Scheduling Kernels via Configuration LP

Dušan Knop
Department of Theoretical Computer Science, Faculty of Information Technology,
Czech Technical University in Prague, Czech Republic

Martin Koutecký
Computer Science Institute of Charles University, Prague, Czech Republic

Abstract

Makespan minimization (on parallel identical or unrelated machines) is arguably the most natural and studied scheduling problem. A common approach in practical algorithm design is to reduce the size of a given instance by a fast preprocessing step while being able to recover key information even after this reduction. This notion is formally studied as kernelization (or simply, kernel) – a polynomial time procedure which yields an equivalent instance whose size is bounded in terms of some given parameter. It follows from known results that makespan minimization parameterized by the longest job processing time $p_{\max}$ has a kernelization yielding a reduced instance whose size is exponential in $p_{\max}$. Can this be reduced to polynomial in $p_{\max}$?

We answer this affirmatively not only for makespan minimization, but also for the (more complicated) objective of minimizing the weighted sum of completion times, also in the setting of unrelated machines when the number of machine kinds is a parameter.

Our algorithm first solves the Configuration LP and based on its solution constructs a solution of an intermediate problem, called huge N-fold integer programming. This solution is further reduced in size by a series of steps, until its encoding length is polynomial in the parameters. Then, we show that huge N-fold IP is in NP, which implies that there is a polynomial reduction back to our scheduling problem, yielding a kernel.

Our technique is highly novel in the context of kernelization, and our structural theorem about the Configuration LP is of independent interest. Moreover, we show a polynomial kernel for huge N-fold IP conditional on whether the so-called separation subproblem can be solved in polynomial time. Considering that integer programming does not admit polynomial kernels except for quite restricted cases, our "conditional kernel" provides new insight.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Integer programming

Keywords and phrases Scheduling, Kernelization

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.73

Related Version *Full Version*: https://arxiv.org/abs/2003.02187

Funding *Dušan Knop*: Supported by the OP VVV MEYS funded project "Research Center for Informatics" (nr. CZ.02.1.01/0.0/0.0/16_019/0000765).
Martin Koutecký: Partially supported by Charles University project UNCE/SCI/004, by the Czech Science Foundation (GA ČR) project 19-27871X, and by the Israel Science Foundation grant 308/18.

Acknowledgements Authors wish to thank the Lorentz Center for making it possible to organize the workshop Scheduling Meets Fixed-Parameter Tractability, which output resulted in this paper. The contribution of the Lorentz Center in stimulating suggestions, giving feedback and taking care of all practicalities, helped us to focus on our research and to organize a meeting of high scientific quality. Furthermore, the authors thank Stefan Kratsch for bringing Proposition 5 to their attention.

1 Introduction

Kernelization, data reduction, or preprocessing: all of these refer to the goal of simplifying and reducing (the size of) the input in order to speed up computation of challenging tasks. Many heuristic techniques are applied in practice, however, we seek a theoretical understanding in the form of a procedure with guaranteed bounds on the sizes of the reduced data. We use the notion of kernelization from parameterized complexity (cf. [39, 9]), where along with an input instance I we get a positive integer k expressing the *parameter value*, which may be the size of the sought solution or some structural limitation of the input. A kernel is an algorithm running in time $\text{poly}(|I| + k)$ which returns a reduced instance I' of the same problem of size bounded in terms of k; we sometimes also refer to I' as the kernel.

It is well known [6] that a problem admits a kernel if and only if it has an algorithm running in time $f(k)\,\text{poly}(|I|)$ for some computable function f (i.e., if it is fixed-parameter tractable, or FPT, parameterized by k). The "catch" is that this kernel may be very large (exponential or worse) in terms of k, while for many problems, kernels of size polynomial in k are known. This raises a fundamental question for any FPT problem: does it have a polynomial kernel? Answering this question typically provides deep insights into a problem and the structure of its solutions.

Parameterized complexity has historically focused primarily on graph problems, but it has been increasingly branching out into other areas. Kernelization, as arguably the most important subfield of parameterized complexity (cf. a recent monograph [15]), follows suit. Scheduling is a fundamental area in combinatorial optimization, with results from parameterized complexity going back to 1995 [3]. Probably the most central problem in scheduling is makespan minimization on identical machines, denoted as $P||C_{\max}$, which we shall define soon. It took until the seminal paper of Goemans and Rothvoß [20] to get an FPT algorithm for $P||C_{\max}$ parameterized by the number of job types (hence also by the largest job). Yet, the existence of a polynomial kernel for $P||C_{\max}$ remained open, despite being raised by Mnich and Wiese in 2013 [38] and reiterated by van Bevern[1]. Here, we give an affirmative answer for this problem:

▶ **Corollary 1.** *There is a polynomial kernel for $P||C_{\max}$ when parameterized by the longest processing time $p_{\max}$.*

Let us now introduce and define the scheduling problems $P||C_{\max}$ and $P||\sum w_jC_j$. There are n *jobs* and m identical *machines*, and the goal is to find a schedule minimizing an objective. For each job $j \in [n]$, a *processing time* $p_j \in \mathbb{N}$ is given and a *weight* w_j are given; in the case of $P||C_{\max}$ the weights play no role and can be assumed to be all zero. A *schedule* is a mapping which to each job $j \in [n]$ assigns some machine $i \in [m]$ and a closed interval of length p_j, such that the intervals assigned to each machine do not overlap except for their endpoints. For each job $j \in [n]$, denote by C_j its *completion time*, which is the time when it finishes, i.e., the right end point of the interval assigned to j in the schedule. In the *makespan minimization* ($C_{\max}$) problem, the goal is to find a schedule minimizing the time when the last job finishes $C_{\max} = \max_{j\in[n]} C_j$, called the *makespan*. In the *minimization of sum of weighted completion times* ($\sum w_jC_j$), the goal is to minimize $\sum w_jC_j$. (In the rest of the paper we formally deal with *decision* versions of these problems, where the task is to decide whether there exists a schedule with objective value at most k. This is a necessary approach when speaking of kernels and complexity classes like NP and FPT.)

[1] The question was asked at the workshop "Scheduling & FPT" at the Lorentz Center, Leiden, in February 2019, as a part of the opening talk for the open problem session.

In fact, our techniques imply results stronger in three ways, where we handle:

1. the much more complicated $\sum w_j C_j$ objective function involving possibly large job weights,
2. the unrelated machines setting (denoted $R||C_{\max}$ and $R||\sum w_j C_j$), and
3. allowing the number of jobs and machines to be very large, known as the *high-multiplicity* setting.

For this, we need further notation to allow for different kinds of machines. For each machine $i \in [m]$ and job $j \in [n]$, a *processing time* $p_j^i \in \mathbb{N}$ is given. For a given scheduling instance, say that two jobs $j, j' \in [n]$ are of the same *type* if $p_j^i = p_{j'}^i$ for all $i \in [m]$ and $w_j = w_{j'}$, and say that two machines $i, i' \in [m]$ are of the same *kind* if $p_j^i = p_j^{i'}$ for all jobs $j \in [n]$. We denote by $\tau \in \mathbb{N}$ and $\kappa \in \mathbb{N}$ the number of job types and machine kinds, respectively, call this type of encoding the *high-multiplicity* encoding, and denote the corresponding problems $R|HM|C_{\max}$ and $R|HM|\sum w_j C_j$.

