ORIGINAL RESEARCH

Approximation algorithms for coupled task scheduling minimizing the sum of completion times

David Fischer[1] · Péter Györgyi[2]

Abstract

In this paper we consider the coupled task scheduling problem with exact delay times on a single machine with the objective of minimizing the total completion time of the jobs. We provide constant-factor approximation algorithms for several variants of this problem that are known to be $\mathcal{NP}$-hard, while also proving $\mathcal{NP}$-hardness for two variants whose complexity was unknown before. Using these results, together with constant-factor approximations for the makespan objective from the literature, we also introduce the first results on bi-objective approximation in the coupled task setting.

Keywords Single machine scheduling · Coupled task problem · Approximation algorithms · Total completion times

1 Introduction

The problem of scheduling coupled tasks with exact delays (CTP) was introduced by Shapiro (1980) more than forty years ago. In this particular scheduling problem, each job has two separate tasks and a delay time. The goal is to schedule these tasks such that no tasks overlap, and the two tasks of a job are scheduled with exactly their given delay time in between them, while optimizing some objective function. This problem has several practical applications, e.g., in pulsed radar systems, where one needs to receive the reflections of the transmitted pulses after a given period of time (Elshafei et al., 2004; Farina, 1980), in improving the performance of submarine torpedoes (Simonin et al., 2011), or in chemistry (Simonin et al., 2007).

Research interest in the coupled task problem is strongly increasing in recent years, see Khatami et al. (2020) for a current, detailed overview of the topic. This research focuses

✉ Péter Györgyi
  gyorgyi.peter@sztaki.hu

  David Fischer
  da.fischer@tuhh.de

[1] Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

[2] Institute for Computer Science and Control, Eötvös Loránd Research Network, Kende u. 13-17, Budapest 1111, Hungary

mainly on variants of the general CTP with some additional restrictions on the job properties, and mainly tries to optimize the makespan (Khatami et al., 2020). Coupled task problems are often $\mathcal{NP}$-hard even in very special cases, but polynomial-time approximation algorithms with constant approximation factors have been developed for a number of them (Ageev and Kononov, 2006; Ageev & Baburin, 2007; Ageev and Ivanov, 2016). Other objective functions have virtually not been considered though, until recently, when Chen and Zhang (2021) drew an almost full complexity picture for the problems of single-machine scheduling of coupled tasks, with the objective of minimizing the total sum of job completion times. However, they did not give any approximation algorithms for $\mathcal{NP}$-hard CTP variants with this particular objective function. We fill this gap by giving a number of constant-factor approximation algorithms for most of these CTP variants. Additionally, we introduce two new, interesting variants, which we also prove to be $\mathcal{NP}$-hard, and also approximate one of these with a constant factor.

Formally, we are given a set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$, where each job j has two tasks: a_j and b_j. We call a_j the first task, and b_j the second task of job j. In order to simplify our notations, we will denote the processing time of these tasks also by a_j and b_j; the meaning of these notations will be clear from context. The sum $(a_j + b_j)$ is then called the total processing time of a job j. These tasks have to be scheduled on a single machine with a given delay time L_j in-between, which means if the machine completes a_j at some time point t, then we have to schedule b_j to start exactly at $t + L_j$. Preemption is not allowed. Note that it is possible to schedule other tasks on the machine during this delay time, but the tasks themselves cannot overlap. Our objective is to find a feasible schedule σ that minimizes the total of job completion times, where a feasible schedule is defined as a schedule that fulfills all of the requirements above. Such a σ is then called optimal schedule or optimal solution for the CTP instance. For a given schedule σ, the starting time S_j of j is the starting time of a_j, while the completion time C_j of j is the completion time of b_j. An example of CTP is visualized in Fig. 1. For a schedule σ, a *gap* is a period between time points t_1 and t_2 such that the machine is idle between t_1 and t_2 and busy at both t_1 and t_2. The length of a gap is the length of this time window. A partial schedule σ^p is a schedule for a subset of the jobs $\mathcal{J}$.

We say a job is the jth finishing job in a schedule if its second task is scheduled after the second tasks of exactly $j - 1$ many jobs have finished. We say a job is the jth starting job in a schedule if its first task is scheduled after the first tasks of exactly $j - 1$ many jobs have been scheduled. Let C^{OPT} be the sum of completion times of an optimal solution. For a fixed optimal schedule, let $C_j^{OPT_f}$ be the completion time of the jth finishing job in that schedule. Analogously, let $C_j^{OPT_s}$ be the completion time of the jth starting job in that schedule. Observe that $C^{OPT} = \sum_{j=1}^n C_j^{OPT_f} = \sum_{j=1}^n C_j^{OPT_s}$. In any proof in this work, we denote the sum of completion times of the solution produced by the currently used
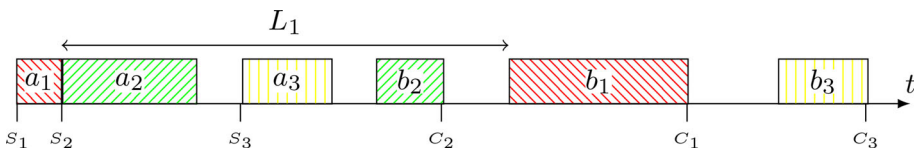

Fig. 1 An example for a feasible solution for an instance of CTP with $n = 3$. The patterns are matching for the two tasks of each job j. For simplicity, the delay time is only visualized for job 1. The total completion time of the solution is $C_1 + C_2 + C_3$

algorithm as C^{ALG}. In the schedule σ^{ALG} created by this algorithm, S_j and C_j denote the starting time and completion time of job j, respectively.

Throughout this paper, we will use the classic $\alpha\|\beta\|\gamma$ notation system of Graham et al. (1979), with α representing the machine environment, β representing the characteristics of the jobs, and γ representing the objective function. $1\|(a_j, L_j, b_j)\|\sum C_j$ then denotes the general CTP for minimizing the sum of completion times, where each job j consists of a pair of tasks of processing times a_j and b_j, respectively, with an exact time delay L_j between the completion time of its first task and the start time of its second task. As we also look at more restricted variants of CTP, we fix some naming conventions to easily express these restrictions in Graham notation. If in a restricted CTP environment, some task of the jobs are fixed or even constant for each job j, we denote this task without the subscript 'j', or by the specific constant value; e.g. CTP, where the delay time L_j is fixed to some L for all jobs j is denoted as $1\|(a_j, L, b_j)\|\sum C_j$. If in a restricted CTP environment, some tasks of the same job always have the same value, we denote them by p instead of their usual descriptor; e.g., CTP where the first and second task of each job j have the same processing time $(a_j = b_j, \forall j)$ is denoted as $1\|(p_j, L_j, p_j)\|\sum C_j$. This is in line with the standard notation for the coupled task scheduling problems, as seen for example in Chen and Zhang (2021). Another way to restrict CTP is to fix the processing sequence of the first tasks of the jobs, this is indicated by π_a in the β-field of the Graham notation.

In this paper we extend the complexity results of Chen and Zhang (2021) and Kubiak (2022) by proving the strong $\mathcal{NP}$-hardness of $1\|(p_j, L, p_j)\|\sum C_j$ and $1\|(1, L_j, 1, \pi_a)\|\sum C_j$. To achieve the former, we first prove strong $\mathcal{NP}$-hardness of the corresponding makespan variant $1\|(p_j, L, p_j)\|C_{\max}$, strengthening a result by Ageev and Ivanov (2016), who prove weak $\mathcal{NP}$-hardness of this problem. We also give constant-factor approximations for most CTP variants in a single machine environment with the sum of completion times objective function, see Fig. 2. The existence of a constant-factor approximation algorithm for the variants $1\|(a_j, L_j, b_j)\|\sum C_j$, $1\|(a, L_j, b_j)\|\sum C_j$, $1\|(a_j, L_j, b)\|\sum C_j$, and $1\|(p_j, L_j, p_j)\|\sum C_j$ is still open (see the upper part of the figure).

We also look at bi-objective optimization for CTP with both the makespan and the sum of completion times objectives, under the goal of minimizing both objectives without prioritization. For this, we use the concept of $(\rho_1, \ldots, \rho_z)$-approximation, as introduced by Jiang et al. (2023) for simultaneously minimizing z objectives, where ρ_i is the approximation factor of the ith objective function to be minimized, for $i = 1, \ldots, z$. This concept is a generalization of the bi-objective (ρ_1, ρ_2)-approximation of scheduling problems minimizing makespan and sum of completion times, as described first by Stein and Wein (1997). As far as we are aware, there are no results of minimizing the two objectives makespan and sum of completion times simultaneously in a coupled task setting, even though this topic is well researched in other scheduling environments. We start to close this gap by directly using results of Stein and Wein (1997), together with constant-factor approximations of the makespan(Ageev and Kononov, 2006; Ageev & Baburin, 2007; Ageev and Ivanov, 2016) and our approximation results on the sum of completion times objective, to give a number of (ρ_1, ρ_2)-approximation results for this problem. The general bi-objective CTP is denoted as $1\|(a_j, L_j, b_j)\|(C_{\max}, \sum C_j)$ in $\alpha\|\beta\|\gamma$ notation system, with the β-field following the previously discussed naming conventions depending on the considered variant's restrictions.

