Minimizing the Weighted Number of Tardy Jobs Is W[1]-Hard

Klaus Heeger
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Danny Hermelin
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

We consider the $1 \mid\mid \sum w_j U_j$ problem, the problem of minimizing the weighted number of tardy jobs on a single machine. This problem is one of the most basic and fundamental problems in scheduling theory, with several different applications both in theory and practice. Using a reduction from the Multicolored Clique problem, we prove that $1 \mid\mid \sum w_j U_j$ is W[1]-hard with respect to the number $p_\#$ of different processing times in the input, as well as with respect to the number $w_\#$ of different weights in the input. This, along with previous work, provides a complete picture for $1 \mid\mid \sum w_j U_j$ from the perspective of parameterized complexity, as well as almost tight complexity bounds for the problem under the Exponential Time Hypothesis (ETH).

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Combinatorial optimization

Keywords and phrases single-machine scheduling, number of different weights, number of different processing times

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.68

Related Version *Full Version*: https://arxiv.org/abs/2401.01740 [10]

Funding Supported by the ISF, grant No. 1070/20.

1 Introduction

In this paper, we consider the following fundamental scheduling problem: we are given a set of n jobs $\{x_1, \ldots, x_n\}$, where each job x_j is defined by a three integer-valued *characteristic* consisting of: a *processing time* $p(x_j) \in \mathbb{N}$, a *weight* $w(x_j) \in \mathbb{N}$, and a *due date* $d(x_j) \in \mathbb{N}$. We have a single machine to process all jobs $\{x_1, \ldots, x_n\}$ non-preemptively. Thus, in this setting a *schedule* for $\{x_1, \ldots, x_n\}$ is a permutation $\Pi : \{x_1, \ldots, x_n\} \to \{1, \ldots, n\}$ that specifies the processing order of each job. In this way, we schedule in Π a job x_j *starting at time* $R(x_j) = \sum_{\Pi(y)<\Pi(x_j)} p(y)$; that is, the total processing time of jobs preceding x_j in Π. The *completion time* $C(x_j)$ of x_j is then defined by $C(x_j) = R(x_j) + p(x_j)$. Job x_j is said to be *tardy* in Π if $C(x_j) > d(x_j)$, and *early* otherwise. Our goal is to find a schedule Π where the total weight of tardy jobs, i.e. $\sum_{x_j : C(x_j) > d(x_j)} w(x_j)$, is minimized. Following Graham [9], we denote this problem by $1 \mid\mid \sum w_j U_j$.

The $1 \mid\mid \sum w_j U_j$ problem models a very basic and natural scheduling scenario, and is thus very important in practice. However, it also plays a prominent theoretical role, most notably in the theory of scheduling algorithms. For instance, it is one of the first scheduling problems shown to be NP-hard, already included in Karp's famous initial list of 21 NP-hard problems [15]. The algorithm by Lawler and Moore [17] which solves the problem in $O(Pn)$ or $O(Wn)$ time, where P and W are the total processing times and weights of all jobs, is one of the first examples of pseudo-polynomial dynamic programming (see [13] for

recent improvements on this algorithm). Sahni [24] used $1 \mid\mid \sum w_jU_j$ as one of the three first examples to illustrate the important concept of a fully polynomial time approximation scheme (FPTAS) in the area of scheduling. To that effect, several generalizations of the $1 \mid\mid \sum w_jU_j$ problem have been studied in the literature, testing the limits to which these techniques can be applied [1].

Another reason why $1 \mid\mid \sum w_jU_j$ is such a prominent problem is that it is a natural generalization of two classical problems in combinatorial optimization. Indeed, the special case of $1 \mid\mid \sum w_jU_j$ where all jobs have a common due date (i.e. $d(x_1) = \cdots = d(x_n) = d$) translates directly to the dual version of Knapsack [15]: in $1 \mid\mid \sum w_jU_j$ our goal is to minimize the total weight of jobs that complete after d, where in Knapsack we wish to maximize the total weight of jobs that complete before d. (Here d corresponds to the Knapsack size, the processing times correspond to item sizes, and the weights correspond to item values.) When in addition to $d(x_1) = \cdots = d(x_n) = d$, we also have $p(x) = w(x)$ for each job x, the $1 \mid\mid \sum w_jU_j$ problem becomes Subset Sum. The $1 \mid\mid \sum p_jU_j$ problem, a generalization of Subset Sum and a special case of $1 \mid\mid \sum w_jU_j$, has recently received attention in the research community as well [2, 16, 25].

1.1 Parameterized complexity of $1 \mid\mid \sum w_jU_j$

In this paper, we focus on the $1 \mid\mid \sum w_jU_j$ problem from the perspective of parameterized complexity [4, 5]. Thus, we are interested to know whether there exists some algorithm solving $1 \mid\mid \sum w_jU_j$ in $f(k) \cdot n^{O(1)}$ time, for some computable function f and some problem-specific parameter k. In parameterized complexity terminology, this equates to asking whether $1 \mid\mid \sum w_jU_j$ is *fixed-parameter tractable* with respect to parameter k. If we take k to be the total weight of tardy jobs in an optimal schedule, then $1 \mid\mid \sum w_jU_j$ is trivially fixed-parameter tractable by using the aforementioned pseudo-polynomial time algorithms that exist for the problem. In fact, these pseudo-polynomial time algorithms show that the $1 \mid\mid \sum w_jU_j$ is only hard in the *unbounded setting*, i.e. the case where the processing times, weights, and due dates of the jobs may be super-polynomial in the number n of jobs. This is the case we focus on throughout the paper.

In the unbounded setting, the most natural first step is to analyze $1 \mid\mid \sum w_jU_j$ through the "number of different numbers" lens suggested by Fellows *et al.* [6]. In this framework, one considers problem instances with a small variety of numbers in their input. Three natural parameters arise in the context of the $1 \mid\mid \sum w_jU_j$ problem: the number of different due dates $d_\# = |\{d(x_1), \ldots, d(x_n)\}|$, the number of different processing times $p_\# = |\{p(x_1), \ldots, p(x_n)\}|$, and the number of different weights $w_\# = |\{w(x_1), \ldots, w(x_n)\}|$. Regarding parameter $d_\#$, the situation is rather clear. Since $1 \mid\mid \sum w_jU_j$ is essentially equivalent to the NP-hard Knapsack problem already for $d_\# = 1$ [15], there is no $f(k) \cdot n^{O(1)}$ time algorithm for the problem unless P=NP.

▶ **Theorem 1** ([15]). *$1 \mid\mid \sum w_jU_j$ is not fixed-parameter tractable with respect to $d_\#$ unless P=NP.*

What about parameters $p_\#$ and $w_\#$? This question was first studied in [11]. There it was shown $1 \mid\mid \sum w_jU_j$ is polynomial time solvable when either $p_\#$ or $w_\#$ are bounded by a constant. This is done by generalizing the algorithms of Moore [20] and Peha [21] for the cases of $w_\# = 1$ or $p_\# = 1$. Moreover, the authors in [11] show that any instance of $1 \mid\mid \sum w_jU_j$ can be translated to an integer linear program whose number of variables depends solely on $d_\# + p_\#$, $d_\# + w_\#$, or $p_\# + w_\#$. Thus, using fast integer linear program solvers such as Lenstra's celebrated algorithm [18], they proved that $1 \mid\mid \sum w_jU_j$ is fixed parameter tractable with respect to all possible combinations of parameters $d_\#$, $p_\#$, and $w_\#$.