Our approach is indirect: taking an instance I of scheduling, we produce a small equivalent instance I' of a the so-called *huge N-fold integer programming* problem with a quadratic objective function (see more details below). This is known as *compression*, i.e., a polynomial time algorithm producing from I a small equivalent instance of a different problem:

▶ **Theorem 2.** *The problems $R|HM|C_{\max}$ and $R|HM|\sum w_j C_j$ parameterized by the number of job types τ, the longest processing time $p_{\max}$, and the number of machine kinds κ admit a polynomial compression to quadratic huge N-fold IP parameterized by the number of block types $\bar{\tau}$, the block dimension t, and the largest coefficient $\|E\|_\infty$.*

If we can then find a polynomial reduction from quadratic huge N-fold IP to our scheduling problems, we are finished. For this, it suffices to show NP membership, as we do in Lemma 13.

Configuration LP. Besides giving polynomial kernels for some of the most fundamental scheduling problems, we wish to highlight the technique behind this result, because it is quite unlike most techniques used in kernelization and is of independent interest. Our algorithm essentially works by solving the natural Configuration LP of $P||C_{\max}$ (and other problems), which can be done in polynomial time when $p_{\max}$ is polynomially bounded, and then using powerful structural insights to reduce the scheduling instance based on the Configuration LP solution. The Configuration LP is a fundamental tool in combinatorial optimization which goes back to the work of Gilmore and Gomory in 1961 [18]. It is known to provide high-quality results in practice, in fact, the "modified integer round-up property (MIRUP)" conjecture states that the natural Bin Packing Configuration LP always attains a value x such that the integer optimum is at most $\lfloor x \rfloor + 1$ [40]. The famous approximation algorithm of Karmarkar and Karp [28] for Bin Packing is based on rounding the Configuration LP, and many other results in approximation use the Configuration LP for their respective problems as the starting point.

In spite of this centrality and vast importance of the Configuration LP, there are only few structural results providing deeper insight. Perhaps the most notable is the work of Goemans and Rothvoß [20] and later Jansen and Klein [26] who show that there is a certain set of "fundamental configurations" such that in any integer optimum, all but few machines (bins, etc.) will use these fundamental configurations. Our result is based around a theorem which shows a similar yet orthogonal result and can be informally stated as follows:

▶ **Theorem 3** (informal; see the full version)**.** *There is an optimum of the Configuration IP where all but few configuration are those discovered by the Configuration LP, and the remaining configurations are not far from those discovered by the Configuration LP.*

We note that our result, unlike the ones mentioned above [20, 26], also applies to arbitrary separable convex functions. This has a fundamental reason: the idea behind both previous results is to shift weight from the inside of a polytope to its vertices without affecting the objective value, which only works for linear objectives.

Huge N-fold IP. Finally, we highlight that the engine behind our kernels, a conditional kernel for the so-called quadratic huge N-fold IP, is of independent interest. Integer programming is a central problem in combinatorial optimization. Its parameterized complexity has been recently intensely studied [11, 31, 33, 10, 7]. However, it turns out that integer programs cannot be kernelized in all but the most restricted cases [34, 35, 26]. We give a positive result about a class of block-structured succinctly encoded IPs with a quadratic objective function, so-called quadratic huge N-fold IPs, which was used to obtain many interesting FPT results [29, 31, 4, 17, 32, 5]. However, our result is conditional on having a polynomial algorithm for the so-called *separation subproblem* of the Configuration LP of the quadratic huge N-fold IP, so there is a price to pay for the generality of this fragment of IP. The separation subproblem is to optimize a certain objective function (which varies) over the set of configurations. In the cases considered here, we show that this corresponds to (somewhat involved) variations of the knapsack problem with polynomially bounded numbers; in other problems expressible as n-fold IP, the separation subproblem corresponds to a known hard problem. Informally, our result reads as follows:

▶ **Theorem 4** (informal; see the full version)**.** *If the separation subproblem can be solved in polynomial time, then quadratic huge N-fold IP has a polynomial kernel parameterized by the block dimensions, the number of block types, and the largest coefficient.*

Because huge N-fold IP is essentially equivalent to the Configuration IP, the kernel of Theorem 3 can be viewed as a validation of the industry common wisdom that column generation works really well. Another view is that methods from mathematical programming, so far underrepresented in kernelization, deserve more attention.

One aspect of the algorithm above is reducing the quadratic objective function. The standard approach, also used in kernelization of weighted problems [13, 8, 2, 19, 42, 41, 21] is to use a theorem of Frank and Tardos [16] which "kernelizes" a linear objective function if the dimension is a parameter. However, we deal with

1. a quadratic convex (non-linear) function,
2. over a space of large dimension.

We are able to overcome these obstacles by a series of steps which first "linearize" the objective, then "aggregate" variables of the same type, hence shrinking the dimension, then reduce the objective using the algorithm of Frank and Tardos, and then we carefully reverse this process (see the full version for more details). This result has applications beyond this work: for example, the currently fastest strongly FPT algorithm for $R||\sum w_j C_j$ (i.e., an algorithm whose number of arithmetic operations does not depend on the weights w_j) has dependence of $m^2 \operatorname{poly}\log(m)$ on the number of machines m; applying our new result instead of [11, Corollary 69] reduces this dependence to $m \operatorname{poly}\log(m)$.

Other Applications. Theorem 4 can be used to obtain kernels for other problems which can be modeled as huge N-fold IP. First, we may also optimize the ℓ_p norms of times when each machine finishes, a problem known as $R|HM|\ell_p$. Our results (Corollary 11) show that also in this setting the separation problem can be solved quickly. Second, the $P||C_{\max}$ problem is identical to Bin Packing (in their decision form), so our kernel also gives a kernel for

Bin Packing parameterized by the largest item size. Moreover, also the Bin Packing with Cardinality Constraints problem has a huge N-fold IP model [30, Lemma 54] for which Corollary 11 indicates that the separation subproblem can be solved quickly. Third, Knop et al. [30] give a huge N-fold IP model for the Surfing problem, in which many "surfers" make demands on few different "services" provided by few "servers", where each surfer may have different costs of getting a service from a server; one may think of internet streaming with different content types, providers, and pricing schemes for different customer types. The separation problem there is polynomially solvable for an interesting reason: its constraint matrix is totally unimodular because it is the incidence matrix of the complete bipartite graph. Thus, Theorem 4 gives polynomial kernels for all of the problems above with the given parameters.

Related Work – Scheduling. Let us finally review related results in the intersection of parameterized complexity and scheduling; for a more comprehensive survey of parameterized results in scheduling see, e.g., [37]. First, to the best of our knowledge, the first to study scheduling problems from the perspective of multivariate complexity were Bodlaender and Fellows [3]. Fellows and McCartin [14] study scheduling on single machine of unit length jobs with (many) different release times and due dates. Single machine scheduling where two agents compete to schedule their private jobs is investigated by Hermelin et al. [22]. There are few other result [43, 27, 24, 23] focused on identifying tractable scenarios for various scheduling paradigms (such as flow-shop scheduling or e.g. structural limitations of the job–machine assignment).

