Exponentially faster fixed-parameter algorithms for high-multiplicity scheduling

David Fischer∗ Julian Golak† Matthias Mnich‡

Abstract

We consider so-called $N$-fold integer programs (IPs) of the form $\max\{c^T x : Ax = b, \ell \leq x \leq u, x \in \mathbb{Z}^n\}$, where $A \in \mathbb{Z}^{(r+s)n \times nt}$ consists of $n$ arbitrary matrices $A^{(i)} \in \mathbb{Z}^{r \times t}$ on a horizontal, and $n$ arbitrary matrices $B^{(j)} \in \mathbb{Z}^{s \times t}$ on a diagonal line. Several recent works design fixed-parameter algorithms for $N$-fold IPs by taking as parameters the numbers of rows and columns of the $A$- and $B$-matrices, together with the largest absolute value $\Delta$ over their entries. These advances provide fast algorithms for several well-studied combinatorial optimization problems on strings, on graphs, and in machine scheduling.

In this work, we extend this research by proposing algorithms that additionally harness a partition structure of submatrices $A^{(i)}$ and $B^{(j)}$, where row indices of non-zero entries do not overlap between any two sets in the partition. Our main result is an algorithm for solving any $N$-fold IP in time $nt \log(nt)L(S_A)O^{r+s}(p_Ap_B\Delta)p_Ap_BSP_Ap_B+SP_Ap_B)$, where $p_A$ and $p_B$ are the size of the largest set in such a partition of $A^{(i)}$ and $B^{(j)}$, respectively, $S_A$ is the number of parts in the partition of $A = (A^{(1)}, \ldots, A^{(n)})$, and $L = (\log(||u - \ell||_\infty) \cdot (\log(\max_{x, \ell \leq x \leq u} |c^T x|)))$ is a measure of the input.

We show that these new structural parameters are naturally small in high-multiplicity scheduling problems, such as makespan minimization on related and unrelated machines, with and without release times, the Santa Claus objective, and the weighted sum of completion times. In essence, we obtain algorithms that are exponentially faster than previous works by Knop et al. (ESA 2017) and Eisenbrand et al./Koutecký et al. (ICALP 2018) in terms of the number of job types. We also improve the complexity of minimum sum coloring on graphs of bounded neighborhood diversity.

Keywords — $N$-Fold, Integer Programming, High-Multiplicity Scheduling

1 Introduction

An integer program (IP) is a mathematical program of the form

$$\max\{c^T x : Mx = b, \ell \leq x \leq u, x \in \mathbb{Z}^n\}$$

which is described by a constraint matrix $M \in \mathbb{Z}^{m \times n}$, an objective function vector $c \in \mathbb{Z}^n$, a right-hand side vector $b \in \mathbb{Z}^m$ and lower and upper bounds $\ell \leq x \leq u$. Many fundamental problems can be formulated as IPs, which has therefore become one of the most important paradigms in optimization. As solving these problems is generally NP-hard, IPs have received much attention in the field of parameterized complexity [5, 11, 17, 19]. In parameterized complexity, the problem input is augmented by some

∗Hamburg University of Technology, Institute for Algorithms and Complexity, Germany. da.fischer@tuhh.de. Supported by DFG grant MN 59/4-1.
†Hamburg University of Technology, Institute for Algorithms and Complexity, Germany. julian.golak@tuhh.de. Supported by HIDSS-0002 DASHH (Data Science in Hamburg - HELMHOLTZ Graduate School for the Structure of Matter).
‡Hamburg University of Technology, Institute for Algorithms and Complexity, Germany. matthias.mnich@tuhh.de

arXiv:2203.03600v1 [cs.DS] 7 Mar 2022.
parameter $k \in \mathbb{N}$. The run time of an algorithm solving this problem is then expressed in terms of the instance size $|I|$ as well as a computable function $f$ of $k$. The major goal is to multiplicatively separate the dependence on $k$ and $|I|$, to obtain fixed-parameter algorithms of run time $f(k)|I|^{O(1)}$; the problems solved by such algorithms are called fixed-parameter tractable.

A natural way to construct such fixed-parameter algorithms is to impose some structure on IP formulations, such that solving these IPs becomes fixed-parameter tractable in parameters measuring the size of those structures. One example of such a structure is the so-called $N$-fold block structure, which leads to IPs of the form

$$\max\{c^T x : Ax = b, \ell \leq x \leq u, x \in \mathbb{Z}^{nt}\}$$

(1)

$$A = \begin{pmatrix}
A^{(1)} & A^{(2)} & \cdots & A^{(n)} \\
B^{(1)} & 0 & \cdots & 0 \\
0 & B^{(2)} & \cdots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & B^{(n)}
\end{pmatrix},$$

(2)

where $A^{(i)} \in \mathbb{Z}^{r \times t}$, $B^{(i)} \in \mathbb{Z}^{s \times t}$ for $i = 1, \ldots, n$, $A \in \mathbb{Z}^{(r+s)n \times nt}$ and $b \in \mathbb{Z}^{nt}$. Further, we call the program a uniform $N$-fold IP if $A^{(1)} = A^{(2)} = \ldots = A^{(n)}$ and $B^{(1)} = B^{(2)} = \ldots = B^{(n)}$.

In recent literature in parameterized complexity, the $N$-fold block structure has been used very successfully to obtain new algorithmic results on various problems [3, 12, 19]. Let $\varphi$ be the largest binary encoding length of any coefficient of $c$, and let $\Delta$ be an upper bound on the absolute value of each entry of the constraint matrix $A$. First results on the subject of parameterized complexity of $N$-fold IPs were made by Hemmecke et al. [13] using augmentation methods based on so-called Graver bases. This translated to the first fixed-parameter algorithm for $N$-fold IPs parameterized by $r, s, t$ and $\Delta$, with a run time of $n^2 \varphi \Delta O(1/(\ell_1+\ell_2))$. That result was improved by removing the exponential dependence on $t$ by Eisenbrand et al. [5] and Koutecký et al. [21]. The currently best run times for general $N$-fold IPs are

- $nt \log(nt)L(r s \Delta)O(r^2 s + n^2)$ by Eisenbrand et al. [5],
- $nt \varphi^2 \log_2 O(1)(nt)(r s \Delta)O(r^2 + n^2)$ by Jansen et al. [16], and
- $2O(r^2 s)(r s \Delta)O(r^2 + n^2)(nt)^{1+o(1)}$ by Cslovjescek et al. [4].

Here, $L$ denotes a certain measure of the input, as defined by Eisenbrand et al. [5]. Specifically, $L := (\log(||u - \ell||_\infty) - \log(c_{\max}))$, where $c_{\max} := \max_{x, \ell \leq x \leq u} |c^T x|$ for the objective function $c$. Note that $L \leq |I|^2$, for the encoding size $|I|$ in bits of a problem instance $I$ that is described by the IP. Eisenbrand et al. [5] develop an algorithmic framework based on the idea of iterative augmentation by using the Graver basis of the matrix $A$. Through small upper bounds on the $\ell_1$-norm of the elements of the Graver basis, which depend on the block parameters $r, s, t$ and the largest coefficient $\Delta$ of the constraint matrix, they obtain fast algorithms for solving $N$-fold IPs. Jansen et al. [16] introduce a technique based on color-coding, which allows to compute the augmentation steps more efficiently in regard to the number of variables in the input. Finally, the improvements by Cslovjescek et al. [4] rely on parametric search and new proximity bounds.

