Approximation Algorithms for Maximum Weighted Throughput on Unrelated Machines

George Karakostas
Department of Computing & Software, McMaster University, Hamilton, Canada

Stavros G. Kolliopoulos[1]
Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece

Abstract

We study the classic weighted maximum throughput problem on unrelated machines. We give a $(1 - 1/e - \varepsilon)$-approximation algorithm for the preemptive case. To our knowledge this is the first ever approximation result for this problem. It is an immediate consequence of a polynomial-time reduction we design, that uses any ρ-approximation algorithm for the single-machine problem to obtain an approximation factor of $(1 - 1/e)\rho - \varepsilon$ for the corresponding unrelated-machines problem, for any $\varepsilon > 0$. On a single machine we present a PTAS for the non-preemptive version of the problem for the special case of a constant number of distinct due dates or distinct release dates. By our reduction this yields an approximation factor of $(1 - 1/e) - \varepsilon$ for the non-preemptive problem on unrelated machines when there is a constant number of distinct due dates or release dates on each machine.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases scheduling, maximum weighted throughput, unrelated machines, approximation algorithm, PTAS

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.5

Category APPROX

Funding *Stavros G. Kolliopoulos*: Travel support was provided by the Faculty of Science of the National and Kapodistrian University of Athens.

1 Introduction

We study the classic scheduling problem of maximizing weighted throughput. We are given a set $\mathcal{J}$ of n jobs, where job j has release date r_j, due date d_j and nonnegative weight w_j. The jobs are to be scheduled on a set $\mathcal{M}$ of m machines so that no two jobs are ever processed simultaneously on the same machine. Job j completes when it has been processed for an amount equal to its processing time. If *preemption* is allowed, the processing of a job may be interrupted and resumed later at no cost. The objective is to maximize the total weight of jobs that complete by their due dates. This basic problem has been studied for decades, but for most of its versions we have no tight approximability results.

We employ the standard 3-field notation $\alpha|\,\beta|\,\gamma$ of [18], where α stands for the machine environment, β for the job characteristics and γ for the objective function. When $\beta = pmtn$ preemption is allowed. When $\alpha = 1$ there is a single machine; $\alpha = P$ stands for m *identical* parallel machines: job j has processing time p_j on every machine i. When $\alpha = R$ the m machines are *unrelated:* job j has processing time p_{ij} on machine i. In this paper we focus on the most general machine environment of unrelated machines. The problem of maximizing weighted throughput on unrelated machines without preemption is denoted by $R|\,r_j|\,\sum w_j\bar{U}_j$.

[1] Corresponding author

We provide a polynomial-time reduction that shows that a ρ-approximation algorithm for the single-machine problem $1|\, r_j,\, \beta|\, \sum w_j \bar{U}_j$ yields a $((1-1/e)\rho - \varepsilon)$-approximation for $R|r_j, \beta|\sum w_j \bar{U}_j$, for any $\varepsilon > 0$ (cf. Theorem 2). For the preemptive version and using different ideas, Kalyanasundaram and Pruhs gave a 6-approximate reduction from $P|r_j, pmtn|\sum w_j \bar{U}_j$ to the single-machine problem [13]. Pruhs and Woeginger [19] designed an FPTAS for $1|\, r_j, pmtn|\, \sum w_j \bar{U}_j$, using a pseudopolynomial algorithm by Lawler [17]. Therefore, our framework produces a $(1-1/e-\varepsilon)$-approximation algorithm for $R|\, r_j,\, pmtn|\, \sum w_j \bar{U}_j$. To our knowledge this is the first approximation result for preemptive throughput maximization on unrelated machines (see also [7]). Our reduction is based on a simple idea. We write a Configuration LP for the parallel-machine setting and use the single-machine algorithm as an approximate separation oracle for the dual. Except for the definition of what constitutes a configuration, the LP and its dual are essentially the same as the ones in [8]. In the latter a Knapsack subroutine was used as a separation oracle. We round the fractional primal LP solution using the dependent rounding method of [21, 9]. Despite its simplicity, our reduction is powerful enough to allow machine-dependent release and due dates: on machine i, job j can be processed in the interval $[r_{ij}, d_{ij}]$. Moreover, for the preemptive case, our algorithm produces a schedule on the unrelated machines with the desirable property of no migrations: every job is processed in its entirety on a single machine. For the non-preemptive case, our approach shows the existence of a $((1-1/e)\rho_* - \varepsilon)$-approximation algorithm, where ρ_* is the approximability threshold of $1|\, r_j|\, \sum w_j \bar{U}_j$. Despite decades of research, the latter threshold is yet to be determined; we only know that $\rho_* \leq 1/2$ [4, 2]. The best known bound for the unweighted case (i.e., $w_j = 1, \forall j$) is 0.6448 [12]. Currently, our reduction does not improve on the best known non-preemptive algorithms for $R|\, r_j|\, \sum w_j \bar{U}_j$, which achieve a ratio of $1/2 - \varepsilon$ [2, 4]. Since the single-machine problem is far from well-understood, special cases have received considerable attention. Our method can translate all existing optimal algorithms or approximation schemes for special cases of the single-machine problem to an $(1-1/e-\varepsilon)$ guarantee on unrelated machines. See Corollary 1 for a list of results.

The unrelated machines setting can be further extended by adding constraints on the grouping of jobs. We provide one such extension. Every job j has a *type* t_j from a finite set T of types. Machines need to undergo preparation to accommodate different job types. For every machine i, we are thus given a set $E_i \subseteq 2^T$ that specifies the allowed combinations of types that may be processed on i in a feasible schedule. Denote this problem as $R|\, r_j$, types $|\, \sum_j w_j \bar{U}_j$. Using a ρ-approximate oracle for $1|\, r_j,\, \beta|\, \sum w_j \bar{U}_j$ we obtain a $((1-1/e)\rho - \varepsilon)$ guarantee for $R|\, r_j,\, \beta$, types $|\, \sum_j w_j \bar{U}_j$ in time polynomial in $\sum_{i \in \mathcal{M}} |E_i|$. To avoid clutter we present in Theorem 2 the result where all jobs have the same type and provide the details for the more general setting in the Appendix.

Our algorithm for $R|\, r_j, pmtn|\, \sum w_j \bar{U}_j$ uses the FPTAS of [17, 19] for the preemptive case on a single machine as a black box. For the single-machine non-preemptive case, the dynamic program of Lawler [17] can produce an FPTAS only for the case of *similarly ordered* release and due dates, i.e., $d_i < d_j$ implies $r_i \leq r_j$ for each pair of jobs i and j. The dynamic program in [17] yields thus an FPTAS for the non-preemptive version when all jobs have a common due date or a common release date. It takes significant effort and new ideas to overcome the restriction of similarly ordered release and due dates. We show how to do this and obtain a PTAS for the single-machine, non-preemptive version, when there is a constant number of distinct due dates (or release dates) (cf. Theorems 8 and 9). At a high level our PTAS combines the FPTAS of [17] with the ideas from the PTAS of Khan et al. [15] for the Generalized Assignment Problem with a constant number of bins. More specifically, our approach can be summarized as follows (we restrict our description to the case of a constant number of distinct due dates; the arguments are symmetric for the constant number of release dates). By guessing the jobs straddling the due dates, we split the schedule into

Table 1 Summary of our main results for weighted throughput on unrelated machines; ρ_* is the approximability threshold of $1|\,r_j|\,\sum w_j \bar{U}_j$.

Preemption	m	# distinct d_j	# distinct r_j	Previous	This work
$\surd$	any	any	any	–	$1-1/e-\varepsilon$ (Cor. 1)
–	any	any	any	$1/2-\varepsilon$ [2, 4]	$(1-1/e)\rho_*-\varepsilon$ (Thm. 2)
–	1	$O(1)$	any	$1/2-\varepsilon$ [2, 4]	PTAS (Thm. 8)
–	1	any	$O(1)$	$1/2-\varepsilon$ [2, 4]	PTAS (Thm. 9)
–	any	$O(1)$	any	$1/2-\varepsilon$ [2, 4]	$1-1/e-\varepsilon$ (Cor. 1)
–	any	any	$O(1)$	$1/2-\varepsilon$ [2, 4]	$1-1/e-\varepsilon$ (Cor. 1)

bins of two different kinds: (i) bins with jobs whose release dates occur before the bin, and (ii) bins with jobs whose release date may occur inside the bin. By partitioning the jobs of $\mathcal{J}$, we split the problem into a constant number of subproblems each defined on a disjoint subset of jobs and where each subproblem contains only one bin of the second kind. This allows us to apply the ideas of [17] on the bin of kind (ii), and the GAP ideas of [15] on the bins of kind (i) *simultaneously*. The combination of all "parallel" subproblems defines the dynamic programming state of our PTAS.

