Improved Approximation Algorithms for Non-preemptive Throughput Maximization*

Alexander Armbruster†
alexander.armbruster@tum.de
Technical University of Munich
Munich, Germany

Fabrizio Grandoni‡
fabrizio@idsia.ch
IDSIA, USI-SUPSI
Lugano, Switzerland

Antoine Tinguely§
antoine.tinguely@tum.de
IDSIA, USI-SUPSI
Lugano, Switzerland

Andreas Wiese
andreas.wiese@tum.de
Technical University of Munich
Munich, Germany

Abstract

The (Non-Preemptive) Throughput Maximization problem is a natural and fundamental scheduling problem. We are given n jobs, where each job j is characterized by a processing time and a time window, contained in a global interval $[0, T)$, during which j can be scheduled. Our goal is to schedule the maximum possible number of jobs non-preemptively on a single machine, so that no two scheduled jobs are processed at the same time. This problem is known to be strongly NP-hard. The best-known approximation algorithm for it has an approximation ratio of $1/0.6448 + \varepsilon \approx 1.551 + \varepsilon$ [Im, Li, Moseley IPCO'17], improving on an earlier result in [Chuzhoy, Ostrovsky, Rabani FOCS'01]. In this paper we substantially improve the approximation factor for the problem to $4/3+\varepsilon$ for any constant $\varepsilon > 0$. Using pseudo-polynomial time $(nT)^{O(1)}$, we improve the factor even further to $5/4 + \varepsilon$. Our results extend to the setting in which we are given an arbitrary number of (identical) machines.

CCS Concepts

• **Theory of computation → Scheduling algorithms**; **Approximation algorithms analysis**.

Keywords

Approximation algorithms, scheduling, throughput

ACM Reference Format:
Alexander Armbruster, Fabrizio Grandoni, Antoine Tinguely, and Andreas Wiese. 2026. Improved Approximation Algorithms for Non-preemptive Throughput Maximization. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC '26), June 22–26, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3798129.3800912

*A full version is available at https://arxiv.org/abs/2603.29451.

†Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project 551896423.
‡Partially supported by the Swiss National Science Foundation (SNSF) Grants 200021-200731/1 and 200021-236706.
§Supported by the Swiss National Science Foundation (SNSF) Grant 200021-200731/1.

STOC '26, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-2536-4/2026/06
https://doi.org/10.1145/3798129.3800912

1 Introduction

In this paper we study the *(Non-Preemptive) Throughput Maximization* problem (also known as *Job Interval Scheduling*) which is one of the most basic scheduling problems. We are given a set of n jobs J, where each job $j \in J$ is characterized by its *processing time* $p_j \in \mathbb{N}$, its *release time* $r_j \in \mathbb{N}$, and its *deadline* $d_j \in \mathbb{N}$. For each job $j \in J$ we define its *time window* by $\mathrm{tw}(j) \coloneqq [r_j, d_j)$. The goal is to select a subset $J' \subseteq J$ of the jobs and to compute a non-preemptive schedule for J' on one machine. More formally, we seek to compute a start time $s(j) \in \mathbb{N}$ for each job $j \in J'$ such that $[s(j), s(j)+p_j) \subseteq \mathrm{tw}(j)$, meaning that we execute j during $[s(j), s(j)+p_j)$. We require that for any two distinct jobs $j, j' \in J'$ their intervals $[s(j), s(j)+p_j)$ and $[s(j'), s(j')+p_{j'})$ are disjoint. The objective is to maximize the number of scheduled jobs, i.e., to maximize $|J'|$. Not surprisingly, the problem and its variants and generalization have several applications, see, e.g., [14, 23, 33, 42] and references therein.

Throughput Maximization is (strongly) NP-hard [24, 45] which motivates studying approximation algorithms for it. However, from this point of view, it is not very well understood. The currently best known approximation factor in polynomial time (and even in pseudo-polynomial or quasi-polynomial time) is $1/0.6448 + \varepsilon \approx 1.551 + \varepsilon$ for any constant $\varepsilon > 0$, due to Im, Li and Moseley [36, 37]. This improves a previous result by Chuzhoy, Ostrovisky and Rabani [21, 22] which achieves an approximation ratio of $\frac{e}{e-1}+\varepsilon \approx 1.582+\varepsilon$ (in fact, even for a more general version where for each job we are given an explicit set of possible execution intervals, instead of our implicitly defined intervals of the form $[s(j), s(j)+p_j) \subseteq \mathrm{tw}(j)$) and other previous results [10, 11, 13, 48].

However, Throughput Maximization is *not* known to be APX-hard and, hence, it might still admit a PTAS! We find intriguing that the approximability status of such a basic problem is still rather unclear.

1.1 Our Results and Techniques

In this paper we substantially improve the best known approximation ratio for Throughput Maximization. More specifically, we obtain the following two main results.

Theorem 1.1. *For any constant $\varepsilon > 0$, there is a polynomial-time randomized $(4/3 + \varepsilon)$-approximation algorithm for Throughput Maximization.*

Using pseudo-polynomial time, we can do even better. We define $T := \max_{j \in J} d_j$ and observe that, hence, each job $j \in J$ must be scheduled within the interval $[0, T)$.

Theorem 1.2. *For any constant $\varepsilon > 0$, there is a randomized $(5/4 + \varepsilon)$-approximation algorithm for Throughput Maximization with a running time of $(nT)^{O_\varepsilon(1)}$.*

In the following, we illustrate the main ideas behind our results. Our starting point is the approach by Chuzhoy et al. [22]. The authors present a polynomial-time procedure that partitions the time horizon $[0, T)$ into a collection of at most $\varepsilon|\text{OPT}|$ intervals which we will call *blocks*. The blocks are defined such that there is a $(1 + \varepsilon)$-approximate solution in which

- each job is scheduled entirely within one block and
- inside each block at most $O_\varepsilon(1)$ jobs are scheduled.

Based on the blocks, they define a configuration-LP that has a configuration for each combination of a block B and a set of at most $O_\varepsilon(1)$ jobs that can be scheduled within B; in particular, the LP has a polynomial number of variables and constraints.

Given an optimal fractional solution to the configuration-LP, they sample independently one configuration for each block. It might happen that some job $j \in J$ appears in more than one sampled configuration; in this case, it can still be scheduled only once in the computed solution. However, one can show that if a job $j \in J$ appeared fractionally in y^*_j configurations of the optimal LP-solution, then it appears in at least one sampled configurations with probability at least $\frac{e-1}{e} y^*_j$ which yields the mentioned approximation ratio of $\frac{e}{e-1} + \varepsilon \approx 1.582 + \varepsilon$.

In both of our algorithms, we use a similar configuration-LP, but we define the blocks in a different way and also invoke a different rounding procedure. We start with our $(4/3 + \varepsilon)$-approximation algorithm. Like in the algorithm by Chuzhoy et al. [22], we sample a configuration for each block according to the optimal LP-solution. However, we do not directly assign the jobs according to the sampled configurations. Instead, if a sampled configuration schedules a job j during some time interval $[s(j), s(j) + p_j)$, then we interpret this interval as a *slot* during which we might schedule *some* job j' whose time window and processing time allow to process it completely during $[s(j), s(j) + p_j)$ (possibly $p_{j'} < p_j$ and then the machine would remain idle during some parts of the interval). Then, we use a bipartite matching routine to compute the largest set of jobs that can be assigned in this way to the slots, i.e., each slot gets at most one compatible job assigned to it and each job is assigned to at most one slot. In particular, if a job j appears in two sampled configurations, then it creates two slots such that we can assign j to one of them and potentially another job j' to the other slot. In contrast, in [22] the second slot was kept empty in this case and, hence, it was lost. On a high level, our algorithm might not seem too different from [22]; however, we show that in expectation our resulting matching yields an approximation ratio of only $4/3 + \varepsilon < 1.334 + \varepsilon$.

One key idea for proving this is a more sophisticated definition of our blocks. We ensure that they still have the properties described above. In addition, we define a second partition of $[0, T)$ into *superblocks*, where each superblock is the union of a large (constant) number of consecutive blocks. We show that we can compute such a partition for which there is a $(1 + \varepsilon)$-approximate solution with a set of jobs OPT' in which each job $j \in \text{OPT}'$ is scheduled either (see Figure 1):

- in the leftmost or the rightmost block that $\text{tw}(j)$ intersects; we call these blocks the *boundary blocks* for j, or
- in one of the superblocks S *spanned* by j, i.e., such that $S \subseteq \text{tw}(j)$.

In order to prove that there exists a sufficiently large matching, a particularly simple case is when in the optimal solution to the configuration-LP, each job $j \in J$ appears only in configurations corresponding to its boundary blocks. Then, we can easily show that it appears in at least one sampled block-configuration with probability at least $\frac{3}{4} y^*_j$. Hence, we can we simply match each job j to one of "its own" slots like [22] and obtain an approximation ratio of $4/3 + \varepsilon$.

However, we need to show that we can achieve the same approximation factor also when the jobs might be fractionally scheduled in non-boundary blocks. This is substantially more complex, and one of the main contributions of this paper. At a high level, we show that in expectation there exists a large-enough fractional (bipartite) matching between jobs and slots. Standard matching theory then implies that the maximum matching is also large enough.

In more detail, if a job j belongs to the sampled configuration for at least one of its boundary blocks, say B, we simply integrally match j with the slot created by j in B. The remaining jobs are fractionally matched to slots in the superblocks spanned by them. We need to show that the latter jobs yield sufficiently many fractional edges. To that aim, we use a fractional version of the classical harmonic grouping technique and concentration arguments that critically exploit the fact that each of our superblocks contains many blocks. For each superblock S, we apply harmonic grouping based on the fractional solution to the configuration-LP for its contained blocks. This yields $O(1/\varepsilon)$ groups of jobs where each group contains essentially the same (fractional) number of jobs, and the processing times of the jobs in one group are not larger than the processing times of the next group. Notice that in harmonic grouping one typically forms groups of items of similar cardinality, while we form the groups so that they have a similar *fractional cardinality* in terms of the computed LP-solution. Since the configuration for each of the (many) blocks contained in S were sampled independently, we can show via concentration arguments that, with sufficiently large probability, for each of the $O(1/\varepsilon)$ job groups, the number of sampled slots is essentially the same as the fractional number of jobs in that group (and, hence, also in the next group). When this event happens, we fractionally match the jobs from each job group to the slots of the next job group (with larger processing time). For this it is crucial that for each matched job j its time window $\text{tw}(j)$ contains the entire superblock S; due to this property, we can schedule j arbitrarily within S.