This work is structured as follows. We first give a brief literature review of the topic in Sect. 2. We present our complexity results in Sect. 3. Our approximation results are stated in Sect. 4. There, we give detailed descriptions and run time analyses of our algorithms, as well as proofs on approximation factors for problem variants whose instances can be solved by these algorithms. We then use these results in Sect. 5 to give (ρ_1, ρ_2)-approximations for the
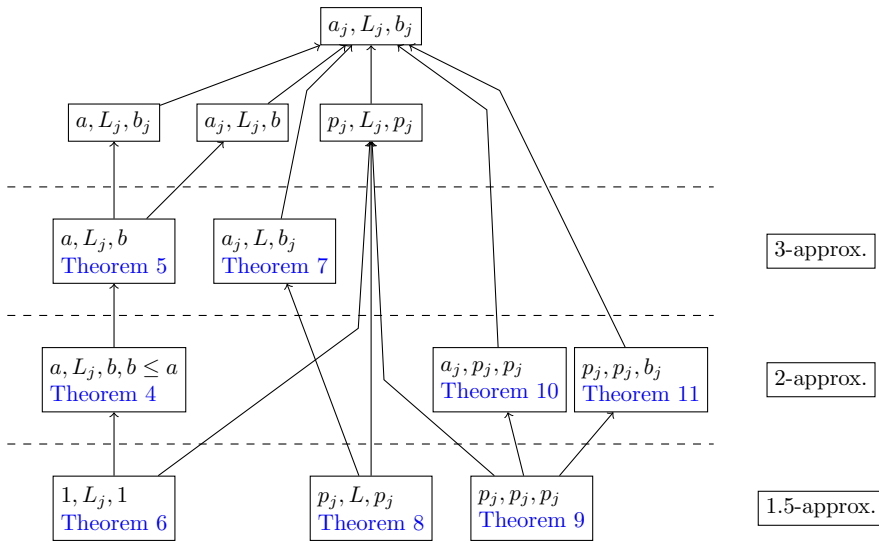
Fig. 2 Overview of our approximation results for different variants of $1\|(a_j, L_j, b_j)\|\sum C_j$. The variants, identified by their special constraints, are grouped into layers of equal approximation factors, with the respective Theorem proving this approximation factor linked next to it. A directed edge from variant "A" to variant "B" indicates that "B" is a generalization of "A"

bi-objective CTP with both the makespan and sum of completion time objectives. Finally, we give concluding remarks and an outlook on future research in Sect. 6.

2 Literature review

Research on coupled task scheduling on a single machine began when Shapiro (1980) proved the $\mathcal{NP}$-hardness of the general problem $1\|(a_j, L_j, b_j)\|C_{\max}$, where both tasks, as well as the delay time between them, can be different for each job, and the makespan is to be minimized.

In subsequent years, the $\mathcal{NP}$-hardness was also shown for more restricted variants of this problem, specifically $1\|(p_j, p_j, p_j)\|C_{\max}$, $1\|(a_j, L, b)\|C_{\max}$, $1\|(a, L, b_j)\|C_{\max}$ and $1\|(p, L_j, p)\|C_{\max}$ by Orman and Potts (1997). Some CTP variants minimizing the makespan are $\mathcal{NP}$-hard even when the processing times of all jobs are fixed to 1, as shown by Yu et al. (2004) for $1\|(1, L_j, 1)\|C_{\max}$. Condotta and Shakhlevich (2012) showed $\mathcal{NP}$-hardness for the even more restricted variant $1\|(1, L_j, 1, \pi_a)\|C_{\max}$, where π_a indicates a fixed processing sequence for the first tasks of all jobs.

For most of these problems, polynomial-time constant-factor approximation algorithms have been developed. Ageev and Kononov (2006) give such algorithms, as well as inapproximability bounds, for the general $1\|(a_j, L_j, b_j)\|C_{\max}$ problem, and the restricted variants $1\|(a_j, L_j, b_j, a_j \leq b_j)\|C_{\max}$, $1\|(a_j, L_j, b_j, a_j \geq b_j)\|C_{\max}$, and $1\|(p_j, L_j, p_j)\|C_{\max}$. Related to this work, Ageev and Baburin (2007) give an approximation algorithm for the $1\|(1, L_j, 1)\|C_{\max}$ variant. Additionally, Ageev and Ivanov (2016) give approximation algorithms and inapproximability bounds for $1\|(a_j, L, b_j)\|C_{\max}$, $1\|(a_j, L, b_j, a_j \leq b_j)\|C_{\max}$ and $1\|(p_j, L, p_j)\|C_{\max}$.

for other restricted variants, polynomial-time algorithms do exist. This was shown by Orman and Potts ([1997]) for the variants $1\|(p, p, b_j)\|C_{\max}$, $1\|(a_j, p, p)\|C_{\max}$, and $1\|(p, L, p)\|C_{\max}$, as well as Hwang and Lin ([2011]) for the variant $1\|(p_j, p_j, p_j)$, fjs$\|C_{\max}$, where "fjs" denotes that the sequence of jobs in the schedule is fixed.

Research interest in the topic of coupled task scheduling remained high also in the last years. Békési et al. ([2022]) recently introduced and gave a constant-factor approximation algorithm for the novel problem variant $1\|(1, L_j, 1), L_j \in \{L_1, L_2\}\|C_{\max}$, where there are only two different delay times in an instance; the complexity status of this variant is still unknown. Khatami and Salehipour ([2021a]) tackle the coupled task scheduling problem differently, giving upper and lower bounds on the solution through different procedures, and proposing a binary heuristic search algorithm for CTP. The same authors give optimal solutions under certain conditions, and a general heuristic for the problem variant with fixed first tasks and delay times, but time-dependent processing times for the second tasks(Khatami & Salehipour, [2021].) Bessy and Giroudeau ([2019]) investigate CTP under parameterized complexity, with the considered parameter k relating to the question if k coupled tasks have a completion time before a fixed due date.

Interest is also high in scheduling coupled tasks in *2-machine flow shop* environments, denoted by $F2$ in the machine environment notation. For scheduling coupled tasks in this environment, we are given two machines instead of one, and each of the two tasks of one job is additionally assigned one of these two machines to be processed on. $\mathcal{NP}$-hardness is shown for a number of flow shop problems minimizing the makespan, e.g., for $F2\|(1, L_j, 1)\|C_{\max}$ by Yu et al. ([2004]), but $\mathcal{NP}$-hardness is also known for variants minimizing the total completion time, e.g., $F2\|(a_j, L, b_j)\|\sum C_j$, as shown by Leung et al. ([2007]). Several flow shop problem variants minimizing the total completion time are also polynomial solvable, as proven by Leung et al. ([2007]) and Huo et al. ([2009]).

All of the mentioned literature for scheduling coupled tasks on a single machine only considers the objective of minimizing the makespan though, and, as Khatami et al. ([2020]) note in their survey of CTP, "there has been no published research investigating the single-machine setting with an objective function other than the makespan, except for those in the cyclic setting." This task is finally tackled by Chen and Zhang ([2021]), who draw a nearly full complexity picture of problem of minimizing the total of job completion times. However, they do not give any approximation algorithms for problem variants they prove to be $\mathcal{NP}$-hard. Recently, Kubiak ([2022]) slightly extended these complexity results by proving $\mathcal{NP}$-hardness of $1\|(1, L_j, 1)\|\sum C_j$ and $1\|\langle 1, L_j, 1\rangle\|\sum C_j$. In the latter problem variant, the delay time between the two tasks does not have to be exactly, but at most L_j.

In scheduling theory, there is also a great interest in bi-objective and multi-objective optimization. Here, instead of trying to optimize just one objective function in a given problem setting, one aims to optimize two or more objective functions at the same time, see Deb (Deb, [2014]) or Hoogeveen ([2005]) for an overview. Since, until recently, virtually only the makespan objective has been considered for coupled tasks scheduling problems, we do not know any such results in the coupled task environment. This is not true for other scheduling environments though, where especially bi-objective optimization is intensively researched, particularly for the two objectives of minimizing makespan and sum of completion times. Here, many approaches focus on establishing a trade-off relationship between the two competing objectives, either by *Pareto optimization* (finding one or all Pareto optimal solutions) or *simultaneous optimization* (minimizing all objectives without prioritization) (Jiang et al., [2023]). Since these problems are generally $\mathcal{NP}$-hard (see e.g. Hoogeveen ([2005])), approximation is a popular method for both mentioned approaches. Angel et al. ([2003]) give fully polynomial time approximation schemes for the Pareto curve of single-machine batching

problems and parallel machine scheduling problems on the two objectives. Bampis and Kononov (2005) consider (ρ_1, ρ_2)-approximations of the two objectives for scheduling problems with communication delays. A very recent work by Jiang et al. (2023) is concerned with (ρ_1, ρ_2)-approximations for scheduling on parallel machines, with different approximation ratios for different fixed numbers of machines.