Algorithms for $N$-fold IPs can be used to design fixed-parameter algorithms in various domains. One such domain is high-multiplicity scheduling, where one seeks an assignment of jobs to machines so as to minimize some objective. The notion of high-multiplicity scheduling was introduced by Hochbaum and Shamir [12]; it refers to the fact that the input is not lavishly encoded by listing all $n$ jobs one by one with their properties, but rather is specified as a small number $d \ll n$ of job types with their properties, and numbers $n_1, \ldots, n_d$ telling the number of jobs of each particular type. Algorithms which are efficient for high-multiplicity encodings must therefore be much more efficient than those for lavish encodings, as $d$ is usually much smaller than $n = n_1 + \ldots + n_d$. Typical parameters for fixed-parameter algorithms in high-multiplicity scheduling are the number $d$ of distinct job types, and the largest processing time $p_{\text{max}}$ of any job. For those parameters, several fixed-parameter algorithms have been suggested in various

\footnote{That work extends two works from the ICALP 2018 proceedings by Eisenbrand et al. and Koutecký et al.}
scheduling scenarios; we refer to a recent survey by Mnich and van Bevern [23]. Knop and Koutecký [17] devised an N-fold formulation for makespan minimization on m uniformly related machines, to solve the problem in time \((p_{\max})^{O(n)} m^{1/|l|}\). Thus, theirs is a fixed-parameter algorithm for parameters \(d\) and \(p_{\max}\); it has the currently best dependencies for these parameters. The major advances for solving N-fold IPs by Eisenbrand et al. [6], Jansen et al. [10] and Cseh and Koutecký [4] did not translate to improvements in the parameter dependence of the run time of makespan minimization on uniformly related machines. Notable other related work was done by Knop et al. [18], who gave fixed-parameter algorithms for solving high-multiplicity scheduling problems on uniformly related machines based on the configuration IP.

Our contributions

The main contribution of this paper is threefold.

First, we prove new bounds on \(\ell_1\)-norms on the elements of the Graver bases of the constraint matrix \(A\), which not only depend on the block parameters \(r, s, t\) and the largest coefficient \(\Delta\) of \(A\), but also some partition parameters of the \(A\) and \(B\)-matrices.

Those partition parameters characterize the overlap of non-zero entries of rows, both in general IPs and in N-fold IPs. Given any matrix \(M\), we consider a partition on its rows, \(P_1, \ldots, P_{S_M}\), for some \(S_M \in \mathbb{N}\). Two rows are in the same partition class if they share at least one entry with a non-zero element. Let \(p_M\) be the size of the largest set in the partition. Thus, the partition parameters of the submatrices \(A\) and \(B\) of an N-fold IP are \(S_A, p_A\) and \(p_B\), respectively. Precisely, we show that the \(\ell_1\)-norm of an element of the Graver basis of a given matrix \(M \in \mathbb{Z}^{m \times n}\) is bounded by \((2p_M \Delta + 1)^{p_{AB}}\).

Second, through the refined bounds on the \(\ell_1\)-norm of Graver basis elements of the constraint matrix we obtain an algorithm for solving N-fold IPs whose run time is expressible in both the block parameters and partition parameters:

**Theorem 1.** There is an algorithm which solves any N-fold IP in time \(nt \log(nt) L(S_A)^{O(r+s)} (p_{APB} \Delta)^{O(p_{APB} + sp_{APB})}\).

Here, \(P_A\) is a partition of \(A = (A(1), \ldots, A(n))\) as described above, with its respective parameters \(p_A\), as well as \(p_B = \max_{i=1, \ldots, n} p_{Bi}\). As \(S_A\) and \(p_A\) are at most \(r\), and \(p_B\) is at most \(s\), the run time of this algorithm is asymptotically always smaller than or equal to the run time of the aforementioned result by Eisenbrand et al. [6] for solving N-fold IPs.

Third, we show that several classical high-multiplicity scheduling problems exhibit partitions where those parameters are small. Combining these results with using [Theorem 1] and the augmentation approach by Eisenbrand et al. [6], we obtain algorithms for those problems that are exponentially faster than previous algorithms when measured in terms of the number \(d\) of job types. Let us remark that in our model, the number of machines is polynomially bounded (as we specify for each machine its speed).

In contrast, Knop et al. [18] allow the number \(m\) of machines may be exponential in the input size, yet the run-times in that work are not competitive with our results in terms of \(d\) and \(p_{\max}\).

Another application of our results is the MINIMUM SUM COLORING problem in graphs of neighborhood diversity at most \(k\). We give an overview of the improved run times in Table 1.

2 The augmentation approach for solving IPs

For positive integers \(n, r, s, t\) and \(i = 1, \ldots, n\), we define matrices \(A(i) \in \mathbb{Z}^{r \times t}, B(i) \in \mathbb{Z}^{s \times t}\), with \(\|A(i)\|_{\infty}, \|B(i)\|_{\infty} \leq \Delta\) for some constant \(\Delta\). Furthermore, let \(A\) be an \((r + ns) \times nt\)-matrix, subject to an N-fold block structure of \(A(i)\) and \(B(i)\). We write the \(i\)th entry of a vector \(v\) as \(v_i\). The support \(\text{supp}(v)\) of a vector \(v\) is defined as the set of indices \(i\) of all non-zero entries \(v_i\) of \(v\), the support \(\text{supp}(M)\) of a matrix \(M\) is defined as the set of indices of all non-zero columns of \(M\). With \(\log(x)\) we denote the logarithm \(\log_2(x)\) with basis 2 for any number \(x\). Recall that \(L\) is defined as \(L = (\log(\|u - \ell\|_{\infty}) \cdot \log(\max_i x_i))\).

Graver bases and augmentation. Two vectors \(u, v \in \mathbb{R}^n\) are called sign-compatible if \(u_i, v_i \geq 0\) for all \(i\). For any matrix \(M \in \mathbb{Z}^{m \times n}\), its kernel \(\ker(M)\) is the set of all vectors \(y\) such that \(My = 0\). If \(y\) is an integer vector, it is called a cycle of \(M\). Such a cycle \(y\) is called indecomposable, if it is not the sum of two sign-compatible and non-zero cycles of \(M\). The set of these indecomposable cycles \(y \in \ker(M)\) is called the Graver basis \(G(M)\) of \(M\) [10].
With this definition of a Graver basis, we can define the general Graver-best augmentation procedure that gives the optimal solution \( x^* \) for any IP, given a feasible solution \( x^0 \). For an IP of the form \( \max \{ c^T x : Mx = b, \ell \leq x \leq u, x \in \mathbb{Z}^n \} \), we call \( y \) an augmenting step if \( c^T (x + y) > c^T x \) and \( x + y \) is feasible, which means that \( M(x + y) = b, \ell \leq x + y \leq u \). Then, a tuple \( (y, \lambda) \) with \( \lambda \in \mathbb{Z} \) is a Graver-best step, if \( y \) is an augmenting step and \( \lambda g^T (x + \lambda y) \leq c^T (x + \lambda y) \), for all \( \lambda \in \mathbb{Z}, g \in \mathcal{G}(M) \). The Graver-best augmentation procedure can now be defined as follows:

1. If there is no Graver-best step for \( x^0 \), return it as optimal.
2. If \( x^0 = x^0 + \lambda y \) and jump to 1.

Eisenbrand et al. [6] introduce a relaxation of the Graver-best step: A so-called halving. An augmenting step \( h \) is a halving for \( x \) if \( c^T x + c(x + h) \geq \frac{1}{2} (c^T x + c(x + \lambda g)) \) for all feasible step pairs \( (g, \lambda) \). The halving augmentation procedure is then defined as follows:

1. If there is no Graver-best step for \( x^0 \), return it as optimal.
2. If there exists a halving \( h \) for \( x^0 \), set \( x^0 = x^0 + h \) and jump to 1.

The main advantage of the halving augmentation procedure over the Graver-best step augmentation procedure is that it converges faster to an optimal solution in the parameter \( n \), as shown in the following result. Let \( g_1(A) \) be an upper bound on the \( \ell_1 \)-norm of the Graver basis \( \mathcal{G}(A) \) of an \( N \)-fold matrix \( A \).

**Proposition 1** (Eisenbrand et al. [6, Theorem 77, Lemma 90]). There is an algorithm realizing the halving augmentation procedure for \( N \)-fold IPs in time \( (\Delta g_1(A))^{O(v + s)} n \log(n) L \).