▶ **Theorem 2** ([11]). *The $1 \mid\mid \sum w_j U_j$ problem is solvable in polynomial-time when $p_\# = O(1)$ or $w_\# = O(1)$. Moreover, it is fixed-parameter tractable with respect to parameters $d_\# + p_\#$, $d_\# + w_\#$, or $p_\# + w_\#$.*

Thus, both the aforementioned Knapsack and Subset Sum problems are both fixed-parameter tractable in the number of different numbers viewpoint. What about $1 \mid\mid \sum w_j U_j$? The parameterized complexity status of $1 \mid\mid \sum w_j U_j$ parameterized by either $p_\#$ or $w_\#$ was left open in [11], and due to Theorem 1 and Theorem 2, these are the only two remaining cases. Thus, the main open problem in this context is

"Is $1 \mid\mid \sum w_j U_j$ fixed-parameter tractable with respect to either $p_\#$ or $w_\#$?"

1.2 Our contribution

In this paper, we resolve the open question above negatively, by showing that $1 \mid\mid \sum w_j U_j$ is W[1]-hard with respect to either $p_\#$ or $w_\#$. This means that unless the central hypothesis of parameterized complexity (i.e. FPT≠W[1]) is false, $1 \mid\mid \sum w_j U_j$ is neither fixed-parameter tractable with respect to $p_\#$ nor with respect to $w_\#$.

▶ **Theorem 3.** *$1 \mid\mid \sum w_j U_j$ parameterized by either $p_\#$ or $w_\#$ is W[1]-hard.*

Thus, Theorem 3 together with Theorem 1 and Theorem 2 provide a complete picture of the parameterized complexity landscape of $1 \mid\mid \sum w_j U_j$ with respect to parameters $\{p_\#, w_\#, d_\#\}$, and any of their combinations.

We prove Theorem 3 using an elaborate application of the "multicolored clique technique" [7] which we discuss later on. The proof gives one of the first examples of a single machine scheduling problem which is hard by the number of different processing times or weights. The only other example we are aware of is in [12] for a generalization of $1 \mid\mid \sum w_j U_j$ involving release times and batches. Indeed, there are several open problems regarding the hardness of scheduling problems with a small number of different processing times or weights. The most notable example is arguably the $P \mid \text{HM} \mid C_{\max}$ problem, whose parameterized complexity status is open for parameter $p_\#$ (despite the famous polynomial-time algorithm for the case of $p_\# = O(1)$ [8]). Further, Mnich and van Bevern [19] list three scheduling with preemption problems that are also open for parameter $p_\#$. We believe that ideas and techniques used in our proof can prove to be useful for some of these problems as well.

Regarding exact complexity bounds for $1 \mid\mid \sum w_j U_j$, the best known algorithms for the problem with respect to $p_\#$ and $w_\#$ have running times of the form $O(n^{k+1})$ for either $k = p_\#$ or $k = w_\#$ [11]. How much can we improve on these algorithms? A slight adaptation of our proof which we discuss in the last part of paper gives an almost complete answer to this question. In particular, we can show that the above upper bounds are tight up to a factor of $O(\lg k)$, assuming the Exponential Time Hypotheses (ETH) of Impagliazzo and Paturi [14].

▶ **Corollary 4.** *$1 \mid\mid \sum w_j U_j$ cannot be solved in $n^{o(k/\lg k)}$ time, for either $k = p_\#$ or $k = w_\#$, unless ETH is false.*

1.3 Technical overview

We next give a brief overview of the proof of Theorem 3. As the case of parameter $p_\#$ and $w_\#$ are rather similar, let us focus on parameter $p_\#$. On a high level, our proof follows the standard "multicolored clique technique" introduced in [7]. In this framework, one

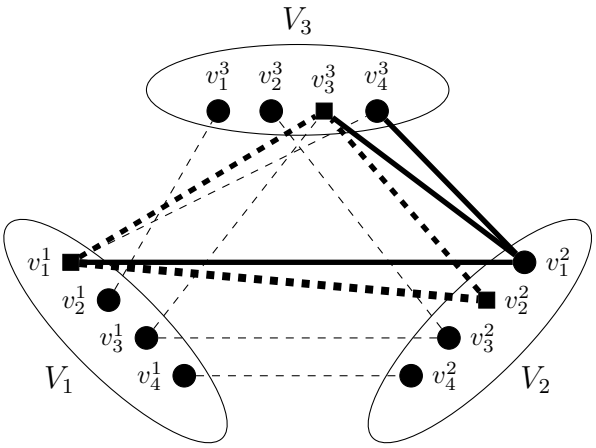


Figure 1 An example nice 3-partite graph (that is, each color class has the same size, and the number of edges between any pair of color classes is the same) with $n = 4$ (the size of each color class) and $m = 4$ (the number of edges between any pair of color classes). The selected vertices are squared. Lexicographically larger or equal edges are dashed, while smaller or equal edges are in bold.

designs a parameterized reduction from k-Multicolored Clique, where we are given a k-partite graph $G = (V_1 \uplus \cdots \uplus V_k, E)$, and we wish to determine whether G contains a clique that includes one vertex from each *color class* V_i of G (see Figure 1). Given an instance of k-Multicolored Clique, our goal is to construct in $f(k) \cdot n^{O(1)}$ time an equivalent instance of $1 \mid\mid \sum w_j U_j$ such that $p_{\#} = g(k)$ for some computable functions f and g.

Our reduction essentially consists of three gadgets: one gadget for the vertices of G, and two gadgets for the edges of G. The gadget for the vertices of G, which we refer to as the *vertex selection gadget*, consists of a set of jobs whose role is to encode the selection of a single vertex from each color class of G (that is, if G contains a multicolored clique, then the selected vertices shall form such a multicolored clique). Since we may assume that each color class of G includes exactly n vertices, we essentially need to encode the selection of k integers $n_1, \ldots, n_k \in \{1, \ldots, n\}$. The crux is that we need to do this using jobs that have only $f(k)$ different processing times. This will be done as follows: For each color class, there are $2n$ jobs with two different, very large processing times in total. As all these jobs have not only a very large processing time but also a very large weight, we know that exactly n of these jobs need to be scheduled for every color class. The jobs with larger processing time also have slightly larger weight than the jobs with smaller processing times. Thereby, scheduling the shorter jobs early results in the jobs from the edge gadgets being started earlier, while selecting the larger jobs early results in a smaller weighted processing time of the late jobs inside the vertex selection gadget. This trade-off between a smaller weighted number of tardy jobs inside the vertex selection gadget and a smaller processing time of the early jobs in the vertex selection gadget encodes the selection of a vertex: selecting n_i jobs of the first kind and $n - n_i$ jobs of the second kind encodes the selection of the i-th vertex of the color class.