3 Complexity results

In this section we prove that both $1\|(p_j, L, p_j)\| \sum C_j$ and $1\|(1, L_j, 1, \pi_a)\| \sum C_j$ are strongly $\mathcal{NP}$-hard. We use reductions from corresponding makespan minimization problems for both results. As we need strong $\mathcal{NP}$-hardness of the corresponding problems for both our reductions, we additionally prove strong $\mathcal{NP}$-hardness of $1\|(p_j, L, p_j)\|C_{\max}$; weak $\mathcal{NP}$-hardness was already proven for this problem by Ageev and Ivanov (2016).

Theorem 1 $1\|(p_j, L, p_j)\|C_{\max}$ *is strongly $\mathcal{NP}$-hard.*

Proof We reduce the well-known strongly $\mathcal{NP}$-hard problem 3- PARTITION (Garey and Johnson, 1979) to $1\|(p_j, L, p_j)\|C_{\max}$. The reduction is similar to the idea by Ageev and Ivanov (2016) for reducing the weakly $\mathcal{NP}$-hard PARTITION problem to $1\|(p_j, L, p_j)\|C_{\max}$. First, let us formally state the 3- PARTITION problem.

3- PARTITION

Instance: A set $Q = \{1, \ldots, 3q\}$, and for each element $i \in Q$, a corresponding positive integer e_i such that $\sum_{i \in Q} e_i = qE$, for some positive integer E, and $E/4 < e_i < E/2$.

Question: Does the set Q partition into q disjoint subsets $Q_1, \ldots, Q_q$ such that $\sum_{i \in Q_j} e_i = E$, for $j = 1, \ldots, q$?

3- PARTITION remains strongly $\mathcal{NP}$-hard even if we assume q is even. Consider an instance I of 3- PARTITION where q is even. We define an instance I' of $1\|(p_j, L, p_j)\|C_{\max}$ with $4q$ jobs as follows:

- jobs $i = 1, \ldots, 3q$ have $p_i = e_i$ and $L = R + E$ (small jobs),
- jobs $i = 3q + 1, \ldots, 4q$ have $p_i = R$ and $L = R + E$ (large jobs),

for some $R > 3qE$. We prove there is a solution for I if and only if there is a solution for I' with makespan $C_{\max} \leq z$, with $z := q(3E + 2R)$. The theorem follows from this statement.

1. Assume there is a solution for I. Then there exist $Q_1, \ldots, Q_q$, such that $\sum_{i \in Q_j} e_i = E$, for $j = 1, \ldots, q$. In this case, we create a schedule σ for I' with makespan at most z. We schedule the jobs in blocks. Let B be an arbitrary block. It consists of 2 large jobs (i and i') and 6 small jobs, which corresponds to items from some Q_j and $Q_{j'}$. Let (j_1, j_2, j_3) and (j'_1, j'_2, j'_3) be the small jobs corresponding to the items in Q_j and $Q_{j'}$, respectively. Assume that $p_{j_1} \geq p_{j_2} \geq p_{j_3}$ and $p_{j'_1} \leq p_{j'_2} \leq p_{j'_3}$. We schedule $a_{i'}$ directly before b_i. We schedule b_{j_1}, b_{j_2} and b_{j_3} in this order in the gap between a_i and $a_{i'}$ and $a_{j'_1}, a_{j'_2}$ and $a_{j'_3}$ in this order in the gap between b_i and $b_{i'}$, see Fig. 3. Observe that the job tasks do not intersect, because the length of the gap between a_i and $a_{i'}$ as well as the gap between b_i and $b_{i'}$ is exactly E. The length of block B is at most $4R + 6E$, because a_{j_1} starts at most $2E$ before a_i, and $b_{j'_3}$ completes at most $2E$ after $b_{i'}$. We create such blocks for all jobs, resulting in $q/2$ blocks in total, and schedule these blocks directly one after another. Thus, the resulting schedule σ has makespan $C_{\max} \leq q/2(4R + 6E) = z$.

2. Now assume that the instance I' has a schedule σ with makespan $C_{\max} \leq z$. Due to the fixed delay times, the order is the same for the first and the second tasks in σ. Consider
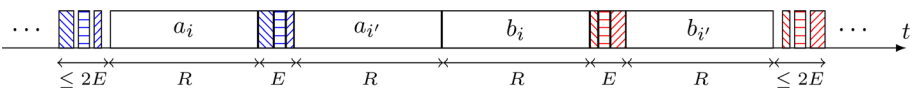
Fig. 3 A block B of jobs in σ. The blue tasks are j_1, j_2 and j_3, while the red tasks are j_1', j_2' and j_3' (in this order)

an arbitrary large job i. There has to be exactly one task of another large job i' between a_i and b_i. There cannot be more than one task of a large job between a_i and b_i. If there was no task of a large job between a_i and b_i, the makespan of σ would be larger than z, as the total processing time of the large jobs is $2qR$, and $L = R + E$, resulting in a minimum makespan of $C_{\max} \geq 2qR + (R + E) > q(2R + 3E) = z$, due to $R > 3qE$. Observe that if a task of job i' is scheduled in the delay time of i, then a task of i is scheduled in the delay time of i'. This means we can partition the large jobs into pairs where the jobs within each pair are interleaved in the above way. Consider an arbitrary pair of large jobs (i, i') and assume that $S_i < S_{i'}$. There cannot be any task of a small job scheduled between $a_{i'}$ and b_i, as a first task of a small job would imply an intersection of its second task with $b_{i'}$, and a second task of a small job would imply an intersection of its first task with a_i, due to the fixed delay times. Consider an arbitrary small job j. If none of its tasks is scheduled in the gap between a_i and $b_{i'}$ of any pair of large jobs (i, i'), then there is also no task of a large job scheduled in the delay time of j due to the fixed delay times. This implies the makespan of σ is at least $2qR + 2e_j + R + E > z$, because the total processing time of the large jobs together with j is $2qR + 2e_j$, and the delay time of j is $R + E$. Therefore, for any small job j, there is a pair of large jobs (i, i'), where at least one of the tasks of j is scheduled in the gap between a_i and $a_{i'}$ or in the gap between b_i and $b_{i'}$. For any pair of large jobs the length of the gap between a_i and $a_{i'}$ is at most E, the same holds for the length of the gap between b_i and $b_{i'}$. Thus, the total length of these gaps is at most $q/2 \cdot (2E) = qE$, which is exactly half of the total processing time of the small jobs. Therefore, the length of each such gap must be exactly E, and they must be completely filled with tasks of small jobs. As $E/4 < e_j < E/2$, there are always exactly three tasks of small jobs in each such gap. This partitions the small jobs into q sets Q_j of 3 jobs each, with $\sum_{x \in Q_j} e_x = E$ for each $j = 1, \ldots, q$, which gives us a feasible solution for the 3-PARTITION instance I.

□

Theorem 2 $1\|(p_j, L, p_j)\| \sum C_j$ *is strongly $\mathcal{NP}$-hard.*

Proof We reduce the strongly $\mathcal{NP}$-hard problem $1\|(p_j, L, p_j)\|C_{\max}$, proven to be strongly $\mathcal{NP}$-hard in Theorem 1, to $1\|(p_j, L, p_j)\| \sum C_j$. Consider an instance I of $1\|(p_j, L, p_j)\|C_{\max}$. We define an instance I' of $1\|(p_j, L, p_j)\| \sum C_j$ as follows. There are $n + M$ jobs in I', where M is a sufficiently large number.

- the first n jobs are the same as the jobs in I (small jobs),
- for the remaining M jobs, we have $p_j = \sum_{i=1}^{n} p_i + nL$, $j = n + 1, \ldots, n + M$ (large jobs).

We prove that there is a solution σ for I with makespan at most C if and only if there is a solution for I' with a total completion time of at most $z := (n + M)C + hM(M + 1)/2$, where $h := 2\left(\sum_{i=1}^{n} p_i + nL\right) + L = a_j + L + b_j$ is the time required for scheduling a large job j. The Theorem follows from this statement.
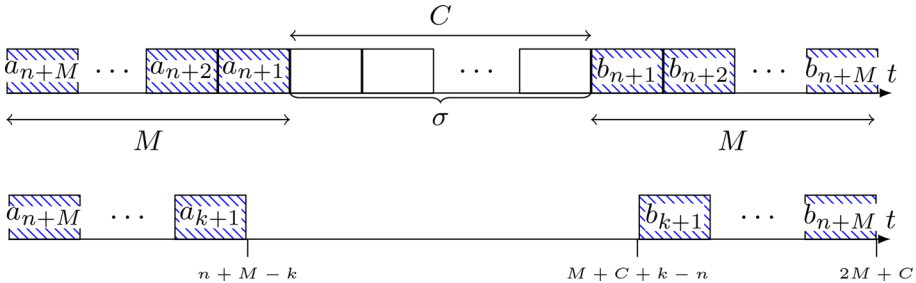
Fig. 4 Schedule σ' created from schedule σ

Fig. 5 Above: schedule σ' created from schedule σ. The original jobs are white, the helper jobs are blue. Below: jobs $k+1, \ldots, n+M$ in schedule $\hat{\sigma}$

If there is such an σ, then we define σ' (a solution of I') from σ as follows. The small jobs start exactly at the same time as they start in σ, while the large jobs start right after them as soon as possible (in arbitrary order), see Fig. 4. Observe that the completion time of any small job in σ' is at most C, while the total completion time of the large jobs is $MC + hM(M+1)/2$. Therefore, the total completion time of σ' is at most z.