Using this result, and the fact that, for this procedure, constructing an initial feasible solution is as hard as optimization [6, Corollary 55, Corollary 89], the authors obtain the following result:

**Proposition 2** (Eisenbrand et al. [6, Corollary 91]). Any \( N \)-fold IP can be solved in time \( nt \log(nt) L(r s \Delta)^{O(v + s + v + s)} \).

As the run time of this procedure is directly dependent on a bound \( g_1(A) \) on the \( \ell_1 \)-norm of Graver basis elements, an improvement of the lower bound directly implies an improvement of the run time.

Important for our results is also the so-called Steinitz Lemma, developed by Steinitz [22]. Let \( \| \cdot \| \) denote an arbitrary norm.

**Lemma 1** (Steinitz Lemma). Let \( v_1, \ldots, v_N \in \mathbb{R}^m \) be vectors with \( \| v_i \| \leq \Delta \) for \( i = 1, \ldots, N \). If \( \sum_{i=1}^{N} v_i = 0 \), then there exists a reordering \( \pi \), such that for each \( k \in \{1, \ldots, N\} \) the partial sum \( \sum_{i=1}^{k} v_{\pi(i)} \) satisfies \( \| \sum_{i=1}^{k} v_{\pi(i)} \| \leq m \Delta \).
3 The Graver basis $\ell_1$-norm and partition parameters

Let $M \in \mathbb{Z}^{m \times n}$ be an integer matrix with $m$ rows and $n$ columns. Let $R_M$ be the set of rows of the matrix $M$. We define a partition $\mathcal{P}_M$ of $R_M$ into non-empty sets of rows, such that the following holds for all elements $P_1, P_2$ of $\mathcal{P}_M$: if $r_1 \in P_1, r_2 \in P_2$, then their supports $\text{supp}(r_1)$ and $\text{supp}(r_2)$ are disjoint. We call such a partition $\mathcal{P}_M$ of $M$ column-independent. Recall that the size of the largest set of $\mathcal{P}_M$ is denoted as $p_M$, and the number of parts in the partition $\mathcal{P}_M$ as $S_M$. Note that both $p_M, S_M \in \{1, \ldots, m\}$.

**Lemma 2.** Let $M \in \mathbb{Z}^{m \times n}$ be an integer matrix and let $\Delta$ be an upper bound on the absolute value of $M$. Let $y$ be a Graver basis element of $M$, and let $\mathcal{P}_M = (P_1, \ldots, P_{S_M})$ be a column-independent partition of $M$. Then $||y||_1 \leq (2p_M \Delta + 1)^{p_M}$.

**Proof.** Let $R_i \in \mathbb{Z}^{|P_i| \times n}$ be the matrix induced by the rows in $P_i$, for $i = 1, \ldots, S_M$. We decompose $y$ into $S_M$ many vectors $y_i \in \mathbb{Z}^n$, with $y_i := y|_{P_i}$, if $j \in \text{supp}(R_i)$, and $y_i := 0$ otherwise, for $j = 1, \ldots, n$. By this definition, $\text{supp}(R_i) = \text{supp}(y_i)$; additionally, induced by the column-independent partition, $\text{supp}(y_i)$ and $\text{supp}(y_j)$ are disjoint for distinct $i, k, l = 1, \ldots, S_M$. It then holds that $y = y_1 + \ldots + y_{S_M}$. Furthermore, since $R_i \cdot y = 0$, it also follows that $R_i \cdot y_i = 0$.

We next proceed with an argument by Eisenbrand et al. [3]. We consider some $P_i$ and its corresponding $y_i$, with the latter’s $\ell_1$-norm denoted as $||y_i||_1$, and define a sequence of vectors $u_1, \ldots, u_{||y_i||_1} \in \mathbb{Z}^{|P_i|}$ in the following manner: If $y_i \geq 0$, we add $y_i$ copies of the $j^{th}$ column of $R_i$ to the sequence, if $y_i < 0$, we add $|y_i|$ negative copies of the column $j$ of $R_i$ to the sequence.

Clearly, the $u_j$ sum up to zero, and their $\ell_\infty$-norm is bounded by $\Delta$. Using Lemma 1 there is a reordering $u_1, \ldots, u_{||y_i||_1}$ of this sequence, with $u_j = v_{\pi(j)}$ for some permutation $\pi$, such that each partial sum $\sum_{j=1}^k u_j$, for $k \leq ||y_i||_1$, is bounded by $|P_i| \Delta$ in the $\ell_\infty$-norm. Note that

$$||x \in \mathbb{Z}^{|P_i|} : ||x||_\infty \leq |P_i| \Delta|| = (2|P_i| \Delta + 1)^{|P_i|}.$$

Suppose, for sake of contradiction, that $||y_i||_1 > (2|P_i| \Delta + 1)^{|P_i|}$. Then, at least two of these partial sums of the sequence $(u_1, \ldots, u_{||y_i||_1})$ are the same. Let $k_1, k_2 \in \{1, \ldots, ||y_i||_1\}$ be the indices of two of such equal partial sums, meaning that $\sum_{j=1}^{k_1} u_j = \sum_{j=1}^{k_2} u_j$. Assume, without loss of generality, that $k_1 < k_2$. Then, since $\sum_{j=k_1}^{k_2} u_j = 0$, both sequences $(u_{k_1+1}, \ldots, u_{k_2})$ and $(u_1, \ldots, u_{||y_i||_1}) \setminus (u_{k_1+1}, \ldots, u_{k_2})$ add up to zero.

We now construct two vectors $y''', y''' \in \mathbb{Z}^n$: For both vectors, we start with the 0-vector in $\mathbb{Z}^n$, and, for each $u_i \in u_{k_1+1}, \ldots, u_{k_2}$, we add 1 to the $j^{th}$ entry of $y'''$, in the expression $\pi^{-1}(u_i) = v_j$, the value $v_j$ is a positive copy of the $j^{th}$ column in $M$, or subtract 1, if $v_j$ is a negative copy of the $j^{th}$ column in $M$. We do the same for $y'''$ and $u_{k_1+1}, \ldots, u_{||y_i||_1} \setminus u_{k_1+1}, \ldots, u_{k_2}$.

We now have two vectors $y''', y'''$ satisfying the following properties: (i) both are non-zero, (ii) both $R_i \cdot y''' = 0$ and $R_i \cdot y''' = 0$ hold, (iii) $y''' + y''' = y'$, and (iv) both $y'''$ and $y'''$ are sign-compatible with $y$. It then directly follows from (iv) that both $y'''$ and $y'''$ are also sign-compatible with $y$.

Hence, as also $\text{supp}(R_i) \neq \text{supp}(R_{i+k})$ for all $i \neq k$, we can decompose $y$ into two sign-compatible cycles $y', y''$ of $M$: $y' := y - y' + y''$, and $y'' := y'''$. This contradicts the assumption that $y$ is a Graver basis element of $M$. Therefore, $||y'||_1 \leq (2|P_i| \Delta + 1)^{|P_i|}$ must hold.

Assume now that, for $y = y_1 + \ldots + y_{S_M}$, more than one $y_i$ has non-zero entries. Let $y_{i_1}$ and $y_{i_2}$ be two of those vectors with non-zero entries, for some indices $i_1$ and $i_2$. Let $P_{i_1}$ and $P_{i_2}$ be the respective sets of rows in the column-independent partition for $i_1$ and $i_2$. As $\text{supp}(P_{i_1}) \cap \text{supp}(P_{i_2}) = \emptyset$ and $y_{i_1}$ and $y_{i_2}$ are cycles of $R_{i_1}$ and $R_{i_2}$ respectively, both $y_{i_1}$ and $y_{i_2}$ are cycles of $M$. Since they are also both sign-compatible with $y$, it then is possible to decompose $y$ into two sign-compatible cycles $y_{i_1}$ and $y_{i_2}$. This is a contradiction to $y$ being a Graver basis element. Therefore, at most one $y_i$ can have non-zero entries. This implies $||y||_1 \leq ||y'||_1$ for each non-zero $y_i$. As $|P_i| \leq p_M$ for all $i$, we have that

$$||y||_1 \leq ||y'||_1 \leq (2p_M \Delta + 1)^{p_M},$$

proving the lemma. \hfill \square
4 The Graver basis $\ell_1$-norm of a partitioned $N$-fold matrix

In this section, we generalize the preceding results to obtain a lower bound of Graver basis elements of any $N$-fold matrix $A \in \mathbb{Z}^{(r+n)\times nt}$, with its submatrices $A^{(i)} \in \mathbb{Z}^{r\times t}$, and $B^{(i)} \in \mathbb{Z}^n\times t$.