There is a subtle technical issue. In order to achieve the claimed approximation factor, we use that the fractional amount by which we assign a job j to slots in a superblock S spanned by j depends on the sampled configurations for the boundary blocks for j. More specifically, if j belongs to one such configuration, it cannot be also fractionally matched to a slot in a spanned superblock. Hence, these fractional amounts for the different jobs are not independent

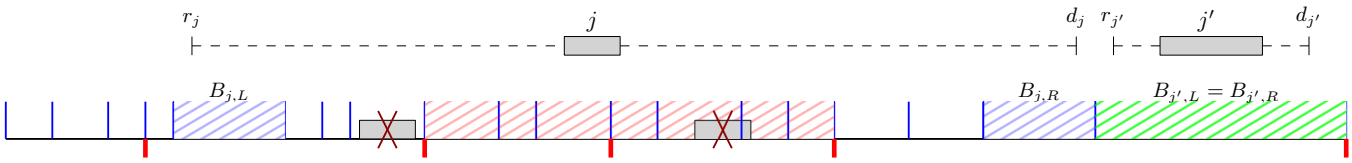

Figure 1: The blue and red lines delimitate the blocks and superblocks, respectively. Job j is global and can be scheduled within its boundary blocks $B_{j,L}$ and $B_{j,R}$ (marked blue) or within one of the two superblocks that it spans (marked red). It cannot be scheduled between a spanned superblock and $B_{j,L}$ or $B_{j,R}$, and cannot intersect two (or more) blocks. Job j' is local. Its (unique) boundary block is marked in green.

random variables, since two jobs might have the same boundary block! Therefore, we cannot use the standard Chernoff's bound to prove that the relevant random variables are sufficiently concentrated around their respective means. However, we are able to show that the impact of such dependencies is sufficiently small. More specifically, we can still use the concentration bounds for *read-k* families of random variables in [26] to obtain the desired properties.

A closer look into our analysis reveals that our approximation ratio is even $1+\varepsilon$ (compared to the optimal LP solution) with respect to the profit of jobs j whose time window $\mathrm{tw}(j)$ is contained in some block; we call such jobs *local*. For the other (*global*) jobs, our approximation ratio is $4/3+\varepsilon$, again compared to the optimal fractional solution. Therefore, we design a second rounding routine that achieves an approximation ratio of $1+\varepsilon$ w.r.t. the global jobs (but does not schedule any local job). The best of the two solutions then has an approximation ratio of only $5/4+\varepsilon$.

For this second algorithm, we need a different block decomposition in which each block contains intuitively $\Theta_\varepsilon(\log T)$ jobs on average from some near-optimal solution. However, then our configuration-LP has $n^{\Theta_\varepsilon(\log T)}$ variables and we can no longer solve it trivially in polynomial time. Therefore, we use the ellipsoid method with a new separation oracle for the dual LP. The separation problem is a *weighted* generalization of our (initial) Throughput Maximization problem, however with the additional constraint that any solution can contain *at most* $\Theta_\varepsilon(\log T)$ jobs. We show how to solve the latter problem in pseudopolynomial time with a suitable dynamic program and the color coding technique [2].

Another major difference in our second algorithm is the LP-rounding procedure. Instead of independently sampling a configuration for each block, we independently assign a job to a block B according to the marginal probabilities induced by the fractional solution. It might be that there is a block B whose assigned jobs cannot all be scheduled within B. However, we show that if we remove $O_\varepsilon(\log T)$ intuitively relatively "long" jobs from each block B, then due to concentration arguments the remaining jobs can (most likely) be scheduled within B. More precisely, we remove a job j if its processing time p_j is relatively large compared to the length of its time window within B, i.e., p_j is large compared to the length of $B \cap \mathrm{tw}(j)$. Since our blocks contain $\Theta_\varepsilon(\log T)$ jobs on average from some near-optimal solution, we can argue that the removed jobs determine only an ε-fraction of the optimal profit.

We remark that a similar random assignment was used by Im, Li and Moseley [37]. However, they had to disallow to schedule a job in its boundary blocks which lost the corresponding profit of the LP. In contrast, since we allow a block to contain up to $O_\varepsilon(\log T)$ jobs and we remove the relatively long jobs, we *can* assign a job to its boundary intervals too and, hence, we obtain a much better approximation ratio of $5/4+\varepsilon$.

Our results extend to the generalization of Throughput Maximization on m (identical) machines (see Appendix 4 for details).

Theorem 1.3. *For any constant $\varepsilon > 0$, there is a polynomial-time randomized $(4/3+\varepsilon)$-approximation algorithm as well as a pseudopolynomial-time randomized $(5/4+\varepsilon)$ approximation algorithm for Throughput Maximization on m machines.*

1.2 Other related Work

The first approximation algorithm for Throughput Maximization was a 2-approximation algorithm due to Spieksma [48]. While the problem is NP-hard in general [24, 45], it can be solved in polynomial time if all the processing times are identical [25] and in pseudo-polynomial time if preemption is allowed [41]. The latter setting also admits an FPTAS [46].

There are natural generalizations of Throughput Maximization to multiple machines and/or to the weighted case in which each job has a weight and we wish to maximize the total weight of scheduled jobs. There is an algorithm by Bar-Noy, Guha, Naor and Schieber [11] that achieves a $\frac{(1+1/m)^m}{(1+1/m)^m-1}$-approximation for any number of machines m (also in the weighted case in pseudo-polynomial time); note that this ratio converges to $\frac{e}{e-1}$ for $m \to \infty$. Essentially the same ratio was obtained in [13] in polynomial time. There is also a $(2+\varepsilon)$-approximation algorithm for the weighted case on multiple machines using the local ratio technique [10]. The algorithm by Chuzhoy et al. [22] yields the ratio of $\frac{e}{e-1}+\varepsilon$ for any number of machines but only in the unweighted case. The algorithm by Im et al. [37] yields the mentioned approximation ratio of $1/0.6448+\varepsilon < 1.551+\varepsilon$ for any number of machines in the unweighted case, and the authors provide also an $(1+\varepsilon)$-approximation for the weighted case if the number of machines m is sufficiently large (for any constant $\varepsilon > 0$). If we allow resource augmentation, i.e., shortening the processing time of each input job by a factor of $1+\varepsilon$, there is even a $(1+\varepsilon)$-approximation in quasi-polynomial time known in the weighted case and for any number of machines [38].

The Unsplittable Flow on a Path with Time Windows problem (UFPTW) is another generalization of Throughput Maximization where each job is in addition specified by a certain *demand* for a shared resource whose capacity might vary over time. In

this problem, multiple jobs can be processed at the same time, provided that their total demand is within the available capacity at that time. Throughput Maximization with m machines is the special case where all the demands are 1, and the available capacity is uniformly m. For UFPTW there is a polynomial time $O(\log n/\log\log n)$-approximation algorithm for the weighted case and an $O(1)$-approximation in the unweighted case [27], improving on [17]. These results hold for a more general setting of explicitly given possible execution intervals and corresponding demands, similar to [22]. Also, there is a quasi-polynomial time $(2+\varepsilon)$-approximation algorithm under resource augmentation for the weighted case [5]. In [5] it is also shown that the problem is APX-hard (even in the unweighted setting); the hardness reduction critically exploits that the tasks can have different demands, and, hence, it does not extend to Throughput Maximization. The special case of the above problem where the length of each time window equals the respective processing time is the classical and very well-studied Unsplittable Flow on a Path problem [3, 7, 8, 12, 15, 28, 30–32] which admits a PTAS [29].

Other scheduling problems on one or multiple machines include for example makespan minimization [34, 39, 40], minimizing the average (weighted) completion times of the given jobs [19], minimizing the average (weighted) job's flowtimes [6], or scheduling jobs with even more general cost functions [4, 9, 20, 35]. We refer to [1, 16, 42, 44] for overviews on the scheduling literature.

2 Polynomial time approximation algorithm

In this section, we present our polynomial time $(4/3+\varepsilon)$-approximation algorithm. We assume w.l.o.g. that $\min_{j\in J} r_j = 0$ and recall that $T = \max_{j\in J} d_j$. Let $\varepsilon > 0$ be a sufficiently small constant and assume that $1/\varepsilon \in \mathbb{N}$. For each $k \in \mathbb{N}$, we define $[k] := \{1,\dots,k\}$.

In Section 2.1, we present a method to partition $[0,T)$ into subintervals such that there exist a $(1+\varepsilon)$-approximate solution that is "aligned" with this partition and satisfies certain additional properties that we will exploit later in our approximation algorithm. Based on this, in Section 2.2 we define a modified configuration LP. In Section 2.3 we present our rounding algorithm, which is then analyzed in Section 2.4.

2.1 Construction of blocks and superblocks

As a first step, we invoke a method from [22] to partition $[0,T)$ into *blocks* where we define a block to be an interval of the form $[a,b)$ for some $a,b \in [0,T]$ with $a < b$. Given a set of jobs $J' \subseteq J$ and a corresponding schedule with a starting time $s(j) \in \{0,\dots,T\}$ for each job $j \in J'$, we say that a job $j \in J'$ is *scheduled in a block* B if $[s(j), s(j)+p_j) \subseteq B$. We remark that the running time bound in [22] holds for a discrete version of the problem and that property (A4) below is not explicitly stated in [22]. Hence for completeness we sketch a proof of the following lemma in the appendix.