Now suppose that there is no such solution for I, i.e., the makespan $C_{\max}$ of any solution σ is at least $C + 1$. Consider an arbitrary optimal solution $\hat{\sigma}$ for I'. Observe that no job task can be scheduled between the first and the second task of any large job in $\hat{\sigma}$, since the processing times of both tasks of any large job are larger than L. Furthermore, no large job can precede any of the small jobs, otherwise, we would get a better schedule by moving that large job to the end of the schedule (cf., the definition of the processing times of the large jobs). Therefore, the large jobs start right after the small jobs, i.e., after $C_{\max}$. Hence,

$$\sum_{j=1}^{n+M} \hat{C}_j = \sum_{j=1}^{n} \hat{C}_j + MC_{\max} + hM(M+1)/2$$
$$> M(C+1) + hM(M+1)/2 > z,$$

where the last inequality follows if M is larger than nC.

$\square$

Theorem 3 $1\|(1, L_j, 1, \pi_a)\| \sum C_j$ *is strongly* $\mathcal{NP}$-*hard.*

Proof We reduce the known strongly $\mathcal{NP}$-hard problem $1\|(1, L_j, 1, \pi_a)\|C_{\max}$, proven to be strongly $\mathcal{NP}$-hard by Condotta and Shakhlevich (2012), to $1\|(1, L_j, 1, \pi_a)\| \sum C_j$. Recall that π_a fixes a scheduling sequence for the first tasks of all jobs. Consider an instance I of $1\|(1, L_j, 1, \pi_a)\|C_{\max}$. We want to know if I has a solution with $C_{\max} \leq C$. We define an instance I' of $1\|(1, L_j, 1, \pi_a)\| \sum C_j$ as follows. There are $n + M$ jobs in I', where M is a sufficiently large number.

- the first n jobs are the same as the jobs in I (original jobs),

- for the remaining M jobs, we have $L_j = C + 2(j - n - 1)$, $j = n + 1, \ldots, n + M$ (helper jobs).

The fixed sequence of the first tasks in I' is defined as π'_a, with

- $\pi'_a := (n + M, n + M - 1, \ldots, n + 1, \{\pi_a\})$.

We prove that there is a solution σ for I with makespan at most C if and only if there is a solution for I' with a total completion time of at most

$$z := n(M + C) + M^2 + \frac{M(M + 1)}{2} + MC.$$

If there is such an σ, then we define σ' (a solution of I') from σ as follows. The helper jobs are scheduled in decreasing order of their indices one after another as soon as possible, while the original jobs are scheduled in the exact same way as in σ, but with their starting time increased by M each. See the first schedule of Fig. 5 for an illustration. This schedule respects the order given by π'_a. Observe that the completion time of any original job in σ' is at most $M + C$, while the total completion time of the helper jobs is exactly $M^2 + \frac{M(M+1)}{2} + MC$. Therefore, the total completion time of σ' is at most z.

Now suppose that there is no such solution for I, i.e., the makespan $C_{\max}$ of any solution σ is at least $C + 1$. Consider an arbitrary optimal solution $\hat{\sigma}$ for I'. Suppose for the sake of contradiction that the total completion time of $\hat{\sigma}$ is at most z for any M.

Due to the fixed order of the first tasks π'_a we can assume that a_{n+M} starts at 0 in $\hat{\sigma}$. Observe that the total completion time of the helper jobs is at least $M^2 + \frac{M(M+1)}{2} + MC$. The original jobs start after the first tasks of the helper jobs, i.e., after M, thus their total completion time is at least nM. If any of the original jobs completes after b_{n+M} in $\hat{\sigma}$, then its completion time is larger than $2M$, and the total completion time of the jobs in $\hat{\sigma}$ is larger than $M^2 + \frac{M(M+1)}{2} + MC + (n + 1)M > z$, if $M > nC$. Thus, each original job has to start before b_{n+M}.

If there is no gap among the first tasks of the helper jobs in $\hat{\sigma}$, then the machine is busy with the helper jobs in $[0, M]$ and in $[M + C, 2M + C]$. Since each original job has to be completed before b_{n+M} (i.e., before $2M + C$), these jobs have to be scheduled in $[M, M + C]$. However, this is a contradiction to the the makespan of σ.

Therefore, in the following we suppose there is a gap between some first tasks of the helper jobs in $\hat{\sigma}$. Let k, $n + 1 \leq k \leq n + M - 1$, be the largest index such that there is a gap before a_k. Then, the machine is busy with jobs $n + M, n + M - 1, \ldots, k + 1$ in $[0, n + M - k]$ and in $[M + C + k - n, 2M + C]$, see the second schedule in Fig. 5. Since a_k starts later than $n + M - k$, b_k starts later than $M + C + k - n$, which means b_k starts later than $2M + C$ to avoid intersection.

Let j be a job in $\{n + 2, \ldots, k\}$ whose second task b_j is scheduled after b_{n+M}. Since $L_{j-1} = L_j - 2$, and a_{j-1} has to be scheduled after a_j according to π'_a, b_{j-1} is scheduled either right before, or at some time after b_j. In both cases, b_{j-1} has to be scheduled after b_{n+M} to avoid intersection with b_{n+M}. This particularly implies that b_{n+1} is scheduled after b_{n+M}, i.e., $C_{n+1} \geq C_{n+M} + 1 = 2M + C + 1$.

The total completion time of jobs $n + 2, \ldots, n + M$ is then at least $(M + C + 2) + (M + C + 3) + \ldots + (2M + C) = (M - 1)(M + C + 2) + (M - 1)(M - 2)/2$. The total completion time of the original jobs is at least nM. Thus we have the following lower bound on the total completion time of $\hat{\sigma}$:

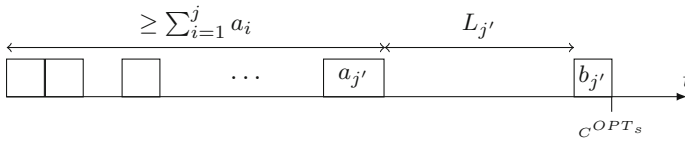$$(2M + C + 1) + \left((M - 1)(M + C + 2) + \frac{(M - 1)(M - 2)}{2}\right) + (nM)$$

Fig. 6 Illustration of the bound on $C_j^{OPT_s}$

$$= M^2 + M + CM + \frac{M^2 + M}{2} + nM > z,$$

if $M > nC$, proving the Theorem. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \square$

4 Approximation results

As in the following, we only look at CTP with the objective of minimizing the total completion times, we call this problem only "CTP" from now on for simplicity reasons. In this section we give polynomial-time approximation algorithms for a number of CTP variants. All of these variants are proven to be $\mathcal{NP}$-hard either by Chen and Zhang (2021), by Kubiak (2022), or by the results of Sect. 3. We start this section with two useful lemmas that provide lower bounds on the objective value of any optimal solution for the general CTP. Sections 4.2 and 4.3 consider variants with fixed processing times and fixed delay times, respectively. Section 4.4 examines variants where there exists some relation between the processing times and the delay time of each job.

4.1 Lower bounds on the optimum

Recall the definitions $C_j^{OPT_f}$ and $C_j^{OPT_s}$ as the completion time of the jth finishing and jth starting job of some optimal schedule, respectively. The next lemma is straightforward from the definition of $C_j^{OPT_f}$, since there are j jobs that complete until $C_j^{OPT_f}$ in an optimal schedule.

Lemma 1 *Let the jobs of a CTP instance be indexed in non-decreasing $a_i + b_i$ order. Then, for any optimal schedule for this instance, we have*

1. $C_j^{OPT_f} \geq \sum_{i=1}^{j}(a_i + b_i)$, $j = 1, \ldots, n$ and
2. $C^{OPT} \geq \sum_{j=1}^{n}\sum_{i=1}^{j}(a_i + b_i)$.

The second lemma is analogous, and follows from the observation that there are $j - 1$ first tasks that finish until the first task of the job corresponding to $C_j^{OPT_s}$ starts in some optimal schedule. See Fig. 6 for an illustration.

Lemma 2 *Let the jobs of a CTP instance be indexed in non-decreasing a_i order. Then, for any optimal schedule for this instance, where j' is the jth starting job, we have*

1. $C_j^{OPT_s} \geq \sum_{i=1}^{j} a_i + L_{j'} + b_{j'}$, $j = 1, \ldots, n$ and
2. $C^{OPT} \geq \sum_{j=1}^{n}\sum_{i=1}^{j} a_i + \sum_{j=1}^{n} L_j + \sum_{j=1}^{n} b_j$.

4.2 CTP with fixed processing times

In this section we assume $a_j = a$ and $b_j = b$ for each job $j \in \mathcal{J}$. Consider Algorithm A.

Algorithm A

Input: a CTP instance with $a_j = a$ and $b_j = b$ for each job j
Output: a schedule σ for this instance

1. Sort the jobs in non-decreasing L_j order. In the following, let the jobs be indexed in this order.
2. Schedule the jobs one-by-one as soon as possible without intersection.

