and $\{t(C,i) \mid \ C \in \mathcal{C}, i \in \mathbb{Z}_c\}$ to represent the constraints defined by Equations (5) and (6), respectively.

Matrices B_r for $1 < r < m$. Matrix B_r is of dimension $(3c+1) \times (c \cdot 2^{\ell+1})$. The first c rows are defined by Equation (6). For $j \in [c]$, in i^{th} row, we have coefficients of Y_{C_r} from $t(C_{r-1}, i)$. In the $(c+1)^{st}$ row of B_r, we have coefficients of Y_{C_r} from $f(C_r)$. For $i \in [c]$, the rows indexed by $c+1+i$ and $2c+1+i$ are defined as follows. In the $(c+1+i)^{th}$ row of B_r, we have coefficients of Y_{C_r} from $s(C_r, i)$ while in the $(2c+1+i)^{th}$ row of B_r, we have coefficients of Y_{C_r} from $t(C_r, i)$. This completes the definition of B_r. By their role in the reduction, the matrix B_r is partitioned in to four parts. The part composed of the first c rows is called the *predecessor matching part*. The part composed of the row indexed by $c+1$ is called the *evaluation part* of B_1. The part composed of rows indexed by $c+2, c+3, \ldots, 2c+1$ is called *selection part* and the part composed of last c rows is called *successor matching part*. That is the entries of B_1 are as follows, where $i \in \mathbb{Z}_c$ and $a \in \mathbb{Z}_L$.

The predecessor matching part is defined by

$$B_r[i+1, i\cdot 2^{\ell+1} + 2a + 1] = B_r[i+1, i\cdot 2^{\ell+1} + 2a + 2] = L - 1 - a. \tag{7}$$

The evaluation part is defined by

$$B_r[c+1, i\cdot 2^{\ell+1} + 2a + 2] = 0, \tag{8}$$

and

$$B_r[c+1, i\cdot 2^{\ell+1} + 2a + 1] = \begin{cases} 1, & \text{if } \phi_a(X_i) \text{ satisfies } C_r, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The selection part for B_r is defined as

$$B_r[c+2+i, i\cdot 2^{\ell+1} + 2a + 1] = B_r[c+2+i, i\cdot 2^{\ell+1} + 2a + 2] = 1, \tag{10}$$

The successor matching part for B_r is defined as

$$B_r[2c+2+i, i\cdot 2^{\ell+1} + 2a + 1] = B_r[2c+2+i, i\cdot 2^{\ell+1} + 2a + 2] = j. \tag{11}$$

All other entries in B_r, which are not listed above, are zero. That is, for all $i, i' \in \mathbb{Z}_c$ and $g \in [2^{\ell+1}]$ such that $i \neq i'$,

$$B_r[i+1, i' \cdot 2^{\ell+1} + g] = 0, \tag{12}$$

$$B_r[c+2+i, i' \cdot 2^{\ell+1} + g] = 0, \text{ and} \tag{13}$$

$$B_r[2c+2+i, i' \cdot 2^{\ell+1} + g] = 0. \tag{14}$$

Matrices B_1 and B_m. These have a slightly different structure. Informally, B_1 and B_m can be defined like B_r, $1 < r < m$, but we delete first c rows to get B_1 and delete last c rows to get B_m. A brief description of B_1 and B_m is given below.

Matrix B_1 is of dimension $(2c+1) \times (c \cdot 2^{\ell+1})$. In the first row of B_1, we have coefficients of Y_{C_1} from $f(C_1)$. For $i \in \mathbb{Z}_c$, the rows indexed by $2+i$ and $c+2+i$ are defined as follows. In the $(2+i)^{th}$ row of B_1, we have coefficients of Y_{C_1} from $s(C_1, i)$ while in the $(c+2+i)^{th}$ row of B_1, we have coefficients of Y_{C_1} from $t(C_1, i)$.

Matrix B_m is of dimension $(2c+1) \times (c \cdot 2^{\ell+1})$. For $j \in [c]$, in i^{the} row, we have coefficients of Y_{C_m} from $t(C_{m-1}, i)$. In the $(c+1)^{st}$ row of B_r, we have coefficients of Y_{C_m} from $f(C_m)$. In the $(c+1+i)^{th}$ row of B_m, we have coefficients of Y_r from $s(C_m, i)$.

Matrix A. Now we explain how the matrix A is formed from $B_1, \ldots, B_m$. The matrices $B_1, \ldots, B_m$ are disjoint submatrices of A and they cover all non zero entries of A. Informally, the submatrices $B_1, \ldots, B_m$ form a chain such that the rows corresponding to the successor matching part of B_r will be the same as the rows in the predecessor matching part of B_{r+1} (because of Equation (6)). Formally, let $I_1 = [2c+1]$ and $I_m = [(m-1)(2c+1) + (c+1)] \setminus [(m-1)(2c+1) - c]$. For every $1 < r < m$, let $I_r = [r(2c+1)] \setminus [(r-1)(2c+1) - c]$, and for $r \in [m]$, let $J_r = [r \cdot c \cdot 2^{\ell+1}] \setminus [(r-1) \cdot c \cdot 2^{\ell+1}]$. Now for each $r \in [m]$, the matrix $A[I_r, J_r] := B_r$. All other entries of A not belonging to any of the submatrices $A[I_r, J_r]$ are zero.