By the aforementioned reduction our PTAS yields a $(1-1/e-\varepsilon)$-approximation on unrelated machines when on every machine the number of distinct due dates or release dates is constant. Even for these special cases, we are not aware of any previous results that improve upon the $1/2-\varepsilon$ ratio of [2, 4]. We summarize our main results in Table 1.

Other related work. The single-machine problem without release dates $1|\,|\,\sum_j w_j \bar{U}_j$ is equivalent in optimality with the problem of minimizing the total weight of late jobs. The latter was one of Karp's 21 NP-complete problems [14]. Without release dates, preemption is of no use; therefore, $1|\,pmtn|\,\sum_j w_j \bar{U}_j$ is also NP-complete. Several versions of the unweighted problem are hard. $1|\,r_j|\,\sum_j \bar{U}_j$ is strongly NP-complete [10], and $R|\,r_j|\,\sum_j \bar{U}_j$ is MAX SNP-hard [3]. $P|\,pmtn|\,\sum \bar{U}_j$ is NP-complete [16]. Chuzhoy et al. [6] provide a $(1-1/e-\varepsilon)$-approximation algorithm for the Job Interval Selection Problem, in which for every job we are given an explicit list of allowed intervals and the intervals selected for all scheduled jobs must be disjoint. The objective is to maximize the number of scheduled jobs. Because of the explicit list requirement, the running time of the algorithm of [6] becomes pseudopolynomial when applied to $R|\,r_j|\,\sum_j \bar{U}_j$. Hyatt-Denesik et al. [11] provide a PTAS that works for unweighted jobs on identical machines, when m, the number of distinct release dates, and the number of distinct due dates, are all bounded by a constant. A number of further results for identical machines are given in [4, 12].

The paper is structured as follows. In Section 2 we provide the reduction from the unrelated machines to the single-machine problem and the improved approximation bounds that are obtained as a consequence. In Section 3 we present the PTAS for the single-machine case when the number of distinct due dates is fixed. In Section 4 we outline the necessary modifications to the algorithm from Section 3 in order to obtain a PTAS for the single-machine when the number of distinct release dates is fixed.

2 Reduction to the single-machine case

In this section we provide a polynomial-time approximation-preserving reduction from the problem with machine-dependent release and due dates, $R|\,r_{ij},\,d_{ij},\,\beta|\,\sum w_j \bar{U}_j$, to $1|\,r_j,\,\beta|\,\sum w_j \bar{U}_j$. We provide first a Configuration LP relaxation for $R|\,r_{ij},\,d_{ij},\,\beta|\,\sum w_j \bar{U}_j$.

For machine $i \in \mathcal{M}$ we define a set $\mathcal{C}(i)$ of configurations. A configuration $C \in \mathcal{C}(i)$ is a schedule of a subset $J \subseteq \mathcal{J}$ of jobs on machine i such that (i) all jobs in J respect their release and due dates on i (ii) there is no unnecessary idle time (iii) the β-constraints are met. $\mathcal{C}(i)$ is a finite set whose size can be naively bounded by $(n!)2^n$. We slightly abuse notation and we also view a configuration C as the set of jobs in the corresponding schedule. Without loss of generality we can assume that $\mathcal{C}(i) \cap \mathcal{C}(j) = \varnothing$ for $i, j \in \mathcal{M}$, $i \neq j$. The configuration LP, here denoted (CLP), is the formulation (1) of [8] and has a variable x_C for each machine i and configuration $C \in \mathcal{C}(i)$:

$$\max \sum_{j\in\mathcal{J}} w_j \left(\sum_{i\in\mathcal{M}} \sum_{C\in\mathcal{C}(i):j\in C} x_C \right) \tag{CLP}$$

$$\sum_{C\in\mathcal{C}(i)} x_C \le 1 \qquad \forall i \in \mathcal{M} \tag{1}$$

$$\sum_{i\in\mathcal{M}} \sum_{C\in\mathcal{C}(i):j\in C} x_C \le 1 \qquad \forall j \in \mathcal{J} \tag{2}$$

$$x_C \ge 0 \qquad \forall i \in \mathcal{M}, \forall C \in \mathcal{C}(i) \tag{3}$$

The set of constraints (1) ensures that to each machine is assigned at most one configuration. Constraints (2) ensure that each job is assigned at most once. Clearly, an integer solution to (CLP) corresponds to a feasible schedule for $R|\,\beta|\,\sum w_j\bar{U}_j$. For a configuration C, let $w(C) := \sum_{j\in C} w_j$. The dual of (CLP) is the following:

$$\min \sum_{i\in\mathcal{M}} y_i + \sum_{j\in\mathcal{J}} z_j \tag{D-CLP}$$

$$y_i + \sum_{j\in C} z_j \ge w(C) \qquad \forall i \in \mathcal{M}, \forall C \in \mathcal{C}(i) \tag{4}$$

$$y, z \ge 0 \tag{5}$$

In what follows, we will use the notion of a *ρ-approximate separation oracle* for (D-CLP) (see e.g., [8]), i.e., a polynomial-time algorithm that, given values y, z, either returns a violated constraint, or guarantees that values $y/\rho, z$ satisfy constraints (4)-(5). It is well-known (cf. Lemma 2.2 in [8]) that such an oracle, combined with binary search on the optimal value, implies a polynomial-time $(\rho - \delta)$-approximate algorithm for solving (D-CLP), and hence (CLP), for any constant $\delta > 0$ (without any constraint violations). The facet complexity ϕ of (D-CLP) is $O(n \log w_{\max})$, where $w_{\max} = \max_{j\in\mathcal{J}} w_j$. Since the coefficients of the objective function are all 1, by well-known arguments the optimal value of (D-CLP) can be represented by $O(n^2\phi)$ bits and hence binary search on the optimum runs in polynomial-time (see Corollary 10.2a in [20]). Our reduction is based on the following simple fact.

▶ **Lemma 1.** *If there is a polynomial-time ρ-approximation algorithm A_β for the problem $1|\,r_j,\,\beta|\,\sum w_j\bar{U}_j$, then there is a ρ-approximate separation oracle for* (D-CLP).

Proof. Given a candidate solution (y, z) to (D-CLP), the separation oracle has to solve $|M|$ instances of $1|\,\beta|\,\sum w_j\bar{U}_j$, one instance I_i for each $i \in \mathcal{M}$.

We define instance I_i. We have a single machine and the n jobs in $\mathcal{J}$ where job j has processing time p_{ij}, release date r_{ij}, due date d_{ij}. Job j has a (possibly negative) *value* $v_j := w_j - z_j$. There is a violated constraint (4) corresponding to machine i iff there is a feasible schedule for I_i where the total value of the on-time jobs exceeds y_i. Any such schedule can contain only jobs of positive value; therefore, we discard any job j with $v_j \leq 0$. If there are no jobs of positive value, then the constraint cannot be violated for any configuration in $\mathcal{C}(i)$ and we move to examine the next instance I_{i+1}.

We run algorithm A_β on instance I_i. A_β will return a feasible schedule whose value is at least ρ-times the maximum value, for $\rho \leq 1$. If this value exceeds y_i we have identified a violated constraint. Else the vectors $(y/\rho, z)$ satisfy all constraints (4) for machine i. ◀

▶ **Theorem 2.** *If there is a polynomial-time ρ-approximation algorithm A_β for the problem $1|\, r_j,\ \beta|\sum w_j\bar{U}_j$, then there is a $((1-1/e)\rho - \varepsilon)$-approximation algorithm for $R|\, r_{ij},\ d_{ij},\ \beta|\sum w_j\bar{U}_j$ where $\varepsilon > 0$ is any constant of our choice.*

Proof. By Lemma 1 and Lemma 2.2 in [8] the ρ-approximation algorithm A_β for $1|\beta|\sum w_j\bar{U}_j$, can be used to compute a $(\rho-\delta)$-approximate solution x^* to (CLP), for some $\delta > 0$. It remains to round the fractional solution x^* to an integer one. We use the dependent rounding method of [21, 9] in the setting where we have a collection of m star graphs, each corresponding to a machine $i \in M$ and the subset of configurations from $C(i)$ that have been assigned by a non-zero amount by x^* to i. We sample independently from m distributions, one for each star, to produce an integer assignment $\hat{x}$ of configurations to the machines. Each distribution and the corresponding sampling method are designed using the technique of [21], so that, with probability 1, each machine gets one configuration and the assignment of configurations to a machine respects negative correlation. Moreover, the expectation of $\hat{x}_C$ equals x^*_C. Adapting slightly the analysis for the maximum coverage problem in [21] we obtain that the expected total weight of the jobs that appear in at least one configuration in the support of $\hat{x}$ is at least $(1-1/e)$ the value of the x^* solution. For the sake of completeness we include in Appendix A the relevant details from [21]. Every configuration chosen by $\hat{x}$ corresponds to a feasible schedule on the corresponding machine. To obtain the final schedule, for every job j that appears in more that one configurations we arbitrarily delete all but one of its occurrences. Note that every machine gets exactly one configuration by $\hat{x}$, therefore the occurrences of j happen each on a different machine. Since x^* is $(\rho-\delta)$-approximate we obtain the theorem. ◀

The upcoming corollary collects applications of Theorem 2 to settings where an (F)PTAS exists for the single machine case. The *additive laxity* of a job j is defined as $d_j - p_j - r_j$. For each application we provide the reference for the single-machine result. Note that the algorithm in [1] finds an optimal solution.

