An Approximation Algorithm for the Three-Machine Scheduling Problem with the Routes Given by the Same Partial Order

Richard Quibell and Vitaly A. Strusevich
School of Computing and Mathematical Sciences, University of Greenwich,
Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, U.K.
e-mail: dick@quibell.demon.co.uk, V.Strusevich@greenwich.ac.uk

Abstract

The paper considers a three-machine shop scheduling problem to minimize the makespan, in which the route of a job should be feasible with respect to a machine precedence digraph with three nodes and one arc. For this NP-hard problem that is related to the classical flow shop and open shop models, we present a simple 1.5-approximation algorithm and an improved 1.4-approximation algorithm.

Keywords: shop scheduling, makespan minimization, partially ordered route, approximation

1 Introduction

In multi-stage scheduling problems, we are given a set $N = \{1, 2, \ldots, n\}$ of jobs that have to be processed in a shop consisting of m machines $M_1, M_2, \ldots, M_m$. Processing each job involves several operations, and each operation has to be performed on a specific machine. The processing times of all operations are given. The order of operations of an individual job are defined by the *processing routes*. The classical scheduling models classified according to a type of processing route are as follows:

flow shop: all jobs have the same route, usually given by the sequence $(M_1, M_2, \ldots, M_m)$;

job shop: the jobs are in advance given different routes defined by arbitrary sequences of machines; some machines are allowed to be missing in a route, some are allowed to be visited more than once;

open shop: the routes are not fixed and the operations of a job can be performed in an arbitrary order, different jobs being allowed to obtain different routes.

See books [1, 2, 3] and surveys [4, 5] for the review of major results on classical shop scheduling.

There are several types of enhanced shop models. One type of such an enhancement allows jobs with both fixed and non-fixed routes. In a *mixed* shop, some jobs are processed according to the same processing route (as in a flow shop) and the other jobs for which the routes are not fixed (as in an open shop). A more general model, sometimes called the

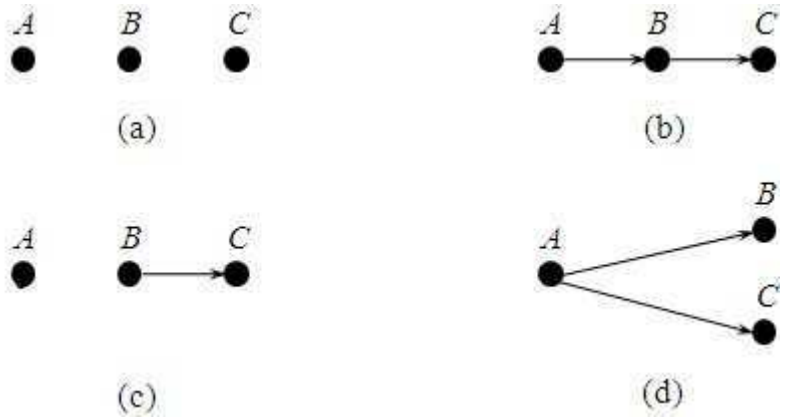

Figure 1: Three machine dags for: (a) open shop; (b) flow shop; (c) combo1 shop and (d) combo2 shop

super shop, can be seen as a job shop with some extra jobs which are processed as in an open shop. See [6] and [7] for studies on mixed shop and super shop problems.

Another type of enhancement allows the processing routes to be given by partially ordered sequences of the machines. The classical models correspond to two extreme types of order: the linear order for the flow shop and job shop, and no order for the open shop. For the machine-enhanced shop scheduling models, each job should be assigned a route that is feasible with respect to given partial order. Such an order is usually represented by a directed machine precedence graph, in which the set of vertices coincides with the set of machines, and the arc goes from vertex M_p to vertex M_q if and only if in any feasible schedule the job has to be first processed on machine M_p and then on machine M_q. Such a graph must be acyclic, and all transitive arcs can be removed from it without any loss of information. Since for the described model the routes are given in the form of directed acyclic graphs (d.a.g.), some authors call this model the *dag* shop.

In this paper, we mainly deal with a three-machine shop models, and call the machines A, B and C. The model of our primary concern is one of the simplest three-machine dag shop models, which bears some features of the flow shop and the open shop. The only restriction on the processing routes is that each job must visit machine B before machine C, different jobs being allowed to be assigned different feasible routes. Thus, for all jobs the routes are given by the same dag that contains exactly one arc going from vertex B to vertex C. We call this model the combo1 shop, as opposed to the combo2 shop, for which the routes are given by the same dag, that contains exactly two arcs going from vertex A. Figure 1 shows the machine precedence graphs for the all three-machine models in which for all jobs the processing routes are defined by the same dag.

ftbpFU3.557in1.913in0ptThree machine dags for: (a) open shop; (b) flow shop; (c) combo1 shop and (d) combo2 shopFigShopsFigure

Given a feasible schedule S which satisfies all processing requirements of the chosen scheduling system, let $C_{\max}(S)$ denote the *makespan* of schedule S, i.e., the maximum completion time by which all jobs are completed on all machines. For all scheduling problems considered in this paper the objective is to minimize the makespan. The main purpose of this paper is to present an algorithm that for the three-machine combo1 shop problem finds a schedule with a makespan that is at most 1.4 times the optimal value.