We know from Lemma 2 that the $\ell_1$-norm of every Graver basis element $y^j$ of $B^{(i)}$ is bounded by $L_{B^{(i)}} := (2pB^{(i)} + 1)^{|B^{(i)}|}$. We define $L := \max_{i=1,\ldots,n}(L_{B^{(i)}})$ and $p := \max_{i=1,\ldots,n}(p_{B^{(i)}})$. It then follows for every $y^j$ that $$||y^j||_1 \leq L_B \leq (2pB + 1)^{pA}.$$ 

Let $A = (A^{(1)},\ldots, A^{(n)})$ and let $P_A = (P_1, \ldots, P_{SA})$ be a column-independent partition of $A$, with $p_A$ again being the size of the largest set in the partition, and $S_A$ being the number of parts in the partition. Let $R_i \in \mathbb{Z}^{P_i\times n}$ be again defined as the matrix induced by rows of $P_i$, with $R_i = R_i^{(1)}, \ldots, R_i^{(n)}$ exhibiting the same block-structure as $A$.

Let $y$ be a Graver basis element of $A$ and let $P_A$ be a decomposition of $y$ into $S_A$ many vectors $y^i \in \mathbb{Z}^n$, $y = y^1 + \ldots + y^S$ with $y^j = y_{ij}$, if $j \in \text{supp}(P_i)$, and $y^j = 0$, otherwise, for $j = 1, \ldots, n$. It holds that $R_i \cdot y^i = 0$, for all $i = 1, \ldots, S_A$.

**Lemma 3.** There exists a decomposition of each $y^i$ into vectors $(y_k^{(i)} + \ldots + y_N^{(i)})$, for some $N_1, \ldots, N_n \in \mathbb{N}$, such that $||y_k^{(i)}||_1 \leq L_B$ for $j \in \{1, \ldots, n\}, k \in \{1, \ldots, N_j\}$.

**Proof.** Let $g$ be a Graver basis element of $B^{(i)}$. As $||g||_1 \leq L_B$ and $||A^{(i)}||_\infty \leq \Delta$, the $\ell_\infty$-norm of the vector $A^{(i)}g$ is bounded by

$$||A^{(i)}g||_\infty \leq \Delta L_B.$$ \hfill (4)

We can also further split $y^i$ into bricks according to the matrices $B^{(i)}$, i.e. $y^i = (y^{(1)} + \ldots + y^{(N)})$. We can do the same split for $y$, $y = (y^{(1)} + \ldots + y^{(N)})$, with each $y^{(j)} \in \mathbb{Z}^n$ being a cycle of $B^{(j)}$. Hence, each $y^{(j)}$ can be decomposed into Graver basis elements $y^{(i)}_k$ of $B^{(i)}$, e.g. $y^{(j)} = y^{(j)}_1 + \ldots + y^{(j)}_{N_j}$ for some $N_j \in \mathbb{N}$.

Similarly, we can decompose each corresponding $y^{(j)}$ as follows. Let $y^{(j)}_k$ be the same as $y^{(j)}$, except for the entries where $y^{(j)}$ equals 0: Set these entries of $y^{(j)}$ to 0 as well. From this procedure, it follows that $y^{(j)} = y^{(j)}_1 + \ldots + y^{(j)}_{N_j}$.

Since $||y^i_j|| \leq ||y^i||$ for all $j \in \{1, \ldots, n\}$ and $y^{(j)}_k \leq L_B$, it also holds that $||y^{(j)}_k||_1 \leq ||y^{(j)}_k||_1 \leq L_B$, for $k = 1, \ldots, N_j$ and $j = 1, \ldots, n$. Therefore, the lemma follows.

**Lemma 4.** For any Graver basis element $y$ of $A$, it holds that $||y||_1 \leq |P_A|L_B(2p_A\Delta L_B + 1)^{p_A}$.

**Proof.** Consider some $y^i$, for $i \in \{1, \ldots, S_A\}$. According to Lemma 3 there exists a decomposition of $y^i$ into vectors $(y_k^{(i)} + \ldots + y_{N}^{(i)})$, $(y_n^{(i)} + \ldots + y_{N}^{(i)})$ such that $||y_k^{(i)}||_1 \leq L_B$ for $j \in \{1, \ldots, n\}, k \in \{1, \ldots, N_j\}$.

Thus, we have a decomposition of $R_i y^i$, which again follows by an argument by Eisenbrand et al. 3:

$$0 = R_i y^i = (R_i^{(1)}, \ldots, R_i^{(n)}) y^i = R_i^{(1)} y^{(1)} + \ldots + R_i^{(n)} y^{(n)} = R_i^{(1)} y^{(1)} + \ldots + R_i^{(n)} y^{(n)} + \ldots + R_i^{(n)} y^{(n)} = v_1 + \ldots + v_N \in \mathbb{Z}^{P_i},$$

for some $N = \sum_{i=1}^{P_i} N_j$ and $||v_k||_\infty \leq \Delta L_B$ for $k = 1, \ldots, N$ using inequality (4). Now we again apply Lemma 1 to reorder the $v_j$, such that each partial sum is bounded by $|P_i|\Delta L_B$ in the $\ell_\infty$-norm.

We can use the same argument as in Lemma 2 that, if $N > (2|P_i|\Delta L_B + 1)^{|P_i|}$, two or more partial sums of the sequence $v_1, \ldots, v_N$ would be the same, and we could decompose $y^i$, and therefore $y$, into at least two sign-compatible, non-zero cycles of $A$—a contradiction to the fact that $y$ is a Graver basis element of $A$ and consequently indecomposable. Thus, it holds that $N \leq (2|P_i|\Delta L_B + 1)^{|P_i|}$. As $y^i$ is the sum of at most $N$ vector of $\ell_\infty$-norm $L_B$,

$$||y^i||_1 \leq L_B(2|P_i|\Delta L_B + 1)^{|P_i|}.$$

6
Since $|P_i| \leq p_A$, for all $i = 1, \ldots, S_A$, it follows that

$$||y||_1 \leq \sum_{i=1}^{S_A} ||y^i||_1 \leq S_A L_B (2p_A \Delta L_B + 1)^{p_A},$$

proving the lemma.

Note that, from [Lemma 4] it follows that

$$||y||_1 \leq S_A L_B (2p_A \Delta L_B + 1)^{p_A}$$
$$\leq S_A (2p_B \Delta + 1)^{p_B} (2p_A \Delta (2p_B \Delta + 1)^{p_B} + 1)^{p_B}$$
$$\in S_A (p_{APB}\Delta)^{O(p_{APB})}.$$

We therefore have a new, improved upper bound $g'(A) \in S_A (p_{APB}\Delta)^{O(p_{APB})}$ on the Graver basis of any $N$-fold matrix $A$. We can use this bound in the algorithm from [Proposition 1] to solve the halving augmentation procedure in time $nt \log(nt) L(S_A)^{O(r+\epsilon)} (p_{APB}\Delta)^{O(rp_{APB} + sp_{APB})}$. Together with the results on constructing an initial feasible solution in at most the same time by Eisenbrand et al. [6, Corollary 55, Corollary 89], this proves [Theorem 1].