Paper Organization. We begin with preliminaries (Section 2) which, besides introducing necessary notation and definitions, also states several results which we use later in the proofs of our results. Section 3 discusses how our scheduling problems are modeled as huge N-fold IP (3.1) and how to solve the ConfLP quickly for those models (3.2 and 3.3). Finally, in the full version, we show our "conditional kernel" for quadratic huge N-fold IP, which first reduces all parts of the instance except for the objective function, and finally deals with the objective. In the full version, we conclude with a short section giving an interpretation of our algorithm for $P||C_{\max}$ and $R||C_{\max}$. A word of caution: it is at first tempting to think that most of the machinery of our algorithm is not necessary for our simplest considered problem $P||C_{\max}$; however, as we discuss in the full version in detail, while some minor simplifications are possible, the only truly avoidable step is the objective reduction which is needed for the $\sum w_j C_j$ objective.

2 Preliminaries

We consider zero to be a natural number, i.e., $0 \in \mathbb{N}$. We write vectors in boldface (e.g., $\boldsymbol{x}, \boldsymbol{y}$) and their entries in normal font (e.g., the i-th entry of a vector $\boldsymbol{x}$ is x_i). For positive integers $m \leq n$ we set $[m, n] := \{m, \ldots, n\}$ and $[n] := [1, n]$, and we extend this notation for vectors: for $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n$ with $\boldsymbol{l} \leq \boldsymbol{u}$, $[\boldsymbol{l}, \boldsymbol{u}] := \{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$ (where we compare component-wise). For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{z} = \max\{\boldsymbol{x}, \boldsymbol{y}\}$ is defined coordinate-wise, i.e., $z_i = \max x_i, y_i$ for all $i \in [n]$, and similarly for $\min\{\boldsymbol{x}, \boldsymbol{y}\}$.

If A is a matrix, $A_{i,j}$ denotes the j-th coordinate of the i-th row, $A_{i,\bullet}$ denotes the i-th row and $A_{\bullet,j}$ denotes the j-th column. We use $\log := \log_2$, i.e., all our logarithms are base 2. For an integer $a \in \mathbb{Z}$, we denote by $\langle a \rangle := 1 + \lceil \log(|a| + 1) \rceil$ the binary encoding length of a; we extend this notation to vectors, matrices, and tuples of these objects. For example,

$\langle A, \boldsymbol{b}\rangle = \langle A\rangle + \langle \boldsymbol{b}\rangle$, and $\langle A\rangle = \sum_{i,j}\langle A_{i,j}\rangle$.[2] For a function $f\colon \mathbb{Z}^n \to \mathbb{Z}$ and two vectors $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n$, we define $f_{\max}^{[\boldsymbol{l},\boldsymbol{u}]} := \max_{\boldsymbol{x}\in[\boldsymbol{l},\boldsymbol{u}]} |f(\boldsymbol{x})|$; if $[\boldsymbol{l}, \boldsymbol{u}]$ is clear from the context we omit it and write just $f_{\max}$.

▶ **Proposition 5** ([15, Theorem 1.6]). *Let $(Q, \kappa), (R, \lambda)$ be parameterized problems such that Q is NP-hard and R is in NP. If (Q, κ) admits a polynomial compression into (R, λ), then it admits a polynomial kernel.*

The above observation is useful when dealing with NP-hard problems. The proof simply follows by pipelining the assumed polynomial compression with a polynomial time (Karp) reduction from R to Q.

2.1 Scheduling Notation

Overloading the convention slightly, for each $i \in [\kappa]$ and $j \in [\tau]$, denote by p_j^i the processing time of a job of type j on a machine of kind i, by w_j the weight of a job of type j, by n_j the number of jobs of type j, by m_i the number of machines of kind i, and denote $\boldsymbol{n} = (n_1, \ldots, n_\tau)$, $\boldsymbol{m} = (m_1, \ldots, m_\kappa)$, $\boldsymbol{p} = \left(p_1^1, \ldots, p_\tau^1, p_1^2, \ldots, p_\tau^\kappa\right)$, $\boldsymbol{w} := (w_1, \ldots, w_\tau)$, $p_{\max} := \|\boldsymbol{p}\|_\infty$, and $w_{\max} := \|\boldsymbol{w}\|_\infty$. We denote the high multiplicity versions of the previously defined problems $R|HM|C_{\max}$ and $R|HM|\sum w_jC_j$.

For an instance I of $R||C_{\max}$ or $R||\sum w_jC_j$, we define its size as $\langle I\rangle := \sum_{i=1}^{\kappa}\sum_{j=1}^{\tau}\langle p_j^i, w_j\rangle$, whereas for an instance I of $P|HM|C_{\max}$ or $P|HM|\sum w_jC_j$ we define its size as $\langle I\rangle = \langle \boldsymbol{n}, \boldsymbol{m}, \boldsymbol{p}, \boldsymbol{w}\rangle$. Note that the difference in encoding actually leads to different problems: for example, an instance of $R|HM|C_{\max}$ with 2^k jobs with maximum processing time $p_{\max}$ can be encoded with $\mathcal{O}(k\tau\kappa \log p_{\max})$ bits while an equivalent instance of $R||C_{\max}$ needs $\Omega(2^k \log p_{\max})$ bits, which is exponentially more if $\tau, \kappa \in k^{\mathcal{O}(1)}$. The membership of high-multiplicity scheduling problems in NP was open for some time, because it is not obvious whether a compactly encoded instance also has an optimal solution with a compact encoding. This question was considered by Eisenbrand and Shmonin, and we shall use their result. For a set $X \subseteq \mathbb{Z}^d$ define the *integer cone of X*, denoted $\mathrm{cone}_{\mathbb{N}}(X)$, to be the set $\mathrm{cone}_{\mathbb{N}}(X) := \left\{\sum_{\boldsymbol{x}\in X}\lambda_{\boldsymbol{x}}\boldsymbol{x} \mid \lambda \in \mathbb{N}^X\right\}$, where $\mathbb{N}^X$ is the set of functions mapping $X \to \mathbb{N}$ viewed as vectors.

▶ **Proposition 6** (Eisenbrand and Shmonin [12, Theorem 2]). *Let $X \subseteq \mathbb{Z}^d$ be a finite set of integer vectors and let $\boldsymbol{b} \in \mathrm{cone}_{\mathbb{N}}(X)$. Then there exists a subset $\tilde{X} \subseteq X$ such that $\boldsymbol{b} \in \mathrm{cone}_{\mathbb{N}}(\tilde{X})$ and the following holds for the cardinality of $\tilde{X}$:*

1. *if all vectors of X are nonnegative, then $|\tilde{X}| \le \langle \boldsymbol{b}\rangle$,*
2. *if $M = \max_{\boldsymbol{x}\in X}\|\boldsymbol{x}\|_\infty$, then $|\tilde{X}| \le 2d(\log 4dM)$.*

One can use Proposition 6 to show that the decision version of $R|HM|C_{\max}$ and $R|HM|\sum w_jC_j$ have short certificates and thus belong to NP. We will later derive the same result as a corollary of the fact that both of these scheduling problems can be encoded as a certain form of integer programming, which we will show to have short certificates as well.