The remainder of this paper is organized as follows. We start with a concise survey of complexity and approximability results for the classical shop scheduling problems, followed by a formal description of the three-machine combo1 shop problem. Further, the complexity issue of the combo1 shop problem is resolved. A $\frac{7}{5}-$approximation algorithm for the combo1 shop problem, analysis of its performance and the tightness issues are contained in three subsequent sections.

2 Shop Problems: A Review

In this section, we give a concise overview of complexity and approximability results for the shop scheduling problems to minimize the makespan. We restrict our attention to the models, in which no machine appears twice in the processing route of any job.

We are given a set $N = \{1, 2, \dots n\}$ of jobs to be processed on m machines M_1, $M_2, \dots, M_m$. Each job $j \in N$ consists of at most m operations $O_{1,j}$, $O_{2,j}, \dots, O_{m,j}$. Operation $O_{i,j}$ is to be processed on machine M_i, and this takes $p_{i,j}$ time. For job j, the order of operations is $(O_{1,j}, O_{2,j}, \dots, O_{m,j})$ (for the flow shop), or is given by a predefined sequence (for the job shop), or is not fixed in advance (for the open shop). It is not allowed to process more than one operation of the same job at a time. Also, a machine processes at most one operation at a time. The objective is to find a schedule that minimizes the makespan $C_{\max}$.

In this paper, we assume that in the processing of any operation preemption is not allowed, i.e., once started, every operation is performed to completion without interruption. Following [4], we use notation $\alpha m|op \le m'|C_{\max}$ to refer to $m-$machine shop scheduling problems to minimize the makespan, where α in the first field denotes a type of machine environment ($\alpha = F$ for the flow shop, $\alpha = J$ for the job shop, and $\alpha = O$ for the open shop), while $op \le m'$ reflects a requirement that the number of operations in a route does not exceed the given value $m' \le m$ (if it is missing, there are up to m operations in the processing route of any job).

Problems $F2|\,|C_{\max}$ and $J2|op \le 2|C_{\max}$ are solvable in $O(n \log n)$ time due to Johnson [8] and Jackson [9], respectively. Several linear time algorithms are known for problem $O2|\,|C_{\max}$, the historically the first belongs to Gonzalez and Sahni [10]. Each of the two-machine mixed shop and super shop problems admits an $O(n \log n)$-time algorithm, see [6] and [7], respectively.

Problem $Fm|\,|C_{\max}$ is NP-hard in the strong sense for $m \ge 3$ as proved by Garey, Johnson and Sethi [11]. Problem $F3|op \le 2|C_{\max}$ remains NP-hard in the strong sense [12], while the complexity status of problem $O3|op \le 2|C_{\max}$ is still open. Problem $O3|\,|C_{\max}$ is NP-hard in the ordinary sense, as proved by Gonzalez and Sahni [10]. It is still unknown whether problem $Om|\,|C_{\max}$ with a fixed number of machines $m \ge 3$ is NP-hard in the strong sense. If the number of machines is variable (part of the input) then the open shop problem is NP-hard in the strong sense. In fact, for both the flow shop and the open shop problems with variable number of machines and integer processing times, Williamson et al. [13] show that the decision problem to verify whether there exists a schedule S with $C_{\max}(S) \le 4$ is NP-complete in the strong sense.

Since most of shop scheduling problems with three and more machines are NP-hard, the design and analysis of approximation algorithms is an appealing topic of research. Usually the quality of approximation algorithms is measured by their worst-case performance ratios. An algorithm H that creates a schedule S_H is said to provide a *ratio performance guarantee*

ρ, if for any instance of the problem the inequality

$$C_{\max}(S_H)/C_{\max}(S^*) \leq \rho$$

holds. A performance guarantee is called *tight* if there exists an instance of the problem such that either $C_{\max}(S_H)/C_{\max}(S^*) = \rho$ or at least $C_{\max}(S_H)/C_{\max}(S^*) \to \rho$ when some of the processing times approach zero or infinity. A polynomial-time heuristic with a worst-case performance ratio of ρ is called a $\rho-$approximation algorithm. A polynomial-time approximation scheme ($PTAS$) is a family of $(1+\varepsilon)-$approximation algorithms such that their running time is polynomial for fixed m and fixed positive ε.