▶ **Corollary 1.** *There is a polynomial-time $(1 - 1/e - \varepsilon)$-approximation algorithm for $R|\, r_{ij},\ d_{ij},\ \beta|\sum_j w_j\bar{U}_j$ where jobs have machine-dependent release and due dates and β can stand for the following:* (i) *preemption is allowed [17, 19]* (ii) *on every machine the number of distinct release dates or the number of distinct due dates is constant (Theorems 8, 9)* (iii) *on every machine the number of distinct processing times is constant and all jobs have unit weights [11]* (iv) *for every machine i and job j, $p_{ij} = p_i$, i.e., all jobs have the same processing time on machine i [1]* (v) *on every machine the jobs have equal additive laxity [5]* (vi) *on every machine release dates and due dates are similarly ordered [17, 19].*

In all the results of Corollary 1, we used as a lower bound for the optimum the value of (CLP), i.e., the value of a (fractional) schedule in which a job may be simultaneously executed on more than one machine. On the other hand the algorithm of Theorem 2 produces an integer solution where every scheduled job runs completely on one machine, i.e., in the preemptive case it produces a schedule with no migrations. The following is immediate.

▶ **Corollary 2.** *The approximation ratio of* $(1-1/e-\varepsilon)$ *for* $R|\, r_{ij},\, d_{ij},\, pmtn|\, \sum_j w_j \bar{U}_j$ *holds both for the problem version where a job may be simultaneously executed on two or more machines and for the version where this is not allowed.*

The above can be extended to the setting where jobs have different types, see Appendix B.

3 Constant number of distinct due dates

In this section we provide a PTAS for the setting where we have a single machine and the number of distinct due dates is constant.

3.1 Preliminaries

For a set of jobs S, we denote by $w(S) = \sum_{j \in S} w_j$ its total weight, $p(S) = \sum_{j \in S} p_j$ its total processing time. Wlog only jobs that meet their due dates are processed in a feasible schedule. Given a schedule σ in which a set S of jobs is processed we denote by $w(\sigma)$ the quantity $w(S)$. For a time interval B we denote its length by $|B|$. The jobs *scheduled in* B are jobs that start and finish in the interval. We slightly abuse notation and denote by $w(B)$ the total weight of the jobs scheduled in B if an associated schedule is clear from the context.

We assume that the due dates of the n jobs take a value out of k possible numbers $D_1 \leq D_2 \leq \ldots \leq D_k$. Let $D_0 := 0$. We assume that time 0 coincides with the earliest release date. Wlog the time horizon $T = D_k$. For an integer m, the notation $m \in (i, k]$ (resp. $m \in [i, k]$) stands for $m = i+1, \ldots, k$ (resp. $m = i, \ldots, k$). Similarly $m \in [i, k)$ is a shorthand for $m = i, \ldots, k-1$. We denote by R_i, $i \in [1, k]$, the set of jobs j s.t. $D_{i-1} \leq r_j < D_i$.

3.2 A structure lemma

In this section we devise structural properties of a near-optimal solution and show how to pre-compute in polynomial time some of its features.

Let OPT be an optimal solution for the problem $1|\, r_j|\, \sum w_j \bar{U}_j$. Wlog $w(OPT) \geq w_{\max}$. Let $\varepsilon_1 > 0$ be any constant. We apply the classic knapsack rounding in order to reduce the job weights within a polynomial range: by rounding all job weights down to the closest multiple of $\theta := \varepsilon_1 w_{\max}/n$, the total weight is still at least $(1-\varepsilon_1)OPT$, but now all job weights are between 1 and $O(n/\varepsilon_1)$ in units of θ. In addition, we can impose the following special structure:

- We shift all scheduled jobs later, to start as late as possible: the schedule consists of contiguous (i.e., without idle time) blocks of jobs, each finishing at a due date (with the last block finishing at D_k).
- In every interval $[D_{i-1}, D_i)$, we can rearrange the jobs that start and finish within the interval to be scheduled in non-decreasing order of release dates. Note that any jobs from the set R_i that are scheduled in the interval, are scheduled last.

A job j *straddles* time t if j starts at or before t and finishes after t. Let $S_1, S_2, \ldots, S_{k-1}$ be the jobs straddling due dates $D_1, \ldots, D_{k-1}$, and let s_i, f_i be the starting and finishing times of straddler S_i. Note that due to the structure of OPT, some due dates may not have a straddler, just like D_0, D_k; in this case $S_i = \emptyset$. Wlog we may assume that no job straddles more than one due date. We will assume that we know the straddlers by enumerating all $O(n^k)$ possibilities.

Let $K_m = (f_{m-1}, s_m)$ be the interval between the end of S_{m-1} and the beginning of S_m, and $|K_m| \leq D_m - D_{m-1}$ its length. K_m is the concatenation of intervals $B_m^1, B_m^2, \ldots, B_m^m$ in this order, so that B_m^h contains only the jobs of R_h scheduled in K_m. We refer to these intervals as *bins*. Note that the jobs in each B_m^h are scheduled al late as possible, and each bin straddles only release dates. Figure 1 shows bins $B_m^1, \ldots, B_m^{m-2}, B_m^{m-1}, B_m^m$ of K_m. The following lemma will allow us to enumerate the bin sizes in polynomial time for the PTAS.

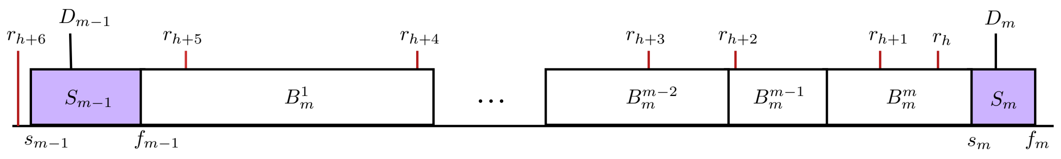

Figure 1 Bins $B_m^1, \ldots, B_m^{m-2}, B_m^{m-1}, B_m^m$ of K_m. Bin B_m^m contains jobs in R_m, i.e., with release dates $r_h, r_{h+1}, \ldots, r_{h+5}$. Jobs with release date r_{h+6} will be scheduled in B_m^{m-1}.

▶ **Lemma 3.** *Let $\varepsilon > 0$ be any given constant. Let be $J \subseteq \mathcal{J}$ be a set of jobs feasibly scheduled in an interval I without idle time where I does not contain a due date. Then there is $\hat{J} \subseteq J$, and a subinterval $\hat{I}$ of I s. t.* (i) *$\hat{I}$ has the same right endpoint as I,* (ii) *$w(\hat{J}) \geq (1-3\varepsilon)w(J)$ and $p(\hat{J}) \leq |\hat{I}| \leq |I|$,* (iii) *the jobs in $\hat{J}$ can be feasibly scheduled without idle time in $\hat{I}$, and* (iv) *$|\hat{I}|$ takes values from a set of size $O(n^{1/\varepsilon^2} \log_{1+\varepsilon} |I|)$.*

Proof. Let σ be the feasible schedule of the jobs of J in I. The set $\hat{J}$ and the corresponding feasible schedule $\hat{\sigma}$ are derived in two phases. We adapt the methods in [15] to account for the presence of release dates in I. In Phase 1, we apply Theorem 11 of [15] for a single bin that corresponds to interval I. We establish the existence of subsets X and Y of J. Define $J' = J - X - Y$. By Theorem 11, for every $j \in J'$, $p_j \leq \varepsilon(|I| - p(X))$. We produce a schedule σ' of $X \cup J'$ by deleting from σ the jobs of Y and replacing them by idle time. Schedule σ' executes in the same interval I as σ. By Theorem 11, $w(X \cup J') \geq (1-\varepsilon)w(J)$.