2.2 Conformal Order and Graver Basis

Let $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{Z}^n$ be two vectors. We say that $\boldsymbol{g}$ *is conformal to* $\boldsymbol{h}$ (we denote it $\boldsymbol{g} \sqsubseteq \boldsymbol{h}$) if both $g_i \cdot h_i \ge 0$ and $|g_i| \le |h_i|$ for all $i \in [n]$. In other words, $\boldsymbol{g} \sqsubseteq \boldsymbol{h}$ if they are in the same

[2] We note that our encoding of integers already contains the delimiter symbol.

orthant (the first condition holds) and $\boldsymbol{g}$ is component-wise smaller than $\boldsymbol{h}$. For a matrix A we define its *Graver basis* (A) to be the set of all $\sqsubseteq$-minimal vectors in $\mathrm{Ker}(A) \setminus \{\mathbf{0}\}$. We define $g_\infty(A) = \max\{\|\boldsymbol{g}\|_\infty \mid \boldsymbol{g} \in (A)\}$ and $g_1(A) = \max\{\|\boldsymbol{g}\|_1 \mid \boldsymbol{g} \in (A)\}$.

We say that two functions $f, g\colon \mathbb{Z}^d \to \mathbb{Z}$ are *equivalent* on a polyhedron $P \subseteq \mathbb{Z}^d$ if $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ if and only if $g(\boldsymbol{x}) \leq g(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y} \in P$. Note that if f and g are equivalent on P, then the sets of minimizers of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ over P coincide.

▶ **Proposition 7** (Frank and Tardos [16]). *Given a rational vector $\boldsymbol{w} \in \mathbb{Q}^d$ and an integer M, there is a polynomial algorithm which finds a $\tilde{\boldsymbol{w}} \in \mathbb{Z}^d$ such that the linear functions $\boldsymbol{w}\boldsymbol{x}$ and $\tilde{\boldsymbol{w}}\boldsymbol{x}$ are equivalent on $[-M, M]^d$, and $\|\tilde{\boldsymbol{w}}\|_\infty \leq 2^{\mathcal{O}(d^3)} M^{\mathcal{O}(d^2)}$.*

The *dual graph* $G_D(A) = (V, E)$ of a matrix $A \in \mathbb{Z}^{m\times n}$ has $V = [m]$ and $\{i, j\} \in E$ if rows i and j contain a non-zero at a common coordinate $k \in [n]$. The dual treewidth $\mathrm{tw}_D(A)$ of A is $\mathrm{tw}(G_D(A))$. We do not define treewidth here, but we point out that $\mathrm{tw}(T) = 1$ for every tree T.

▶ **Proposition 8** (Eisenbrand et al. [11, Theorem 98]). *An IP with a constraint matrix A can be solved in time $(\|A\|_\infty g_1(A))^{\mathcal{O}(\mathrm{tw}_D(A))}\,\mathrm{poly}(n, L)$, where n is the dimension of the IP and L is the length of the input.*

▶ **Proposition 9** (Eisenbrand et al. [11, Lemma 25]). *For an integer matrix $A \in \mathbb{Z}^{m\times n}$, we have $g_1(A) \leq (2\|A\|_\infty m + 1)^m$.*

2.3 N-fold Integer Programming

The Integer Programming problem is to solve:

$$\min f(\boldsymbol{x}) : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \boldsymbol{x} \in \mathbb{Z}^n, \tag{IP}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$, $A \in \mathbb{Z}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Z}^m$, and $\boldsymbol{l}, \boldsymbol{u} \in (\mathbb{Z} \cup \{\pm\infty\})^n$.

A *generalized N-fold IP matrix* is defined as

$$E^{(N)} = \begin{pmatrix} E_1^1 & E_1^2 & \cdots & E_1^N \\ E_2^1 & 0 & \cdots & 0 \\ 0 & E_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_2^N \end{pmatrix}. \tag{1}$$

Here, $r, s, t, N \in \mathbb{N}$, $E^{(N)}$ is an $(r + Ns) \times Nt$-matrix, and $E_1^i \in \mathbb{Z}^{r\times t}$ and $E_2^i \in \mathbb{Z}^{s\times t}$ for all $i \in [N]$, are integer matrices. Problem (IP) with $A = E^{(N)}$ is known as *generalized N-fold integer programming* (generalized N-fold IP). "Regular" N-fold IP is the problem where $E_1^i = E_1^j$ and $E_2^i = E_2^j$ for all $i, j \in [N]$. Recent work indicates that the majority of techniques applicable to "regular" N-fold IP also applies to generalized N-fold IP [11].

The structure of $E^{(N)}$ allows us to divide any Nt-dimensional object, such as the variables of $\boldsymbol{x}$, bounds $\boldsymbol{l}, \boldsymbol{u}$, or the objective f, into N *bricks* of size t, e.g. $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$. We use subscripts to index within a brick and superscripts to denote the index of the brick, i.e., x_j^i is the j-th variable of the i-th brick with $j \in [t]$ and $i \in [N]$. We call a brick *integral* if all of its coordinates are integral, and *fractional* otherwise.

Huge N-fold IP. The *huge N-fold IP* problem is an extension of generalized N-fold IP to the high-multiplicity scenario, where blocks come in types and are encoded succinctly by type multiplicities. This means there could be an *exponential* number of bricks in an instance with a polynomial encoding size. The input to the huge N-fold IP problem with $\bar{\tau}$ *types of blocks* is defined by matrices $E_1^i \in \mathbb{Z}^{r\times t}$ and $E_2^i \in \mathbb{Z}^{s\times t}$, $i \in [\bar{\tau}]$, vectors $\boldsymbol{l}^1, \ldots, \boldsymbol{l}^{\bar{\tau}}$, $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{\bar{\tau}} \in \mathbb{Z}^t$, $\boldsymbol{b}^0 \in \mathbb{Z}^r$, $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^{\bar{\tau}} \in \mathbb{Z}^s$, functions $f^1, \ldots, f^{\bar{\tau}} \colon \mathbb{R}^t \to \mathbb{R}$ satisfying $\forall i \in [\bar{\tau}]$, $\forall \boldsymbol{x} \in \mathbb{Z}^t$ we have $f^i(\boldsymbol{x}) \in \mathbb{Z}$ and given by evaluation oracles, and integers $\mu^1, \ldots, \mu^{\bar{\tau}} \in \mathbb{N}$ such that $\sum_{i=1}^{\bar{\tau}} \mu^i = N$. We say that a brick is of type i if its lower and upper bounds are $\boldsymbol{l}^i$ and $\boldsymbol{u}^i$, its right hand side is $\boldsymbol{b}^i$, its objective is f^i, and the matrices appearing at the corresponding coordinates are E_1^i and E_2^i. Denote by T_i the indices of bricks of type i, and note $|T_i| = \mu_i$ and $|\bigcup_{i\in[\bar{\tau}]} T_i| = N$. The task is to solve (IP) with a matrix $E^{(N)}$ which has μ^i blocks of type i for each i. Knop et al. [30] have shown a fast algorithm solving huge n-fold IP. The main idea of their approach is to prove a powerful proximity theorem showing how one can drastically reduce the size of the input instance given that one can solve a corresponding configuration LP (which we shall formally define later). We will build on this approach here. When f^i are restricted to be separable quadratic (and convex) for all $i \in [\bar{\tau}]$, we call the problem *quadratic huge N-fold IP*.