Recall major results on approximation for relevant scheduling models with a fixed number of machines. For each of the problems $Om|\ |C_{\max}$ and $Fm|\ |C_{\max}$ there exists a PTAS, see Sevastianov and Woeginger [15] and Hall [14], respectively. Recall that a PTAS has been offered for the general problem $Jm|\ |C_{\max}$ with a fixed number of operations per job [16]; moreover, the algorithm can be extended to handle the general dag shop problem. These results provide important theoretical evidence that for the classical shop problems heuristic schedules close to the optimum can be found in polynomial time; in fact, for each model above a PTAS is the best approximability result that one could hope for. Still, the running time of these algorithms, although polynomial, is not acceptable for practical needs even for small number of machines.

If the number of machines m is variable, then there are polynomial-algorithms with $\rho = 2$ for the open shop [17]; with $\rho = \lceil m/2 \rceil$ for the flow shop and with $\rho = m$ for the job shop [18]. For the job shop problem $J|op \leq m'|C_{\max}$ with no repeated machines in any processing route Feige and Schedideler [19] give a polynomial-time algorithm with $\rho = O\left(m'm \log\left(m'm\right) \log\log\left(m'm\right)\right)$, which improves the result by Shmoys et al. [20] developed for a general job shop. On the other hand, as follows from [13], for both the flow shop and the open shop problems there exists no polynomial-time algorithm with $\rho < 5/4$, unless $\mathcal{P} = \mathcal{NP}$.

Fast algorithms are available for problems with a small number of machines. For problem $F3|\ |C_{\max}$ a heuristic from [21] requires $O(n \log n)$ time and guarantees $\rho = 5/3$. Several linear time $3/2-$approximation algorithms for problem $O3|\ |C_{\max}$ are known, see, e.g., [22, 23]. For problem $J3|op \leq 2|C_{\max}$ and $J2|op \leq 3|C_{\max}$ there are algorithms that run in $O(n \log n)$ time and provide $\rho = 3/2$, see [24]. For the three-machine combo2 shop an algorithm from [25] runs in $O(n \log n)$ time and guarantees $\rho = 5/3$.

3 Combo1 Shop: Preliminaries

In this section, we give a formal description of the three-machine combo1 shop scheduling problem, which is the main subject of this study. We also establish relations of our problem with the two-machine flow shop scheduling problem. As a result, we derive a number of lower bounds on the optimal value of the makespan for the combo1 shop problem. These lower bounds are subsequently used in worst-case analysis of our heuristic algorithms.

The combo1 shop model can be defined as follows. We are given a set $N = \{1, 2, \ldots, n\}$ of jobs to be processed in the shop consisting of three machines A, B and C. Processing each job involves three operations $O_{A,j}, O_{B,j}$ and $O_{C,j}$. For a job $j \in N$, operation $O_{A,j}$ is processed on machine A, $O_{B,j}$ is processed on machine B, and $O_{C,j}$ is processed on machine C. The processing times of operations $O_{A,j}, O_{B,j}$ and $O_{C,j}$ are equal to a_j, b_j and c_j, respectively. The operations of the same job are not allowed to overlap. At a time, a

machine may process at most one operation. For any job $j \in N$, operation $O_{B,j}$ must be completed before operation $O_{C,j}$ may start. The order of operation $O_{A,j}$ with respect to operations $O_{B,j}$ and $O_{C,j}$ is not predefined and may be chosen arbitrarily. See Figure 1(c) for the machine precedence graph for this model. The combo1 shop has features of the flow shop (machine B precedes machine C in any feasible route), as well as features of the open shop (each pair of machines, A and B, as well as A and C, essentially forms a two-machine open shop). Therefore, we have chosen to denote the problem of minimizing the makespan for the three-machine combo1 shop by $A(BC)|\ |C_{\max}$.

Problem $A(BC)|\ |C_{\max}$ is NP-hard, even if we know the sequence of jobs on machines B and C in an optimal schedule. Given such a sequence, in order to complete an optimal schedule we still need to schedule the jobs on machine A relative to the optimal sequence on machines B and C. Even if we relax that situation and ignore one of the machines, e.g., machine C, we are still left with the problem of minimizing the makespan on machines A and B, provided that the sequence of jobs on machine B is fixed The latter problem is proved NP-hard in the ordinary sense in [26].