Lemma 2.1. *Assume that* $|\mathrm{OPT}| \geq (\frac{6}{\varepsilon})^3$. *In polynomial time we can compute a partition of* $[0,T)$ *into a set of blocks* $\mathcal{B}_0$ *such that there exists a set of jobs* $\mathrm{OPT}'' \subseteq J$ *and a feasible schedule of them with the following properties:*

(A1) $|\mathrm{OPT}''| \geq (1-\varepsilon)|\mathrm{OPT}|$,
(A2) each job $j \in \mathrm{OPT}''$ *is scheduled in some block* $B \in \mathcal{B}_0$,
(A3) for each block $B \in \mathcal{B}_0$ *there are at most* $K_0 = (1/\varepsilon)^{O(1/\varepsilon \log(1/\varepsilon))}$ *jobs of* OPT'' *that are scheduled in* B,
(A4) $|\mathrm{OPT}''| \geq |\mathcal{B}_0|/\varepsilon$.

To avoid ambiguity later, we refer to the blocks from Lemma 2.1 as *elementary blocks*. We remark that property (A4) of Lemma 2.1 implies that the *average* number of jobs in an elementary block is at least $1/\varepsilon$, while by property (A3) the *maximum* number of jobs in each elementary block is bounded by $K_0 = O_\varepsilon(1)$.

As a next step, we want to compute a partition of $[0,T)$ with stronger properties. Instead of just blocks, we want to compute superblocks such that each superblock is the union of a set of consecutive blocks (which will be constantly many blocks below).

Definition 2.2. A pair $(\mathcal{B},\mathcal{S})$ is a *block-superblock partition* if both $\mathcal{B}$ and $\mathcal{S}$ are sets of blocks that form a partition of $[0,T)$ and each block $S \in \mathcal{S}$ is the union of a set of consecutive blocks $\mathcal{B}(S) \subseteq \mathcal{B}$. We call the blocks in $\mathcal{S}$ also *superblocks*.

Intuitively, we use the blocks in $\mathcal{B}_0$ as a basis and, with a suitable shifting argument, glue them together to form the blocks $\mathcal{B}$. We use a parameter Δ that controls how many elementary blocks we glue together here. For our polynomial-time algorithm in this section we will define $\Delta := 1$, while for the pseudopolynomial-time algorithm in the next section we will use $\Delta = O_\varepsilon(\log T)$. Then, we define each superblock as the union of (constantly) many consecutive blocks in $\mathcal{B}$. Thanks to the mentioned shifting argument, we can ensure that there is a $(1+\varepsilon)$-approximate solution in which each job j may be scheduled only within a specific set of blocks and superblocks. In more detail, given a block-superblock partition $(\mathcal{B},\mathcal{S})$, for each job $j \in J$ we define its *release block* $B_{j,L} \in \mathcal{B}$ such that $r_j \in B_{j,L}$ and its *deadline block* $B_{j,R} = [s,t) \in \mathcal{B}$ such that $d_j \in (s,t]$. The release block and the deadline block of j are also called its *boundary* blocks. We remark that possibly $B_{j,L} = B_{j,R} = B$, in which case $\mathrm{tw}(j) \subseteq B$. Furthermore, we say that j *spans* a block $B \in \mathcal{B}$ or a superblock $S \in \mathcal{S}$ (and B or S are *spanned* by j) if $B \subseteq \mathrm{tw}(j)$ or $S \subseteq \mathrm{tw}(j)$, resp. In the following lemma we use the value K_0 as defined in Lemma 2.1.

Lemma 2.3. *Let* $\Delta \in \mathbb{N}$ *be a given parameter, and assume that* $|\mathrm{OPT}| \geq 5K_0\Delta(2K_0/\varepsilon^5)^{1/\varepsilon}$. *In time* $\Delta^{O(1)}n^{O_\varepsilon(1)}$ *we can compute at most* $1/\varepsilon$ *block-superblock partitions and for each one of them a value* K *with* $K \leq \Delta(2K_0/\varepsilon^5)^{1/\varepsilon} = O_\varepsilon(\Delta)$ *such that for at least one computed partition* $(\mathcal{B},\mathcal{S})$ *there exists a set of jobs* $\mathrm{OPT}' \subseteq J$ *and a feasible schedule of them with the following properties:*

(B1) $|\mathrm{OPT}'| \geq (1-2\varepsilon)|\mathrm{OPT}|$,
(B2) each job $j \in \mathrm{OPT}'$ *is scheduled in* $B_{j,L}$, *or in* $B_{j,R}$, *or in a block* $B \in \mathcal{B}$ *for which* j *spans the superblock* $S \in \mathcal{S}$ *containing* B,
(B3) for each block $B \in \mathcal{B}$ *there are at most* K *jobs from* OPT' *that are scheduled in* B,
(B4) $|\mathrm{OPT}'| \geq \frac{\Delta}{\varepsilon}|\mathcal{B}|$ *and* $|\mathrm{OPT}'| \geq \frac{K}{\varepsilon^6}|\mathcal{S}|$.

In the rest of this section, we will assume $\Delta = 1$ and w.l.o.g. $|\mathrm{OPT}| \geq 5K_0\Delta(2K_0/\varepsilon^5)^{1/\varepsilon}$, since otherwise the problem can be solved exactly in polynomial time by enumeration. Therefore, we can apply Lemma 2.3. Intuitively, among the $1/\varepsilon$ computed block-superblock partitions, we guess one for which properties (B1)-(B4) hold. Formally, we execute the following steps for each of them and finally output the best obtained solution. Hence, in the analysis

we may assume that we know the partition $(\mathcal{B}, \mathcal{S})$, together with its corresponding value K, for which the described solution OPT′ exists.

2.2 The linear program

We define a configuration-LP which intuitively computes a (fractional) solution that satisfies (B2) and (B3). A configuration C is specified by a pair (B_C, J_C), where $B_C = [s,t)$ is a block and J_C is a subset of jobs that can be feasibly scheduled inside B_C according to Lemma 2.3; formally, we require that $|J_C| \leq K$ and that, for each $j \in J_C$, the block B_C is either a boundary block for j or B_C is contained in a superblock S spanned by j (i.e., $S \subseteq \mathrm{tw}(j)$). We denote by $s_C : J_C \to \{s, \dots, t-1\}$ an arbitrary but fixed feasible schedule of J_C inside B_C. For each block B we define the set $\mathcal{C}(B)$ to be the set of all configurations for B and we set $\mathcal{C} \coloneqq \bigcup_{B \in \mathcal{B}} \mathcal{C}(B)$. In our LP, for each configuration $C \in \mathcal{C}$ we introduce a variable x_C representing whether we select C for its corresponding block. Then, we introduce constraints to model that we schedule each job at most once and we select one configuration for each block.

$$\max \sum_{C \in \mathcal{C}} |J_C| \cdot x_C \quad \text{s.t.} \quad \sum_{C \in \mathcal{C}: j \in J_C} x_C \leq 1 \quad \forall j \in J$$
$$\sum_{C \in \mathcal{C}(B)} x_C = 1 \quad \forall B \in \mathcal{B} \qquad \text{(LP)}$$
$$x_C \geq 0 \quad \forall C \in \mathcal{C}.$$

Due to Lemma 2.3, the optimal objective function value of (LP) is at least close to |OPT|.

Lemma 2.4. *Let $(\mathcal{B}, \mathcal{S})$ be a block-superblock partition satisfying properties (B1)-(B4) of Lemma 2.3. Then the optimal objective function value of the associated configuration LP is at least $(1-2\varepsilon)|\mathrm{OPT}|$.*

Obviously, we can solve the above LP in polynomial time when K, i.e., the number of jobs per configuration, is upper bounded by a constant.

Lemma 2.5. *Let K denote the maximum number of jobs in a configuration. If $K = O_\varepsilon(1)$, then in polynomial time one can compute an optimal solution $(x^*_C)_{C \in \mathcal{C}}$ to the configuration LP together with a feasible schedule s_C for each configuration $C \in \mathcal{C}$.*

2.3 Rounding algorithm

Let $(x^*_C)_{C \in \mathcal{C}}$ denote an optimal solution to (LP). We describe now how to round it to an integral solution using randomized rounding, losing at most a factor of $4/3 + O(\varepsilon)$ in the profit. For each block $B \in \mathcal{B}$ independently, we sample one configuration $C^*(B) \in \mathcal{C}(B)$ with respect to the distribution given by the vector $(x^*_C)_{C \in \mathcal{C}(B)}$. More precisely, each configuration $C \in \mathcal{C}(B)$ is sampled with probability x^*_C and, deterministically, exactly one configuration in $\mathcal{C}(B)$ is sampled.

For each block $B \in \mathcal{B}$ and each job $j \in J_{C^*(B)}$, there is an interval $[s_{C^*(B)}(j), s_{C^*(B)}(j) + p_j)$ during which j is scheduled by $s_{C^*(B)}$. We call this interval a *slot* and we define Q to be the set of all slots for all blocks $B \in \mathcal{B}$ and all jobs $j \in J_{C^*(B)}$. Note that the slots in Q are pairwise disjoint. We want to assign a subset of the jobs in J to the slots in Q via a bipartite matching. Possibly, this will assign a job $j \in J$ to a slot $[s,t)$ corresponding to a different job $j' \neq j$.

Formally, we define a bipartite graph $G = (V, E)$ where $V = J \dot\cup Q$ and there is an edge $\{j, [s,t)]\} \in E$ connecting (the vertices corresponding to) a job $j \in J$ and a slot $[s,t) \in Q$ if and only if j can be scheduled during $[s,t)$; the latter conditions holds if and only if $\min\{t, d_j\} - \max\{s, r_j\} \geq p_j$, in which case j can be scheduled during $[\max\{s, r_j\}, \max\{s, r_j\} + p_j)$. We compute a maximum matching $M^* \subseteq E$ in G. For each edge $e = (j, [s,t)) \in M^*$ we select job j and schedule it during $[\max\{s, r_j\}, \max\{s, r_j\} + p_j)$. Finally, we output the resulting solution.