Recall that here, σ^{ALG} denotes the schedule created by Algorithm A and S_j is the starting time of job j. The precise meaning of 'as soon as possible' is the following: for a given (partial) schedule σ^P and a job j, scheduling j as soon as possible means setting it's starting time $S_j :=$ $\min\{t \geq 0 :$ the machine is idle both in $[t, t + a_j]$ and in $[t + a_j + L_j, t + a_j + L_j + b_j]\}$.

Lemma 3 *Suppose that $a_j = a$, $b_j = b$ ($j \in \mathcal{J}$) and $b \leq a$. When Algorithm A schedules job j, where j is the jth job in non-decreasing L_j order, then a_j starts from the earliest time point t', such that the machine is idle in $[t', t' + a]$.*

Proof We prove the statement by an induction on j in the order given by step 1 in Algorithm A; it is trivial for $j = 1$. Suppose that the statement is true for some j, and consider the partial schedule σ^P created by the algorithm before scheduling job $j + 1$. Let t be the earliest time point such that the machine is either idle in $[t', t' + a]$ in σ^P (it is possible that t' is the time point when σ^P finishes). We prove the lemma by showing $S_{j+1} = t$. Due to the definition of t', a_{j+1} cannot intersect with any task in σ^P. We need to prove the same for b_{j+1}, which is scheduled in $[t' + a + L_{j+1}, t' + a + L_{j+1} + b]$.

First, observe that a_i starts before a_{i+1} for each $i \leq j$ due to the induction. This means $a_1, \ldots, a_j$ complete before t'. Let $L_1, \ldots, L_n$ be the delay times of the jobs obtained in step 1 of Algorithm A. Since $L_1, \ldots, L_j \leq L_{j+1}$, we have $C_1, \ldots, C_j \leq t' + L_{j+1} + b \leq t' + L_{j+1} + a$, where the last inequality follows from $b \leq a$. Thus, as each task in σ^P completes before the start of b_{j+1}, there is no intersection of any scheduled tasks. □

Lemma 4 *Algorithm A runs in $\mathcal{O}(n \log n)$ time and it always produces a feasible solution.*

Proof Sorting the jobs requires $\mathcal{O}(n \log n)$ time. When Algorithm A schedules job j, it searches the first gap after the first task of the directly previous scheduled job with a length of at least a (Lemma 3). This means the length of the gap after each task is only checked once during the whole procedure, which requires $\mathcal{O}(n)$ time in total. The feasibility of the schedule is straightforward from the definition of 'as soon as possible'. □

Theorem 4 *Algorithm A is a factor-2 approximation for $1 \| (a, L_j, b, b \leq a) \| \sum C_j$.*

Proof Due to Lemma 4, it remains to prove that $C^{ALG} \leq 2C^{OPT}$.

From Lemma 3 we know that the machine is always busy just before a first task of a job is scheduled (except a_1, which starts at time 0). This means there are at most $j - 1$ gaps before S_j, and the length of any of these gaps is smaller than a. Hence, if we consider the partial schedule at the time when the algorithm schedules job j, we have

$$S_j \leq (j - 1)(a + b) + (j - 1)a, \quad j \in \mathcal{J},$$

because (i) the machine is busy at most $(j-1)(a+b)$ time in $[0, S_j]$ with tasks of the jobs $1, \ldots, j-1$; and (ii) the total length of the gaps before S_j is smaller than $(j-1)a$. Therefore, we have $C_j \leq (j-1)(a+b) + (j-1)a + a + L_j + b$ and

$$C^{ALG} = \sum_{j=1}^{n} C_j \leq \frac{n(n-1)}{2}(a+b) + \frac{n(n-1)}{2}a + n(a+b) + \sum_{j=1}^{n} L_j$$

$$\leq C^{OPT} + \frac{n(n+1)}{2}a + nb + \sum_{j=1}^{n} L_j \leq 2C^{OPT},$$

where the second inequality follows from Lemma 1 and from $n(n-1)/2 + n = n(n+1)/2$, while the third follows from Lemma 2. □

As $1\|(a, L_j, b, b \leq a)\| \sum C_j$ is a more general version of the variant $1\|(p, L_j, p)\| \sum C_j$, Corollary 1 directly follows.

Corollary 1 *Algorithm A is a factor-2 approximation for $1\|(p, L_j, p)\| \sum C_j$.*

The following lemma describes some important attributes of σ^{ALG} in the opposite case of $a \leq b$.

Lemma 5 *Suppose that $a_j = a$, $b_j = b$ ($j \in \mathcal{J}$) and $a \leq b$. Consider the partial schedule σ^P created by Algorithm A, right before it schedules job j starting at S_j. Then,*

 (i) *a_j or b_j start right after another task (for each $j > 1$),*
 (ii) *the number of the gaps in σ^P is at most $j-1$.*
 (iii) *the number of gaps before S_j is at most $j-1$,*
 (iv) *the length of each gap in $[0, S_j]$ is at most b.*

Proof Statement (i) immediately follows from the algorithm. We use induction on j for proving the remaining statements; they are trivial for $j = 1$. Suppose that they are true for $1, 2, \ldots, j-1$, we then prove them for j.

Statement (ii) follows from (i) and the following observation. If there is no gap before a_j, then there will be at most one new gap created by b_j (as it is in the middle of a previous gap or at the end of the schedule after a new gap). If there instead is no gap before b_j, then there will be at most one new gap created by a_j.

Let t denote the completion time of the last task in σ^P. Observe that $S_j \leq t$. Hence, statement (iii) immediately follows from (ii).

For proving (iv), suppose for purpose of showing contradiction that there exists a gap $[g_1, g_2]$ in σ^P before S_j with a length larger than b. We know from the induction that each a_ℓ ($\ell \leq j-1$) completes before g_1, because otherwise, the gap before a_ℓ would already have been larger than b right after we have scheduled job ℓ. If $[g_1, g_2]$ existed however, the algorithm would schedule a_j at time g_1: there would be neither an intersection between a_j and any other task (since $a \leq b < g_2 - g_1$), nor between b_j and any other task (since a_{j+1} starts later than the previously scheduled first tasks, and $L_j \geq L_\ell$ for each $\ell \leq j-1$). The existence of such a gap would contradict the definition of the algorithm, therefore,(iv) follows. □

Theorem 5 *Algorithm A is a factor-3 approximation for $1\|(a, L_j, b)\| \sum C_j$.*

Proof Due to Lemma 4, it remains to prove that $C^{ALG} \leq 3C^{OPT}$.

If $b \leq a$, we can approximate $1\|(a, L_j, b)\| \sum C_j$ with a factor of 2 (Lemma 4). Therefore, w.l.o.g, we assume in the following that $a \leq b$. We show that, in this case, we get an approximation factor of 3.

From Lemma 5 (iii) and (iv) we know that when the algorithm schedules job j, the total idle time of the machine in $[0, S_j]$ is at most $(j - 1)b$. Due to the order of the jobs, the machine is busy in the partial schedule from 0 to S_j for at most $(a + b)(j - 1)$ time; thus we have $S_j \leq (a + b)(j - 1) + b(j - 1)$ and $C_j \leq (a + b)(j - 1) + b(j - 1) + (a + L_j + b)$. Hence,

$$C^{ALG} = \sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} (a + b)(j - 1) + \sum_{j=1}^{n} b(j - 1) + n(a + b) + \sum_{j=1}^{n} L_j$$

$$= n(n - 1)b + \left(\frac{n(n+1)}{2} a + nb + \sum_{j=1}^{n} L_j \right) \leq n(n - 1)b + C^{OPT},$$

where the second inequality follows from Lemma 2. Since $C_j^{OPT_f} \geq (a + b)j$ by Lemma 1, we have $C^{OPT} \geq \frac{n(n+1)}{2}(a + b)$. Therefore,

$$C^{ALG} \leq C^{OPT} + n(n - 1)b \leq 3C^{OPT} .$$

□

Theorem 6 *Algorithm A is a factor-*1.5 *approximation for* $1\|(1, L_j, 1)\| \sum C_j$.

Proof Due to Lemma 4, it remains to prove that $C^{ALG} \leq 1.5C^{OPT}$. From Lemma 3 we know that the algorithm always schedules a_j in the first gap. Thus, the starting time of j ($j \in \mathcal{J}$) is at most $2(j - 1)$, because the total processing time of the jobs $\ell < j$ is $2(j - 1)$ and the other jobs start later. Therefore, $C_j \leq 2(j - 1) + 2 + L_j$, and

$$C^{ALG} \leq 2 \sum_{j=1}^{n} (j - 1) + 2n + \sum_{j=1}^{n} L_j$$

$$\leq C^{OPT} + \sum_{j=1}^{n} (j - 1) + n \leq 1.5C^{OPT},$$

where the second inequality follows from Lemma 2, and the third from Lemma 1. □

4.3 CTP with fixed delay times

In this section we assume $L_j = L$ for each job $j \in \mathcal{J}$. Consider Algorithm B, based on an idea of Ageev and Ivanov (2016). Observe that both the first and the second tasks are in non-decreasing $a_j + b_j$ order in the schedule found by Algorithm B.