The first edge gadget, called the *large edge gadget*, consists of a set of jobs whose role is to count the number of edges that are lexicographically larger or equal to any selected pair $(n_i, n_j) \in \{1, \ldots, n\}^2$. The second edge gadget, referred to as the *small edge gadget*, counts all lexicographically smaller or equal edges. In this way, if the total number of edges counted is $|E| + \binom{k}{2}$, then we know that the vertices indexed by $n_1, \ldots, n_k \in \{1, \ldots, n\}$ form a clique in G. If the total number of counted edges is smaller, then G contains no clique with k vertices. Again, we need to ensure that the jobs in both gadgets have $f(k)$ different processing times.

To ensure all jobs constructed have a small variety of different processing times, we make heavy use of the fact that the processing times (and weights and due dates) can be rather large. Thus, we choose some polynomially large N, and use integers in the range of $\{0, \ldots, N^{f(k)} - 1\}$ for some function f. In this way, considering all integers in their base N representation, allows us to use the different *digits* in the representation to encode various numerical values such as the integers $n_1, \ldots, n_k \in \{1, \ldots, n\}$. We partition each integer in $\{0, \ldots, N^{f(k)} - 1\}$ into *blocks* of $m + 2$ consecutive digits. Each digit in each block has a function that will overall allow us to use the strategy discussed above, and selecting a sufficiently large N ensures that no overflow can occur between adjacent digits. The devil, of course, is in the details.

1.4 Roadmap

The rest of the paper is organized as follows. In Section 2 we briefly review all preliminary results that are necessary for proving our main result, i.e. Theorem 3. Section 3 then contains the proof of Theorem 3 for parameter $p_{\#}$, which is the main technical part of the paper. The discussion of how to adapt the proof of Section 3 to parameter $w_{\#}$ as well as the discussion of our ETH-based lower bounds are deferred to the full version [10]. Further, all proofs of statements marked with $\star$ are deferred to the full version [10].

2 Preliminaries

We use standard notation from graph theory and parameterized complexity. For more details on parameterized complexity, we refer to [3]. Throughout the paper, we will use $<$ to denote the lexicographical order between ordered pairs of integers. Thus,

$$(i, j) < (i_0, j_0) \quad \iff \quad (i < i_0) \text{ or } (i = i_0 \text{ and } j < j_0)$$

for any pair of integers (i, j) and (i_0, j_0).

2.1 The multicolored clique problem

The source W[1]-hard problem in our parameterized reduction used for proving Theorem 3 is the k-Multicolored Clique problem.

▶ **Definition 5.** *Given a k-partite graph $G = (V_1 \uplus \cdots \uplus V_k, E)$, the k-Multicolored Clique problem asks to determine whether G contains a subset of k pairwise adjacent vertices (i.e. a clique of size k).*

For a given a k-partite graph $G = (V_1 \uplus \cdots \uplus V_k, E)$, we let $E_{i,j}$ denote the set of edges between any vertex in V_i and any vertex in V_j, for all $1 \leq i < j \leq k$. We say that a k-partite graph $G = (V_1 \cup \cdots \cup V_k)$ is *nice* if $|V_1| = \cdots = |V_k|$ and $|E_{1,2}| = \cdots = |E_{k-1,k}|$.

▶ **Theorem 6** ([7, 22]). *k-Multicolored Clique is W[1]-hard when parameterized by k, even if the input graph is nice.*

Given a nice k-partite graph $G = (V_1 \uplus \cdots \uplus V_k, E)$, we refer to each $V_i \in \{V_1, \ldots, V_k\}$ as a *color class* of G. We write $V_i = \{v_1^i, \ldots, v_n^i\}$ to denote vertices in V_i for each $1 \leq i \leq k$, and $E_{i,j} = \{e_1^{i,j}, \ldots, e_m^{i,j}\}$ to denote the edges in $E_{i,j}$ for each $1 \leq i < j \leq k$. When considering a specific set of edges $E_{i,j}$, we will often use $\ell_i \in \{1, \ldots, n\}$ and $\ell_j \in \{1, \ldots, n\}$ to respectively denote the index of the vertex in V_i and the index of the vertex of V_j in the ℓ'th edge of $E_{i,j}$. That is, $e_\ell^{i,j} = \{v_{\ell_i}^i, v_{\ell_j}^j\}$.

2.2 EDD schedules

In the $1 \mid\mid \sum w_j U_j$ problem it is frequently convenient to work with what we refer to as an EDD[1] schedule.

▶ **Definition 7.** *A schedule* Π *for a set* $\{x_1, \ldots, x_n\}$ *of jobs is* EDD *if all early jobs in* Π *are scheduled in before all tardy jobs, and the order among early jobs is non-decreasing in due dates. Thus, if* $\Pi(x_i) < \Pi(x_j)$*, then either* x_j *is tardy, or both jobs are early and* $d(x_i) \leq d(x_j)$*.*

The reason EDD schedules are popular when working with the $1 \mid\mid \sum w_j U_j$ problem is that we can always assume that there exists an optimal schedule which is EDD. The following lemma is by now folklore (see e.g. [1]), and can easily be proven by an exchange argument which swaps early jobs that do not satisfy the EDD property in a given optimal schedule.

▶ **Lemma 8.** *Any instance of* $1 \mid\mid \sum w_j U_j$ *has an optimal EDD schedule.*

Thus, throughout our reduction from k-Multicolored Clique, we can restrict our attention to EDD schedules only. Given an EDD schedule Π_0 for a job set $\{x_1, \ldots, x_n\}$, we say that Π is *an extension of* Π_0 *to the set of jobs* $\{y_1, \ldots, y_m\}$ if Π is an EDD schedule for $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ which schedules early all jobs that are scheduled early in Π_0. We write $P(\Pi)$ and $W(\Pi)$ to respectively denote the total processing time and weight of all early jobs in a given EDD schedule Π.

3 Parameter $p_{\#}$

In the following section we present a proof of Theorem 3 for parameter $p_{\#}$. As mentioned above, the proof consists of a parameterized reduction from k-Multicolored Clique parameterized by k to $1 \mid\mid \sum w_j U_j$ parameterized by $p_{\#}$. We use $G = (V = V_1 \uplus \cdots \uplus V_k, E)$ to denote an arbitrary nice k-partite graph given as an instance of k-Multicolored Clique, with $n = |V_1| = \cdots = |V_k|$ and $m = |E_{1,2}| = \cdots = |E_{k-1,k}|$. Before discussing our construction in full detail, we review the terminology that we will use throughout for handling large integers.