Towards upper bounding the path-width of A, we start with some notations. We partition the set of columns of A into m parts $J_1, \ldots, J_m$ (we have already defined these sets) with one part per clause. For each $r \in [m]$, J_r is the set of columns associated with Y_{C_r}. We further divide J_r into c equal parts, one per variable set $Y_{C_r,i}$. These parts are

$$P_{r,i} = \{(r-1)c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + 1, \ldots, (r-1)c \cdot 2^{\ell+1} + (i+1) \cdot 2^{\ell+1}\},\ i \in \mathbb{Z}_c.$$

In other words, $P_{r,i}$ is the set of columns corresponding to $Y_{C_r,i}$ and $|P_{r,i}| = 2^{\ell+1}$. We also put $n' = m \cdot c \cdot 2^{\ell+1}$ to be the number of columns in A.

▶ **Lemma 12.** *The path-width of the column matroid of A is at most $c+4$*

Proof. Recall that $n' = m \cdot c \cdot 2^{\ell+1}$, is the number of columns in A and m' the number of rows in A. To prove that the path-width of A is $\leq c+4$, it suffices to show that for all $j \in [n'-1]$,

$$\dim\langle \text{span}(A|\{1, \ldots, j\}) \cap \text{span}(A|\{j+1, \ldots, n'\})\rangle \leq c+3. \tag{15}$$

The idea for proving Equation (15) is based on the following observation. For $V' = A|\{1, \ldots, j\}$ and $V'' = A|\{j+1, \ldots, n'\}$, let $I = \{q \in [m'] \mid \text{there exist } v' \in V' \text{ and } v'' \in V'' \text{ such that } v'[q] \neq v''[q] \neq 0\}$. Then the dimension of $\text{span}(V') \cap \text{span}(V'')$ is at most $|I|$. Thus to prove (15), for each $j \in [n'-1]$, we construct the corresponding set I and show that its cardinality is at most $c+3$.

We proceed with the details. Let $v_1, v_2, \ldots, v_{n'}$ be the column vectors of A. Let $j \in [n'-1]$. Let $V_1 = \{v_1, \ldots, v_j\}$ and $V_2 = \{v_{j+1}, \ldots, v_{n'}\}$. We need to show that $\dim\langle \text{span}(V_1) \cap$

$\text{span}(V_2)\rangle \leq c+3$. Let $I' = \{q \in [m'] \mid \text{there exists } v \in V_1 \text{ and } v' \in V_2 \text{ such that } v[q] \neq 0 \neq v'[q]\}$. We know that $[n']$ is partitioned into parts $P_{r',i'}, r' \in [m], i' \in \mathbb{Z}_c$. We fix $r \in [m]$ and $i \in \mathbb{Z}_c$ such that $j \in P_{r,i}$.

Let $j = (r-1)c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + g$, where $g \in [2^{\ell+1}]$. Let $q_1 = \max\{0, (r-1)(2c+1) - c\}$, $q_2 = r(2c+1)$, $j_1 = (r-1) \cdot c \cdot 2^{\ell+1}$, and $j_2 = r \cdot c \cdot 2^{\ell+1}$. Then $[q_2] \setminus [q_1] = I_r$ and $[j_2] \setminus [j_1] = J_r$ (recall the definition of sets I_r and J_r).

By the decomposition of matrix A, for every $q > q_2$ and for every vector $v \in V_1$, we have $v[q] = 0$. Also, for every $q \leq q_1$ and for any $v \in V_2$, we have that $v[q] = 0$. This implies that $I' \subseteq [q_2] \setminus [q_1] = I_r$. Now we partition I_r into 4 parts: R_1, R, S, and R_2, These parts are defined as follows.

$$\begin{aligned} R_1 &= \begin{cases} \emptyset, & \text{if } r = 1, \\ \{(r-2)(2c+1) + i' \mid i' \in \mathbb{Z}_c\}, & \text{otherwise,} \end{cases} \\ R &= \{(r-1)(2c+1) + 1\}, \\ S &= \{(r-1)(2c+1) + 2 + i' \mid i' \in \mathbb{Z}_c]\}, \\ R_2 &= \begin{cases} \emptyset, & \text{if } r = m, \\ \{(r-1)(2c+1) + c + 2 + i' \mid i' \in \mathbb{Z}_c\}, & \text{otherwise} \end{cases} \end{aligned} \tag{16}$$

We complete the proof of the lemma by proving the following series of claims. We first show that for each $r' \in [m]$ such that $q \notin I_{r'}$ and $j'' \in J_{r'}$, $v_{j''}[q] = 0$. Following that, we show that $|I' \cap R_1| \leq c - (i-1)$. The final two claims in this series of claims are (i) $|I' \cap R_2| \leq i$, and (ii) $|I' \cap S| \leq 1$.