In Phase 2, we extend the trimming method of [15] to show that there is $R \subseteq J'$ s.t. $w(R) \leq (2\varepsilon/(1-\varepsilon))w(J')$ and there is an integer β s.t. $p(J'-R) \leq (1+\varepsilon)^\beta \leq |I| - p(X)$. See Appendix D.3. We set $\hat{J} = X \cup (J' - R)$. We have that $w(\hat{J}) \geq (1-\varepsilon)(1 - 2\varepsilon/(1-\varepsilon))w(J) = (1-3\varepsilon)w(J)$. The schedule $\hat{\sigma}$ results from σ' by deleting all jobs in R and shifting jobs as needed to the right to remove any idle time. This is feasible, since I doesn't contain any due dates. It suffices to set $\hat{I}$ to an interval that ends at the right endpoint of I such that $|\hat{I}|$ equals $p(\hat{J})$. By Theorem 11, there are $O(n^{1/\varepsilon^2})$ choices for the set X and accordingly that many choices for $p(X)$. The value $p(J'-R)$ can be upper-bounded by a number that takes $O(\log_{1+\varepsilon} |I|)$ many values. Therefore $|\hat{I}|$ takes values from a set of size $O(n^{1/\varepsilon^2} \log_{1+\varepsilon} |I|)$. ◀

We apply Lemma 3 with $\varepsilon := \varepsilon_2$ to the jobs scheduled in OPT in each of the $O(k^2)$ bins B_m^i, $m \in [1,k]$, $i \in [1,m]$. There are sets X_m^i and integers a_m^i such that there is a near-optimal feasible schedule that assigns to each bin B_m^i a set S_m^i of "small" jobs in addition to X_m^i so that $p(S_m^i) \leq (1+\varepsilon_2)^{a_m^i}$. By enumerating all $O(n^{O(k^2/\varepsilon_2^2)}(\log_{1+\varepsilon_2} D_k)^{k^2})$ possibilities, we can assume that we have determined all the X_m^i's and a_m^i's.

Let $\widehat{OPT}$ be the solution of value at least $(1-3\varepsilon_2)(1-\varepsilon_1)w(OPT)$ obtained from OPT by replacing each bin B_m^i by a (smaller) bin $\hat{B}_m^i$ with known size $|\hat{B}_m^i| = p(X_m^i) + (1+\varepsilon_2)^{a_m^i}$. By Lemma 3 $|\hat{B}_m^i| \leq |B_m^i|$. Then K_m has been replaced by a (smaller) $\hat{K}_m$ of size $\hat{K}_m = \sum_{i=1}^m |\hat{B}_m^i|$, and all scheduled jobs are shifted as late as possible, while maintaining $S_1, S_2, \ldots, S_{k-1}$ as straddlers. Figure 2 shows $\widehat{OPT}$.

Expansion of X_m^i. Let $|\hat{B}_m^i|' := |\hat{B}_m^i| - p(X_m^i)$ be the bin space available for jobs other than X_m^i in $\widehat{OPT}$, i.e., job sets J_m^i s.t. $p(J_m^i) \leq |\hat{B}_m^i|'$. In order to be able to apply the techniques of [15] later, we need to keep in $\widehat{OPT}$ only jobs of J_m^i that satisfy a property

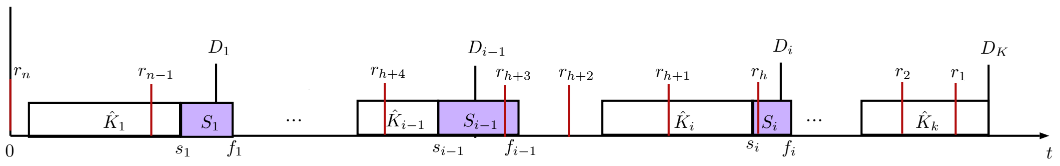

Figure 2 An approximate optimal schedule $\widehat{OPT}$. Note that the length of $\hat{K}_i$ is at most the length of the original K_i. Release dates are in non-increasing order. R_i contains all jobs with release dates $r_h, r_{h+1}, r_{h+2}, r_{h+3}$.

stronger than the property $p_j \leq \varepsilon(|B^i_m| - p(X^i_m))$ guaranteed by Lemma 3 for all jobs of J^i_m. We apply the structure theorem of [15] (Theorem 11 in Appendix D) on each job set $J^i_{i+1}, J^i_{i+2}, \ldots, J^i_k$ for each $i \in [1, k)$, to define job sets W^i_m and U^i_m with $|W^i_m| = O(1/\varepsilon_3^2)$, such that $w(\cup_{i,m} U^i_m) \leq \varepsilon_3 \widehat{OPT}$ and

$$p_j \leq \varepsilon_3(|\hat{B}^i_m|' - p(W^i_m)),\ \forall j \in J^i_m - W^i_m - U^i_m.$$

Note that bins $B^i_i, i = 1, 2, \ldots, k$ do not participate in this process, and, therefore, $W^i_i := \emptyset$. We expand the sets X^i_m to include W^i_m, i.e., we reset $X^i_m := X^i_m \cup W^i_m$ for all i and m. The sets W^i_m can be enumerated in $O(n^{k^2/\varepsilon_3^2})$ time.

▶ **Definition 1.** *A job j is* large *if $j \in \bigcup_{i,m} X^i_m$.*

▶ **Definition 2.** *A job j is* small *for bin $\hat{B}^i_m$ if j is not large and $p_j \leq \varepsilon_3(|\hat{B}^i_m|' - p(W^i_m))$.*

Since $| \cup_{i,m} X^i_m| = O(k^2/(\min\{\varepsilon_2, \varepsilon_3\})^2)$, for the rest of the paper we will assume that we have "guessed" all sets X^i_m of large jobs and their placement in bins. Keeping in $\widehat{OPT}$ only the large and small jobs, as defined in Definitions 1 and 2, $\widehat{OPT}$ still achieves total weight at least $(1 - \varepsilon_3)(1 - 3\varepsilon_2)(1 - \varepsilon_1)OPT$.

Calculation of times s_i, f_i. Since we can enumerate them in polynomial time, in what follows we will assume that we have "guessed" sizes $|\hat{K}_i|$, $i = 1, \ldots, k$. Starting from D_k, and going backwards in time, we use these sizes to calculate times $f_{k-1}, s_{k-1}, f_{k-2}, s_{k-2}, \ldots, f_1, s_1$. We should only be careful in case $s_i - |\hat{K}_i| - p(S_{i-1}) < D_i - D_{i-1}$, i.e., when the beginning of $\hat{K}_i$ is more than $p(S_{i-1})$ time units after D_{i-1}. In this case, we don't allow S_{i-1} to be scheduled after D_{i-1} (as a non-straddler), but we set $s_{i-1} := D_{i-1}$, and continue.

The following lemma summarizes the properties of $\widehat{OPT}$:

▶ **Lemma 4** (Structure Lemma). *There is a feasible schedule $\widehat{OPT}$ of weight at least $(1 - \varepsilon_3)(1 - 3\varepsilon_2)(1 - \varepsilon_1)OPT$, such that:*

- *Job weights are between 1 and $O(n/\varepsilon_1)$.*
- *There are k known straddlers S_i, with known starting and finishing times s_i, f_i, $i \in [1, k]$.*
- *Jobs are scheduled in intervals $\hat{K}_i \subseteq (f_{i-1}, s_i)$ contiguously as late as possible, and in Earliest Release Date (ERD) order. Each $\hat{K}_i$ is adjacent to S_i, and its size $|\hat{K}_i|$ is known.*
- *Only large and small jobs are scheduled; the large jobs can be computed in polynomial-time.*

3.3 A PTAS for $\widehat{OPT}$

We order the jobs in Latest Release Date (LRD) ordering, i.e., in non-increasing release date order $r_1 \geq r_2 \geq \ldots \geq r_n = 0$. The algorithm will schedule the jobs one at a time, and a job scheduled in some bin $\hat{B}^i_m$ of $\hat{K}_i$ is scheduled right *before* the jobs already scheduled in $\hat{B}^i_m$. There are at most k^2 different bins overall. Recall that all job weights are at most $O(n/\varepsilon_1)$, hence there are $O((n^2/\varepsilon_1^2)^{k^2})$ different combinations of total weights for the bins.

3.3.1 Intuition

When there is only one due date D_1, the DP of [17] for non-preemptive maximum throughput is applicable, because its requirement of similarly ordered release and due dates is met. Crucially, the correctness of the algorithm is based on DP subproblems computing the minimum total schedule time needed to achieve a total weight target (cf. Appendix C). Unfortunately, this approach cannot work with more than one due date, because for a given R_i, the notion of minimum total scheduling time is not well-defined for all its bins *simultaneously*.