3.1 Digits and blocks

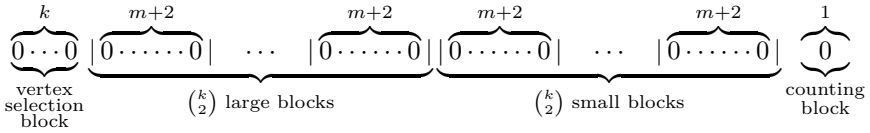
Let N be a polynomially-bounded integer that is chosen to be sufficiently larger than the overall number of jobs in our construction ($N = O(kn + k^2 m)$ is enough). This number will appear frequently in the processing times, weights, and due dates of the jobs in our construction. In particular, it is convenient to view each integer in our construction in its base N representation: each integer will be in the range of $[0, 1, \ldots, N^{k+2\binom{k}{2}\cdot(m+2)+1} - 1]$, and so we can view each integer as a string of length $k + 2\binom{k}{2} \cdot (m+2) + 1$ over the alphabet $\{0, \ldots, N-1\}$. When viewed as such, we will refer to each letter of the string as a *digit*.

Furthermore, we will conceptually partition each integer into *blocks* of consecutive digits as follows (see Example 9): the least significant digit is a block within itself which we refer to as the *counting block*. Following this, there are $\binom{k}{2}$ blocks which we refer to as the *small blocks*, consisting of $m+2$ digits each, where the first (least significant) block corresponds to

[1] EDD here is an acronym for "Earliest Due Date".

the color class pair (V_1, V_2), the second corresponds to (V_1, V_3), and so forth. Following the small blocks are $\binom{k}{2}$ blocks which we dub the *large blocks*, which again consist of $m+2$ digits each, and are ordered similarly to the left blocks. The final block is the *vertex selection block* which consists of the k most significant digits of the given integer.

▶ **Example 9.** As example, the following is the partitioning of integer 0:



The large and small blocks are ordered in increasing lexicographic order of (i, j), so the $(1, 2)$ small block is the first block following the counting block. In each block we order the digits from least significant to most significant, so the first digit in the $(1, 2)$ small block is the second least significant digit overall.

Let $g : \{(i, j) \mid 1 \leq i < j \leq k\} \to \{0, \ldots, \binom{k}{2} - 1\}$ denote the lexicographic ordering function, that is $g(i, j) > g(i_0, j_0)$ iff $(i, j) > (i_0, j_0)$ for all $1 \leq i < j \leq k$. Furthermore, let $G(i, j) = (m + 2) \cdot g(i, j) + 1$ for all $1 \leq i < j \leq k$. Similarly, let $f : \{(i, j) \mid 1 \leq i < j \leq k\} \to \{\binom{k}{2}, \ldots, 2 \cdot \binom{k}{2} - 1\}$ denote the function defined by $f(i, j) = \binom{k}{2} + g(i, j)$, and let $F(i, j) = (m + 2) \cdot f(i, j) + 1$. We will use the following constants in our construction:

- $X_i := N^{(m+2)\cdot 2\binom{k}{2}+i}$ for $i \in \{1, \ldots, k\}$,
- $Y_{i,j} := N^{F(i,j)+m+1}$ for $i < j \in \{1, \ldots, k\}$, and
- $Z_{i,j} := N^{G(i,j)+m+1}$ for $i < j \in \{1, \ldots, k\}$.

Thus, X_i corresponds to the i'th digit in the vertex selection block, $Y_{i,j}$ corresponds to the last digit in the (i, j) large block, and $Z_{i,j}$ corresponds to the last digit in the (i, j) small block.

3.2 Vertex selection gadget

The role of the vertex selection gadget is to encode the selection of k vertices, one from each color class V_i of G. In constructing the vertex selection jobs, we will use the following two values associated with each $i \in \{1, \ldots, k\}$:

- $L(i) = \sum_{j=1}^{i-1} N^{F(j,i)} + \sum_{j=i+1}^{k} N^{F(i,j)+1}$.
- $S(i) = \sum_{j=1}^{i-1} N^{G(j,i)} + \sum_{j=i+1}^{k} N^{G(i,j)+1}$.

Thus, adding $L(i)$ to an integer corresponds to adding a 1 to the first digit of every (j, i) large block with $j < i$, and a 1 to the second digit of any (i, j) large block with $j > i$. Adding $S(i)$ corresponds to adding a 1 to the same digits in the small blocks.

Let $1 \leq i \leq k$, and consider the color class V_i of G. The V_i vertex selection gadget is constructed as follows. Let P_i^V denote the following value:

$$P_i^V \;=\; n \cdot \sum_{j>i} X_j \;=\; n \cdot \sum_{j>i} N^{(m+2)\cdot 2\binom{k}{2}+j}.$$

Thus, P_i^V has n as its j'th most significant digit for $j < i$, and 0 in all of its other digits. We construct $n - 1$ copies of the job pair $\{x_i, \neg x_i\}$ with the following characteristic:

- $p(x_i) = w(x_i) = X_i + L(i)$.
- $p(\neg x_i) = w(\neg x_i) = X_i + S(i)$.
- $d(x_i) = d(\neg x_i) = P_{i-1}^V + N^{(m+2)\cdot 2\binom{k}{2}}$ (where $P_0^V = n \cdot \sum_i X_i$).

In addition to these $n-1$ copies of $\{x_i, \neg x_i\}$, we construct a single job x_i^* with similar processing time and due date as x_i, but with significantly larger weight:

- $p(x_i^*) = p(x_i)$ and $d(x_i^*) = d(x_i)$.
- $w(x_i^*) = (n+1) \cdot X_i + L(i)$.

The jobs x_i^*, x_i, and $\neg x_i$ are called *V_i vertex selection jobs*.

Overall, we have $(2n-1) \cdot k$ vertex selection jobs that together have only $2k$ different processing times, $3k$ different weights, and k different due dates. The vertex selection jobs are constructed in a way so that any schedule with sufficiently large weight of early jobs will schedule n early jobs from $\{x_i^*, x_i, \neg x_i\}$ for each $1 \leq i \leq n$. Due to the large weight of x_i^*, job x_i^* will always be scheduled early, while the number of early jobs x_i and $\neg x_i$ will be used to encode an integer $n_i \in \{1, \ldots, n\}$ corresponding to vertex $v_{n_i}^i \in V_i$.

▶ **Lemma 10** (⋆). *Let $n_1, \ldots, n_k \in \{1, \ldots, n\}$. There exists a schedule $\Pi = \Pi(n_1, \ldots, n_k)$ for the vertex selection jobs such that for each $i \in \{1, \ldots, k\}$ precisely n_i jobs from $\{x_i^*, x_i\}$ and $n - n_i$ copies of $\neg x_i$ are early in Π for each $1 \leq i \leq k$.*

Throughout the remainder of the proof, we will use $\Pi = \Pi(n_1, \ldots, n_k)$ to denote the schedule that schedules x_i^*, exactly $n_i - 1$ jobs x_i, and $n - n_i$ jobs $\neg x_i$ early. Let W_V denote the value

$$W_V = 2n \cdot \sum_i X_i.$$

Then the following corollary follows directly from Lemma 10:

▶ **Corollary 11.** *Let $\Pi = \Pi(n_1, \ldots, n_k)$ for some $n_1, \ldots, n_k \in \{1, \ldots, n\}$. Then*

(i) $P(\Pi) = P_V + \sum_i n_i \cdot L(i) + \sum_i (n - n_i) \cdot S(i)$.