2.4 Configuration LP of Huge N-fold IP

Let a huge N-fold IP instance with $\bar{\tau}$ types be fixed. Recall that μ^i denotes the number of blocks of type i, and let $\boldsymbol{\mu} = (\mu^1, \ldots, \mu^{\bar{\tau}})$. We define for each $i \in [\bar{\tau}]$ the set of configurations of type i as

$$\mathcal{C}^i = \left\{\boldsymbol{c} \in \mathbb{Z}^t \mid E_2^i \boldsymbol{c} = \boldsymbol{b}^i,\, \boldsymbol{l}^i \le \boldsymbol{c} \le \boldsymbol{u}^i\right\} \ .$$

Here we are interested in four instances of convex programming (CP) and convex integer programming (IP) related to huge N-fold IP. First, we have the *Huge IP*

$$\min f(\boldsymbol{x}) :\ E^{(N)}\boldsymbol{x} = \boldsymbol{b},\, \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\, \boldsymbol{x} \in \mathbb{Z}^{Nt} \ . \tag{HugeIP}$$

Then, there is the *Configuration LP* of (HugeIP),

$$\min \boldsymbol{v}\boldsymbol{y} = \sum_{i=1}^{\bar{\tau}} \sum_{\boldsymbol{c}\in\mathcal{C}^i} f^i(\boldsymbol{c}) \cdot y(i,\boldsymbol{c}) \tag{2}$$

$$\sum_{i=1}^{\bar{\tau}} E_1^i \sum_{\boldsymbol{c}\in\mathcal{C}^i} \boldsymbol{c} y(i,\boldsymbol{c}) = \boldsymbol{b}^0$$

$$\sum_{\boldsymbol{c}\in\mathcal{C}^i} y(i,\boldsymbol{c}) = \mu^i \qquad \forall i \in [\bar{\tau}]$$

$$\boldsymbol{y} \ge \boldsymbol{0} \ . \tag{3}$$

Let B be its constraint matrix and $\boldsymbol{d} = (\boldsymbol{b}^0, \boldsymbol{\mu})$ be the right hand side and shorten (2)-(3) to

$$\min \boldsymbol{v}\boldsymbol{y} :\ B\boldsymbol{y} = \boldsymbol{d},\, \boldsymbol{y} \ge \boldsymbol{0} \ . \tag{ConfLP}$$

Finally, by observing that $B\boldsymbol{y} = \boldsymbol{d}$ implies $y(i,\boldsymbol{c}) \le \|\boldsymbol{\mu}\|_\infty$ for all $i \in [\bar{\tau}], \boldsymbol{c} \in \mathcal{C}^i$, defining $C = \sum_{i\in[\bar{\tau}]} |\mathcal{C}^i|$, leads to the *Configuration ILP*,

$$\min \boldsymbol{v}\boldsymbol{y} :\ B\boldsymbol{y} = \boldsymbol{d},\, \boldsymbol{0} \le \boldsymbol{y} \le (\|\boldsymbol{\mu}\|_\infty, \ldots, \|\boldsymbol{\mu}\|_\infty),\, \boldsymbol{y} \in \mathbb{N}^C \ . \tag{ConfILP}$$

The classical way to solve (ConfLP) is by solving its dual using the ellipsoid method and then restricting (ConfLP) to the columns corresponding to the rows encountered while solving the dual, a technique known as column generation. The Dual LP of (ConfLP) in variables $\boldsymbol{\alpha} \in \mathbb{R}^r$, $\boldsymbol{\beta} \in \mathbb{R}^{\bar{\tau}}$ is:

$$\begin{aligned} \max \quad & \boldsymbol{b}^0\boldsymbol{\alpha} + \sum_{i=1}^{\bar{\tau}} \mu^i \beta^i & & \\ \text{s.t.} \quad & (\boldsymbol{\alpha} E_1^i)\boldsymbol{c} - f^i(\boldsymbol{c}) \leq -\beta^i & \forall i \in [\bar{\tau}],\, \forall \boldsymbol{c} \in \mathcal{C}^i & \end{aligned} \tag{4}$$

To verify feasibility of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for $i \in [\bar{\tau}]$, we need to maximize the left-hand side of (4) over all $\boldsymbol{c} \in \mathcal{C}^i$ and check if it is at most $-\beta^i$. This corresponds to solving the following *separation problem*: find integer variables $\boldsymbol{c}$ which for a given vector $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ solve

$$\min f^i(\boldsymbol{c}) - (\boldsymbol{\alpha} E_1^i)\boldsymbol{c} \,:\, E_2^i \boldsymbol{c} = \boldsymbol{b}^i,\, \boldsymbol{l}^i \leq \boldsymbol{c} \leq \boldsymbol{u}^i, \boldsymbol{c} \in \mathbb{Z}^t \ . \tag{$\mathfrak{sep}$-IP}$$

Denote by $\mathfrak{sep}(\boldsymbol{l}^i, \boldsymbol{u}^i, f^i_{\max}, E^i_1, E^i_2)$ the time needed to solve ($\mathfrak{sep}$-IP).

▶ **Lemma 10** (Knop et al. [30, Lemma 12])**.** *An optimal solution $\boldsymbol{y}^*$ of* (ConfLP) *with $|\operatorname{supp}(\boldsymbol{y}^*)| \leq r + \bar{\tau}$ can be found in $(rt\bar{\tau}\langle f_{\max}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{\mu}\rangle)^{\mathcal{O}(1)} \cdot \max_{i \in [\bar{\tau}]} \mathfrak{sep}(\boldsymbol{l}^i, \boldsymbol{u}^i, f^i_{\max}, E^i_1, E^i_2)$ time.*

Since ($\mathfrak{sep}$-IP) is an IP, it can be solved using Proposition 8 in time $g_1(E_2^i)^{\operatorname{tw}_D(E_2^i)} \cdot \operatorname{poly}(\langle \boldsymbol{l}^i, \boldsymbol{u}^i, \boldsymbol{b}^i, \|E_1^i\|_\infty f_{\max}\rangle, t, \bar{\tau})$. Hence, together with Lemma 10, we get the following corollary:

▶ **Corollary 11.** *An optimal solution $\boldsymbol{y}^*$ of* (ConfLP) *with $|\operatorname{supp}(\boldsymbol{y}^*)| \leq r + \bar{\tau}$ can be found in time $(rt\bar{\tau}\langle f_{\max}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{\mu}\rangle)^{\mathcal{O}(1)} \cdot \max_{i \in [\bar{\tau}]} g_1(E_2^i)^{\operatorname{tw}_D(E_2^i)}$.*

We later show how that for our formulations of $R|HM|C_{\max}$ and $R|HM|\sum w_j C_j$, indeed $g_1(E_2^i)$ is polynomial in $\tau, p_{\max}$, and $\operatorname{tw}_D(E_2^i) = 1$, hence the (ConfLP) optimum can be found in polynomial time.

3 Compressing High Multiplicity Scheduling to Quadratic N-fold IP

In this section we are going to prove Theorem 2. To that end, we use the following assumption, mainly to simplify notation.

▶ Remark 12. From here on, we assume $\tau \geq \|\boldsymbol{p}\|_\infty$, i.e., there is a job type for each possible job length $\{1, 2, \ldots, \|\boldsymbol{p}\|_\infty\}$. This is without loss of generality in our regime since both quantities are parameters.

▶ **Theorem 2** (repeated)**.** *The problems $R|HM|C_{\max}$ and $R|HM|\sum w_j C_j$ parameterized by the number of job types τ, the longest processing time $p_{\max}$, and the number of machine kinds κ admit a polynomial compression to quadratic huge N-fold IP parameterized by the number of block types $\bar{\tau}$, the block dimension t, and the largest coefficient $\|E\|_\infty$.*

Recall that in order to use Theorem 2 to provide *kernels* for selected scheduling problems (which are NP-hard) we want to utilize Proposition 5. Thus, we have to show that the "target problem" quadratic huge N-fold IP is in NP.