Lemma 6 *Algorithm B runs in* $\mathcal{O}(n \log n)$ *time and always produces a feasible solution.*

Proof The run time of Algorithm B is straightforward. Together with the observation that in step 3iii, we schedule the current job after all the other tasks, it is clear that there are no intersections in the schedule produced by the algorithm. □

Algorithm B

Input: a CTP instance with $L_j = L$ for each job j
Output: a schedule σ for this instance

1. Sort the jobs in non-decreasing $a_j + b_j$ order. In the following, let the jobs be indexed in this order.
2. Schedule a_1 from 0 and b_1 from $a_1 + L$. Let $s := 1$ and $j_s := 1$.
3. For $j = 2, \ldots, n$ do the following:

    i) If it is possible to start a_j right after a_{j-1} without b_j intersecting any job task on the schedule, then do it and consider the next job.
    ii) If it is possible to start b_j right after b_{j-1} without a_j intersecting any job tasks on the schedule, then do it and go to consider the next job.
    iii) Otherwise, start a_j right after b_{j-1}. Let $s := s + 1$ and $j_s := j$.
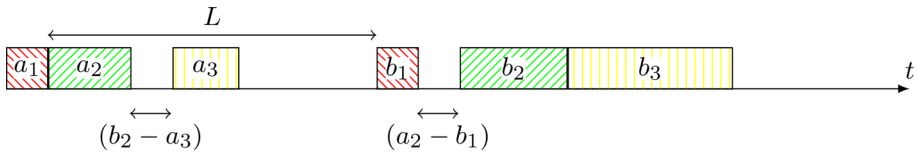
Fig. 7 Example for gaps in different cases

Let $B_s := \{j_s, j_s + 1, \ldots, j_{s+1} - 1\}$ be the sth *block* $(1 \leq s \leq n)$ and let $H_s := C_{j_{s+1}-1} - S_{j_s}$ denote the length of B_s. The next lemma describes an important observation on the gap sizes within a block and follows directly from the equal delay times and some simple algebraic calculations (see Fig. 7 for illustration).

Let G_i and G'_i be the length of the gap directly after a_i and b_i, respectively. If there is no gap, then the corresponding value is 0.

Lemma 7 *If jobs j and $j + 1$ are in the same block, then a_{j+1} starts right after a_j or b_{j+1} starts right after b_j. In the former case $G'_j \leq a_{j+1} - b_j \leq a_{j+1}$, while in the latter case $G_j \leq b_j - a_{j+1} \leq b_j$.*

Theorem 7 *Algorithm B is a factor-3 approximation for $1\|(a_j, L, b_j)\| \sum C_j$.*

Proof Due to Lemma 6, it remains to prove that $C^{ALG} \leq 3C^{OPT}$. The length of a block B_s is the sum of the following: (i) the lengths of the first tasks of the jobs in B_s, (ii) the lengths of the gaps among these tasks, (iii) L, and (iv) the length of the second task of the last job in B_s, i.e., $H_s = \sum_{i=j_s}^{j_{s+1}-1} a_i + \sum_{i=j_s}^{j_{s+1}-2} G_i + L + b_{j_{s+1}-1}$. From Lemma 7, we have

$$H_s \leq \sum_{i=j_s}^{j_{s+1}-1} (a_i + b_i) + L. \tag{1}$$

Observe that the algorithm starts a new block every time it cannot schedule the next upcoming job in steps 3i) and 3ii). Therefore, there can be two reasons why j_{s+1} cannot be scheduled in B_s: (a) the length of the gap between $a_{j_{s+1}-1}$ and b_{j_s} is smaller than $a_{j_{s+1}}$, i.e., $G_{j_{s+1}-1} < a_{j_{s+1}}$ or (b) the completion time of $b_{j_{s+1}-1}$ minus the starting time of b_{j_s} is larger than L, see Fig. 8.
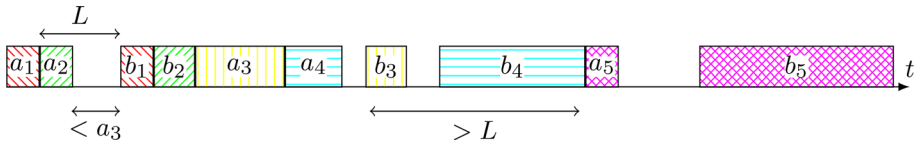
Fig. 8 Reasons of starting a new block: a_3 does not fit between a_2 and b_1, while the total processing times of the second tasks in the second block ($b_3 + b_4$) and the gap between them is larger than L

In case of (a), we know that b_{j_s} starts at $S_{j_s} + a_{j_s} + L$ and $a_{j_{s+1}-1}$ completes at $S_{j_s} + \sum_{i=j_s}^{j_{s+1}-1} a_i + \sum_{i=j_s}^{j_{s+1}-2} G_i$, thus

$$a_{j_{s+1}} > G_{j_{s+1}-1} = (S_{j_s} + a_{j_s} + L) - \left(S_{j_s} + \sum_{i=j_s}^{j_{s+1}-1} a_i + \sum_{i=j_s}^{j_{s+1}-2} G_i \right)$$

$$= L - \sum_{i=j_s+1}^{j_{s+1}-1} a_i - \sum_{i=j_s}^{j_{s+1}-2} G_i \geq L - \sum_{i=j_s+1}^{j_{s+1}-1} a_i - \sum_{i=j_s}^{j_{s+1}-2} b_i,$$

where the last inequality follows from Lemma 7. After rearrangement, we get $\sum_{i=j_s+1}^{j_{s+1}} a_i + \sum_{i=j_s}^{j_{s+1}-2} b_i > L$.

In case of (b), we have

$$L < \sum_{i=j_s}^{j_{s+1}-1} b_i + \sum_{i=j_s}^{j_{s+1}-2} G_i' \leq \sum_{i=j_s}^{j_{s+1}-1} b_i + \sum_{i=j_s+1}^{j_{s+1}-1} a_i,$$

where the last inequality follows from Lemma 7.

For each block, at least one of the previous inequalities holds, and neither of those inequalities contains any task occurring in any other inequality of another block. Hence, summing the valid inequalities for the first $s - 1$ blocks, we have

$$(s - 1)L < \sum_{i=1}^{j_s-1} (a_i + b_i) + a_{j_s}. \tag{2}$$

Applying Eqs. 1 and 2 to bound the completion time C_j of a job j in B_s, we get

$$C_j \leq \sum_{k=1}^{s-1} H_k + \sum_{i=j_s}^{j} (a_i + b_i) + L \leq \sum_{i=1}^{j} (a_i + b_i) + sL$$

$$< 2 \sum_{i=1}^{j} (a_i + b_i) + L \leq 3 C_j^{OPT_f},$$

where the last inequality follows from Lemma 1 and from $L < C_j^{OPT_f}$. Summing over all jobs, the Theorem follows. □

Theorem 8 *Algorithm B is a factor-1.5 approximation for* $1\|(p_j, L, p_j)\| \sum C_j$.

Proof Due to Lemma 6, it remains to prove that $C^{ALG} \leq 1.5 C^{OPT}$.

Consider an arbitrary optimal solution. Note that, in the present case, the non-decreasing $a_j + b_j$ order is the same as the non-decreasing a_j order. Thus, we can use both lower
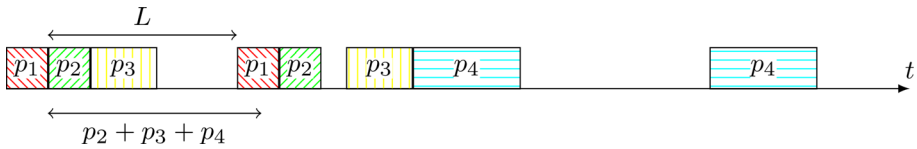
Fig. 9 Since $p_2 + p_3 + p_4 > L$, job j_4 has to be scheduled in a new block

bounds on the optimum described in Sect. 4.1. Also, $C_j^{OPT_f} = C_j^{OPT_s}$ follows directly from all delay times being fixed.

Consider jobs j and $j + 1$ from the same block. From Lemma 7, and from $p_j \leq p_{j+1}$, we know that there is no gap between a_j and a_{j+1} in σ^{ALG}. Thus, the length of a block B_s can be expressed as

$$H_s = \sum_{k=j_s}^{j_{s+1}-1} p_k + L + p_{j_{s+1}-1}. \tag{3}$$

Since j_s could not be scheduled in B_{s-1}, we have

$$\sum_{k=j_{s-1}+1}^{j_s} p_k > L, \ \ s \geq 2, \tag{4}$$

see Fig. 9.

In the remaining part of the proof we compare C_j and $C_j^{OPT_f}$, i.e., the completion time of job j in σ^{ALG} and the completion time of the jth finishing (or starting) job in a fixed optimal schedule. For all $j > j_2$, we will prove $C_j \leq 1.5 C_j^{OPT_f}$, but this inequality does not necessarily always hold true for $j = j_2$, where j_2 is the first job of the second block in σ^{ALG}. However, with a more sophisticated analysis, we still manage to prove $C^{ALG} \leq 1.5 C^{OPT}$ over the total of all jobs.