(ii) $W(\Pi) = W_V + \sum_i n_i \cdot L(i) + \sum_i (n - n_i) \cdot S(i)$.

▶ **Example 12.** Consider the 3-partite graph in Figure 1, where vertex v_i^i is selected for each color class V_i. Then the total processing time of all early vertex selection jobs in this example is:

$$444|\underbrace{000023|}_{\substack{(2,3)\\ \text{large}}}|\underbrace{000013|}_{\substack{(1,3)\\ \text{large}}}|\underbrace{000012|}_{\substack{(1,2)\\ \text{large}}}|\underbrace{000021|}_{\substack{(2,3)\\ \text{small}}}|\underbrace{000031|}_{\substack{(1,3)\\ \text{small}}}|\underbrace{000032|}_{\substack{(1,2)\\ \text{small}}}|0$$

The total weight of all early vertex selection jobs is identical, except that the vertex selection block equals "888" instead of "444".

As mentioned above, the vertex selection jobs are constructed in a way so that any schedule Π for these jobs with sufficiently large weight of early jobs will schedule precisely n early jobs from $\{x_i^*, x_i, \neg x_i\}$ for each $1 \leq i \leq k$. This is formally proven in the following lemma:

▶ **Lemma 13** (⋆). *Let Π be an EDD schedule for the vertex selection jobs with $W(\Pi) \geq W_V$. Then $\Pi = \Pi(n_1, \ldots, n_k)$ for some $n_1, \ldots, n_k \in \{1, \ldots, n\}$.*

3.3 Large edge gadget

We next describe the large edge gadget. The role of this gadget is to "count" all edges that are lexicographically larger or equal to pairs of selected vertices. This is done by constructing a pair of jobs $\{y_\ell^{i,j}, \neg y_\ell^{i,j}\}$ for each edge $e_\ell^{i,j}$ of G, along with some additional filler jobs.

Let $1 \leq i < j \leq k$. The (i,j) large edge gadget is constructed as follows. First we define $P^L_{i,j}$ to be the following value:

$$P^L_{i,j} = \sum_{(i_0,j_0)>(i,j)} \left(m \cdot Y_{i_0,j_0} + n \cdot N^{F(i_0,j_0)+1} + n \cdot N^{F(i_0,j_0)} \right).$$

Thus, the two first digits of the (i,j) large block in $P^L_{i,j}$ equal n, the last digit of this block equals m, and all other digits equal 0. Let $\ell \in \{1,\dots,m\}$, and suppose that the ℓ'th edge between V_i and V_j is the edge $e^{i,j}_\ell = \{v^i_{\ell_i}, v^j_{\ell_j}\}$ for some $\ell_i, \ell_j \in \{1,\dots,n\}$. We construct two jobs $y^{i,j}_\ell$ and $\neg y^{i,j}_\ell$ corresponding to $e^{i,j}_\ell$ with the following characteristic:

- $p(y^{i,j}_\ell) = Y_{i,j}$ and $w(y^{i,j}_\ell) = Y_{i,j} \,/\, N^\ell + 1$.
- $p(\neg y^{i,j}_\ell) = Y_{i,j}$ and $w(\neg y^{i,j}_\ell) = Y_{i,j} \,/\, N^\ell$.
- $d(y^{i,j}_\ell) = P_V + P^L_{i,j} + \ell \cdot Y_{i,j} + \ell_i \cdot N^{F(i,j)+1} + \ell_j \cdot N^{F(i,j)} + N^{F(i,j)-1}$.
- $d(\neg y^{i,j}_\ell) = P_V + P^L_{i,j} + \ell \cdot Y_{i,j} + n \cdot N^{F(i,j)+1} + n \cdot N^{F(i,j)} + N^{F(i,j)-1}$.

Observe that both jobs have the same processing time, which is equal throughout for jobs corresponding to other edges of $E_{i,j}$. Also note that the weight of $y^{i,j}_\ell$ is slightly larger than the weight of $\neg y^{i,j}_\ell$, while the due date of $\neg y^{i,j}_\ell$ is significantly larger than the due date of $y^{i,j}_\ell$.

We will also need to add *filler jobs* that will help us control the total processing times of all early jobs selected from the (i,j) large edge gadget. We construct n copies of the the job pair $\{f^{i,j}_0, f^{i,j}_1\}$ which have the following characteristic:

- $p(f^{i,j}_0) = w(f^{i,j}_0) = N^{F(i,j)}$.
- $p(f^{i,j}_1) = w(f^{i,j}_1) = N^{F(i,j)+1}$.
- $d(f^{i,j}_0) = d(f^{i,j}_1) = P_V + P^L_{i,j} + m \cdot Y_{i,j} + n \cdot N^{F(i,j)+1} + n \cdot N^{F(i,j)} + N^{F(i,j)-1}$.

Thus, altogether, the large edge gadget consists of the job pair $\{y^{i,j}_\ell, \neg y^{i,j}_\ell\}$ for $\ell \in \{1,\dots,m\}$ and n copies of the job pair $\{f^{i,j}_0, f^{i,j}_1\}$, for each $1 \leq i < j \leq k$. Note that the large edge jobs have $3\binom{k}{2}$ different processing times in total. We next prove a lemma regarding the structure of certain schedules for the vertex selection and large edge job. This structure is what allows us to count all edges that are lexicographically larger or equal any selected pair (n_i, n_j). Let Π_V be a schedule for the vertex selection jobs. We say that Π is an *optimal extension* of Π_V to the set of large edge jobs if all jobs that are early in Π_V are also early in Π, and there is no other such schedule with a larger total weight of early jobs.

▶ **Lemma 14** (⋆). *Let $\Pi_V = \Pi(n_1,\dots,n_k)$ be a schedule for the vertex selection jobs for some $n_1,\dots,n_k \in \{1,\dots,n\}$, and let Π be an optimal extension of Π_V to the set of large edge jobs. Then the following properties hold for each $1 \leq i < j \leq k$:*

(a) *The total processing time P of all vertex selection jobs and all (i_0,j_0) large jobs for $(i_0,j_0) > (i,j)$ which are early in Π satisfies*

$$P \geq P_V + P^L_{i,j} + n_i \cdot N^{F(i,j)+1} + n_j \cdot N^{F(i,j)}$$

and

$$P \leq P_V + P^L_{i,j} + n_i \cdot N^{F(i,j)+1} + n_j \cdot N^{F(i,j)} + N^{F(i,j)-1}.$$

(b) *For each $\ell \in \{1,\dots,m\}$ we have that either job $y^{i,j}_\ell$ or job $\neg y^{i,j}_\ell$ is early in Π, but not both. Job $y^{i,j}_\ell$ is early iff $(n_i,n_j) \leq (\ell_i,\ell_j)$, where $e^{i,j}_\ell = \{v^i_{\ell_i}, v^j_{\ell_j}\}$ is the ℓ'th edge in $E_{i,j}$.*

(c) *Precisely $n - n_i$ copies of job $f^{i,j}_1$ and $n - n_j$ copies of job $f^{i,j}_0$ are scheduled early in Π.*