2.4 Analysis

By construction, we schedule $|M^*|$ jobs in total. Hence, it remains to show that M^* is sufficiently large. To formalize this, we call a job $j \in J$ *local* if $\mathrm{tw}(j) \subseteq B$ for some block $B \in \mathcal{B}$ (note that then $B_{j,L} = B_{j,R}$) and *global* otherwise. We denote by J_{local} and J_{global} the local and global jobs in J, respectively. We prove that, compared to $(x^*_C)_{C \in \mathcal{C}}$, we lose a factor of $4/3 + O(\varepsilon)$ in the profit of the global jobs and a factor of $1 + O(\varepsilon)$ for the local ones. For convenience, for each job $j \in J$ we define $y^*_j \coloneqq \sum_{C \in \mathcal{C}: j \in J_C} x^*_C$ which is the total fractional extent to which j is selected in $(x^*_C)_{C \in \mathcal{C}}$.

Lemma 2.6. *For any $\Delta \geq 1$ we have*

$$\mathbb{E}[|M^*|] \geq (1 - O(\varepsilon)) \sum_{j \in J_{\mathrm{local}}} y^*_j + (3/4 - O(\varepsilon)) \sum_{j \in J_{\mathrm{global}}} y^*_j$$
$$\geq (3/4 - O(\varepsilon))|\mathrm{OPT}|.$$

In the remainder of this section we prove Lemma 2.6. For this, we construct a feasible fractional bipartite matching, i.e., a function $f : E \to [0,1]$ such that $\sum_{e: v \in e} f(e) \leq 1$ for each $v \in V$. We will prove that its expected size $\mathbb{E}\left[\sum_{e \in E} f(e)\right]$ is at least the lower bound we want to prove for $\mathbb{E}[|M^*|]$. Since the standard LP-relaxation of the bipartite matching problem is integral (see [47]), there exists also an integral matching with at least $\sum_{e \in E} f(e)$ edges.

Consider a job $j \in J$. If the sampled configurations $C^*(B_{j,L})$ for its release block $B_{j,L}$ contains j, then we match j integrally to the corresponding slot. Formally, in this case we denote by $[s,t) \coloneqq [s_{C^*(B_{j,L})}(j), s_{C^*(B_{j,L})}(j) + p_j)$ the slot corresponding to j; we define $f(\{j, [s,t)\}) \coloneqq 1$. If this is not the case but the sampled configurations $C^*(B_{j,R})$ for its deadline block $B_{j,R}$ contains j, then, similarly, we match j to the corresponding slot. We do this operation for each job $j \in J$. Let $J_{\mathrm{bnd}} \subseteq J$ denote the set of all jobs that were (integrally) matched in this way.

We want to define a fractional matching for the remaining jobs $J \setminus J_{\mathrm{bnd}}$. We do this separately for each superblock $S \in \mathcal{S}$. Let $J_S \subseteq J$ denote all jobs in J that span S. Note that this might include jobs in J_{bnd}. We want to define a fractional matching for the jobs in $J_S \setminus J_{\mathrm{bnd}}$. For each job $j \in J_S$ we define the total fractional amount by which j is assigned to S in $(x^*_C)_{C \in \mathcal{C}}$ by $y^*_{j,S} \coloneqq \sum_{B \in \mathcal{B}(S)} \sum_{C \in \mathcal{C}(B): j \in J_C} x^*_C$. Similar to harmonic grouping [43, 50], we partition the jobs in J_S into $1/\varepsilon$ groups $J_1, \dots, J_{1/\varepsilon}$ ordered non-increasingly by their processing times such that the jobs from each group contribute almost the same to the profit of $(x^*_C)_{C \in \mathcal{C}}$ within S.

Lemma 2.7. *There exists a partition $J_1, \dots, J_{1/\varepsilon}$ of J_S such that*

(C1) for each $\ell \in [1/\varepsilon - 1]$, each job $j \in J_\ell$, and each job $j' \in J_{\ell+1}$ we have that $p_j \geq p_{j'}$,

(C2) for each $\ell \in [1/\varepsilon]$ we have that $\varepsilon \cdot \sum_{j \in J_S} y^*_{j,S} - 1 \leq \sum_{j \in J_\ell} y^*_{j,S} \leq 1 + \varepsilon \cdot \sum_{j \in J_S} y^*_{j,S}$.

For each $\ell \in [1/\varepsilon]$ we define $n_\ell := \sum_{j \in J_\ell} y^*_{j,S}$. Note that Lemma 2.7 implies that $n_{\ell'} - 2 \leq n_\ell \leq n_{\ell'} + 2$ for any $\ell, \ell' \in [1/\varepsilon]$. Intuitively, when we define the fractional matching for the jobs in $J_S \setminus J_{\text{bnd}}$ we will ignore the jobs in $J_1 \setminus J_{\text{bnd}}$ and for each $\ell \geq 2$ we will match the jobs in $J_\ell \setminus J_{\text{bnd}}$ fractionally to the slots corresponding to jobs in $J_{\ell-1}$. In particular, here we may use slots that correspond to jobs in J_{bnd} (and to which we have not yet assigned any jobs). We would like that after sampling the configurations for the blocks, for each group J_ℓ we obtain (essentially) n_ℓ slots in S corresponding to jobs in J_ℓ. If n_ℓ is sufficiently large, we can show that this is indeed the case with concentration arguments, using that each superblock contains many blocks and the block's configurations are sampled independently. On the other hand, if n_ℓ is small then we can simply ignore the profit of jobs in J_S within S: indeed, by Lemma 2.3, each superblock contains on average $\frac{K}{\varepsilon^6}$ jobs. Formally, for each $\ell \in [1/\varepsilon]$ we define $\bar{N}_\ell$ to be the (random) number of slots corresponding to jobs in J_ℓ in the sampled configurations $\{C^*(B)\}_{B \in \mathcal{B}: B \subseteq S}$.

Lemma 2.8. *Let $\ell \in [1/\varepsilon]$. With probability at least $1 - \varepsilon^2$ we have that $\bar{N}_\ell \geq (1-\varepsilon) n_\ell - 2K/\varepsilon^3$.*

If the event due to Lemma 2.8 does not happen for some $\ell \in [1/\varepsilon]$, then we simply do not match the jobs in J_S to the slots contained in S in our fractional matching. Since this happens only with probability ε, this influences our expected profit only marginally. Otherwise, for each $\ell \in [1/\varepsilon]$ we have essentially n_ℓ slots available corresponding to the jobs in J_ℓ. Therefore, since $n_{\ell+1} \approx n_\ell$ we can (fractionally) match essentially $n_{\ell+1}$ jobs from $J_{\ell+1}$ to these slots. Recall that $\sum_{j \in J_{\ell+1}} y^*_{j,S} = n_{\ell+1}$. Hence, we could match each job $j \in J_{\ell+1}$ to a fractional extent of $y^*_{j,S}$. Since the jobs in $J_{\ell+1} \cap J_{\text{bnd}}$ are already matched, we can even match each remaining job $j \in J_{\ell+1} \setminus J_{\text{bnd}}$ to a larger fractional extent than only $y^*_{j,S}$.

Let us define this increased extent. For each job $j \in J$, let $y^*_{j,L} := \sum_{C \in \mathcal{C}(B_{j,L}): j \in J_C} x^*_C$ and $y^*_{j,R} := \sum_{C \in \mathcal{C}(B_{j,R}): j \in J_C} x^*_C$, i.e., the probabilities that j is contained in the sampled configuration for $B_{j,L}$ and $B_{j,R}$, resp. Hence, each job $j \in J_{\ell+1}$ is *not* contained in J_{bnd} with probability $(1 - y^*_{j,L})(1 - y^*_{j,R})$. Thus, in expectation, the sum of the values $y^*_{j,S}$ for the (remaining) jobs in $J_{\ell+1} \setminus J_{\text{bnd}}$ equals $\mathbb{E}\left[\sum_{j \in J_{\ell+1} \setminus J_{\text{bnd}}} y^*_{j,S}\right] = \sum_{j \in J_{\ell+1}} y^*_{j,S}(1 - y^*_{j,L})(1 - y^*_{j,R})$. Therefore, we try to match each job $j \in J_{\ell+1} \setminus J_{\text{bnd}}$ even to an increased extent of $\frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})}$. In expectation, the sum of those values is then

$$\mathbb{E}\left[\sum_{j \in J_{\ell+1} \setminus J_{\text{bnd}}} \frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})}\right] = \sum_{j \in J_{\ell+1}} y^*_{j,S} = n_{\ell+1}.$$

Via concentration arguments, we can argue that the sum of those values is also sufficiently concentrated around $n_{\ell+1}$, implying that we can find a matching of size close to n_ℓ in expectation. As mentioned in the introduction, the dependencies among the variables do not allow us to apply the standard Chernoff's bound. However, we are able to show that the impact of such dependencies is sufficiently small, hence we have sufficient concentration. To define our matching formally, let $Q_S \subseteq Q$ denote all slots in the blocks $\mathcal{B}(S)$ corresponding to jobs in J_S and let $E_S \subseteq E$ denote all edges in E that connect a vertex (corresponding to a job) in $J_S \setminus J_{\text{bnd}}$ with a slot in Q_S.

Lemma 2.9. *For each superblock $S \in \mathcal{S}$, there exists a fractional matching $f_S : E_S \to [0,1]$ such that*

- $\mathbb{E}[\sum_{e \in E_S} f_S(e)] \geq (1 - O(\varepsilon)) \sum_{j \in J_S} y^*_{j,S} - 2K/\varepsilon^5$ *and*
- *for each $j \in J_S \setminus J_{\text{bnd}}$ we have*

$$\sum_{q: \{j,q\} \in E_S} f_S(\{j,q\}) \leq \frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})} \leq 1.$$

We explain how to obtain this in more detail in Section 2.5. We combine all these matchings for the superblocks $S \in \mathcal{S}$, together with the values for f that we defined already for the jobs in J_{bnd}. Formally, for each each superblock $S \in \mathcal{S}$ and each edge $e \in E_S$ we define $f(e) := f_S(e)$. For each edge $e' \in E$ for which we have not defined the value $f(e')$ yet, we set $f(e') := 0$.