For a different approach that allows for simultaneous scheduling across all $\hat{K}_m$'s, one may turn to [15], who give a very interesting PTAS for the Generalized Assignment Problem (GAP) with a *constant* number of bins (cf. Appendix D). When discretizing the state space of the dynamic program the algorithm of [15] augments bin capacities by a $(1+\varepsilon)$ factor. It then employs trimming (see Appendix D.2) to remove a low-weight contiguous set of jobs, which exists by the Pigeonhole Principle, so that the remaining jobs fit in the bins with the original capacities restored. In our case, increasing the bin capacities may entail that in some $\hat{K}_m$, jobs may end up scheduled to start *before* their release dates in (f_{m-1}, s_m). (We remind the reader that $f_{m-1} > D_{m-1}$ is the completion time of the $(m-1)$th straddler and $s_m \leq D_m$ is the start time of the mth straddler). Unfortunately, the trimming part of [15] may fail when applied to such a block of jobs: it can only work if the jobs removed are the latest ones in the block so that *all* the previous ones can be shifted later and meet their release date. On the other hand, if all release dates of the jobs in the block occur *before* D_{m-1}, then the whole procedure works as for GPA, without any problems. The two key observations for deriving a PTAS for our problem are the following:

Obs1 We can break the problem into a *constant* number of subproblems that schedule disjoint subsets of jobs: we are going to have one subproblem for each set $R_i, i = 1, 2, \ldots, k$.

Obs2 The subproblem for R_i consists of at most k bins $\hat{B}^i_i, \hat{B}^i_{i+1}, \ldots, \hat{B}^i_k$, each being a "slice" of $\hat{K}_i, \hat{K}_{i+1}, \ldots, \hat{K}_k$ (recall that in each $\hat{K}_m$, jobs in R_g are scheduled before jobs in R_h when $g < h$). Note that the release dates of jobs in R_i do not affect the scheduling of bins $\hat{B}^i_{i+1}, \ldots, \hat{B}^i_k$, since these release dates occur before D_i and all those intervals start after D_i; for these bins the PTAS of [15] for GAP is applicable, when the bin capacities $|\hat{B}^i_{i+1}|, \ldots, |\hat{B}^i_k|$ are given. This leaves bin $\hat{B}^i_i$, which is scheduled using the minimum total processing time idea of [17], while respecting the total processing time and weight targets of the rest of the bins and so that the release dates of R_i are respected.

Obs1 above together with the total processing times, $|\hat{B}^i_i|, \ldots, |\hat{B}^i_k|$, for the bins for each R_i, allows us to combine all "parallel" subproblems in a common DP state, computed by k "parallel" applications of Obs2.

3.3.2 Scheduling of the jobs

Recall that we have "guessed" the large jobs X^i_m that have already been slotted for each bin $\hat{B}^i_m$. We consider jobs one-by-one for scheduling in a Latest Release Date (LRD) ordering. If large and small jobs have the same release date, the large jobs are scheduled first. Recall that Lemma 3 allowed us to "guess" bin sizes $|\hat{B}^i_m|$. Let $|\hat{B}^i_m|' := |\hat{B}^i_m| - p(X^i_m)$ be the bin space available for jobs small for $\hat{B}^i_m$, as defined in Definition 2. Note that given the (guessed) $|\hat{B}^i_m|'$ and W^i_m the algorithm can determine whether a job j is small for bin $\hat{B}^i_m$.

Resource augmentation. We increase the bin capacities allocated to small jobs $|\hat{B}^i_m|'$ in each bin $\hat{B}^i_m, m > i$, and set $|\hat{B}^i_m|'' := (1+\varepsilon_3)|\hat{B}^i_m|'$, and round the job processing times according to the resource augmentation scheme of [15] (cf. Appendix D.1). The bin capacity allocated to the large jobs of the bin is not augmented. Now small jobs may have different processing times for different bins; let p_{mj} be the processing time of a small job $j \in R_i$ for bin $\hat{B}^i_m$ ($p_{mj} = \infty$ if j cannot be scheduled in $\hat{B}^i_m$, e.g., when it is not small for this bin, according to definition 2). Let $\widehat{OPT}_a$ be the maximum throughput schedule for the resource-augmented instance. The following lemma is a direct corollary of Lemma 1 in [15].

▶ **Lemma 5.** *Schedule $\widehat{OPT}$ is feasible for the resource-augmented instance, hence $w(\widehat{OPT}_a) \geq w(\widehat{OPT})$.*

DP states and state transitions. Recall that jobs are scheduled one by one, in LRD order. The DP table records whether a given state is feasible. It is initialized with value ∞, indicating infeasibility, everywhere except for base-case-state $(0, 0, \ldots, 0)$ that is initialized to value 1, indicating feasibility. Table entries that cannot be reached by legitimate state transitions as described below will remain infeasible throughout the execution.

Assume that the first $j-1$ jobs have been considered, and we are now considering job $j \in R_i$. For brevity, we present the portion (substate) $\mathcal{S}^i(j-1)$ of the DP state $\mathcal{S}(j-1) = (\mathcal{S}^1(j-1), \mathcal{S}^2(j-1), \ldots, \mathcal{S}^k(j-1))$ that corresponds to the i-th "parallel" subproblem, defined for jobs in R_i. It has the form of the following tuple:

$$\mathcal{S}^i(j-1) = (Z^{j-1}_i, W^{j-1}_i, Z^{j-1}_{i+1}, W^{j-1}_{i+1}, \ldots, Z^{j-1}_k, W^{j-1}_k),$$

where Z^{j-1}_m is the processing time used by the jobs in $\{1, 2, \ldots, j-1\} \cap R_i$ scheduled in $\hat{B}^i_m$, and W^{j-1}_m is their total weight. Scheduling job $j \in R_i$ produces the following transitions $\mathcal{S}(j-1) \to \mathcal{S}(j)$:

- Large job $j \in X^i_m$ is scheduled in $\hat{B}^i_m$, $m \geq i$: There is a transition to state $\mathcal{S}(j)$ with

$$\mathcal{S}^i(j) = (Z^{j-1}_i, W^{j-1}_i, \ldots, Z^{j-1}_m + p_{mj}, W^{j-1}_m + w_j, \ldots, Z^{j-1}_k, W^{j-1}_k).$$

- Job j is scheduled as small in a $\hat{B}^i_m$, $m > i$: We check whether the following conditions hold: (i) d_j is not violated, (ii) large jobs in X^i_m with release dates smaller than r_j can still be feasibly scheduled in $\hat{B}^i_m$, (iii) $Z^{j-1}_m + p_{mj} \leq |\hat{B}^i_m|''$, and (iv) there isn't already a feasible state $\mathcal{T}(j-1)$ with

$$\mathcal{T}^i(j-1) = (T, W^{j-1}_i, \ldots, Z^{j-1}_m + p_{mj}, W^{j-1}_m + w_j, \ldots, Z^{j-1}_k, W^{j-1}_k)$$

such that $T < Z^{j-1}_i$. Note that large jobs with release dates at least r_j have already been scheduled before j, so given Z^i_m it is easy to check condition (iii) above. If all conditions hold, then there is a transition to state $\mathcal{S}(j)$ with

$$\mathcal{S}^i(j) = (Z^{j-1}_i, W^{j-1}_i, \ldots, Z^{j-1}_m + p_{mj}, W^{j-1}_m + w_j, \ldots, Z^{j-1}_k, W^{j-1}_k),$$

otherwise j is not scheduled in $\hat{B}^i_m$.

- Job j is scheduled as small in $\hat{B}^i_i$: We check whether the following conditions hold: (i) large jobs in X^i_i with release dates smaller than r_j can still be feasibly scheduled in LRD order in $\hat{B}^i_i$, (ii) $Z^{j-1}_i + p_{ij} \leq |\hat{B}^i_i|''$, and (iii) the scheduling of j doesn't violate r_j. If all conditions hold, then we consider the state $\mathcal{T}(j-1)$ with

$$\mathcal{T}^i(j-1) = (Z, W^{j-1}_i + w_j, Z^{j-1}_{i+1}, W^{j-1}_{i+1}, \ldots, Z^{j-1}_k, W^{j-1}_k)$$

that, after scheduling the $j-1$ first jobs, achieves total weight $W^{j-1}_i + w_j$ in $\hat{B}^i_i$, while the rest of the state is the same as $\mathcal{S}(j-1)$. If state $\mathcal{T}(j-1)$ is not feasible, then we schedule

j in $\hat{B}_i^i$ as in the previous case. Otherwise, Z is well defined as a sum of processing times. Then there is a transition to state $\mathcal{S}(j)$ with

$$\mathcal{S}^i(j) = \begin{cases} (Z_i^{j-1} + p_{ij}, W_i^{j-1} + w_j, Z_{i+1}^{j-1}, W_{i+1}^{j-1}, \ldots, Z_k^{j-1}, W_k^{j-1}), & Z_i^{j-1} + p_{ij} < Z \\ \mathcal{T}^i(j-1), & Z_i^{j-1} + p_{ij} \geq Z. \end{cases} \quad (6)$$

- Job j is not scheduled: In this case $\mathcal{S}(j) = \mathcal{S}(j-1)$.