Using Lemma 14, we can again derive the total processing time and weight of all early jobs in any optimal EDD schedule Π for vertex selection jobs and the large edge jobs. Let W^L be the following value:

$$W_L = \sum_{(i,j)} \left(\sum_{\ell} Y_{i,j} \,/\, N^{\ell} + n \cdot N^{F(i,j)+1} + n \cdot N^{F(i,j)} \right).$$

Define $P_L = P^L_{0,0}$. Moreover, for $1 \leq i < j \leq k$ and $n_i, n_j \in \{1, \ldots, n\}$, define $m^L_{i,j}(n_i, n_j)$ to be the total number of edges in $E_{i,j}$ that are lexicographically larger or equal to (n_i, n_j). That is, the total number of edges $e^{i,j}_{\ell} = (v^i_{\ell_i}, v^j_{\ell_j}) \in E_{i,j}$ with $(n_i, n_j) \leq (\ell_i, \ell_j)$. Then the following holds:

▶ **Corollary 15** ($\star$). *Let $\Pi_V = (n_1, \ldots, n_k)$ be a schedule for the vertex selection jobs for some $n_1, \ldots, n_k \in \{1, \ldots, n\}$, and let Π be an optimal extension of Π_V to the set of vertex selection and large edge jobs. Then*

(i) $P(\Pi) = P_V + P_L + \sum_i (n - n_i) \cdot S(i)$.

(ii) $W(\Pi) = W_V + W_L + \sum_i (n - n_i) \cdot S(i) + \sum_{(i,j)} m^L_{i,j}(n_i, n_j)$.

▶ **Example 16.** Recall the schedule of Example 12. After optimally scheduling all jobs from the $(2, 3)$ large edge gadget (including the filler jobs), the total processing time of all early jobs is:

$$444|\underbrace{400044|}_{\substack{(2,3)\\ \text{large}}}|\underbrace{000013|}_{\substack{(1,3)\\ \text{large}}}|\underbrace{000012|}_{\substack{(1,2)\\ \text{large}}}|\underbrace{000021|}_{\substack{(2,3)\\ \text{small}}}|\underbrace{000031|}_{\substack{(1,3)\\ \text{small}}}|\underbrace{000032|}_{\substack{(1,2)\\ \text{small}}}|0$$

The total weight of all early jobs is

$$888|\underbrace{111144|}_{\substack{(2,3)\\ \text{large}}}|\underbrace{000013|}_{\substack{(1,3)\\ \text{large}}}|\underbrace{000012|}_{\substack{(1,2)\\ \text{large}}}|\underbrace{000021|}_{\substack{(2,3)\\ \text{small}}}|\underbrace{000031|}_{\substack{(1,3)\\ \text{small}}}|\underbrace{000032|}_{\substack{(1,2)\\ \text{small}}}|2$$

as $m^L_{2,3}(2, 3) = 2$ in the example.

3.4 Small edge gadget

We next describe the small edge gadget. Analogous to the large edge gadget, the role of the small edge gadget is to count all edges that are lexicographically smaller or equal to pairs of selected vertices. It is constructed similarly to the large edge gadget, except that we focus on the small blocks of the integers. We start by defining $P^S_{i,j}$, which is analogous to the value $P^L_{i,j}$ used in the large edge gadget:

$$P^S_{i,j} \;=\; \sum_{(i_0,j_0)>(i,j)} \left(m \cdot Z_{i_0,j_0} + n \cdot N^{G(i_0,j_0)+1} + n \cdot N^{G(i_0,j_0)} \right).$$

For each $1 \leq i < j \leq j$, we construct the (i, j) *small edge gadget* as follows: let $\ell \in \{1, \ldots, m\}$, and suppose that $e^{i,j}_{\ell} = \{v^i_{\ell_i}, v^j_{\ell_j}\} \in E_{i,j}$ is the ℓ'th edge in $E_{i,j}$. We construct two jobs $z^{i,j}_{\ell}$ and $\neg z^{i,j}_{\ell}$ associated with $e^{i,j}_{\ell}$ that have the following characteristic:

- $p(z^{i,j}_{\ell}) = Z_{i,j}$ and $w(z^{i,j}_{\ell}) = Z_{i,j} \,/\, N^{\ell} + 1$.
- $p(\neg z^{i,j}_{\ell}) = Z_{i,j}$ and $w(\neg z^{i,j}_{\ell}) = Z_{i,j} \,/\, N^{\ell}$.
- $d(z^{i,j}_{\ell}) = P_V + P_L + P^S_{i,j} + \ell \cdot Z_{i,j} + (n - \ell_i) \cdot N^{G(i,j)+1} + (n - \ell_j) \cdot N^{G(i,j)} + N^{G(i,j)-1}$.
- $d(\neg z^{i,j}_{\ell}) = P_V + P_L + P^S_{i,j} + \ell \cdot Z_{i,j} + n \cdot N^{G(i,j)+1} + n \cdot N^{G(i,j)} + N^{G(i,j)-1}$.

We will also add *filler jobs* as done in the large edge gadgets. We construct n copies of the job pair $\{g_0^{i,j}, g_1^{i,j}\}$ which have the following characteristic:

- $p(g_0^{i,j}) = w(g_0^{i,j}) = N^{G(i,j)}$.
- $p(g_1^{i,j}) = w(g_1^{i,j}) = N^{G(i,j)+1}$.
- $d(g_0^{i,j}) = d(g_1^{i,j}) = P_V + P_L + P_{i,j}^S + \ell \cdot Z_{i,j} + n \cdot N^{G(i,j)+1} + n \cdot N^{G(i,j)}$.

Thus, altogether, the (i,j) small edge gadget consists of all job pairs $\{z_\ell^{i,j}, \neg z_\ell^{i,j}\}$ for $\ell \in \{1,\dots,m\}$, and all n copies of the job pair $\{g_0^{i,j}, g_1^{i,j}\}$. Note that all these jobs have three different processing times in total.

Lemma 17 below is analogous to Lemma 14, except that the structure that it conveys allows us to count all edges that are lexicographically smaller or equal (as opposed to larger or equal) to pairs of selected vertices. We have the following:

▶ **Lemma 17** (⋆). *Let $\Pi_V = (n_1,\dots,n_k)$ be a schedule for the vertex selection jobs for some $n_1,\dots,n_k \in \{1,\dots,n\}$, and let Π_L be an optimal extension of Π_V to the set of large edge jobs. Let Π be an optimal extension of Π_L to the set of small edge jobs. Then the following properties hold for each $1 \le i < j \le k$:*

(a) *The total processing time P of all vertex selection jobs, all large edge jobs, and all (i_0,j_0) small jobs for $(i_0,j_0) > (i,j)$, which are early in Π satisfies*

$$P \;\ge\; P_V + P_L + P_{i,j}^S + (n-n_i)\cdot N^{G(i,j)+1} + (n-n_j)\cdot N^{G(i,j)}$$

and

$$P \;\le\; P_V + P_L + P_{i,j}^S + (n-n_i)\cdot N^{G(i,j)+1} + (n-n_j)\cdot N^{G(i,j)} + N^{G(i,j)-1}.$$