Let job j be a job in some block B_s, where $s \geq 2$ and $j \neq j_2$. Then the completion time C_j of j can be expressed as:

$$C_j = \sum_{i=1}^{s-1} H_i + \sum_{k=j_s}^{j} p_k + L + p_j.$$

Using Eq. 3, we get

$$C_j < \sum_{i=1}^{s-1} \left(\sum_{k=j_i}^{j_{i+1}-1} p_k + L + p_{j_{i+1}-1} \right) + \sum_{k=j_s}^{j} p_k + L + p_j \leq \sum_{i=1}^{j} p_i + \sum_{i=1}^{s-1} p_{j_{i+1}-1} + sL + p_j.$$

Applying Inequality 4 once for blocks $B_1, \ldots, B_{s-1}$, we get:

$$C_j < p_1 + 2 \sum_{k=2}^{j_s} p_k + \sum_{i=1}^{s-1} p_{j_{i+1}-1} + \sum_{k=j_s+1}^{j-1} p_k + 2p_j + L.$$

Applying it again for B_1, and then using $p_{j_2-1} \leq p_j$ (since $j > j_2$), we have

$$C_j < p_1 + 3 \sum_{k=2}^{j_2} p_k + 2 \sum_{k=j_2+1}^{j_s} p_k + \sum_{i=1}^{s-1} p_{j_{i+1}-1} + \sum_{k=j_s+1}^{j-1} p_k + 2p_j$$

$$\leq p_1 + 3\sum_{k=2}^{j_2} p_k + 2\sum_{k=j_2+1}^{j_s} p_k + \sum_{i=2}^{s-1} p_{j_{i+1}-1} + \sum_{k=j_s+1}^{j-1} p_k + 3p_j$$

$$\leq 3\sum_{k=1}^{j} p_k.$$

Using Lemma 1 on this statement, we have $C_j < 1.5C_j^{OPT_f}$ and thus,

$$\sum_{j>j_2} C_j < 1.5\sum_{j>j_2} C_j^{OPT_f}. \tag{5}$$

If j is in B_1, we have $C_j = \sum_{k=1}^{j} p_k + L + p_j \leq C_j^{OPT_f}$, where the last inequality follows from Lemma 2.

Now, if $j = j_2$, we then have $C_{j_2} = H_1 + 2p_{j_2} + L = \sum_{k=1}^{j_2} p_k + p_{j_2-1} + p_{j_2} + 2L$ (from Eq. 3) and $C_{j_2}^{OPT_f} \geq \sum_{k=1}^{j_2} p_k + L + p_{j_2}$ (from Lemma 2). Hence,

$$1.5\left(C_1^{OPT_f} + C_{j_2}^{OPT_f}\right) \geq 1.5\left(2p_1 + L + \sum_{k=1}^{j_2} p_k + L + p_{j_2}\right)$$

$$= 3p_1 + 3L + 1.5p_{j_2} + 1.5\sum_{k=1}^{j_2} p_k \geq 2p_1 + 3L + p_{j_2} + \sum_{k=1}^{j_2} p_k + 0.5(p_{j_2-1} + p_{j_2}) \tag{6}$$

$$\geq 2p_1 + 3L + p_{j_2} + \sum_{k=1}^{j_2} p_k + p_{j_2-1},$$

where the last inequality follows from $p_{j_2-1} \leq p_{j_2}$. Thus, we have

$$\sum_{k=1}^{j_2} C_k = \left(C_1 + C_{j_2}\right) + \sum_{k=2}^{j_2-1} C_j$$

$$\leq \left(2p_1 + L + \sum_{k=1}^{j_2} p_k + p_{j_2-1} + p_{j_2} + 2L\right) + \sum_{k=2}^{j_2-1} C_j^{OPT_f}$$

$$\leq 1.5\left(C_1^{OPT_f} + C_{j_2}^{OPT_f}\right) + \sum_{k=2}^{j_2-1} C_j^{OPT_f} \leq 1.5\sum_{k=1}^{j_2} C_j^{OPT_f},$$

where the second inequality follows from Eq. 6. Therefore, $C^{ALG} = \sum_{j=1}^{n} C_j \leq 1.5\sum_{j=1}^{n} C_j^{OPT_f} = 1.5C^{OPT}$, following from the previous statement and Eq. 5. □

4.4 CTP with related processing and delay times

In this section we consider variants where at least one of the tasks has a processing time equal to the delay time. We first reuse Algorithm A:

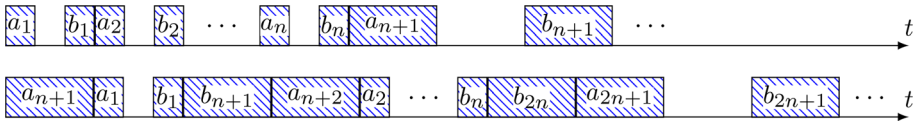**Theorem 9** *Algorithm A is a factor-1.5 approximation for* $1\|(p_j, p_j, p_j)\|\sum C_j$.

Fig. 10 The schedules considered in Remark 1

Proof Due to Lemma 4, it remains to prove that $C^{ALG} \leq 1.5 C^{OPT}$. Observe that both first and the second tasks are in non-decreasing p_j order in σ^{ALG}. This means the completion time C_j of job j is at most $3 \sum_{i=1}^{j} p_i$. Thus, we have $C^{ALG} = \sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} \left(3 \sum_{i=1}^{j} p_i \right) \leq 1.5 C^{OPT}$, where the last inequality follows from Lemma 1. $\square$

Remark 1 Consider an instance with $2n + \lfloor \sqrt{2}n \rfloor$ jobs. For $j = 1, \ldots, n$, let $p_j = 1 + j\varepsilon$, while for $j > n$ let $p_j = 3 + 3(j - n)\varepsilon$. For this instance, the algorithm creates a schedule, where the jobs are in increasing index order and the first task of a job always starts right after the second task of the previous job. The objective function of this schedule tends to $(18 + 12 \cdot \sqrt{2})n^2 + O(n)$ as n tends to infinity and ε tends to 0. Consider the schedule where job $n + 1$ starts from 0, each first task of a job with $j > n + 1$ starts right after the second task of job $j - 1$, and the first task of a job with $j \leq n$ starts right after the first task of job $j + n$, see Fig. 10. The objective function of this schedule tends to $(18 + 9 \cdot \sqrt{2})n^2 + O(n)$ as n tends to infinity and ε tends to 0. This shows that the approximation factor of our algorithm can generally not be better than $(2 + \sqrt{2})/3 \approx 1.138$.

Now, consider Algorithm C.

Algorithm C

Input: a CTP instance with $L_j = b_j$ for each job j
Output: a schedule σ for this instance

1. Sort the jobs in non-decreasing $a_j + p_j$ order. In the following, let the jobs be indexed in this order.
2. Schedule the jobs one-by-one as soon as possible without intersection.

Theorem 10 *Algorithm C is a factor-2 approximation for* $1\|(a_j, p_j, p_j)\| \sum C_j$.

Proof It is straightforward that Algorithm C runs in $\mathcal{O}(n \log n)$ time and always produces a feasible solution. In the worst case, the algorithm schedules each job right after the second task of the previously scheduled job. Hence, we have $C^{ALG} \leq \sum_{j=1}^{n} \sum_{i=1}^{j} (a_j + 2p_j)$. Since $C^{OPT} \geq \sum_{j=1}^{n} \sum_{i=1}^{j} (a_j + p_j)$ from Lemma 1, the Theorem follows. $\square$

Remark 2 Consider an instance with $2n$ jobs. For $k = 1, \ldots, n$, let $a_j = p_j = 1 + (k - 1)\varepsilon$, if $j = 2k - 1$; and $a_j = 1 + k\varepsilon$ and $p_j = 1 + (k - 1)\varepsilon$, if $j = 2k$. For this instance, the algorithm creates a schedule where the first task of a job starts right after the second task of the previously scheduled job. The objective value of this schedule is $6n^2 + \mathcal{O}(n)$. There exists a solution, by scheduling the jobs as soon as possible in decreasing index order, where the objective value is $4n^2 + \mathcal{O}(n)$. This shows that the approximation factor of our algorithm can generally not be better than 1.5.

If we modify the input, as well as the first step of Algorithm C, such that it takes instances of CTP with $L_j = a_j$, and sorts the jobs in non-decreasing $p_j + b_j$ order, we can approximate $1\|(p_j, p_j, b_j)\| \sum C_j$ with factor of 2. The proof is analogous to the proof of Theorem 10.

Theorem 11 *Modified Algorithm C is a factor-*2 *approximation for* $1\|(p_j, p_j, b_j)\| \sum C_j$.

Remark 3 An instance similar to the one described in Remark 2 shows that the approximation factor of the modified algorithm can generally not be better than 1.5.

5 Bi-objective approximation

In this section, we give constant-factor (ρ_1, ρ_2)-approximations for all variants of the bi-objective $1\|(a_j, L_j, b_j)\|\{C_{\max}, \sum C_j\}$ problem for which we gave constant factor approximations on, the $\sum C_j$-objective in this work.