5 Applications

5.1 High-Multiplicity Scheduling on Uniformly Related Machines

A well-studied problem in high-multiplicity scheduling is the scheduling of jobs on uniformly related machines so as to minimize or maximize some objective function. Formally, we are given $m$ machines with integer speeds $s_1 \leq \ldots \leq s_m$, on which we have to schedule jobs of $d$ distinct types. All jobs of the same type $j$ have the same processing time $p_j$, and there are $n_j$ jobs of type $j$. This problem is denoted as $Q||C_{\text{max}}$ in the classical Graham-notation [9]. We will keep consistent with the Graham-notation for all upcoming scheduling problems. We denote the largest processing time in an instance as $p_{\text{max}}$. It takes $p_j/s_i$ time units to process a job of type $j$ on machine $i$. The objective is to find a schedule that minimizes the makespan $C_{\text{max}}$, which is the latest completion time of any of the machines. A well-known special case of this is $P||C_{\text{max}}$, where one has parallel (identical) machines, which all have the same speed $s_1 = \ldots = s_m = 1$.

The decision variant of $Q||C_{\text{max}}$ asks whether there exists a schedule with makespan at most $T$ for a given integer $T$. The minimum makespan is found by iteratively solving the decision variant for increasing values of $T$. The decision variant is captured by the following $N$-fold IP, which was first proposed by Knop and Koutecký [13, Section 4.1]:

$$\sum_{i=1}^{m} x_j^i = n_j, \quad \text{for } j = 1, \ldots, d,$$
$$\sum_{j=1}^{d} p_j x_j^i \leq s_i T, \quad \text{for } i = 1, \ldots, m,$$

where $x_j^i$ are the decision variables denoting the number of jobs of type $j$ scheduled on machine $i$. Knop and Koutecký also only use $p_{\text{max}}$ as the parameter, and bound the number of job types $d$ by $p_{\text{max}}$. We instead use $d$ as an additional parameter, which can be much smaller than $p_{\text{max}}$.

We now want to bound $L$ for scheduling problems of this form. As we are looking at the decision variant, we just have to compute a feasible solution for the IP to be done. As Eisenbrand et al. have shown [3, Theorem 7], this can be done by introducing an IP slack variables and a objective function $c = (1, \ldots, 1)$ that can be solved in the same time as the original IP. We can then bound $c_{\text{max}}$ by $n$, as each $c$ is the 1-vector and each entry of $x$ can be at most $n$, as at most all jobs can be assigned to one machine. We can further bound $||u - \ell||_{\infty}$ by $n$, as each entry of $\ell$ is greater or equal 0, since no negative number of jobs can be assigned to a machine, and each entry of $u$ is smaller or equal than $n$, again as at most all jobs can be assigned to one machine. Therefore, $L \leq \log^2(n)$. This is true for all considered scheduling problems with the makespan as their objective.
A well-known variation is to schedule the jobs in order to maximize the minimum machine load, denoted as \(Q|C_{\text{min}}\). The problem has received much attention in previous literature, see \cite{1,2,3}. To our knowledge, there has not been made any advances to propose a fixed-parameter algorithm. We propose a procedure to solve an instance of \(Q|C_{\text{min}}\) to again solve an \(N\)-fold formulation of the decision variant:

\[
\sum_{i=1}^{m} x_{ij}^i = n_j, \quad \text{for } j = 1, \ldots, d, \\
\sum_{j=1}^{d} p_j x_{ij}^j \geq s_i T, \quad \text{for } i = 1, \ldots, m.
\]

**Corollary 1.** \(Q|C_{\text{max}}\) and \(Q|C_{\text{min}}\) can be solved in time \(p_{\text{max}}^{O(d)} n \log(m) \log^2(n)\), where \(m\) is the number of machines, \(p_{\text{max}}\) the largest processing time of any job, and \(d\) the number of job types in the input.

**Proof.** Note that the parameters characterizing the \(N\)-fold IP are equal for both objective functions. In the formulation, the submatrices \(A\) are identity matrices in \(\mathbb{Z}^{dxd}\). Consequently, consider a column-independent partition where each partition consists of a single row. Furthermore, every \(B^{(i)}\) matrix is a single row. This leads to the parameters summarized in Table 2. Using Theorem 1, we obtain the claimed run time.

This also improves the result by Mnich and Wiese \cite{24}, by which \(P|C_{\text{min}}\) can be solved in time \(2^{O(p_{\text{max}})} n^{O(1)}\).

### 5.1.1 Sum of weighted completion times

Another objective function that was focus of previous publications is the weighted sum of completion times, e.g. in Knop et al. \cite{17}. Additionally to processing times \(p_j\), we are given a weight \(w_j\) for each job type \(j\), and aim to minimize the sum over all completion times \(C_j\) for each job \(j\), multiplied by its weight \(w_j\). We denote this objective function as \(\sum w_j C_j\).

Knop and Koutecký formulate the following result. Let \(\rho^i(j) = w_j/p_j\) for job \(j\) on machine \(i\).

**Proposition 3** (Knop and Koutecký \cite{17} Corollary 1). Given integers \(x_1^i, \ldots, x_d^i\) representing numbers of jobs of each type scheduled to run on machine \(i\) and a permutation \(\pi^i : \{1, \ldots, d\} \to \{1, \ldots, d\}\) such that \(\rho^i(\pi^i(j)) \geq \rho^i(\pi^i(j+1))\) for all \(j = 1, \ldots, d-1\), the optimal schedule has value

\[
f^i(x^i, z^i) := \sum_{j=1}^{d} \left(\frac{1}{2}(z_{\pi^i(j)}^i)^2 (\rho^i(\pi^i(j)) - \rho^i(\pi^i(j+1))) + \frac{1}{2} x_{\pi^i(j)}^i p_{\pi^i(j)}^i w_{\pi^i(j)}\right),
\]

where \(z_{\pi^i(j)}^i = \sum_{\ell=1}^{j} p_{\pi^i(\ell)}^i x_{\pi^i(\ell)}^i\).

The objective function for an \(N\)-fold formulation for minimizing the weighted sum of completion times takes the form \(f(x, z) = \sum_{i=1}^{m} f^i(x^i, z^i)\). Note that \(f(x, z)\) is separable convex \cite{17}; recall that a function \(f : \mathbb{R}^n \to \mathbb{R}\) is separable convex if it can be written as \(f(x) = \sum_{i=1}^{n} f_i(x_i)\) for convex functions \(f_1, \ldots, f_n\). As Proposition 1 also holds for separable convex functions, with the same run time \cite{6, Theorem 77}, we can also use Theorem 1 for solving \(N\)-fold IPs with separable convex objective functions. We can therefore use \(f(x, z)\) as the objective function, and the same constraints as before. For upper bounding \(L\), we can use again the same bound on \(\|u - \ell\|_{\infty}\) as for the makespan objective. The
value $f_{\text{max}}$ we can upper bounded by $n^2p_{\text{max}}$, as this is an upper bound on the sum of completion times of a schedule where all jobs are scheduled on one machine, and all have processing time $p_{\text{max}}$. Thus, $L \in O(\log^2(n))$, and we obtain the following corollary:

**Corollary 2.** $Q||\sum w_jC_j$ can be solved in time $(p_{\text{max}})^C(m \log(m)) \log^2(n)$, where $m$ is the number of machines, $p_{\text{max}}$ the largest processing time of any job, and $d$ the number of job types in the input.

### 5.1.2 Problem Generalizations

The $N$-fold method does not only provide a faster way of solving basic high-multiplicity scheduling problems, but is flexible enough to solve various generalizations. Real-life applications may motivate additional constraints and restrictions that may further be imposed on scheduling problems. This can entail individual and job-dependent capacity constraints on the machines, ordering of jobs and release time or deadlines on jobs. The presented $N$-fold IPs for high-multiplicity scheduling can be extended to such generalizations by including related constraints in the IP. We now present some of these generalizations with additional constraints.