(b) *For each $\ell \in \{1,\dots,m\}$ we have that either job $z_\ell^{i,j}$ or job $\neg z_\ell^{i,j}$ is early in Π, but not both. Job $z_\ell^{i,j}$ is early if and only if $(n_i,n_j) \ge (\ell_i,\ell_j)$, where $e_\ell^{i,j} = \{v_{\ell_i}^i, v_{\ell_j}^j\}$ is the ℓ'th edge in $E_{i,j}$.*

(c) *Precisely n_i copies of job $g_1^{i,j}$ and n_j copies of job $g_0^{i,j}$ are scheduled early in Π.*

For $1 \le i < j \le k$ and $n_i, n_j \in \{1,\dots,n\}$, define $m_{i,j}^S(n_i,n_j)$ to be the total number of edges in $E_{i,j}$ that are lexicographically smaller or equal to (n_i,n_j). That is, the total number of edges $e_\ell^{i,j} = (v_{\ell_i}^i, v_{\ell_j}^j) \in E_{i,j}$ with $(n_i,n_j) \ge (\ell_i,\ell_j)$. Let W_S denote the following value:

$$W_S = \sum_{(i,j)} \left(\sum_{\ell} Z_{i,j} \,/\, N^\ell + n \cdot N^{G(i,j)+1} + n \cdot N^{G(i,j)} \right).$$

We have the following corollary of Lemma 17.

▶ **Corollary 18** (⋆). *Let $\Pi_V = (n_1,\dots,n_k)$ be a schedule for the vertex selection jobs for some $n_1,\dots,n_k \in \{1,\dots,n\}$, let Π_L be an optimal extension of Π_V to the set of large edge jobs, and let Π be an optimal extension of Π_L to the set of small edge jobs. Then*

$$W(\Pi) = W_V + W_L + W_S + \sum_{(i,j)} m_{i,j}^L(n_i,n_j) + \sum_{(i,j)} m_{i,j}^S(n_i,n_j).$$

▶ **Example 19.** Consider the schedule of Example 16. After scheduling all remaining large edge jobs, and all jobs from the $(2,3)$ small edge gadget (including the filler jobs), the total processing time of all early jobs is:

$$444|\underbrace{400044|}_{\substack{(2,3)\\ \text{large}}}|\underbrace{400044|}_{\substack{(1,3)\\ \text{large}}}|\underbrace{400044|}_{\substack{(1,2)\\ \text{large}}}|\underbrace{400044|}_{\substack{(2,3)\\ \text{small}}}|\underbrace{000031|}_{\substack{(1,3)\\ \text{small}}}|\underbrace{000032|}_{\substack{(1,2)\\ \text{small}}}|0$$

The total weight of all early jobs is

$$888|\underbrace{111144|}_{\substack{(2,3)\\ \text{large}}}|\underbrace{111144|}_{\substack{(1,3)\\ \text{large}}}|\underbrace{111144|}_{\substack{(1,2)\\ \text{large}}}|\underbrace{11114|}_{\substack{(2,3)\\ \text{small}}}|\underbrace{000031|}_{\substack{(1,3)\\ \text{small}}}|\underbrace{000032|}_{\substack{(1,2)\\ \text{small}}}|12$$

as $\sum_{(i,j)} m^L_{i,j}(i,j) + m^S_{2,3}(2,3) = 12$ in the example.

3.5 Correctness

We have completed the description of all jobs in our $1 \mid\mid \sum w_j U_j$ instance. Table 1 provides a compact list of the characteristics of all these jobs. Lemma 20 below, along with Theorem 6, completes our proof of Theorem 3 for parameter $p_\#$. Theorem 3 for parameter $w_\#$ follows by a similar reduction; for details we refer to the full version [10].

Table 1 The weights, processing times, due dates, and multiplicities of all jobs in our construction. Here, $P^L_{i,j}(\ell)$ is shorthand notation for $P_V + P^L_{i,j} + \ell \cdot Y_{i,j} + N^{F(i,j)-1}$, and $P^S_{i,j}(\ell) = P_V + P_L + P^S_{i,j} + \ell \cdot Z_{i,j} + N^{G(i,j)-1}$.

Job	Proc. Time	Weight	Due Date	Mult.
x^*_i	$X_i + L(i)$	$(n+1) \cdot X_i + L(i)$	$P^V_{i-1} + N^{(m+2)\cdot 2\binom{k}{2}}$	1
x_i	$X_i + L(i)$	$X_i + L(i)$	$d(x^*_i)$	$n-1$
$\neg x_i$	$X_i + S(i)$	$X_i + S(i)$	$d(x^*_i)$	n
$y^{i,j}_\ell$	$Y_{i,j}$	$Y_{i,j} \,/\, N^\ell + 1$	$P^L_{i,j}(\ell) + \ell_i \cdot N^{F(i,j)+1} + \ell_j \cdot N^{F(i,j)}$	1
$\neg y^{i,j}_\ell$	$Y_{i,j}$	$Y_{i,j} \,/\, N^\ell$	$P^L_{i,j}(\ell) + n \cdot N^{F(i,j)+1} + n \cdot N^{F(i,j)}$	1
$f^{i,j}_1$	$N^{F(i,j)+1}$	$N^{F(i,j)+1}$	$P^L_{i,j}(m) + n \cdot N^{F(i,j)+1} + n \cdot N^{F(i,j)}$	n
$f^{i,j}_0$	$N^{F(i,j)}$	$N^{F(i,j)}$	$d(f^{i,j}_1)$	n
$z^{i,j}_\ell$	$Z_{i,j}$	$Z_{i,j} \,/\, N^\ell + 1$	$P^S_{i,j}(\ell) + (n-\ell_i) \cdot N^{G(i,j)+1} + (n-\ell_j) \cdot N^{G(i,j)}$	1
$\neg z^{i,j}_\ell$	$Z_{i,j}$	$Z_{i,j} \,/\, N^\ell$	$P^S_{i,j}(\ell) + n \cdot N^{G(i,j)+1} + n \cdot N^{G(i,j)}$	1
$g^{i,j}_1$	$N^{G(i,j)+1}$	$N^{G(i,j)+1}$	$P^S_{i,j}(m) + n \cdot N^{G(i,j)+1} + n \cdot N^{G(i,j)}$	n
$g^{i,j}_0$	$N^{G(i,j)}$	$N^{G(i,j)}$	$d(g^{i,j}_1)$	n

▶ **Lemma 20.** *There is a parameterized reduction from k-Multicolored Clique (restricted to nice k-partite graphs) parameterized by k to $1 \mid\mid \sum w_j U_j$ parameterized by $p_\#$.*

Proof. The reduction is as described throughout the section. It is in fact a reduction to the equivalent problem of $1 \mid\mid \sum w_j U_j$ where the goal is to maximize the weight of early jobs. The reduction can be carried out in polynomial-time, and the total number of different processing-times $p_\#$ in the resulting $1 \mid\mid \sum w_j U_j$ instance is $2k + 6\binom{k}{2}$ (see Table 1). To complete the proof of the lemma, we argue that the graph $G = (V = V_1 \uplus \cdots \uplus V_k, E)$ of the input k-Multicolored Clique instance has a clique of size k iff the constructed $1 \mid\mid \sum w_j U_j$ instance has a schedule where the total weight of early jobs is at least $W_V + W_L + W_S + (m+1) \cdot \binom{k}{2}$.