After the feasibility of all DP states $\mathcal{S}(j) = (\mathcal{S}^1(j), \mathcal{S}^2(j), \ldots, \mathcal{S}^k(j))$ has been computed for all j, we keep as a solution the feasible state $\mathcal{S}(n)$ with the maximum sum of total weights in all bins, as well as the schedule S that achieves it.

▶ **Lemma 6.** *The DP algorithm correctly outputs a feasible schedule S that is as good as $\widehat{OPT}_a$, if the latter exists.*

Proof. First note that $\widehat{OPT}_a$ complies with the structure of Lemma 4. Let $\widehat{OPT}_a(j)$ be the sub-schedule of $\widehat{OPT}_a$ containing only jobs $1, \ldots, j$ in the LRD order. Note that $\widehat{OPT}_a(j)$ continues to comply with the structure of Lemma 4. Let $\bar{Z}_m^i(j), \bar{W}_m^i(j)$ be the total processing time and total weight scheduled in $\hat{B}_m^i$ by $\widehat{OPT}_a(j)$.

▷ Claim 7. For $j = 0, 1, \ldots, n$, there is a feasible state $\mathcal{S}(j)$ with substates

$$\mathcal{S}^i(j) = (Z_i^i(j), W_i^i(j), Z_{i+1}^i(j), W_{i+1}^i(j), \ldots, Z_k^i(j), W_k^i(j)),\ i = 1, \ldots, k$$

corresponding to jobs in R_i, produced by the DP after considering jobs $1, 2, \ldots, j$, such that

$$Z_m^i(j) = \bar{Z}_m^i(j), W_m^i(j) = \bar{W}_m^i(j),\ \forall i, m : m > i, \ \text{ and } \ Z_i^i(j) \leq \bar{Z}_i^i(j), W_i^i(j) = \bar{W}_i^i(j),\ \forall i.$$

Proof. The proof is by induction on j. For the base case $j = 0$, the feasible DP state $\mathcal{S}(0) = (0, 0, \ldots, 0)$ proves the claim trivially true. We assume that the claim is true up to job $j-1$, and we consider the case $j \in R_i$. Let $\mathcal{S}(j-1)$ be the state which by the inductive hypothesis corresponds to $\widehat{OPT}_a(j-1)$. There are three cases to consider.

If j is not scheduled in the transition from $\widehat{OPT}_a(j-1)$ to $\widehat{OPT}_a(j)$ then setting $\mathcal{S}(j) = \mathcal{S}(j-1)$ satisfies the claim.

If $\widehat{OPT}_a(j)$ is obtained from $\widehat{OPT}_a(j-1)$ by scheduling j in $\hat{B}_m^i$ for some $m > i$, then the transition from the state $\mathcal{S}(j-1)$ to a state $\mathcal{S}(j)$ with the same placement of j as $\widehat{OPT}_a(j-1)$ is feasible, since it is feasible for $\widehat{OPT}_a(j)$, and $\mathcal{S}(j)$ satisfies the claim.

If $\widehat{OPT}_a(j)$ is obtained from $\widehat{OPT}_a(j-1)$ by scheduling j in $\hat{B}_i^i$, then the transition from state $\mathcal{S}(j-1)$ of the inductive hypothesis to a state $\mathcal{S}(j)$ when the same placement of j as in $\widehat{OPT}_a(j-1)$ is tried, can follow one of two possibilities: (i) If there is another feasible state $\mathcal{T}(j-1)$ with Z_i^i value T smaller than $Z_i^i(j-1) + p_j \leq \bar{Z}_i^i(j-1) + p_j$, W_i^i value U equal to $W_i^i(j-1) + w_j$, and all other values equal to the values of $\mathcal{S}(j-1)$, then the transition followed by the DP (bottom branch of (6)) sets $\mathcal{S}(j) = \mathcal{T}(j-1)$ (which means that j is not scheduled in $\mathcal{S}(j)$), and the claim is satisfied. (ii) If there is no such state $\mathcal{T}(j-1)$, then by the inductive hypothesis $Z_i^i(j-1) \leq \bar{Z}_i^i(j-1)$ which implies that $Z_i^i(j-1) + p_j \leq \bar{Z}_i^i(j-1) + p_j$. In addition, the transition of the DP for this case (top branch of (6)) is feasible, since it was feasible for $\widehat{OPT}_a(j-1)$, and the new state $\mathcal{S}(j)$ it produces again satisfies the claim. ◁

The claim proves that there is a path of feasible states of the DP that achieves the same total weight as $\widehat{OPT}_a(j)$ for all j, and, therefore, there is a feasible state $\mathcal{S}(n)$ of total weight equal to $w(\widehat{OPT}_a(n)) = w(\widehat{OPT}_a)$. ◀

Trimming. Schedule S corresponds to scheduling in a resource-augmented set of bins $\hat{B}^i_m, m > i$. Following [15], we apply trimming (cf. Appendix D.2) to each one of them with $\gamma := 1 + \varepsilon_3$, $\delta := \varepsilon_3, \varepsilon := \varepsilon_3$, for a total loss of at most $2\varepsilon_3 \widehat{OPT}_a$ total weight. The resulting schedule is feasible for the original problem without speed-up, and its total weight is at least $(1 - \varepsilon)OPT$, where $\varepsilon \le 1 - (1 - \varepsilon_1)(1 - 3\varepsilon_2)(1 - 3\varepsilon_3)$.

Running time. Let us focus first on the subproblem for a fixed R_i. There are $C := O(n^{O(k/\varepsilon^2)}(\log_{1+\varepsilon} D_k)^k)$ choices for all the large sets and the bin sizes. For each choice we run the DP for resource-augmented bin sizes, with rounded weight values, which gives a total of $O((n^2/\varepsilon^2)^k \cdot C)$ choices for the coordinates, except for the first one, of the substate $\mathcal{S}^i()$. By Claim 7, for every such combination of coordinates we keep one substate. To place job $j \in R_i$ we examine $O(k)$ transitions. Taking into account that there are k sets $R_1, \ldots, R_k$, we obtain a total running time of $O((n \log D_k)^{O(k^2/\varepsilon^2)})$.

▶ **Theorem 8.** *There is a PTAS for the maximum weighted throughput problem on a single machine with a constant number of distinct due dates.*

Theorem 8 is used in part (ii) of Corollary 1.

4 Constant number of distinct release dates

The ideas behind Theorem 8 can easily be applied to provide a PTAS for the case of a constant number of release dates. Let $d_{\max} = \max_{j \in \mathcal{J}} d_j$. The role of due dates is now played by the k different release dates $0 = r_1 \le r_2 \le \ldots \le r_k$, that define k intervals $[r_i, r_{i+1})$, $i \in [1, k-1]$, and $[r_k, d_{\max})$. Note that release dates are now ordered in non-decreasing order. Let S_i be the straddler of r_i, $i \in [2, k]$ and $S_1 = \emptyset$.

After the rounding of the job weights, we consider an optimal schedule OPT where jobs are now shifted as *early* as possible. Intervals $K_m = (f_m, s_{m+1})$ between straddlers S_m, S_{m+1} are defined as before, and Lemma 3 applies to show the existence of a subset of the scheduled jobs of K_m that can now be scheduled in (smaller) intervals $\hat{K}_m$ starting also at f_m and with polynomially many possible sizes. This is ensured by Lemma 3, modified to apply to bins straddling only due dates, and by scheduling jobs within the bins in Earliest Due Date (EDD) order and as early as possible. When empty space is created within a bin, we shift jobs to the left on the time axis as needed so as to eliminate idle time. The only additional change to Lemma 4 is that now jobs within each interval are scheduled in EDD order.