Stein and Wein (1997) defined two simple conditions on scheduling problems: *Truncation* (deleting jobs from a valid schedule results in a valid partial schedule) and *Composition* (a simple way of appending two valid partial schedules results in a valid schedule) and proved the following:

Proposition 1 (Stein and Wein (1997), Corollary 3) *For any scheduling problem satisfying the conditions Truncation and Composition, if there exists an α-approximation algorithm for the minimization of makespan and a β-approximation algorithm for the minimization of sum of completion times, there exists an $(\alpha(1 + \delta), \beta(\frac{\delta+1}{\delta}))$-algorithm for any $\delta > 0$.*

Note that all considered coupled task problem variants fulfill these conditions. With this result in hand, we can now combine our β-approximation algorithms for the sum of completion times with previous α-approximation algorithms for the minimization of makespan to get (ρ_1, ρ_2)-approximations for all approximated $\sum C_j$ problems. We choose δ in such a way that the maximum of the two approximation factor $\max(\rho_1, \rho_2)$ is minimized. This is a common choice in bi-objective optimization as the goal is to get the best balanced result for both objectives simultaneously.

We give these results in Table 1. The first column of the table specifies the specific variant of $1\|(a_j, L_j, b_j)\|\{C_{\max}, \sum C_j\}$ to be approximated, with the variant identified by its job characteristics. The second column gives the (ρ_1, ρ_2)-approximation factor for each variant. As in our case ρ_1 always equals ρ_2, we just give one value in this column. In the remaining columns we give the specific α and β values used in Proposition 1, with a reference to their origin, as well as our choice of δ.

The run time of this algorithm implied by Proposition 1 is the sum of the run times of both the α- and the β-approximation algorithms. As all used approximation algorithms for both $C_{\max}$ and $\sum C_j$ problems run in polynomial time, all (ρ_1, ρ_2)-approximations given in Table 1 can be computed in polynomial time as well.

6 Conclusion

In this paper, we deal with the single machine coupled task scheduling problem, with the minimization of the total completion time as our objective function. Our work extends the complexity results of Chen and Zhang (2021) and Kubiak (2022) by introducing two new $\mathcal{NP}$-hard variants, and provides several polynomial-time constant-factor approximation algorithms. To do this, we were able to modify several known algorithmic concepts used in coupled task makespan minimization, but some of our proofs on approximation factors required more sophisticated ideas. E.g., in the proof of Lemma 8, the original idea for the approximation factor only worked for jobs scheduled after a certain number of other jobs had already been

Table 1 Results on bi-objective (ρ_1, ρ_2)-approximation for CPT variants with $C_{\max}$ and $\sum C_j$ objectives

	$\rho_1 = \rho_2$	α	β	δ
(a, L_j, b)	6.5	3.5 (Ageev and Kononov, 2006)	3 (Theorem 5)	6/7
(a_j, L, b_j)	6	3 (Ageev and Ivanov, 2016)	3 (Theorem 7)	1
$(a, L_j, b, b \le a)$	5.5	3.5 (Ageev and Kononov, 2006)	2 (Theorem 4)	4/7
(a_j, p_j, p_j)	5.5	3.5 (Ageev and Kononov, 2006)	2 (Theorem 10)	4/7
(p_j, p_j, b_j)	5.5	3.5 (Ageev and Kononov, 2006)	2 (Theorem 11)	4/7
(p_j, p_j, p_j)	4	2.5 (Ageev and Kononov, 2006)	1.5 (Theorem 9)	3/5
$(1, L_j, 1)$	3.25	1.75 (Ageev & Baburin, 2007)	1.5 (Theorem 6)	6/7
(p_j, L, p_j)	3	1.5 (Ageev and Ivanov, 2016)	1.5 (Theorem 8)	1

processed, and a careful analysis of the approximation factor for the jobs before this cut-off was needed to get the result on hand.

We also give the first results on bi-objective approximation in the coupled task setting: we use a result from Stein and Wein (1997), together with constant-factor approximations on the makespan objective taken from the literature, to give bi-objective constant-factor approximations for the problem of both minimizing the sum of completion time and the makespan simultaneously. We do this for all variants of CTP with the sum of completion times objective that we managed to approximate with a constant factor.

Although we did manage to provide approximation algorithms for several coupled task scheduling problem variants with the sum of completion times objective, it is still unknown if there is a constant-factor approximation algorithm for a few important cases: for the most general case; for the cases where only one of the tasks has a fixed processing time; and for the (p_j, L_j, p_j) variant. This stems from the fact that all algorithms presented in this work make use of some unique ordering of the jobs implied by the job characteristics on either the task lengths or delay lengths. In the aforementioned cases, there exists no such unique ordering using only task lengths or delay lengths. Inapproximability results also are of interest for the problems presented in this paper, as they do exist for most CTP variants with the makespan objective (Ageev and Kononov 2006; Ageev and Ivanov 2016). The ideas of these papers were useful for achieving complexity results, but it seems to us that new approaches are needed for the total completion time objective. To our best knowledge, there are no such results. While we have proved some lower bounds on the best approximation factors, the tightness of our algorithms is still open, which invites for more sophisticated analyses. We point to these three open questions as suggestions for future research.

Author Contributions Both authors contributed equally to this work.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Ageev, A., & Ivanov, M. (2016). Approximating coupled-task scheduling problems with equal exact delays. In *Discrete optimization and operations research, volume 9869 of lecture notes in computer science* (pp. 259–271).

Ageev, A. A., & Baburin, A. E. (2007). Approximation algorithms for UET scheduling problems with exact delays. *Operations Research Letters, 35*(4), 533–540.

Ageev, A. A., & Kononov, A. V. (2006). Approximation algorithms for scheduling problems with exact delays. In *Approximation and online algorithms, volume 4368 of Lecture Notes in Computer Science* (pp. 1–14).

Angel, E., Bampis, E., & Kononov, A. (2003). On the approximate tradeoff for bicriteria batching and parallel machine scheduling problems. *Theoretical Computer Science, 306*(1), 319–338.

Bampis, E., & Kononov, A. (2005). Bicriteria approximation algorithms for scheduling problems with communications delays. *Journal of Scheduling, 8*(4), 281–294.

Bessy, S., & Giroudeau, R. (2019). Parameterized complexity of a coupled-task scheduling problem. *Journal of Scheduling, 22*(3), 305–313.

Békési, J., Dósa, G., & Galambos, G. (2022). A first fit type algorithm for the coupled task scheduling problem with unit execution time and two exact delays. *European Journal of Operational Research, 297*(3), 844–852.

Chen, B., & Zhang, X. (2021). Scheduling coupled tasks with exact delays for minimum total job completion time. *Journal of Scheduling, 24*(2), 209–221.

Condotta, A., & Shakhlevich, N. V. (2012). Scheduling coupled-operation jobs with exact time-lags. *Discrete Applied Mathematics, 160*(16–17), 2370–2388.

Deb, K. (2014). *Multi-objective Optimization* (pp. 403–449).

Elshafei, M., Sherali, H. D., & Smith, J. C. (2004). Radar pulse interleaving for multi-target tracking. *Naval Research Logistics, 51*, 72–94.

Farina, A. (1980). Multitarget interleaved tracking for phased-array radar. In *IEE Proceedings F (Communications, Radar and Signal Processing)* (Vol. 127, No. 4, pp. 312-318). IET Digital Library.

Garey, M., & Johnson, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326.

Hoogeveen, H. (2005). Multicriteria scheduling. *European Journal of Operational Research, 167*(592–623), 12.

Huo, Y., Li, H., & Zhao, H. (2009). Minimizing total completion time in two-machine flow shops with exact delays. *Computers and Operation Research, 36*(6), 2018–2030.

Hwang, F. J., & Lin, B. M. T. (2011). Coupled-task scheduling on a single machine subject to a fixed-job-sequence. *Computers and Industrial Engineering, 60*(4), 690–698.

Jiang, X., Lee, K., & Pinedo, M. L. (2023). Approximation algorithms for bicriteria scheduling problems on identical parallel machines for makespan and total completion time. *European Journal of Operational Research, 305*(2), 594–607.

Khatami, M., & Salehipour, A. (2021a). A binary search algorithm for the general coupled task scheduling problem. *4OR, 19*, 1–19.

Khatami, M., & Salehipour, A. (2021). Coupled task scheduling with time-dependent processing times. *Journal of Scheduling, 24*, 223–236.

Khatami, M., Salehipour, A., & Cheng, T. (2020). Coupled task scheduling with exact delays: Literature review and models. *European Journal of Operational Research, 282*(1), 19–39.

Kubiak, W. (2022). A note on scheduling coupled tasks for minimum total completion time. *Annals of Operations Research* 1–4.

Leung, J.Y.-T., Li, H., & Zhao, H. (2007). Scheduling two-machine flow shops with exact delays. *International Journal of Foundations of Computer Science, 18*, 341–359.

Orman, A. J., & Potts, C. N. (1997). On the complexity of coupled-task scheduling. *Discrete Applied Mathematics, 72*(1–2), 141–154.

Shapiro, R. D. (1980). Scheduling coupled tasks. *Naval Research Logistics Quarterly, 27*(3), 489–498.

Simonin, G., Giroudeau, R., & König, J. (2011). Complexity and approximation for scheduling problem for a torpedo. *Computers & Industrial Engineering, 61*(2), 352–356.

Stein, C., & Wein, J. (1997). On the existence of schedules that are near-optimal for both makespan and total weighted completion time. *Operations Research Letters, 21*(3), 115–122.

Yu, W., Hoogeveen, H., & Lenstra, J. K. (2004). Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *Journal of Scheduling, 7*(5), 333–348.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.