Suppose G has a clique of size k with $v^1_{n_1} \in V_1, \ldots, v^k_{n_k} \in V_k$. Then $\sum_{(i,j)} m^L_{i,j}(n_i, n_j) + m^S_{i,j}(n_i, n_j) = (m+1) \cdot \binom{k}{2}$. Thus, according to Corollary 18, the optimal extension Π of $\Pi_V = \Pi(n_1, \ldots, n_k)$ to the set of large and small edge jobs has total weight of early jobs $W(\Pi) = W_V + W_L + W_S + (m+1) \cdot \binom{k}{2}$. Conversely, suppose that there is a schedule Π for the $1 \mid\mid \sum w_j U_j$ instance with $W(\Pi) \geq W_V + W_L + W_S + (m+1) \cdot \binom{k}{2}$. Let Π_V be the restriction of Π to the vertex selection jobs. Then as $N^{(m+2)\cdot 2\binom{k}{2}-1}$ is larger than the total weight of all large and small edge jobs, we have $W(\Pi_V) \geq W_V$ as otherwise we have $W(\Pi_V) < W_V - X_1 + N^{(m+2)\cdot 2\binom{k}{2}-1}$, implying $W(\Pi) < W_V$. Thus, $\Pi_V = \Pi(n_1, \ldots, n_k)$

for some $n_1, \ldots, n_k \in \{1, \ldots, n\}$ according to Lemma 13. We may assume without loss of generality that Π is an optimal extension of Π_V to set of large and small edge jobs. It follows then from Corollary 18 that $\sum_{(i,j)} m^L_{i,j}(n_i, n_j) + m^S_{i,j}(n_i, n_j) = (m+1) \cdot \binom{k}{2}$, which means that there are $\binom{k}{2}$ edges in G between vertices in $\{v^1_{n_1}, \ldots, v^k_{n_k}\}$. Thus, $v^1_{n_1}, \ldots, v^k_{n_k}$ is a clique of size k in G. ◀

Slightly adapting the reduction, we also get an ETH-based lower bound (we refer to the full version [10] for details).

▶ **Corollary 4.** *$1 \mid\mid \sum w_j U_j$ cannot be solved in $n^{o(k/\lg k)}$ time, for either $k = p_\#$ or $k = w_\#$, unless ETH is false.*

4 Conclusions

In the current paper we completely resolved the parameterized complexity status of $1 \mid\mid \sum w_j U_j$ with respect to parameters $p_\#$, $w_\#$, and $d_\#$. Our result also gives almost ETH tight bounds in the case when only one of $p_\#$ or $w_\#$ is bounded by a constant. However, there still remains several research directions to explore regarding the $1 \mid\mid \sum w_j U_j$ problem, and its variants. Below we list a few questions that still remain open:

- Can the gap between lower and upper bound in Corollary 4 be closed? That is, can one show a lower bound of $n^{o(k)}$ or can $1 \mid\mid \sum w_j U_j$ be solved in $n^{O(k/\lg k)}$ time, for $k = p_\#$ or $k = w_\#$?
- The current FPT algorithms solving $1 \mid\mid \sum w_j U_j$ for parameters $k = p_\# + w_\#$, $k = p_\# + d_\#$, or $k = w_\# + d_\#$ have running times of the form $2^{O(k \lg \lg k)} \cdot n^{O(1)}$ using the recent ILP-algorithm by Reis and Rothvoss [23]. Can any of these running times be improved to $2^{O(k)} \cdot n$, or can one show a $2^{\Omega(k \lg \lg k)} \cdot n^{O(1)}$ lower-bound?
- Our result shows that $1 \mid\mid \sum w_j U_j$ is $W[1]$-hard with respect to parameters $p_\#$ and $w_\#$, but it does not show that the problem is *in* $W[1]$ for any of these parameters. Is $1 \mid\mid \sum w_j U_j$ contained in $W[t]$ for some $t \geq 1$?

References

1. Muminu O. Adamu and Aderemi O. Adewumi. A survey of single machine scheduling to minimize weighted number of tardy jobs. *Journal of Industrial and Management Optimization*, 10(1):219–241, 2014.
2. Karl Bringmann, Nick Fischer, Danny Hermelin, Dvir Shabtay, and Philip Wellnitz. Faster minimization of tardy processing time on a single machine. *Algorithmica*, 84(5):1341–1356, 2022.
3. Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. *ACM Transactions on Algorithms*, 15(1):14:1–14:25, 2019.
4. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.
5. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
6. Michael R. Fellows, Serge Gaspers, and Frances A. Rosamond. Parameterizing by the number of numbers. *Theory of Computing Systems*, 50(4):675–693, 2012.
7. Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

8 Michel X. Goemans and Thomas Rothvoss. Polynomiality for bin packing with a constant number of item types. *Journal of the ACM*, 67(6):38:1–38:21, 2020. doi:10.1145/3421750.

9 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

10 Klaus Heeger and Danny Hermelin. Minimizing the weighted number of tardy jobs is W[1]-hard. *CoRR*, abs/2401.01740, 2024. doi:10.48550/arXiv.2401.01740.

11 Danny Hermelin, Shlomo Karhi, Michael L. Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, 298(1):271–287, 2021.

12 Danny Hermelin, Matthias Mnich, and Simon Omlor. Single machine batch scheduling to minimize the weighted number of tardy jobs. *CoRR*, abs/1911.12350, 2019.

13 Danny Hermelin, Hendrik Molter, and Dvir Shabtay. Minimizing the weighted number of tardy jobs via $(\max, +)$-convolutions. *INFORMS Journal on Computing - to appear*, 2024.

14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

15 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

16 Kim-Manuel Klein, Adam Polak, and Lars Rohwedder. On minimizing tardy processing time, max-min skewed convolution, and triangular structured ILPs. In *Proc. of the 34th ACM-SIAM Symposium On Discrete Algorithms, SODA 2023*, pages 2947–2960, 2023.

17 Eugene L. Lawler and James M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16(1):77–84, 1969.

18 Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

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

20 James M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15(1):102–109, 1968.

21 Jon M. Peha. Heterogeneous-criteria scheduling: Minimizing weighted number of tardy jobs and weighted completion time. *Computers and Operations Research*, 22(10):1089–1100, 1995.

22 Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 67(4):757–771, 2003. doi:10.1016/S0022-0000(03)00078-3.

23 Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *Proc. of the 64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 974–988, 2023. doi:10.1109/FOCS57990.2023.00060.

24 Sartaj K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

25 Baruch Schieber and Pranav Sitaraman. Quick minimization of tardy processing time on a single machine. In *Proc. of the 18th international Workshop on Algorithms and Data Structures, WADS 2023*, pages 637–643, 2023.