With the help of these claims, we can conclude the following. $|I'| = |I' \cap I_r|$ (since $I' \subseteq I_r$) and $|I' \cap I_r| = |I' \cap R_1| + |I' \cap R| + |I' \cap S| + |I' \cap R_2|$ (by (16)), which implies that $|I'| \leq c - (i-1) + 1 + 1 + i = c + 3$. This completes the proof of the lemma. ◀

5 Conclusion

While Theorems 3 and 4 come close to the bound of Proposition 1, the precise multivariate complexity of (IP) with respect to the parameters n, m, Δ, and $\|b\|_\infty$ is not fully clear and our work leaves some unanswered questions regarding the landscape of tradeoffs between the parameters. For instance, is it possible to solve (IP) in time $(m \cdot n \cdot \Delta)^{o(m)} \cdot (\|b\|_\infty)^{\mathcal{O}(1)}$, or $(m \cdot n \cdot \Delta \cdot \|b\|_\infty)^{o(m)}$? Or could one improve our lower bound results to rule out such algorithms? While our SETH-based lower bounds for (IP) with non-negative constraint matrix are tight for path-width parameterization, there is a "$(\|b\|_\infty + 1)^k$ to $(\|b\|_\infty + 1)^{2k}$ gap" between lower and upper bounds for branch-width parameterization. Closing this gap is a natural question.

The bottleneck in the algorithm of Cunningham and Geelen is the following subproblem. We are given two vector sets A and B of partial solutions, each set of size at most $(\|b\|_\infty + 1)^k$. We need to construct a new vector set C of partial solutions, where the set C will have size at most $(\|b\|_\infty + 1)^k$ and each vector from C is the *sum* of a vector from A and a vector from B. Thus to construct the new set of vectors, one has to go through all possible pairs of vectors from both sets A and B, which takes time roughly $(\|b\|_\infty + 1)^{2k}$.

A tempting approach towards speeding up this particular step could be the use of *fast subset convolution* or *matrix multiplication* tricks, which work very well for "join" operations in dynamic programming algorithms over tree and branch decompositions of graphs [5, 18, 4], see also [3, Chapter 11]. Unfortunately, we have reason to suspect that these tricks may *not* help for matrices: solving the above subproblem in time $(\|b\|_\infty + 1)^{(1-\epsilon)2k} n^{\mathcal{O}(1)}$ for any $\epsilon > 0$ would imply that 3-SUM is solvable in time $n^{2-\epsilon}$, which is believed to be unlikely.

References

1. William H. Cunningham and Jim Geelen. On integer programming and the branch-width of the constraint matrix. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 4513 of *Lecture Notes in Comput. Sci.*, pages 158–166. Springer, 2007.
2. 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. In *Proceedings of the 27th IEEE Conference on Computational Complexity (CCC)*, pages 74–84. IEEE, 2012. doi:10.1109/CCC.2012.36.
3. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
4. Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE, 2011.
5. Frederic Dorn. Dynamic programming and fast matrix multiplication. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, volume 4168 of *Lecture Notes in Comput. Sci.*, pages 280–291. Springer, Berlin, 2006.
6. Friedrich Eisenbrand and Robert Weismantel. Proximity results and faster algorithms for integer programming using the steinitz lemma. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 808–816. SIAM, 2018.
7. G. B. Horn and Frank R. Kschischang. On the intractability of permuting a block code to minimize trellis complexity. *IEEE Trans. Information Theory*, 42(6):2042–2048, 1996. doi:10.1109/18.556701.
8. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
9. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *J. Computer and System Sciences*, 63(4):512–530, 2001.
10. K. Jansen and L. Rohwedder. On Integer Programming and Convolution. *ArXiv e-prints*, 2018. arXiv:1803.04744.
11. Jisu Jeong, Eun Jung Kim, and Sang-il Oum. Constructive algorithm for path-width of matroids. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1695–1704. SIAM, 2016. doi:10.1137/1.9781611974331.ch116.
12. Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.
13. Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.
14. Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 777–789. SIAM, 2011.
15. Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. arXiv:toc:v006/a005.
16. Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981. doi:10.1145/322276.322287.
17. Neil Robertson and Paul D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combinatorial Theory Ser. B*, 52(2):153–190, 1991. doi:10.1016/0095-8956(91)90061-N.
18. Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, volume 5757 of *Lecture Notes in Comput. Sci.*, pages 566–577. Springer, 2009.