▶ **Lemma 13.** *The decision version of quadratic huge N-fold IP belongs to* NP.

Proof. We will use Proposition 6 to show that there exists an optimum whose number of distinct configurations is polynomial in the input length. Such a solution can then be encoded by giving those configurations together with their multiplicities, and constitutes a polynomial certificate. Recall that (ConfILP) corresponding to the given instance of huge N-fold is

$$\min \boldsymbol{vy} : B\boldsymbol{y} = \boldsymbol{d},\ \mathbf{0} \leq \boldsymbol{y} \leq (\|\boldsymbol{\mu}\|_\infty, \ldots, \|\boldsymbol{\mu}\|_\infty),\ \boldsymbol{y} \in \mathbb{N}^C \ .$$

Let X be the set of columns of the matrix B extended with an additional coordinate which is the coefficient of the objective function $\boldsymbol{v}$ corresponding to the given column, that is, $\boldsymbol{vb}$ for a column $\boldsymbol{b}$ (i.e., the objective value of configuration $\boldsymbol{b}$). Hence $X \subseteq \mathbb{Z}^{r+\bar{\tau}+1}$ and $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{l}, \boldsymbol{u}, f_{\max}\|_\infty =: M$ for any $\boldsymbol{x} \in X$. Applying Proposition 6, part 2, to X, yields that there exists an optimal solution $\boldsymbol{y}$ of (ConfILP) with $\mathrm{supp}(\boldsymbol{y}) = \tilde{X}$ satisfying $|\tilde{X}| \leq 2(r + \bar{\tau} + 1)\log(4(r + \bar{\tau} + 1)M)$, hence polynomial in the input length of the original instance. ◀

▶ Remark 14. Clearly Lemma 13 holds for any huge N-fold IP whose objective is restricted by some, not necessarily quadratic, polynomial. Moreover, using the newer technique of Aliev et al. [1] it is likely possible to remove any restrictions on the objective.

Using Theorem 2. Before we move to the proof of Theorem 2 we first derive two simple yet interesting corollaries.

▶ **Corollary 15.** *The problems $R||C_{\max}$ and $R||\sum w_j C_j$ admit polynomial kernelizations when parameterized by $\tau, \kappa, p_{\max}$.*

Proof. Let $\mathfrak{obj} \in \{C_{\max}, \sum w_j C_j\}$. We describe a polynomial compression from $R||\mathfrak{obj}$ to quadratic huge N-fold IP which, by Lemma 13, yields the sought kernel, since $R||\mathfrak{obj}$ is NP-hard and huge N-fold with a quadratic objective is in NP.

We first perform the high-multiplicity encoding of the given instance I of $R||\mathfrak{obj}$, thus obtaining an instance I_{HM} of $R|HM|\mathfrak{obj}$ with the input encoded as $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{p}, \boldsymbol{w})$. Now, we can apply Theorem 2 and obtain an instance $\bar{I}_{\text{huge } N\text{-fold}}$ equivalent to I_{HM} with size bounded by a polynomial in $\kappa, \tau, p_{\max}$. ◀

Proof of Corollary 1. This is now trivial, since it suffices to observe that $P||C_{\max}$ is a special case of $R||C_{\max}$, where there is only a single machine kind (i.e., $\kappa = 1$) and $\tau \leq p_{\max}$ job types. Our claim then follows by Corollary 15 (combined with the fact that $P||C_{\max}$ is NP-hard and $R||C_{\max}$ is in NP). ◀

3.1 Huge n-fold IP Models

Denote by $\boldsymbol{n}_{\max}$ the τ-dimensional vector whose all entries are $\|\boldsymbol{n}\|_\infty$. It was shown [29, 30] that $R|HM|C_{\max}$ is modeled as a feasibility instance of huge n-fold IP as follows. Recall that we deal with the decision versions and that k is the upper bound on the value of the objective(s). We set $\boldsymbol{b}^0 = \boldsymbol{n}$, the number of block types is $\bar{\tau} = \kappa$, $E_1^i := (I\ \mathbf{0}) \in \mathbb{Z}^{\tau \times (\tau+1)}$, $E_2^i := (\boldsymbol{p}^i\ 1)$, $\boldsymbol{l}^i = \mathbf{0}, \boldsymbol{u}^i = (\boldsymbol{n}, \infty)$, $\boldsymbol{b}^i = k$, for $i \in [\kappa]$, and the multiplicities of blocks are $\boldsymbol{\mu} = \boldsymbol{m}$. The meaning is that the first type of constraints expressed by the E_1^i matrices ensures that every job is scheduled somewhere, and the second type of constraints expressed by the E_2^i matrices ensures that every machine finishes in time $C_{\max}$.

In the model of $R|HM|\sum w_j C_j$, for each machine kind $i \in [\kappa]$, we define $\preceq^i$ to be the ordering of jobs by the w_j/p_j^i ratio non-increasingly, and let $\boldsymbol{a} = (a_1, a_2, \ldots, a_\tau)$ be a reordering of $\boldsymbol{p}^i$ according to $\preceq^i$. We let

$$G^i := \begin{pmatrix} a_1 & 0 & 0 & \dots & 0 \\ a_1 & a_2 & 0 & \dots & 0 \\ a_1 & a_2 & a_3 & \dots & 0 \\ \vdots & & & \ddots & \\ a_1 & a_2 & a_3 & \dots & a_\tau \end{pmatrix}, \qquad H := -I,$$

with I the $\tau \times \tau$ identity, and define $F^i := (G^i \ H)$ in two steps. Denote by $I^{\preceq^i}$ a matrix obtained from the $\tau \times \tau$ identity matrix by permuting its columns according to $\preceq^i$. The model is then $\boldsymbol{b}^0 = \boldsymbol{n}$, the number of block types is again $\bar{\tau} = \kappa$, for each $i \in [\kappa]$ we have $E_1^i = (I^{\preceq^i} \ \mathbf{0}) \in \mathbb{Z}^{\tau \times 2\tau}$, $E_2^i = \bar{F}^i$, $\boldsymbol{l}^i = \mathbf{0}$, $\boldsymbol{u}^i = (\boldsymbol{n}_{\max}, p_{\max}\tau\boldsymbol{n}_{\max})$, $\boldsymbol{b}^i = \mathbf{0}$, f^i is a separable convex quadratic function (whose coefficients are related to the w_j/p_j^i ratios), and again $\boldsymbol{\mu} = \boldsymbol{m}$. Intuitively, in each brick, the first τ variables represent numbers of jobs of each type on a given machine, and the second τ variables represent the amount of processing time spend by jobs of the first j types with respect to the ordering $\preceq^i$.