Consider the high-multiplicity scheduling problem on parallel machines $P||C_{\text{max}}$ with capacity constraints as discussed in Zhang et al. [26]. We augment this problem with individual machine speeds and define following problem: Additionally to an instance of $Q||C_{\text{max}}$, we are given a capacity constraint $c_i$ for each machine $i$. A feasible solution must satisfy that on each machine $i$ there are at most $c_i$ jobs scheduled. We denote this variant as $Q|\text{cc}|C_{\text{max}}$. The $N$-fold IP of $Q||C_{\text{max}}$ can then be enhanced with the following local capacity constraint

$$\sum_{j=1}^{d} x_{ij} \leq c_i, \quad \text{for } i = 1, \ldots, m.$$  

This results in only one additional constraint per block in the $B^{(i)}$ matrices, and therefore only parameter $s$ is incremented, also only by one. This directly leads to the following corollary.

**Corollary 3.** $Q|\text{cc}|C_{\text{max}}$ can be solved in time $(p_{\text{max}})^C(m \log(m)) \log^2(n)$, where $m$ is the number of machines, $p_{\text{max}}$ the largest processing time of any job, and $d$ the number of job types in the input.

In the problem $Q|r_j|C_{\text{max}}$ the input is augmented by release times for each job type. Formally, we are given release times $r_j$ for each job type $j = 1, \ldots, d$. A job of type $j$ is not allowed to be scheduled before its release time $r_j$ on any machine. For the $N$-fold IP formulation, let the job types $j = 1, \ldots, d$ be ordered in non-decreasing order of $r_j$. If there exists an optimal schedule with makespan at most $T$ that respects the release time of all jobs, then there also exists such a schedule where the jobs are scheduled in non-decreasing order of $r_j$ on every machine. As in $Q||C_{\text{max}}$, we introduce decision variables $x_{ij}$, where $x_{ij}$ denotes the number of jobs of type $j$ scheduled on machine $i$. Additionally, we introduce variables $s_{ij}$, where $s_{ij}$ represents the starting time of the first job of type $j$ on machine $i$. Consider the following $N$-fold IP formulation of $Q|r_j|C_{\text{max}}$:

$$\sum_{i=1}^{m} x_{ij} = n_j, \quad \text{for } j = 1, \ldots, d, \quad \text{(5)}$$

$$s_{ij} - (s_{ij-1} + p_{(j-1)x_{ij-1}}) \geq 0, \quad \text{for } j = 1, \ldots, d, i = 1, \ldots, m, \quad \text{(6)}$$

$$s_{ij} + p_{\text{max}}x_{ij} \leq s_i T, \quad \text{for } i = 1, \ldots, m, \quad \text{(7)}$$

$$s_{ij} \geq r_j, \quad \text{for } j = 1, \ldots, d, i = 1, \ldots, m. \quad \text{(8)}$$

Constraints (5) ensure that each job is processed, and constraints (6) ensure that each starting time is later than its release time. The order of jobs in each machine is captured by constraints (8), where it is also made sure that jobs of a certain type are not scheduled before the previously scheduled jobs have finished. Constraints (7) then ensure that the makespan of each machine is lesser or equal than $T$.

---

2The authors call this problem **Scheduling Machines with Capacity Constraints** in their work.
We can formulate the related problem $Q|d_j|C_{\text{max}}$ in similar fashion. Here, instead of release times, we are given deadlines $d_j$ for each job type $j = 1, \ldots, d$. A job $j$ is not allowed to be scheduled after its deadline on any machine. Let the job types $j = 1, \ldots, d$ be ordered in non-decreasing order of $d_j$. If there exists an optimal schedule with makespan at most $T$ that respects the deadlines of all jobs, then there also exists such a schedule where the jobs are scheduled in non-decreasing order of $d_j$ on every machine. The following $N$-fold formulation of $Q|d_j|C_{\text{max}}$ uses the same decision variables $x_{ij}$ and $s_{ij}$ as $Q|r_j|C_{\text{max}}$. To formulate the problem we use constraints (5) to (7). Additionally, we add the following constraints that ensure that on each machine the jobs are finished before their deadline:

$$s_{ij} + p_j x_{ij} \leq d_j, \quad \text{for } j = 1, \ldots, d \text{ and } i = 1, \ldots, m .$$  

Note that proposed programs are subject to $N$-fold structure as the constraints (5) to (7) with additionally either (8) or (9) are locally uniform for each machine $i$. We have $N$-fold parameters $r = d, s = 2d + 1, t = 2d, \Delta = p_{\text{max}}$ and $n = m$ for both $Q|r_j|C_{\text{max}}$ and $Q|d_j|C_{\text{max}}$. As the $A$-matrices are identity matrices with $d$ additional zero entries each (coefficients of the $s_{ij}$ variables), we can partition them similarly as in Corollary 1. This implies the following corollary.

**Corollary 4.** $Q|r_j|C_{\text{max}}$ and $Q|d_j|C_{\text{max}}$ can be solved in time $(p_{\text{max}})^{O(d^2)} m \log(m) \log^2(n)$, where $m$ is the number of machines, $p_{\text{max}}$ the largest processing time of any job, and $d$ the number of job types in the input.

### 5.2 High-Multiplicity Scheduling on Unrelated Machines

We now consider makespan minimization on unrelated machines, denoted as $R||C_{\text{max}}$. We are given $m$ machines of $K$ different kinds, and again $d$ different job types. Each job type $j$ has $n_j$ jobs. Each of the $K$ machine kinds has a vector of size $d$ of processing times, each entry corresponding to one job type. Knop and Koutecký [17] formulate the problem using a more restricted formulation, namely a uniform $N$-fold formulation. As Theorem 1 can be used to solve general $N$-fold formulations, we instead present a slightly different IP formulation. Furthermore, Knop and Koutecký only use parameters $p_{\text{max}}$ and $K$, and bound the number of job types $d$ by $p_{\text{max}}$. Again, we use $d$ as an additional parameter.

Let a machine $i$ be of kind $k \in K$. Let $p^i_j$ denote the processing time of a job of type $j$ on this machine. To allow restrictions on compatibility of machines and jobs, we introduce an additional processing time representing infinity and therefore there are at most $(d + 1)^K$ many different processing times [17]. We introduce decision variables $x^i_1, \ldots, x^i{(d+1)K}$, where $x^i_j$ denotes the number of jobs of type $j$ scheduled on machine $i$. Problem $R||C_{\text{max}}$ can then be formulated as the following $N$-fold IP:

$$\begin{align*}
\sum_{i=1}^{m} x^i_j &= n_j, \quad \text{for } j = 1, \ldots, (d + 1)^K, \\
\sum_{j=1}^{(d+1)K} a^i_j x^i_j &\leq T, \quad \text{where } a^i_j = p^i_j, \text{ if } p^i_j < \infty, \text{ and } a^i_j = 0 \text{ else, for } i = 1, \ldots, m, \\
\sum_{j=1}^{(d+1)K} b^i_j x^i_j &= 0, \quad \text{where } b^i_j = 1, \text{ if } p^i_j = \infty, \text{ and } b^i_j = 0 \text{ else, for } i = 1, \ldots, m .
\end{align*}$$

Using Proposition 2, this $N$-fold IP can be solved in $(p_{\text{max}})^{O(K(d^2)^K)} m \log(m) \log^2(n)$ time, which is already an improvement on the run time given by Knop and Koutecký [17] for this problem. Using Theorem 1 we can improve the run time for solving $R||C_{\text{max}}$ even further.

Again observing that the submatrices $A$ are identity matrices, the same reasoning as in the proof of Corollary 1 applies. The parameters are summarized in Table 3.

We thus obtain the following result.