It remains to argue that $\sum_{e \in E} f(e)$ is sufficiently large in expectation. For each job $j \in J$ we define the extent to which j is matched in f by $g(j) := \sum_{q \in Q: \{j,q\} \in E} f(\{j,q\})$. We can easily show for each (local) job $j \in J_{\text{local}}$ that $g(j) = 1$ with probability y^*_j.

Lemma 2.10. *For each job $j \in J_{\text{local}}$ we have $\Pr[g(j) = 1] = y^*_j$.*

Proof. There is a unique block $B \in \mathcal{B}$ containing $\text{tw}(j)$. Hence, $g(j) = 1$ iff for B we sampled a configuration C containing j. This happens with probability $\sum_{C \in \mathcal{C}(B): j \in J_C} x^*_C = y^*_j$. □

Consider a (global) job $j \in J_{\text{global}}$. At the beginning, we matched j to one slot in $B_{j,L}$ or $B_{j,R}$ with probability $y^*_{j,L} + y^*_{j,R} - y^*_{j,L} \cdot y^*_{j,R}$. On the other hand, the LP-solution obtains a profit of $y^*_{j,L} + y^*_{j,R}$ from assigning j fractionally to $B_{j,L}$ and $B_{j,R}$. However, since $y^*_{j,L} + y^*_{j,R} \leq 1$ the latter profit is by at most a factor of $4/3$ larger than the probability that we matched j to some slot in $B_{j,L}$ or $B_{j,R}$.

Proposition 2.11. *For each job $j \in J$ we have that*

$$y^*_{j,L} + y^*_{j,R} \leq \frac{4}{3}(y^*_{j,L} + y^*_{j,R} - y^*_{j,L} \cdot y^*_{j,R}).$$

Assume now that j is not matched to a slot in $B_{j,L}$ nor in $B_{j,R}$, which happens with probability $(1-y^*_{j,L})(1-y^*_{j,R})$. Roughly speaking, for each superblock $S \in \mathcal{S}$ spanned by j, we match j fractionally to a total extent of $(1 - O(\varepsilon)) \frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})}$ to the blocks in $\mathcal{B}(S)$. More precisely, this holds on average over all the global jobs j spanning each superblock S. Using this, we can prove the following bound for the fractionally matched global jobs.

Lemma 2.12. *For any $\Delta \geq 1$ we have that*
$\mathbb{E}\left[\sum_{j \in J_{\text{global}}} g(j)\right] \geq \frac{3}{4}(1 - O(\varepsilon)) \sum_{j \in J_{\text{global}}} y^*_j - O(\varepsilon)|OPT'|$.

Now Lemma 2.6 can be shown using Lemmas 2.4, 2.10, 2.12, and the integrality of the bipartite matching polytope. Also Theorem 1.1 can be shown by choosing $\Delta = 1$ and combining Lemmas 2.3, 2.5, and 2.6.

2.5 Proof of Lemma 2.9

Let us focus on a specific superblock $S \in \mathcal{S}$. We would like to match a job j to an extend of

$$Y_j := \begin{cases} \frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})} & \text{if } j \in J_S \setminus J_{\text{bnd}} \\ 0 & \text{if } j \in J_{\text{bnd}} \end{cases}$$

Consider the sets J_ℓ as defined in Lemma 2.7, and define $N_\ell := \sum_{j \in J_\ell} Y_j$ for $\ell \in [1/\varepsilon] \setminus \{1\}$. First, we show some technical properties.

Proposition 2.13. *For each $j \in J_S$ we have (deterministically) $Y_j \leq 1$ and for each $\ell \in [1/\varepsilon]$ we have $\mathbb{E}[N_\ell] = n_\ell$.*

As already mentioned, we want to match a job $j \in J_\ell$ to an extend of Y_j to slots corresponding to jobs in $J_{\ell-1}$. As the time window of j spans S and each slot corresponding to a job in $J_{\ell-1}$ has a length of at least p_j due to Lemma 2.7, we can match j to any slot corresponding to any job in $J_{\ell-1}$. The total extend to which we want to match jobs is N_ℓ and the total number of slots is $\bar{N}_{\ell-1}$. As we have a complete bipartite graph between these sets of nodes, the maximum matching has a size of $\min\{N_\ell, \bar{N}_{\ell-1}\}$. Therefore, the major part of the proof is devoted to show that $\min\{N_\ell, \bar{N}_{\ell-1}\}$ is in expectation close to n_ℓ.

In the following paragraph, we give some intuition. By Lemma 2.8 we already know that $\bar{N}_{\ell-1}$ is close to n_ℓ with probability $1 - \varepsilon^2$. So the goal is to obtain a similar bound for N_ℓ. We know that $\mathbb{E}[N_\ell] = n_\ell$ by Proposition 2.13, so we only need some kind of concentration for N_ℓ. Recall that Y_j depends on the sampled configurations for the boundary blocks for j. This dependence also implies that the random variables Y_j are not independent. Hence we cannot directly apply standard Chernoff's bound to obtain the desired concentration for N_ℓ. To our advantage, the correlation between the random variables is still quite low: If two random variables Y_j and $Y_{j'}$ are not independent, then there must be a block which is a boundary block for both j and j'. We make this dependence even weaker. We define a new random variable $\tilde{Y}_j$, which equals Y_j except if there is a boundary block B of j with $\Pr[j \in J_{C^*(B)}] < \varepsilon$ and nevertheless we sample a configuration $C^*(B)$ with $j \in J_{C^*(B)}$. In this case $\tilde{Y}_j$ behaves the same as Y_j behaves when we sample a configuration $C^*(B)$ with $j \notin J_{C^*(B)}$. As this event only occurs with probability ε this sacrifices only an ε fraction of the profit in expectation. The advantage of this is that for a job j, there are now only $2K/\varepsilon$ other jobs j' for which Y_j and $Y_{j'}$ are not independent. Indeed, each such job j' needs to share a boundary block with j, and j' must be scheduled in this boundary block with probability at least ε. Since at most K fractional jobs can be scheduled in a block, there can be at most K/ε jobs fractionally assigned to a block by an amount of at least ε. This is enough independence to obtain the desired concentration for N_ℓ, using the results by [26].

Now we make this formal. First we introduce the concept of read-k families (see [26]), which are defined as follows.

Definition 2.14 (Read-k families [26]). Let $C_1, \dots, C_b$ be independent random variables and let $k \in \mathbb{N}$ and J' be a finite set. For each $j \in J'$, let $A_j \subseteq [b]$ and let $f_j : (C_p)_{p \in A_j} \to [0,1]$. Assume that $|\{j : b' \in A_j\}| \leq k$ for every $b' \in [b]$. Then the random variables $Z_j = f_j\big((C_p)_{p \in A_j}\big)$ for $j \in J'$ are called a *read-k family*.

Intuitively, in our setting the random variables $C_1, \dots, C_b$ are the sampled configurations $C^*(B)$ for the blocks $B \in \mathcal{B}$. For every job $j \in J_S$, the set A_j is a subset of the boundary blocks for j, and Z_j is a random variable *close to* Y_j. For each job $j \in J_S$, we would like to choose A_j as the two boundary blocks for j and Z_j as the random variable Y_j, as the value of Y_j is determined by the sampled configurations for two boundary blocks for j, i.e.$B_{j,L}$ and $B_{j,R}$. But this doesn't result in a read-k-family (for a reasonable value of k) as it might be that all jobs are released within the same block and thus all have the same boundary block and all depend on the same sampled configuration. Therefore, we will define new random variables $\tilde{Y}_j$ with stronger independence properties (which will form the read-k-family) as follows:

$$\tilde{Y}_j := \begin{cases} 0 & \text{if } y^*_{j,L} \geq \varepsilon \text{ and } j \in J_{C^*(B_{j,L})} \\ 0 & \text{if } y^*_{j,R} \geq \varepsilon \text{ and } j \in J_{C^*(B_{j,R})} \\ \frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})} & \text{otherwise} \end{cases}$$

and let $\tilde{N}_\ell := \sum_{j \in J_\ell} \tilde{Y}_j$. Using these random variables, we obtain the following properties.

Lemma 2.15. *The following holds.*

- *For each $\ell \in [1/\varepsilon]$ we have $\tilde{N}_\ell \geq N_\ell$ and $\mathbb{E}[\tilde{N}_\ell - N_\ell] \leq O(\varepsilon) \cdot \sum_{j \in J_\ell} y^*_{j,S}$.*
- *The random variables $\tilde{Y}_j$ for $j \in J_S$ are a read-(K/ε) family.*

As we now have a read-K/ε family of random variables, we can apply [26, Theorem 1.1]. This states that the adjustment of Chernoff's bound also holds for read-k families, when the exponent is divided by k.

Lemma 2.16 ([26]). *Let $Y_1, \dots, Y_r$ be a family of read-k variables taking values in $[0,1]$. Then for any $\delta > 0$ we have*

$$\Pr\Big[\sum_{s=1}^{r} Y_s < (1-\delta)\mathbb{E}\Big[\sum_{s=1}^{r} Y_s\Big]\Big] \leq \exp\left(-\frac{\delta^2}{2 \cdot k} \cdot \mathbb{E}\Big[\sum_{s=1}^{r} Y_s\Big]\right).$$

Now we apply the above concentration bound to $\tilde{N}_\ell$ and obtain the following result.

Lemma 2.17. *Let $\ell \in [1/\varepsilon]$. We have that*

$$\Pr\left[\tilde{N}_\ell \leq (1-\varepsilon) n_\ell - 2K/\varepsilon^4\right] \leq \varepsilon.$$

Now we can prove Lemma 2.9.

Proof of Lemma 2.9. For each $\ell \in [1/\varepsilon] \setminus \{1\}$, let $f^\ell : E_S \to [0,1]$ denote a maximal fractional matching between the jobs in J_ℓ and slots corresponding to jobs in $J_{\ell-1}$, where each job $j \in J_\ell$ is can be matched only to an extend of Y_j, i.e., which fulfills:

- For each $j \in J_S \setminus J_\ell$ and each $q \in Q$ we have $f^\ell(\{j,q\}) = 0$.
- For each $j \in J_\ell$ and each slot $q \in Q_S$ not corresponding to a job in $J_{\ell-1}$ we also have $f^\ell(\{j,q\}) = 0$.
- For each $j \in J^\ell$ we have $\sum_{q, \in Q_S} f^\ell(\{j,q\}) \leq Y_j$.