The PTAS description is virtually the same, with the role of R_i played by D_i, the set of jobs with due dates in $[r_i, r_{i+1})$. Bins $\hat{B}^i_m$, $i \in [m, k]$, of $\hat{K}_m$ are defined similarly to Section 3.3, where $\hat{B}^i_m$ contains jobs scheduled in $\hat{K}_m$ with due date in $\hat{K}_i$. These bins appear from left to right in $\hat{K}_m$, i.e., in order of increasing i. The DP transitions are also defined similarly, to always respect (i) the due dates, (ii) the bin capacities, and (iii) the invariant that the jobs from D_i that are scheduled in $\hat{B}^i_i$ occupy the minimum processing time, *given* the processing time target for bins $\hat{B}^i_m$, $m \in [1, i)$, and the total weight targets for bins $\hat{B}^i_m$, $m \in [1, i]$. By the latter invariant, Lemma 6 applies also to this case. Hence we have:

▶ **Theorem 9.** *There is a PTAS for the maximum weighted throughput problem on a single machine with a constant number of distinct release dates.*

Theorem 9 is used in part (ii) of Corollary 1.

References

1 P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *Journal of Scheduling*, 2(6):245–252, 1999.

2 A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

3 A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.*, 31(2):331–352, 2001.

4 P. Berman and B. DasGupta. Improvements in throughout maximization for real-time scheduling. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 680–687. ACM, 2000.

5 M. Böhm, N. Megow, and J. Schlöter. Throughput scheduling with equal additive laxity. *Oper. Res. Lett.*, 50(5):463–469, 2022.

6 J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Math. Oper. Res.*, 31(4):730–738, 2006.

7 F. Eberle, N. Megow, and K. Schewior. Online throughput maximization on unrelated machines: Commitment is no burden. *ACM Trans. Algorithms*, 19(1):10:1–10:25, 2023.

8 L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Math. Oper. Res.*, 36(3):416–431, 2011.

9 R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, 2006.

10 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

11 D. Hyatt-Denesik, M. Rahgoshay, and M. R. Salavatipour. Approximations for throughput maximization. In *31st Int. Symp. on Algorithms and Computation, ISAAC 2020*, volume 181 of *LIPIcs*, pages 11:1–11:17, 2020.

12 S. Im, S. Li, and B. Moseley. Breaking the $1 - 1/e$ barrier for nonpreemptive throughput maximization. *SIAM J. Discret. Math.*, 34(3):1649–1669, 2020.

13 B. Kalyanasundaram and K. Pruhs. Eliminating migration in multi-processor scheduling. *J. Algorithms*, 38(1):2–24, 2001.

14 R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

15 A. Khan, E. Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *CoRR*, abs/2102.05854, 2021. arXiv:2102.05854.

16 E. L. Lawler. Recent results in the theory of machine scheduling. In *Mathematical Programming The State of the Art, XIth Int. Symp. on Math. Programming*, pages 202–234. Springer, 1983.

17 E. L. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Ann. Oper. Res.*, 26(1–4):125–133, 1990.

18 E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Handbooks in Operations Research and Management Science, Vol 4., Logistics of Production and Inventory*, pages 445–522. North Holland, 1993.

19 K. Pruhs and G. J. Woeginger. Approximation schemes for a class of subset selection problems. *Theor. Comput. Sci.*, 382(2):151–156, 2007.

20 A. Schrijver. *Theory of linear and integer programming.* John Wiley and Sons, 1986.

21 A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *42nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 588–597, 2001.

A Technical details for the proof of Theorem 2

We explain how the dependent rounding method of [21, 9] is used in the proof of Theorem 2. The details in this section are a simple application of the work of Srinivasan [21] (see also [9]) and we provide them here for the sake of completeness. Let x^* be a $(\rho - \delta)$-approximate (fractional) solution of (CLP). Without loss of generality we can assume that $\sum_{C \in \mathcal{C}(i)} x^*_C = 1, \quad \forall i \in \mathcal{M}$.

Let $\bar{\mathcal{C}}(i)$ be the configurations in $\mathcal{C}(i)$ with nonzero value in x^*. We define $x^{(i)}$ to be the projection of the x^* vector to the coordinates that correspond to configurations in $\bar{\mathcal{C}}(i)$. Denote $|\bar{\mathcal{C}}(i)|$ by t_i. Srinivasan [21] defines a distribution $D(t_i, x^{(i)})$ over vectors in $\{0,1\}^{t_i}$ such that any vector X sampled from the distribution satisfies the following three properties:

(A1) *(probability preservation)* $\forall C \in \bar{\mathcal{C}}(i), \Pr[X_C = 1] = x^*_C$.

(A2) *(degree preservation)* $\Pr[|\{C \in \bar{\mathcal{C}}(i) \colon X_C = 1\}| = 1] = 1$.

(A3) *(negative correlation)* For all $S \subseteq \bar{\mathcal{C}}(i)$ we have $\Pr[(\bigwedge_{C \in S}(X_C = 0)] \le \prod_{C \in S} \Pr[X_C = 0]$ and $\Pr[(\bigwedge_{C \in S}(X_C = 1)] \le \prod_{C \in S} \Pr[X_C = 1]$.

The existence of the distribution is established algorithmically:

▶ **Theorem 10** ([21]). *Given the vector $x^{(i)}$ there is a linear-time algorithm that generates a sample from distribution $D(t_i, x^{(i)})$.*

The rounding algorithm follows. It takes as input the vector x^*.

Algorithm 1 DependentRounding.

```
For all $i \in \mathcal{M}$, do independently:
    1. Using the algorithm of Theorem 10, sample from $D(t_i; x^{(i)})$ to obtain vector
       $X^{(i)} \in \{0,1\}^{t_i}$. By Property (A2), $X^{(i)}$ has a unique entry equal to 1.
    2. Assign the configuration $C$ that corresponds to the nonzero entry of $X^{(i)}$ to
       machine $i$.
```

For every $i \in \mathcal{M}$ and $C \in \mathcal{C}(i)$ s.t. $X^{(i)} = 1$, we set $\hat{x}_C = 1$. The remaining entries of $\hat{x}$ are set to zero. The rest of the proof simply adapts the analysis of [21] for Maximum-Coverage-type problems. Since the sets $\bar{\mathcal{C}}(i)$ are pairwise disjoint we slightly abuse notation and omit the machine superscript from the vectors $X^{(i)}$. For $j \in \mathcal{J}$, let z_j be the random variable that takes value 1 if job j is assigned by Algorithm DependentRounding to at least one machine and 0 otherwise. Let $\mathcal{C}$ denote $\bigcup_{i \in \mathcal{M}} \mathcal{C}(i)$.

$$\begin{aligned} \Pr[z_j = 1] =& 1 - \Pr[\bigwedge_{C \in \mathcal{C}: C \ni j} (X_C = 0)] \\ \ge& 1 - \prod_{C \in \mathcal{C}: C \ni j} \Pr[X_C = 0] && (7) \\ =& 1 - \prod_{C \in \mathcal{C}: C \ni j} (1 - x^*_C) && (8) \end{aligned}$$

Inequality (7) follows from the negative correlation property (A3), and equality (8) from property (A1). Define $z^*_j := \sum_{C \in \mathcal{C}: C \ni j} x^*_C$. This is the fractional amount by which job j is scheduled, and the objective value of the solution x^* is equal to $\sum_{j \in \mathcal{J}} w_j z^*_j$. Observe that for every j, $z^*_j \le 1$. Using calculus we obtain that the expression in (8) is greater than $(1 - 1/e) \cdot z^*_j$.

B Scheduling with job types

In this section we show how the proofs in Section 2 can be extended to accommodate job types. Only jobs with certain types can be scheduled together on each machine and the actual combination of types affects the job characteristics.

The extended problem is defined as follows. Every job j has a *type* t_j from a finite set T of types. For every machine i, we are given a set $E_i \subseteq 2^T$ that specifies the allowed combinations of types that may be processed on i in a feasible schedule. We assume from now that $\sum_{i \in \mathcal{M}} |E_i|$ is bounded by a polynomial in n and m. For machine i, let $E_i = \{e_i(1), \ldots, e_i(k_i)\}$ be the set of allowed combinations. In a feasible schedule, the set J_i assigned to machine i must meet the following two additional constraints. $\mathsf{C(i)}$: all jobs in J_i must have a type in $e_i(l)$ for some $e_i(l) \in E_i$. We call this type combination $e_i(l)$ *active* for machine i. $\mathsf{C(ii)}$: The jobs in J_i have processing times, release dates, and due dates that depend on the active type $e_i(l)$, and are denoted $p^l_{ij}, r^l_{ij}, d^l_{ij}$ respectively.

We denote by $R|\, r_{ij},\, d_{ij}$, types $|\, \sum_j w_j \bar{U}_j$ the problem of maximizing weighted throughput on unrelated machines under the additional constraints $\mathsf{C(i)}$ and $\mathsf{C(ii)}$. Using a ρ-approximate oracle for $1|\, r_j,\, \beta|\, \sum w_j \bar{U}_j$ we show how to obtain a $((1-1/e)\rho - \varepsilon)$ guarantee for $R|\, r_{ij},\, d_{ij}$, β, types $|\, \sum_j w_j \bar{U}_j$. We outline only the necessary modifications in the proofs of Section 2.