**Corollary 5.** $R||C_{\text{max}}$ can be solved in time $(p_{\text{max}})^{O(Kd^2)} m \log(m) \log^2(n)$, where $m$ is the number of machines, $p_{\text{max}}$ is the size of the largest job, $d$ the number of job types, and $K$ is the number of kinds of machines in the input.
This is also an improvement over the run time obtained for solving $R||C_{max}$ with the well-known result for general scheduling problems by Goemans and Rothvoss [3, Corollary 6.6]: Using this result, $R||C_{max}$ can be solved in time $\log^2(n)$.

5.3 Minimum Sum Coloring

To show that our results cannot only be applied to scheduling, but to a wider range of problems, we now give improvements on a non scheduling related problem. The MINIMUM SUM COLORING problem is a generalization of the well-known GRAPH COLORING problem. It takes as input a graph $G$ and seeks a proper vertex coloring $c : V(G) \rightarrow N$ which minimizes $\sum_{v \in V(G)} c(v)$.

An $N$-fold formulation of the MINIMUM SUM COLORING problem is provided by Gavenčiak et al. [2, Model 10, Section 5.1]. For each vertex $v \in V(G)$, let its neighborhood $N_G(u)$ in $G$ be the set of vertices adjacent to $u$ in $G$. Two vertices $u, v$ are called twins if $N_G(u) \setminus \{v\} = N_G(v) \setminus \{u\}$. The twin equivalence is the relation on vertices of a graph where two vertices are equivalent if and only if they are twins. An equivalence class is always either an independent set (there does not exist an edge between any pair of vertices in this set) or a clique (there exists an edge between every pair of vertices in this set) in $G$. The neighborhood diversity of a graph $G$, denoted by $nd(G)$, is the number $k$ of classes (called types) of the twin equivalence of $G$.

We denote by $V_i$ the classes of twin equivalence on $G$ for $i = 1, \ldots, k$. The type graph $T(G)$ of a graph $G$ is a graph on $k = nd(G)$ vertices, where each $i$ is assigned weight $|V_i|$, and where $\{i, j\}$ is an edge or a loop in $T(G)$ if two distinct vertices of $V_i$ and $V_j$ are adjacent. Any graph $G$ with $nd(G) = k$ can be described in a compressed way using only $O((|V(G)|)^2)$ space by its type graph, which is computable in linear time.

We can now restate the $N$-fold formulation by Gavenčiak et al. [2, Model 10, Section 5.1]:

$$\min \sum_{\alpha=1}^{\frac{|V(G)|}{k}} \left( \sum_{i \in T(G)} \alpha x_i^\alpha \right) + \left( \sum_{i \in T(G)} \alpha |V_i| x_i^\alpha \right),$$

$$\sum_{\alpha=1}^{\frac{|V(G)|}{k}} x_i^\alpha = |V_i|, \quad \forall i \in T(G), G[V_i] \text{ is clique},$$

$$\sum_{\alpha=1}^{\frac{|V(G)|}{k}} x_i^\alpha = 1, \quad \forall i \in T(G), G[V_i] \text{ is independent},$$

$$x_i^\alpha + x_j^\alpha \leq 1, \quad \forall \alpha = 1, \ldots, |V(G)|, \{i, j\} \in E(T(G)) .$$

Note that $L \in \mathcal{O}(\log^2(|V(G)|))$, as each entry of $u$ is bounded by $|V(G)|$ (each color can at most all vertices), and $c_{max}$ is bounded by $|V(G)|^2$ ($x \leq u$ and each color has at most weight $|V(G)|$). The parameters of this formulation are presented in Table 4. Gavenčiak et al. solved this $N$-fold IP in $kO(k^3)|V(G)|^2 \log^2(|V(G)|)$ time [2, Theorem 2 (a)].

We can improve the run-time dependence on $k$ using our partitioning approach, as stated in the following corollary.

**Corollary 6.** MINIMUM SUM COLORING can be solved in time $kO(k^3)|V(G)| \log^3(|V(G)|)$ on $n$-vertex graphs $G$ of neighborhood diversity $k$.

---

In their paper, they state the run time as $kO(k^3)|V(G)|^2 \log^2(|V(G)|)$. This is an error as confirmed by private communication with the authors.
**Proof.** As is also observed by the authors of the \( N \)-fold formulation of the Minimum Sum Coloring problem \cite{Zin01}, the \( A^{(i)} \)-matrices in this formulation are \( k \times k \) identity matrices for all blocks. We can therefore partition these, as seen in Corollary \ref{cor:partition}, into \( k \times k \) sets of size 1 each, giving us \( S_A = k \), and \( p_A = 1 \). We cannot partition the \( B \) matrices any further than into one set of size \( k^2 \) each, giving us \( p_B = k^2 \). Plugging these values, together with the other \( N \)-fold parameter values from Table \ref{table:parameters} into the algorithm from Theorem \ref{thm:main} we obtain the claimed run time. \hfill \( \square \)

### 6 Lower Bounds and Hardness

In this section, we give lower bounds on the run time for the aforementioned problems, to argue about the tightness of our results. Recall that, for a problem instance \( I \), its size in bits is denoted as \(| I |\).

We begin with problem \( Q||C_{\text{max}} \), for which we show that the dependence on \( p_{\text{max}} \) in the run time \( (p_{\text{max}})^{O(d)m}\log(n)^{\log^{O}(n)} \) of Corollary \ref{cor:partition} cannot be significantly reduced. To this end, consider the following proposition:

**Proposition 4** (Koutecký and Zink \cite{Kou10}, Theorem 1). \( Q||C_{\text{max}} \) is NP-hard already for \( 6 \) job types.

Using Proposition \ref{prop:hardness}, we can then show the following lemma.

**Lemma 5.** There is no algorithm for \( Q||C_{\text{max}} \) with run time \( \log^{c}(p_{\text{max}})^{d}|I|^{O(1)} \), for any \( c \in \mathbb{N} \), unless \( P = \text{NP} \).

**Proof.** Proposition \ref{prop:hardness} implies that, if we can solve any instance of \( Q||C_{\text{max}} \) with \( d = 6 \) in time \( |I|^{O(1)} \), \( P = \text{NP} \) follows.

Now assume we can solve \( Q||C_{\text{max}} \) in time \( \log^{c}(p_{\text{max}})^{d}|I|^{O(1)} \) for some \( c \in \mathbb{N} \), meaning in time \( (\log^{c}(p_{\text{max}}))^d|I|^{O(1)} \) as \( d = 6 \). As \( p_{\text{max}} \) is part of the input, it follows that \( p_{\text{max}} \leq 2^{|I|} \). We then obtain a run time of \( (\log^{c}(2^{|I|}))^d|I|^{O(1)} = |I|^{\log^{O}(2^{|I|}^d)} \in |I|^{O(1)} \), proving the lemma. \hfill \( \square \)

Consider the Exponential Time Hypothesis (ETH) as stated by Impagliazzo et al. \cite{Imp99}: There is a positive real \( \delta \) such that 3-SAT with \( n \) variables and \( m \) clauses cannot be solved in time \( 2^{\delta n}(n + m)^{O(1)} \).

We use this hypothesis to show that we cannot expect any significant improvement on the dependence on \( d \), either. Again, first consider a proposition based on the ETH:

**Proposition 5** (Chen et al. \cite{Chen10}, Theorem 2). For any \( \delta > 0 \), there is no \( 2^{O(n^{\delta - 1})} \) time algorithm for \( P||C_{\text{max}} \) with \( n \) jobs even if the processing time of each job is bounded by \( \text{O}(n) \), unless ETH fails.

Using Proposition \ref{prop:eth}, we can show the following lemma.

**Lemma 6.** There is no algorithm that solves \( Q||C_{\text{max}} \) in time \( (p_{\text{max}})^{\log^{c}(d)}|I|^{O(1)} \), for any \( c \in \mathbb{N} \), unless ETH fails.