Let $f_S(e) := \sum_{\ell=2}^{1/\varepsilon} f_\ell(e)$ denote the union of these fractional matchings. As shown before, we have $\sum_{e \in E_S} f_\ell(e) = \min\{N_\ell, \bar{N}_{\ell-1}\}$. For each $j \in J_{\text{bnd}}$ we have that $Y_j = 0$ and thus j is not matched at all. And for each $j \in J_S \setminus J_{\text{bnd}}$ there exists at most one $\ell \in [1/\varepsilon]$ with $j \in J_\ell$ and thus j is matched to an extend of at most $\frac{y^*_{j,S}}{(1-y^*_{j,L})(1-y^*_{j,R})}$.

It remains to compute the expected size of the matching. By construction, we have that $\sum_{e\in E} f_S(e) = \sum_{\ell=2}^{1/\varepsilon} \min\{N_\ell, \bar{N}_{\ell-1}\}$. Let $\ell \in [1/\varepsilon] \setminus \{1\}$. By Lemma 2.8, we have that $\bar{N}_{\ell-1} \geq (1-\varepsilon)n_{\ell-1} - 2K/\varepsilon^3$ holds with probability at least $1-\varepsilon^2$. And by Lemma 2.17, we have that $\tilde{N}_\ell \geq (1-\varepsilon)n_\ell - 2K/\varepsilon^4$ holds also with probability at least $1-\varepsilon$. Thus with probability at least $1-2\varepsilon$ both events occur. Thus we have with probability $1-2\varepsilon$ that

$$\begin{aligned}\min\{\tilde{N}_\ell, \bar{N}_{\ell-1}\} &\geq (1-\varepsilon)\min\{n_\ell, n_{\ell-1}\} - \max\{2K/\varepsilon^4, 2K/\varepsilon^3\}\\ &= (1-\varepsilon)\min\{n_\ell, n_{\ell-1}\} - 2K/\varepsilon^4\end{aligned}$$

Recall that $n_\ell \geq \varepsilon \sum_{j\in J_S} y^*_{j,S} - 1, n_{\ell-1} \geq \varepsilon \sum_{j\in J_S} y^*_{j,S} - 1$ by Lemma 2.7. Thus we obtain

$$\begin{aligned}&\mathbb{E}[\min\{\tilde{N}_\ell, \bar{N}_{\ell-1}\}]\\ &\geq \Pr\left[\min\{\tilde{N}_\ell, \bar{N}_{\ell-1}\} \geq (1-\varepsilon)\min\{n_\ell, n_{\ell-1}\} - 2K/\varepsilon^4\right]\\ &\cdot \left((1-\varepsilon)\min\{n_\ell, n_{\ell-1}\} - 2K/\varepsilon^4\}\right)\\ &\geq (1-2\varepsilon)(1-\varepsilon)(\varepsilon \sum_{j\in J_S} y^*_{j,S} - 1) - (1-2\varepsilon)\cdot 2K/\varepsilon^4\\ &\geq (1-3\varepsilon)\cdot\varepsilon\cdot \sum_{j\in J_S} y^*_{j,S} - 2K/\varepsilon^4.\end{aligned}$$

This implies

$$\begin{aligned}&\mathbb{E}[\sum_{e\in E} f_S(e)]\\ &= \sum_{\ell=2}^{1/\varepsilon} \mathbb{E}[\min\{N_\ell, \bar{N}_{\ell-1}\}]\\ &\geq \sum_{\ell=2}^{1/\varepsilon} \left(\mathbb{E}[\min\{\tilde{N}_\ell, \bar{N}_{\ell-1}\}] - \mathbb{E}[\tilde{N}_\ell - N_\ell]\right)\\ &\overset{\text{Lem. 2.15}}{\geq} (1-3\varepsilon)(1-\varepsilon)\sum_{j\in J_S} y^*_{j,S} - 2K/\varepsilon^5 - O(\varepsilon)\sum_{j\in J_S} y^*_{j,S}\\ &\geq (1-O(\varepsilon))\sum_{j\in J_S} y^*_{j,S} - 2K/\varepsilon^5,\end{aligned}$$

which completes the proof. □

3 A pseudopolynomial time $(5/4+\varepsilon)$-approximation

In this section we show how to improve our approximation ratio to $5/4+\varepsilon$, using pseudopolynomial running time. Lemma 2.6 states that in our previous algorithm, we lose a factor of $4/3+O(\varepsilon)$ on the profit of the global jobs but only a factor of $1+O(\varepsilon)$ on the profit of the local jobs (compared to the optimal LP solution). In this section, we present a different rounding method that loses only a factor of $1+O(\varepsilon)$ on the profit of the global jobs, but does not schedule any local job. Therefore, the best of the two solutions will then yield a $(5/4+O(\varepsilon))$-approximation by a simple computation.

For this section, we define $\Delta := 4(\log_2 T + 1)/\varepsilon^4$. We want to apply Lemma 2.3 and solve the configuration-LP for the resulting block-superblock partition. However, it is not clear how to solve it in (pseudo-)polynomial time since possibly $K = \Theta_\varepsilon(\log T)$ and Lemma 2.5 guarantees polynomial running time only if $K = O_\varepsilon(1)$. Also, it is not clear how to solve instances where $|\mathrm{OPT}| \leq 5\Delta K_0(2K_0/\varepsilon^5)^{1/\varepsilon}$ (see Lemma 2.3) since the latter quantity can be up to $\Theta_\varepsilon(\log T)$. This is discussed in Section 3.1. The alternative rounding algorithm is then presented in Section 3.2.

3.1 Solving the linear program and instances with small optimal solutions

We resolve the above mentioned issues related to our choice of $\Delta = \Theta_\varepsilon(\log T)$ with an algorithmic technique that, intuitively, searches for solutions with only few jobs and computes the best solution of this kind, even in a weighted generalization of our problem. First, we show that we can solve (LP) in time $2^{O(K)}\cdot(nT)^{O(1)}$ in this way. To do this, we invoke the ellipsoid method together with a suitable separation oracle for its dual LP:

$$\begin{aligned}\min \sum_{j\in J}\alpha_j + \sum_{B\in\mathcal{B}}\beta_B \quad \text{s.t.}&\\ \sum_{j\in J_C}\alpha_j + \beta_B \geq |J_C| \quad &\forall B\in\mathcal{B}, C\in\mathcal{C}(B) \quad \text{(dualLP)}\\ \alpha_j, \beta_B \geq 0 \quad &\forall j\in J, B\in\mathcal{B}.\end{aligned}$$

The non-trivial task of the separation oracle is to determine, for a given tentative solution $(\alpha_j, \beta_B)_{j\in J, B\in\mathcal{B}}$ and a given block $B = [s,t) \in \mathcal{B}$, whether there exists a configuration $C \in \mathcal{C}(B)$ with $\sum_{j\in J_C}\alpha_j + \beta_B < |J_C|$. This task is equivalent to computing a configuration $C^* = (B, J^*) \in \mathcal{C}(B)$ of maximum weight, where the weight of each job j is set to $w(j) := 1-\alpha_j$. We show how to solve this task in time $2^{O(K)}\cdot(nT)^{O(1)}$ using the color coding technique [2]. We enumerate over a set of K-colorings of the jobs such that at least one such coloring assigns a distinct color to each job $j \in J^*$. Then the problem reduces to computing a maximum-weight configuration that selects at most one job per color. We solve the latter task via dynamic programming. The idea is to construct a table indexed by a subset $COL \subseteq [K]$ of colors and an integer time $q \in \{s,\dots,t\}$. The associated value corresponds to a maximum-weight subset of jobs, each one with a distinct color from COL, that can be feasibly scheduled within $[s,q)$. Each table entry can be computed in time $O(Kn)$ using previously computed entries for certain other entries. The table entry with $(COL, q) = ([K], t)$ corresponds to the desired solution. With a similar dynamic program, we can optimally solve instances in the unweighted case of Maximum Throughput in time $2^{O(|\mathrm{OPT}|)}(nT)^{O(1)}$ which is pseudopolynomial as long as $|\mathrm{OPT}| = O_\varepsilon(\log T)$.

Lemma 3.1. *In time $2^{O(K)}\cdot(nT)^{O(1)}$, we can compute an optimal solution $(x^*_C)_{C\in\mathcal{C}}$ to (LP) together with a schedule s_C for each configuration $C\in\mathcal{C}$ with $x^*_C > 0$. Also, Maximum Throughput can be solved exactly in time $2^{O(|\mathrm{OPT}|)}(nT)^{O(1)}$.*

3.2 Alternative rounding algorithm

Due to Lemma 3.1, we can optimally solve all instances for which $|\mathrm{OPT}| < 5K_0\Delta(2K_0/\varepsilon^5)^{1/\varepsilon}$. If this is not the case, we apply Lemma 2.3 with our defined value of Δ to compute a block-superblock partition satisfying properties (B1)-(B4) from Lemma 2.3. Based on that we define (LP). Via Lemma 3.1, we compute an optimal solution $(x^*_C)_{C\in\mathcal{C}}$ for it in time $(nT)^{O_\varepsilon(1)}$, using that $K = O_\varepsilon(\log T)$. In the following, we present now an alternative LP-rounding algorithm

which is complementary to the rounding algorithm from Section 2.3. In the latter one, we sampled a configuration for each block and then we assigned the jobs to the blocks. Instead, now we randomly assign each job to a block and then try to find a schedule within each block, possibly by discarding some of its assigned jobs.