3.2 Solving The Separation Problem Quickly: $C_{\max}$

The crucial aspect of complexity of ($\mathfrak{sep}$-IP) is its constraint matrix E_2^i. For $R|HM|C_{\max}$, this is just the vector $(\boldsymbol{p}^i, 1)$. Clearly $tw_D((\boldsymbol{p}^i, 1)) = 1$ since $G_D((\boldsymbol{p}^i, 1))$ is a single vertex. By Proposition 9, $g_1((\boldsymbol{p}^i, 1)) \leq 2\|\boldsymbol{p}\|_\infty + 1$. Moreover, $f_{\max}$ depends polynomially on $\|\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{p}\|_\infty$. Hence, Corollary 11 states that (ConfLP) of the $R|HM|C_{\max}$ model can be solved in time $(rt\bar{\tau}\langle f_{\max}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{\mu}\rangle)^{\mathcal{O}(1)} \cdot \max_i g_1(E_2^i)^{\mathrm{tw}_D(E_2^i)} = \mathrm{poly}(p_{\max}, \tau, \log \|\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{p}\|_\infty)$, which is polynomial in the input.

3.3 Solving The Separation Problem Quickly: $\sum w_j C_j$

The situation is substantially more involved in the case of $R|HM|\sum w_j C_j$: in order to apply Corollary 11, we need to again bound $g_1(E_2^i)$ and $\mathrm{tw}_D(E_2^i)$, but the matrix E_2^i is more involved now. Let

$$\bar{G}^i := \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & a_\tau \end{pmatrix}, \qquad \bar{H} := \begin{pmatrix} -1 & 0 & 0 & \dots & 0 \\ 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & -1 \end{pmatrix},$$

and define $\bar{F}^i = (\bar{G}^i \ \bar{H})$. Now observe that F^i and $\bar{F}^i$ are row-equivalent[3]. This means that we can replace F^i with $\bar{F}^i$ without changing the meaning of the constraints and without changing the feasible set. But while $\mathrm{tw}_D(F_i) = \tau$ (because it is the clique K_τ), we have

▶ **Lemma 16** ($\star$). *For each $i \in [\kappa]$, $\mathrm{tw}_D(\bar{F}^i) = 1$.*

Note that application of Proposition 9 yields $g_1(E_2^i) \leq \mathcal{O}(\tau^\tau)$. This general upper bound, as we shall see, is not sufficient for our purposes, since we need $g_1(E_2^i) \leq \mathrm{poly}(\tau)$. However, we can improve it significantly:

[3] Two matrices A, A' are row-equivalent if one can be transformed into the other using elementary row operations.

▶ **Lemma 17** (Hill-cutting). *We have* $g_\infty(F^i), g_\infty(\bar{F}^i) \leq \mathcal{O}(\tau^4)$ *and* $g_1(F^i), g_1(\bar{F}^i) \leq \mathcal{O}(\tau^5)$ *for every* $i \in [\kappa]$.

Proof. Let $F = F^i$ for some $i \in [\kappa]$. Let $(\boldsymbol{x}, \boldsymbol{z}) \in \mathbb{Z}^{2\tau}$ be some vector satisfying $F \cdot (\boldsymbol{x}, \boldsymbol{z}) = \mathbf{0}$. Our goal now is to show that whenever there exists $k \in [\tau]$ with $|z_k - z_{k-1}| > 2\tau^3 + 1$ (where we define $z_0 := 0$ for convenience), then we can construct a non-zero integral vector $(\boldsymbol{g}, \boldsymbol{h}) \in \mathrm{Ker}_{\mathbb{Z}}(F)$ satisfying $(\boldsymbol{g}, \boldsymbol{h}) \sqsubseteq (\boldsymbol{x}, \boldsymbol{z})$, which shows that $(\boldsymbol{x}, \boldsymbol{z}) \notin (A)$. If no such index k exists, it means that $\|\boldsymbol{x}\|_\infty \leq \mathcal{O}(\tau^3)$ because $z_k - z_{k-1} = a_k x_k$ holds in F (and $\bar{F}^i$). Moreover, if $|z_k - z_{k-1}| \leq 2\tau^3 + 1$ for all $k \in [\tau]$, then $|z_k| \leq \mathcal{O}(\tau^4)$ for every $k \in [\tau]$, hence $\|(\boldsymbol{x}, \boldsymbol{z})\|_\infty \leq \mathcal{O}(\tau^4)$. Note that, since the dimension of $(\boldsymbol{x}, \boldsymbol{z})$ is 2τ, this also implies $\|(\boldsymbol{x}, \boldsymbol{z})\|_1 \leq \mathcal{O}(\tau^5)$. Thus we now focus on the case when $\exists k \in [\tau] : |z_k - z_{k-1}| > 2\tau^3 + 1$.

Let us now assume that $(z_k - z_{k-1})$ is positive and $z_k \geq \tau^3$. There are three other possible scenarios: when $(z_k - z_{k-1})$ is positive but $z_k < \tau^3$, or when $(z_k - z_{k-1})$ is negative and $z_k < -\tau^3$ or $z_k \geq -\tau^3$. We will later show that all these situations are symmetric to the one we consider and our arguments carry over easily, hence our assumption is without loss of generality.

▷ Claim 18 (⋆). If $(z_k - z_{k-1})$ is positive and $z_k \geq \tau^3$, then there exists nonzero $(\boldsymbol{g}, \boldsymbol{h}) \in \mathrm{Ker}_{\mathbb{Z}}(F)$ with $(\boldsymbol{g}, \boldsymbol{h}) \sqsubseteq (\boldsymbol{x}, \boldsymbol{z})$.

Let us consider the remaining symmetric cases. If $z_k - z_{k-1}$ is negative and $z_k < -\tau^3$, then $(-\boldsymbol{x}, -\boldsymbol{z})$ satisfies the original assumption, leading to some $(\boldsymbol{g}', \boldsymbol{h}') \sqsubseteq (-\boldsymbol{x}, -\boldsymbol{z})$, hence $(-\boldsymbol{g}', -\boldsymbol{h}') \sqsubseteq (\boldsymbol{x}, \boldsymbol{z})$ and we are done. If $z_k - z_{k-1}$ is negative but $z_k > -\tau^3$, then we would pick the largest index ℓ smaller than k with $x_\ell > \tau$ and continue as before (the symmetry is that now ℓ is to the left of k rather than to its right; that is, the case distinction from the previous paragraph is according to the value of z_1). Lastly, if $z_k - z_{k-1}$ is positive but $z_k < \tau^3$, negating $(\boldsymbol{x}, \boldsymbol{z})$ gives a reduction to the previous case. ◀

Together with the observation from the previous section and using our newly obtained bounds together with Corollary 11, we obtain:

▶ **Corollary 19.** *Let* $I = (\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{p}, \boldsymbol{w})$ *be an instance of* $R|HM|C_{\max}$ *or* $R|HM|\sum w_j C_j$. *A* (ConfLP) *optimum* $\boldsymbol{y}^*$ *with* $|\mathrm{supp}(\boldsymbol{y}^*)| \leq r + \bar{\tau}$ *can be found in time* $\mathrm{poly}(p_{\max}, \tau, \kappa, \langle \boldsymbol{n}, \boldsymbol{m}, \boldsymbol{p}, \boldsymbol{w} \rangle)$.