**Proof.** In Proposition \ref{prop:eth}, the authors use \( P||C_{\text{max}} \) to talk about the scheduling problem on parallel machines in the non high-multiplicity setting. Since every instance in the non high-multiplicity setting can be translated into an instance in the high-multiplicity setting in polynomial time with a smaller or equal encoding size, the proposition immediately follows for our high-multiplicity definition of \( P||C_{\text{max}} \).

Consider now such an instance of \( P||C_{\text{max}} \), where all processing times are bounded by \( \text{O}(n) \); in particular, \( p_{\text{max}} \in \text{O}(n) \). Assume that there is an algorithm that solves such instances of \( P||C_{\text{max}} \) in time \( 2^{O(p_{\text{max}})}|I|^{O(1)} \) time, for some \( \delta > 0 \). Since \( p_{\text{max}} \in \text{O}(n) \), that algorithm would solve \( P||C_{\text{max}} \) in time \( 2^{O(n^{\delta - 1})}|I|^{O(1)} \), directly contradicting Proposition \ref{prop:eth}. As \( Q||C_{\text{max}} \) is more general than \( P||C_{\text{max}} \), this lower bound on the run time under ETH immediately follows for such instances of \( Q||C_{\text{max}} \).
Assume now that there exists an algorithm which can solve such instances of $Q||C_{\text{max}}$ in time $(p_{\text{max}})^{\log^c(d)}|I|^{O(1)}$ for some $c \in \mathbb{N}$. As $d \leq p_{\text{max}}$, that algorithm would solve these instances in time $2^{O((\log^{c+1}(p_{\text{max}}))|I|^{O(1)})}$, contradicting the previous statement that there exists no algorithm with run time $2^{O(p_{\text{max}}^{c+1})}|I|^{O(1)}$. Thus, the lemma follows.

We now also give a lower bound on the run time of algorithms solving MINIMUM SUM COLORING, assuming ETH. It is well-known that the 3-COLORING problem on $n$-vertex graphs cannot be solved in $2^{o(n)}$ time [22, Theorem 3.1], unless ETH fails. As MINIMUM SUM COLORING is a generalization of 3-COLORING, and $k = nd(G) \leq n$ for all $n$-vertex graphs, one cannot solve MINIMUM SUM COLORING in $2^{o(k)}$ time, unless ETH fails.

7 Conclusions

By introducing a new, improved upper bound $\ell_1$-norm on the Graver basis elements of general and $N$-fold matrices based on the newly introduced partition parameters, we are able to give algorithms with a significant faster run time for well-known $N$-fold IPs, such as those appearing in high-multiplicity scheduling.

In this paper, we mainly focused on scheduling problems, and proved for a number of them that their $N$-fold formulation has small partition parameters. It would be interesting to know if there exist more such classes of problems with small partition parameters, for which the run time then could be improved by our algorithm.

It would also be interesting to know whether the run times for $P||C_{\text{max}}, Q||C_{\text{max}}$ and $Q||C_{\text{min}}$ can be further improved, when measured in terms of $p_{\text{max}}$ and $d$. A major open question is whether one can completely remove the dependence on $p_{\text{max}}$ for solving $P||C_{\text{max}}$ and obtain fixed-parameter algorithms which only depend exponentially on $d$ in the lavish instance encoding; such an algorithm would generalize the fundamental works by Jansen and Klein [14] and Goemans and Rothvoss [8].

Acknowledgements. We thank Matthias Kaul for fruitful discussions and ideas for improvements on the dependence of the Graver basis $\ell_1$-norm on the partition parameters, and thank Tobias Stamm for suggesting the $N$-fold formulation of $Q||C_{\text{min}}$. We also thank Sebastian Berndt for valuable feedback.

References

[1] Y. Azar and L. Epstein. Approximation schemes for covering and scheduling in related machines. In Proc. APPROX 1998, pages 39–47, 1998.
[2] D. Chakrabarty, R. Krishnaswamy, and A. Kumar. The heterogeneous capacitated k-center problem. In Proc. IPCO 2017, volume 10328 of Lecture Notes Comput. Sci., pages 123–135, 2017.
[3] L. Chen, K. Jansen, and G. Zhang. On the optimality of exact and approximation algorithms for scheduling problems. J. Comput. System Sci., 96:1–32, 2018.
[4] J. Cslovjecsek, F. Eisenbrand, C. Hunkenschröder, L. Rohwedder, and R. Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In Proc. SODA 2021, pages 1666–1681, 2021.
[5] F. Eisenbrand, C. Hunkenschröder, and K.-M. Klein. Faster algorithms for integer programs with block structure. In Proc. ICALP 2018, volume 107 of Leibniz Int. Proc. Informatics, pages Art. No. 49, 13. 2018.
[6] F. Eisenbrand, C. Hunkenschröder, K.-M. Klein, M. Koutecký, A. Levin, and S. Onn. An algorithmic theory of integer programming, 2019. https://arxiv.org/abs/1904.01361
[7] T. Gavenčiak, M. Koutecký, and D. Knop. Integer programming in parameterized complexity: Five miniatures. Discrete Optim., page 100596, 2020.
[8] M. X. Goemans and T. Rothvoss. Polynomiality for bin packing with a constant number of item types. *J. ACM*, 67(6), 2020.

[9] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey.

[10] J. E. Graver. On the foundations of linear and integer linear programming. I. *Math. Programming*, 9(2):207–226, 1975.

[11] R. Hemmecke, S. Onn, and L. Romanchuk. n-fold integer programming in cubic time. *Math. Program.*, 137(1-2, Ser. A):325–341, 2013.

[12] D. S. Hochbaum and R. Shamir. Strongly polynomial algorithms for the high multiplicity scheduling problem. *Oper. Res.*, 39(4):648–653, 1991.

[13] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *J. Comput. System Sci.*, volume 63, pages 512–530, 2001.

[14] K. Jansen and K.-M. Klein. About the structure of the integer cone and its application to bin packing. *Math. Oper. Res.*, 45, 2020.

[15] K. Jansen and M. Maack. An EPTAS for scheduling on unrelated machines of few different types. *Algorithmica*, 81(10):4134–4164, 2019.

[16] K. Jansen, A. Lassota, and L. Rohwedder. Near-linear time algorithm for n-fold ILPs via color coding. In *Proc. ICALP 2019*, volume 132 of *Leibniz Int. Proc. Informatics*, pages Art. No. 75, 13. 2019.

[17] D. Knop and M. Koutecký. Scheduling meets n-fold integer programming. *J. Sched.*, 21(5):493–503, 2018.

[18] D. Knop, M. Koutecký, A. Levin, M. Mnich, and S. Onn. Multitype integer monoid optimization and applications, 2019. [https://arxiv.org/abs/1909.07326](https://arxiv.org/abs/1909.07326).

[19] D. Knop, M. Koutecký, and M. Mnich. Combinatorial n-fold integer programming and applications. *Math. Program.*, 184(1-2, Ser. A):1–34, 2020.

[20] M. Koutecký and J. Zink. Complexity of scheduling few types of jobs on related and unrelated machines. In *Proc. ISAAC 2020*, volume 181 of *Leibniz Int. Proc. Informatics*, pages Art. No. 18, 17. 2020.

[21] M. Koutecký, A. Levin, and S. Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *Proc. ICALP 2018*, volume 107 of *Leibniz Int. Proc. Informatics*, pages Art. No. 85, 14. 2018.

[22] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (105):41–71, 2011.

[23] M. Mnich and R. van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Comput. Oper. Res.*, 100:254–261, 2018.

[24] M. Mnich and A. Wiese. Scheduling and fixed-parameter tractability. *Math. Program.*, 154(1-2, Ser. B):533–562, 2015.

[25] E. Steinitz. Bedingt konvergente Reihen und konvexe Systeme. *J. Reine Angew. Math.*, 143:128–176, 1913.

[26] C. Zhang, G. Wang, X. Liu, and J. Liu. Approximating scheduling machines with capacity constraints. In *Proc. FAW 2009*, volume 5598 of *Lecture Notes Comput. Sci.*, pages 283–292, 2009.