We ignore all the (local) jobs J_{local} and do not schedule any of them. For each block B and job $j \in J_{\text{global}}$, let $y^*_{j,B} := \sum_{C \in \mathcal{C}(B): j \in J_C} x^*_C$ be the fractional amount by which j is assigned to block B in the optimal LP solution. For each job $j \in J_{\text{global}}$ independently, we do the following. We randomly decide to assign j to some block $B \in \mathcal{B}$ or to discard j such that j is assigned to each block $B \in \mathcal{B}$ with probability $(1-2\varepsilon)y^*_{j,B}$ and, deterministically, j is assigned to at most one block in $\mathcal{B}$. Therefore, j is not assigned to any block and, hence, discarded with probability $1-(1-2\varepsilon)\sum_{B\in\mathcal{B}} y^*_{j,B}$. For each block $B \in \mathcal{B}$ we define a random variable $Y_{j,B} \in \{0,1\}$, modeling whether we assign j to B or not, such that $\Pr[Y_{j,B}=1] = (1-2\varepsilon)y^*_{j,B}$. Notice that deterministically $\sum_{B\in\mathcal{B}} Y_{j,B} \le 1$.

Consider a block $B = [s,t) \in \mathcal{B}$ and the jobs $J(B) \subseteq J_{\text{global}}$ randomly assigned to B, i.e., all jobs j such that $Y_{j,B} = 1$. In a second phase, we discard some of these jobs $J(B)$ such that the remaining jobs can be scheduled within B, similar to randomized rounding with alteration, see, e.g., [18, 49]. Formally, we define a random variable $Z_{j,B} \in \{0,1\}$ for all jobs $j \in J(B)$, where $Z_{j,B} = 0$ indicates that we discard j (if $Y_{j,B} = 0$, simply define $Z_{j,B} = 0$).

First, we discard all jobs j whose processing time p_j is relatively long compared to $\text{tw}(j) \cap B$, i.e., the portion of $\text{tw}(j)$ inside B. Formally, for each interval $I = [s',t')$, we define its length by $|I| := t'-s'$. The set of *long* jobs is given by $J_{\text{long}}(B) := \{j \in J(B) : p_j > \varepsilon^4 |\text{tw}(j) \cap B|\}$. We can show that in expectation $J_{\text{long}}(B)$ contains at most Δ jobs, which we can afford to discard since on average $(x^*_C)_{C\in\mathcal{C}}$ schedules at least Δ/ε jobs in each block.

Lemma 3.2. *For each block $B \in \mathcal{B}$ we have that $\mathbb{E}[|J_{\text{long}}(B)|] \le 4(\log_2 T+1)/\varepsilon^4 = \Delta$.*

Consider next the remaining *short* jobs $J_{\text{short}}(B) := J(B) \setminus J_{\text{long}}(B)$. For these jobs, intuitively, we have useful concentration properties. Let us initially set $Z_{j,B} = 1$ for all $j \in J_{\text{short}}(B)$. For each such j we define one or two (unlikely) bad events: when any one of those events happens, we set $Z_{j,B} = 0$. The first bad event $\mathcal{E}_{total}$ happens if the total processing time of the short jobs assigned to B is larger than $|B|$, i.e.,

$$\sum_{j' \in J_{\text{short}}(B)} p_{j'} = \sum_{j' \in J_{\text{global}}: p_{j'} \le \varepsilon^4 |\text{tw}(j')\cap B|} p_{j'} \cdot Y_{j',B} > |B|.$$

When $\mathcal{E}_{total}$ happens, we set $Z_{j,B} = 0$ for all $j \in J_{\text{short}}(B)$. The event $\mathcal{E}_{total}$ is the unique bad event for the considered jobs that span B, i.e., such that $B \subseteq tw(j)$.

Assume now that some $j \in J_{\text{short}}(B)$ does not span B but that $\text{tw}(j)$ intersects B from the right, i.e., $s < r_j < t$. Then we define the additional bad event $\mathcal{E}_{right}(j)$ that the total processing time of the jobs $j' \in J_{\text{short}}(B)$ with $r_j \le r_{j'}$ is larger than $|\text{tw}(j) \cap B|$. The intuition is that j and all these jobs need to be processed during $\text{tw}(j) \cap B$. Formally, $\mathcal{E}_{right}(j)$ happens if

$$\sum_{j' \in J_{\text{short}}(B): r_j \le r_{j'}} p_{j'} = \sum_{j' \in J_{\text{global}}: p_{j'} \le \varepsilon^4 |\text{tw}(j')\cap B|, r_j \le r_{j'}} p_{j'} \cdot Y_{j,B} > |\text{tw}(j) \cap B|.$$

If $\mathcal{E}_{right}(j)$ happens, we set $Z_{j,B} := 0$. We define an analogous second bad event $\mathcal{E}_{left}(j)$ if j does not span B but $\text{tw}(j)$ intersects B from the left, i.e., $s < d_j < t$, and we set $Z_{j,B}$ accordingly.

We can show that for each job $j \in J_{\text{short}}(B)$ the defined bad events are very unlikely and, hence, if $Y_{j,B} = 1$ then most likely also $Z_{j,B} = 1$.

Lemma 3.3. *For each block $B \in \mathcal{B}$ and each job $j \in J_{\text{global}}$ with $p_j \le \varepsilon^4 |\text{tw}(j) \cap B|$ and $\Pr[Y_{j,B}=1] > 0$, it holds that $\Pr[Z_{j,B} = 1 | Y_{j,B} = 1] \ge 1-2\varepsilon$.*

Let $APX(B) := \{j \in J_{\text{global}} : Z_{j,B} = 1\} \subseteq J(B)$ be all the jobs that we finally assign to B. We show that we can compute a feasible schedule for them in B (with a simple greedy algorithm).

Lemma 3.4. *For each block $B \in \mathcal{B}$ we can compute a schedule for $APX(B)$ in B in polynomial time.*

We apply the procedure above to each block $B \in \mathcal{B}$. Our solution is the set of jobs $APX := \bigcup_{B\in\mathcal{B}} APX(B)$ together with the associated schedule as described above. Combining the above results yields the following lemma.

Lemma 3.5. *In polynomial time we can compute a feasible solution to the given instance whose expected number of jobs is at least*

$$\left((1-2\varepsilon)\sum_{j\in J_{\text{global}}} y^*_j\right) - 4(\log_2 T+1)/\varepsilon^4 \cdot |\mathcal{B}| \ge (1-O(\varepsilon))\sum_{j\in J_{\text{global}}} y^*_j - O(\varepsilon)\sum_{j\in J} y^*_j.$$

Finally, we output the best of the two solutions computed with the algorithm of this section and with the algorithm from Section 2.3 (using the same optimal solution to the configuration LP for $\Delta = 4(\log_2 T+1)/\varepsilon^4$ in both cases). Then, Lemmas 2.6, 3.1, and 3.5 yield Theorem 1.2.

4 Extension to multiple machines

In this section, we describe how we extend our polynomial time $(4/3+\varepsilon)$-approximation and our pseudo-polynomial time $(5/4+\varepsilon)$-approximation algorithms to the setting of multiple (identical) machines.

First, we define the problem formally on multiple machines. In the input, we are given a set of jobs J with given processing times, release times, and deadlines as in the case of one machine. Additionally, we are given a value $m \in \mathbb{N}$ that denotes the given number of (identical) machines. Our goal is again to compute a subset of jobs $J' \subseteq J$; however, now we also need to compute a partition $J' = J'_1 \dot\cup \ldots \dot\cup J'_m$ where for each $i \in [m]$ the set J'_i corresponds to the jobs we assign to machine i. Like before, for each job $j \in J'$ we need to compute a start time $s(j) \in \mathbb{N}$ such that $[s(j), s(j)+p_j) \subseteq \text{tw}(j)$. We require that for each machine $i \in [m]$ and for any two jobs $j, j' \in J'_i$ that $[s(j), s(j)+p_j) \cap [s(j'), s(j')+p_{j'}) = \emptyset$. Note that we do not require this for two jobs assigned to different machines i, i'.

As before, the objective is to maximize $|J'|$. We call the resulting problem Throughput Maximization on m machines.

It was shown in [37] that for each $\varepsilon > 0$ there is a $(1-O(\sqrt{(\log m)/m})-\varepsilon)^{-1}$-approximation algorithm.

Theorem 4.1 ([37]). *For any $\varepsilon > 0$, there exists a polynomial time $(1 - O(\sqrt{(\log m)/m}) - \varepsilon)^{-1}$ approximation for Throughput Maximization on m machines.*

Hence, for each $\varepsilon > 0$ there is a constant $m_\varepsilon \in \mathbb{N}$, $m = O_\varepsilon(1)$, such that the theorem above yields a $(1+O(\varepsilon))$-approximation algorithm if $m > m_\varepsilon$. Therefore, in the following we assume that we are given a constant $\varepsilon > 0$ and an instance of Throughput Maximization on multiple machines in which $m \leq m_\varepsilon$. For this setting, we give a polynomial time $(4/3 + \varepsilon)$-approximation algorithm.

Theorem 4.2. *For any constants $\varepsilon > 0$ and $m \in \mathbb{N}$ with $m = O_\varepsilon(1)$, there is a polynomial-time randomized $(4/3 + \varepsilon)$-approximation algorithm for Throughput Maximization on m machines.*

Similarly, also the pseudo-polynomial time $(5/4+\varepsilon)$-approximation algorithm extends to a constant number of machines.