4 Conclusions and Research Directions

On the side of theory, one may wonder why not apply the approach developed here to other scheduling problems, in particular those modeled as quadratic huge N-fold IP in [30], such as $R|r_j, d_j|\sum w_j C_j$. The answer is simple: we are not aware of a way to solve the separation problem in polynomial time; in fact, we believe this to be a hard problem roughly corresponding to Unary Vector packing in variable dimension. However, the typical use of Configuration LP is not to obtain an exact optimum (which is often hard), but to obtain an approximation which is good enough. Perhaps a similar approach within our context may lead to so-called *lossy kernels* [36]? However, it is not even clear that an approximate analogue of Theorem 3 holds, because getting an LP solution whose value is close to optimal does not immediately imply getting a solution which is (geometrically) close to some optimum; cf. the discussions on ϵ-accuracy in [11, Definition 31] and [25, Introduction]. Another interesting problem highlighted here is to find a combinatorial algorithm computing the Carathéodory decomposition of the average configuration $\boldsymbol{n}/m$ into machine configurations. The only approach we are aware of so far uses the equivalence of separation and optimization (thus, the ellipsoid method), which is impractical.

References

1 Iskander Aliev, Jesús A. De Loera, Friedrich Eisenbrand, Timm Oertel, and Robert Weismantel. The support of integer optimal solutions. *SIAM Journal on Optimization*, 28(3):2152–2157, 2018.

2 Matthias Bentert, René van Bevern, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. Polynomial-time preprocessing for weighted problems beyond additive goal functions. *CoRR*, abs/1910.00277, 2019. arXiv:1910.00277.

3 Hans L. Bodlaender and Michael R. Fellows. W[2]-hardness of precedence constrained k-processor scheduling. *Operations Research Letters*, 18(2):93–97, 1995. doi:10.1016/0167-6377(95)00031-9.

4 Robert Bredereck, Andrzej Kaczmarczyk, Dušan Knop, and Rolf Niedermeier. High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 505–523. ACM, 2019. doi:10.1145/3328526.3329649.

5 Laurent Bulteau, Danny Hermelin, Dušan Knop, Anthony Labarre, and Stéphane Vialette. The clever shopper problem. *Theory of Computing Systems*, 64(1):17–34, 2020. doi:10.1007/s00224-019-09917-z.

6 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Annal of Pure Appliled Logic*, 84(1):119–138, 1997. doi:10.1016/S0168-0072(95)00020-8.

7 Timothy Chan, Jacob W. Cooper, Martin Koutecký, Daniel Král, and Kristýna Pekárková. A row-invariant parameterized algorithm for integer programming. *CoRR*, abs/1907.06688, 2019. arXiv:1907.06688.

8 Steven Chaplick, Fedor V. Fomin, Petr A. Golovach, Dušan Knop, and Peter Zeman. Kernelization of graph hamiltonicity: Proper h-graphs. In Zachary Friggstad, Jörg-Rüdiger Sack, and Mohammad R. Salavatipour, editors, *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5-7, 2019, Proceedings*, volume 11646 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2019. doi:10.1007/978-3-030-24766-9_22.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. Faster algorithms for integer programs with block structure. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 49:1–49:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.49.

11 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019. arXiv:1904.01361.

12 Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34(5):564–568, 2006.

13 Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer and System Sciences*, 84:1–10, 2017. doi:10.1016/j.jcss.2016.06.004.

14 Michael R. Fellows and Catherine McCartin. On the parametric complexity of schedules to minimize tardy tasks. *Theoretical Computer Science*, 298(2):317–324, 2003. doi:10.1016/S0304-3975(02)00811-3.

15 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

16 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

17 Tomáš Gavenčiak, Dušan Knop, and Martin Koutecký. Integer programming in parameterized complexity: Three miniatures. In Christophe Paul and Michal Pilipczuk, editors, *13th International Symposium on Parameterized and Exact Computation, IPEC 2018, August 20-24, 2018, Helsinki, Finland*, volume 115 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.IPEC.2018.21.

18 P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

19 Steffen Goebbels, Frank Gurski, Jochen Rethmann, and Eda Yilmaz. Change-making problems revisited: a parameterized point of view. *Journal of Combinatorial Optimization*, 34(4):1218–1236, November 2017. doi:10.1007/s10878-017-0143-z.

20 Michel X. Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proc. SODA 2014*, pages 830–839, 2014.

21 Frank Gurski, Carolin Rehs, and Jochen Rethmann. Knapsack problems: A parameterized point of view. *Theoretical Compututer Science*, 775:93–108, 2019. doi:10.1016/j.tcs.2018.12.019.

22 Danny Hermelin, Judith-Madeleine Kubitza, Dvir Shabtay, Nimrod Talmon, and Gerhard J. Woeginger. Scheduling two competing agents when one agent has significantly fewer jobs. In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 55–65. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.55.

23 Danny Hermelin, Michael Pinedo, Dvir Shabtay, and Nimrod Talmon. On the parameterized tractability of single machine scheduling with rejection. *European Journal of Operational Research*, 273(1):67–73, 2019. doi:10.1016/j.ejor.2018.07.038.

24 Danny Hermelin, Dvir Shabtay, and Nimrod Talmon. On the parameterized tractability of the just-in-time flow-shop scheduling problem. *Journal of Scheduling*, 22(6):663–676, 2019. doi:10.1007/s10951-019-00617-7.

25 D. S. Hochbaum and J. G. Shantikumar. Convex separable optimization is not much harder than linear optimization. *J. ACM*, 37(4):843–862, 1990.

26 Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ilps: Treewidth and total unimodularity. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 779–791. Springer, 2015. doi:10.1007/978-3-662-48350-3_65.

27 Klaus Jansen, Marten Maack, and Roberto Solis-Oba. Structural parameters for scheduling with assignment restrictions. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, volume 10236 of *Lecture Notes in Computer Science*, pages 357–368, 2017. doi:10.1007/978-3-319-57586-5_30.

28 Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, FOCS '82, pages 312–320, Washington, DC, USA, 1982. IEEE Computer Society.

29 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21:493–503, 2018.

30 Dušan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. Multitype integer monoid optimization and applications. *CoRR*, abs/1909.07326, 2019. arXiv:1909.07326.

31 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Mathematical Programming*, November 2019. doi:10.1007/s10107-019-01402-2.

32 Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. *ACM Transactions on Economics and Computation*, 8(3), June 2020. doi:10.1145/3396855.

33 Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 85:1–85:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.85.

34 Stefan Kratsch. On polynomial kernels for integer linear programs: Covering, packing and feasibility. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 647–658. Springer, 2013. doi:10.1007/978-3-642-40450-4_55.

35 Stefan Kratsch. On polynomial kernels for sparse integer linear programs. *Journal of Computer and System Sciences*, 82(5):758–766, 2016. doi:10.1016/j.jcss.2015.12.002.

36 Daniel Lokshtanov, Fahad Panolan, MS Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 224–237, 2017.

37 Matthias Mnich and René van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, 2018.

38 Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Math. Program.*, 154(1-2):533–562, 2015.

39 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.

40 Guntram Scheithauer and Johannes Terno. The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, 84(3):562–571, 1995.

41 René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. On (1+\varepsilon)-approximate data reduction for the rural postman problem. In Michael Khachay, Yury Kochetov, and Panos M. Pardalos, editors, *Mathematical Optimization Theory and Operations Research - 18th International Conference, MOTOR 2019, Ekaterinburg, Russia, July 8-12, 2019, Proceedings*, volume 11548 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2019. doi:10.1007/978-3-030-22629-9_20.

42 René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. Parameterized algorithms and data reduction for the short secluded s-t-path problem. *Networks*, 75(1):34–63, 2020. doi:10.1002/net.21904.

43 René van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015. doi:10.1007/s10951-014-0398-5.