The definition of a configuration is now as follows. A configuration $C \in \mathcal{C}(i)$ is a schedule of a subset $J \subseteq \mathcal{J}$ of jobs on machine i such that (i) all jobs in J have types from some set $e_i(l) \in E_i$ (ii) job j in J has processing p^l_{ij} and it must be scheduled in the interval $[r^l_{ij}, d^l_{ij}]$ on i (iii) there is no unnecessary idle time (iv) the β-constraints are met.

In the proof of Lemma 1 instead of a single instance I_i for machine i we define k_i instances, one for every element of E_i. Instance $I_i(l)$ is defined on job set $J_i(l) = \{j \in \mathcal{J} \mid j \text{ has type in } e_i(l)\}$. Job j in $J_i(l)$ has processing time, release date, and due date $p^l_{ij}, r^l_{ij}, d^l_{ij}$ respectively. Running algorithm A_β on each instance $I_i(l)$, $l = 1, \ldots, k_i$, will detect whether there is a configuration C in $\mathcal{C}(i)$ with total value more than y_i/ρ.

The rounding algorithm in the proof of Theorem 2 will return on every machine a configuration C s.t. all jobs in C have a type from a set $e_i(l) \in E_i$. The rest of the proof holds as before, including the analogous extensions of Corollaries 1 and 2 to the setting with types.

C FPTAS for similarly ordered release and due dates [17]

When release and due dates are similarly ordered, Lawler ([17], Section 6) observed that there is a non-preemptive EDD (Earliest Due Date) optimal schedule, that can be computed by the following DP:

$$C_j(w) = \min\{C_{j-1}(w), \max\{r_j, C_{j-1}(w - w_j)\} + p_j\}, \tag{9}$$

where $C_j(w)$ is the minimum total scheduling time needed to achieve total weight at least w, by a feasible schedule of the first j jobs in an EDD order. If the right-hand side of (9) violates any d_i, then $C_j(w) = \infty$. While the running time of this DP is $O(n^2 w_{\max})$, it can be transformed into a FPTAS for maximum weighted throughput by rounding the job weights as in the classic knapsack problem.

The FPTAS can be adapted to work with the setting of Section 3, since there is also an optimal schedule that has all its jobs shifted as late as possible such that within each contiguous block of jobs that does not contain a due date, jobs appear in earliest release date order. The correctness of the algorithm is not affected if we construct the optimal schedule in a "towards the past" direction, starting at $d_{\max}$, and examining the jobs in a Latest Release Date (LRD) order.

For the setting of a constant number of distinct release dates (due dates), we exploit the fact that the algorithm of [17] computes the optimal solution when all jobs have a common due date (common release date).

D PTAS for GAP with a constant number of bins [15]

In the Generalized Assignment Problem (GAP), we are given m bins (machines in the terminology of [15]), with bin i having volume capacity B_i, and n jobs with weight w_j for job j, as well as processing time p_{ij} when scheduled in bin i (note that if job j cannot go to machine i, $p_{ij} = \infty$); the goal is to maximize the total weight of scheduled jobs. There are no release or due dates.

Khan et al. [15] presented a PTAS for the case of constant m. They split the jobs into "big" and small according to their processing times, by proving the following structural theorem for any feasible GAP solution:

▶ **Theorem 11** (Theorem 3 in [15]). *Let S be a feasible solution to GAP, and let $S_i \subseteq S$ be the jobs assigned to the i-th bin. Then for all $\varepsilon > 0$, there exist sets X and Y such that $|X| \leq m/\varepsilon^2$ and $w(Y) \leq \varepsilon w(S)$ and*

$$p_{ij} \leq \varepsilon \left(B_i - \sum_{j \in X \cap S_i} p_{ij} \right), \ \forall i, \ \forall j \in S_i - X - Y.$$

Having guessed the set X together with the assignment of its jobs to bins, [15] turn to the scheduling of the rest of the jobs that are small in the sense of Theorem 11. In order to recover total weight at least $(1-\varepsilon)w(S-X)$, the authors in [15] apply *resource augmentation* for processing times (Section D.1), followed by a DP-based PTAS for scheduling the jobs with their rounded processing times, and, lastly, *trimming* (Section D.2) is used in order to remove a lengthy enough set of cheap jobs so as to restore the bin volume capacities to their original values. We proceed to outline how resource augmentation and trimming are defined in [15] and under what conditions they can be used for the throughput problem.

D.1 Resource augmentation

For every bin i, define a scaling factor $\mu_i := \varepsilon B_i / n$. Also, define new bin volume capacity $B'_i := \lfloor B_i/\mu_i \rfloor + n$ and new job processing times $p'_{ij} := \lceil p_{ij}/\mu_i \rceil$. Then there is an optimal solution to maximum throughput that can be calculated by a PTAS, and the bin capacities used are $B'_j \leq (1+\varepsilon)B_j$. Obviously, this optimal solution is at least as good as the optimal solution that can be obtained with the original bin capacities. Khan et al. [15] show that if the items (jobs) have size at most εB_i after trimming the solution as in Section D.2 the bin capacities can be restored to their original values while losing only an ε-fraction of weight (profit).

Resource augmentation can be applied on a maximum throughput instance restricted to an interval I on the time axis only when there are no release or due dates contained in I.

D.2 Trimming

Khan et al. [15] designed the trimming method that follows for a knapsack instance. Suppose that in bin i, there is a schedule of a set S_i of jobs, without due dates or relase dates, s.t. $p(S_i) = \gamma B_i$, with $\gamma \leq 1 + \delta$ and with total weight $w(S_i)$. Moreover, $p_{ij} \in (0, \varepsilon B_i]$ for $j \in S_i$. Then for parameters δ, ε with $\delta \leq \varepsilon$, [15] show how to create empty space of length at least δB_i, without losing more than $((\delta + \varepsilon)/(\gamma - \varepsilon))w(S_j)$ weight, and the scheduling of the remaining jobs takes no more than B_i processing time.

This trimming technique of [15] can be extended to apply to a maximum throughput schedule with different release dates and a common due date (different due dates and a common release date). In this case, all jobs can be shifted as late (as early) as possible, to create a new schedule ending at the due date (starting at the release date) with no idle time. This can be achieved without violating any release (due) dates. We provide the details for the case of the common due date in our Lemma 3.

D.3 Proof of Lemma 3

We provide the missing details for the existence of set R. We adjust the trimming method of [15] in order to account for the existence in our setting of release dates and achieve the parameters we need in the statement of the lemma. In particular let σ' be the feasible schedule of the set $X \cup J'$ in the interval I where for every $j \in J'$, $p_j \leq \varepsilon(|I| - p(X))$. Let $B = |I| - p(X)$. We consider the set S of the subintervals of I that contain idle time or jobs from J'. Their length may vary and they may not be consecutive on the time axis as they may be intermingled with the jobs of X. Clearly the total length of these subintervals equals B. Let $k = \lfloor (1 - \varepsilon)/(2\varepsilon) \rfloor$. Put a blue marker to the leftmost endpoint of a subinterval in S. Moving to the right, sweep along the time axis within the subintervals of S only and put a red marker after length εB, a blue marker after the next length εB and so on. Therefore the markers alternate between red and blue. Proceed until you have placed $k + 1$ red markers and $k + 1$ blue markers. The last marker will be red. Since

$$(k + 1)\varepsilon B + k\varepsilon B = (\varepsilon + 2\varepsilon \lfloor (1 - \varepsilon)/(2\varepsilon) \rfloor)B \leq B$$

this is always possible. Number the markers from 0 to $2k + 1$ from left to right. Consider the set of jobs S_i, $i \in \{0, 1, \ldots, k\}$ that are scheduled to a non-zero extent after blue marker $2i$ and up to red marker $2i + 1$. These sets are pairwise disjoint. Let $S_i^* = \arg\min_{i=0}^{k} w(S_i)$. It follows that $w(S_{i^*}) \leq \frac{1}{k+1} \sum_{i=0}^{k} w(S_i) \leq (2\varepsilon)/(1 - \varepsilon)w(J')$. Removing $R := S_{i^*}$ will leave empty space of εB. We have that $p(J' - R) \leq (1 - \varepsilon)B$. Define the integer $\beta \geq 0$ such that $(1 + \varepsilon)^{\beta} \leq B < (1 + \varepsilon)^{\beta+1}$. Then $p(J' - R) \leq (1 - \varepsilon^2)(1 + \varepsilon)^{\beta}$.