Theorem 4.3. *For any constant $\varepsilon > 0$, there is a randomized $(5/4 + \varepsilon)$-approximation algorithm for Throughput Maximization on m machines with a running time of $(nT)^{O_{\varepsilon,m}(1)}$.*

References

[1] Alessandro Agnetis, Jean-Charles Billaut, Michael Pinedo, and Dvir Shabtay. 2025. Fifty years of research in scheduling–theory and applications. *European Journal of Operational Research* (2025). doi:10.1016/j.ejor.2025.01.034

[2] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-Coding. *J. ACM* 42, 4 (1995), 844–856. doi:10.1145/210332.210337

[3] A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. 2014. A Mazing 2+ε Approximation for Unsplittable Flow on a Path. In *SODA*. 26–41. doi:10.1145/3242769

[4] Antonios Antoniadis, Ruben Hoeksma, Julie Meißner, José Verschae, and Andreas Wiese. 2017. A QPTAS for the General Scheduling Problem with Identical Release Dates. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 31–1. doi:10.4230/LIPIcs.ICALP.2017.31

[5] Alexander Armbruster, Fabrizio Grandoni, Edin Husić, Antoine Tinguely, and Andreas Wiese. 2025. On the Approximability of Unsplittable Flow on a Path with Time Windows. In *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 29–42. doi:10.1007/978-3-031-93112-3_3

[6] Alexander Armbruster, Lars Rohwedder, and Andreas Wiese. 2023. A PTAS for minimizing weighted flow time on a single machine. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 1335–1344. doi:10.1145/3564246.3585146

[7] N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. 2006. A quasi-PTAS for unsplittable flow on line graphs. In *STOC*. 721–729. doi:10.1145/1132516.1132617

[8] N. Bansal, Z. Friggstad, R. Khandekar, and R. Salavatipour. 2009. A logarithmic approximation for unsplittable flow on line graphs. In *SODA*. 702–709. doi:10.1145/2532645

[9] Nikhil Bansal and Kirk Pruhs. 2014. The Geometry of Scheduling. *SIAM J. Comput.* 43, 5 (2014), 1684–1698. doi:10.1137/130911317

[10] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. 2001. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)* 48, 5 (2001), 1069–1090. doi:10.1145/502102.502107

[11] Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. 2001. Approximating the Throughput of Multiple Machines in Real-Time Scheduling. *SIAM J. Comput.* 31, 2 (2001), 331–352. doi:10.1137/S0097539799354138

[12] J. Batra, N. Garg, A. Kumar, T. Mömke, and A. Wiese. 2015. New Approximation Schemes for Unsplittable Flow on a Path. In *SODA*. 47–58. doi:10.1137/1.9781611973730.5

[13] Piotr Berman and Bhaskar DasGupta. 2000. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization* 4 (2000), 307–323. doi:10.1023/A:1009822211065

[14] Jacek Blazewicz, Klaus H. Ecker, Erwin Pesch, Gunter Schmidt, and Jan Weglarz. 2013. *Scheduling Computer and Manufacturing Processes*. Springer Science & Business Media, Berlin, Heidelberg. doi:10.1007/978-3-662-04363-9

[15] P. Bonsma, J. Schulz, and A. Wiese. 2014. A Constant-Factor Approximation Algorithm for Unsplittable Flow on Paths. *SIAM J. Comput.* 43 (2014), 767–799. doi:10.1137/120868360

[16] Peter Brucker. 2004. *Scheduling algorithms*. Springer. doi:10.1007/978-3-540-69516-5

[17] V. T. Chakaravarthy, A. R. Choudhury, S. Gupta, S. Roy, and Y. Sabharwal. 2014. Improved Algorithms for Resource Allocation under Varying Capacity. In *ESA*. 222–234. doi:10.1007/978-3-662-44777-2_19

[18] Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. 2007. Approximation algorithms for the unsplittable flow problem. *Algorithmica* 47, 1 (2007), 53–78. doi:10.1007/s00453-006-1210-5

[19] Chandra Chekuri, Rajeev Motwani, Balas Natarajan, and Clifford Stein. 2001. Approximation techniques for average completion time scheduling. *SIAM J. Comput.* 31, 1 (2001), 146–166. doi:10.1137/S0097539797327180

[20] Maurice Cheung, Julián Mestre, David B Shmoys, and José Verschae. 2017. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. *SIAM Journal on Discrete Mathematics* 31, 2 (2017), 825–838. doi:10.1137/16M1086819

[21] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. 2001. Approximation Algorithms for the Job Interval Selection Problem and Related Scheduling Problems. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, Las Vegas, Nevada, USA, October 14-17, 2001*. IEEE Computer Society, 348–356. doi:10.1109/SFCS.2001.959909

[22] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. 2006. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research* 31, 4 (2006), 730–738. doi:10.1287/moor.1060.0218

[23] Matteo Fischetti, Silvano Martello, and Paolo Toth. 1987. The Fixed Job Schedule Problem with Spread-Time Constraints. *Operations Research* 35, 6 (1987), 849–858. doi:10.1287/opre.35.6.849

[24] M. R. Garey and D. S. Johnson. 1977. Two-Processor Scheduling with Start-Times and Deadlines. *SIAM J. Comput.* 6, 3 (Sept. 1977), 416–426. doi:10.1137/0206029

[25] M. R. Garey, David S. Johnson, Barbara B. Simons, and Robert Endre Tarjan. 1981. Scheduling Unit-Time Tasks with Arbitrary Release Times and Deadlines. *SIAM J. Comput.* 10, 2 (1981), 256–269. doi:10.1137/0210018

[26] Dmitry Gavinsky, Shachar Lovett, Michael Saks, and Srikanth Srinivasan. 2015. A tail bound for read-k families of functions. *Random Structures & Algorithms* 47, 1 (2015), 99–108. doi:10.1002/rsa.20532

[27] Fabrizio Grandoni, Salvatore Ingala, and Sumedha Uniyal. 2015. Improved approximation algorithms for unsplittable flow on a path with time windows. In *International Workshop on Approximation and Online Algorithms*. Springer, 13–24. doi:10.1007/978-3-319-28684-6_2

[28] F. Grandoni, T. Mömke, and A. Wiese. 2021. Faster (1+ϵ)-Approximation for Unsplittable Flow on a Path via Resource Augmentation and Back. In *ESA*, Vol. 204. 49:1–49:15. doi:10.4230/LIPIcs.ESA.2021.49

[29] F. Grandoni, T. Mömke, and A. Wiese. 2022. A PTAS for unsplittable flow on a path. In *STOC*. ACM, 289–302. doi:10.1145/3519935.3519959

[30] F. Grandoni, T. Mömke, and A. Wiese. 2022. Unsplittable Flow on a Path: The Game!. In *SODA*. SIAM, 906–926. doi:10.1137/1.9781611977073.39

[31] F. Grandoni, T. Mömke, A. Wiese, and H. Zhou. 2017. To Augment or Not to Augment: Solving Unsplittable Flow on a Path by Creating Slack. In *SODA*. 2411–2422. doi:10.1137/1.9781611974782.159

[32] F. Grandoni, T. Mömke, A. Wiese, and H. Zhou. 2018. A (5/3 + ϵ)-approximation for unsplittable flow on a path: placing small tasks into boxes. In *STOC*. 607–619. doi:10.1145/3188745.3188894

[33] Nicholas G. Hall and Michael J. Magazine. 1994. Maximizing the Value of a Space Mission. *European Journal of Operational Research* 78, 2 (1994), 224–241. doi:10.1016/0377-2217(94)90385-9

[34] Dorit S Hochbaum and David B Shmoys. 1987. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)* 34, 1 (1987), 144–162. doi:10.1145/7531.7535

[35] Wiebke Höhn, Julián Mestre, and Andreas Wiese. 2018. How unsplittable-flow-covering helps scheduling with job-dependent cost functions. *Algorithmica* 80, 4 (2018), 1191–1213. doi:10.1007/s00453-017-0300-x

[36] Sungjin Im, Shi Li, and Benjamin Moseley. 2017. Breaking 1 - 1/e Barrier for Non-preemptive Throughput Maximization. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10328)*, Friedrich Eisenbrand and Jochen Könemann (Eds.). Springer, 292–304. doi:10.1007/978-3-319-59250-3_24

[37] Sungjin Im, Shi Li, and Benjamin Moseley. 2020. Breaking 1-1/e barrier for nonpreemptive throughput maximization. *SIAM Journal on Discrete Mathematics* 34, 3 (2020), 1649–1669. doi:10.1137/17M1148438

[38] Sungjin Im, Shi Li, Benjamin Moseley, and Eric Torng. 2015. A dynamic programming framework for non-preemptive scheduling problems on multiple machines. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1070–1086. doi:10.1137/1.9781611973730.72

[39] Klaus Jansen. 2010. An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics* 24, 2 (2010), 457–485. doi:10.1137/090749451

[40] Klaus Jansen, Kim-Manuel Klein, and José Verschae. 2020. Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research* 45, 4 (2020), 1371–1392. doi:10.1287/moor.2019.1036

[41] Eugene L Lawler. 1990. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research* 26, 1 (1990), 125–133. doi:10.1007/BF02248588

[42] Eugene L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and David B. Shmoys. 1993. Sequencing and Scheduling: Algorithms and Complexity. In *Handbooks in Operations Research and Management Science*, S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin (Eds.). Vol. 4. Elsevier, Amsterdam, 445–522. doi:10.1016/S0927-0507(05)80189-6

[43] Chan C Lee and Der-Tsai Lee. 1985. A simple on-line bin-packing algorithm. *Journal of the ACM (JACM)* 32, 3 (1985), 562–572. doi:10.1145/3828.3833

[44] Joseph YT Leung. 2004. *Handbook of scheduling: algorithms, models, and performance analysis.* Chapman and Hall/CRC.

[45] Michael L. Pinedo. 2008. *Scheduling: Theory, Algorithms, and Systems.* Springer. doi:10.1007/978-3-031-05921-6

[46] Kirk Pruhs and Gerhard J Woeginger. 2007. Approximation schemes for a class of subset selection problems. *Theoretical Computer Science* 382, 2 (2007), 151–156. doi:10.1016/j.tcs.2007.03.006

[47] Alexander Schrijver. 2003. *Combinatorial optimization: polyhedra and efficiency.* Vol. 24. Springer.

[48] Frits CR Spieksma. 1999. On the approximability of an interval scheduling problem. *Journal of Scheduling* 2, 5 (1999), 215–227. doi:10.1002/(SICI)1099-1425(199909/10)2:5<215::AID-JOS27>3.0.CO;2-Y

[49] Aravind Srinivasan. 2001. New approaches to covering and packing problems. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms* (Washington, D.C., USA) *(SODA '01)*. Society for Industrial and Applied Mathematics, USA, 567–576.

[50] David P Williamson and David B Shmoys. 2011. *The design of approximation algorithms.* Cambridge University Press.

Received 2025-11-04; accepted 2026-02-01