Temporarily ignore machine A, and consider a two-machine flow shop problem on machines B and C. It is well-known that for the resulting problem $F2|\ |C_{\max}$ there exists an optimal schedule in which the jobs are processed on both machines in the same sequence, and the sequence that minimizes the makespan on these machines can be found in $O(n \log n)$ time by Johnson's algorithm. Recall that Johnson's algorithm outputs the sequence which starts with the jobs with $b_j \leq c_j$ taken in non-decreasing order of b_j, followed by the remaining jobs taken in the non-increasing order of c_j; see [8]. Throughout this paper we assume that the jobs are renumbered in accordance with the Johnson's sequence on machines B and C. For a set of jobs $Q \subseteq N$, we denote by $\Phi_{BC}(Q)$ the optimal makespan of processing these jobs in the flow shop that consists of machines B and C. In particular,

$$\Phi_{BC}(N) = \max_{1 \leq \mu \leq n} \left\{ \sum_{j=1}^{\mu} b_j + \sum_{j=\mu}^{n} c_j \right\}. \tag{1}$$

If the maximum in (1) is attained for $\mu = u$ then job u is called *critical*. For the flow shop, a critical job starts processing on C at the same time its processing on B is completed.

Since every job k contributes at least either b_k or c_k to the overall makespan $\Phi_{BC}(N)$, it follows that

$$\Phi_{BC}(N \setminus \{k\}) + \min\{b_k, c_k\} \leq \Phi_{BC}(N). \tag{2}$$

We now discuss various lower bounds on the value of the makespan for problem $A\,(BC)\,|\ |C_{\max}$. For a non-empty subset $Q \subseteq N$, denote

$$a(Q) = \sum_{j \in Q} a_j,$$

and define $a(\emptyset) = 0$. The values $b(Q)$ and $c(Q)$ are defined analogously.

Let S^* be an optimal schedule. The so-called machine-based bound is apparent:

$$C_{\max}(S^*) \geq a(N), \tag{3}$$

while the job-based bound is given by

$$C_{\max}(S^*) \geq \max\{a_j + b_j + c_j | j \in N\}. \tag{4}$$

Additional lower bounds come from the fact that the original three-machine shop contains a two-machine flow shop, i.e.,

$$C_{\max}(S^*) \geq \Phi_{BC}(N) \geq \max\{b(N), c(N)\}. \tag{5}$$

Define

$$LB = \max\left\{a(N), \Phi_{BC}(N), \max\{a_j + b_j + c_j | j \in N\}\right\}. \tag{6}$$

For any schedule S, let $F_L(Q)$ denote the completion time of the last job of set $Q \subseteq N$ on machine $L \in \{A, B, C\}$. Without loss of clarity, we often write F_L rather than $F_L(N)$. It is clear that $C_{\max}(S) = \max\{F_A, F_C\}$.

4 Initial Schedules and $\frac{3}{2}-$Approximation

Let λ, $0 < \lambda < 1$, be a given number. In this section, we consider two classes of instances of problem $A(BC)|\ |C_{\max}$, that depend on the presence of a job, for which the the processing time of its $A-$operation exceeds $\lambda \cdot LB$. We show that our approach, if implemented with $\lambda = \frac{1}{2}$, immediately leads to a $\frac{3}{2}-$approximation algorithm. We also deduce the conditions that describe the instances of the problem which require additional consideration in order to admit a more accurate $\frac{7}{5}-$approximation algorithm.

4.1 A Long $A-$Operation

Given a λ, $0 < \lambda < 1$, assume that that there exists a job p with

$$a_p \geq \lambda \cdot LB, \tag{7}$$

where LB is defined by (6). The algorithm described below finds a schedule S_p such that

$$\frac{C_{\max}(S_p)}{C_{\max}(S^*)} \leq 2 - \lambda. \tag{8}$$

In the description of this and subsequent algorithm we often refer to processing jobs as blocks. In a *block*, the jobs are processed on a particular machine according to a given sequence without intermediate idle time. The algorithm below starts the long operation $O_{A,p}$ and an optimal flow shop schedule of the remaining jobs on machine B and C at time zero. The other operations are appended to avoid clashes.

Algorithm P

Step 1. Find schedule $S_{BC}(N \setminus \{p\})$, an optimal flow shop schedule of the jobs of set $N \setminus \{p\}$ on machines B and C.

Step 2. From time zero, start job p on A and run schedule $S_{BC}(N \setminus \{p\})$ on B and C.

Step 3. Start the block of jobs $N \setminus \{p\}$ sequenced in any order on A at time $\max\{a_p, \Phi_{BC}(N \setminus \{p\})\}$.

Step 4. Start job p on B as early as possible, i.e., at time $\max\{a_p, b(N \setminus \{p\})\}$. Start job p on C as early as possible, i.e., at time $\max\{F_B, \Phi_{BC}(N \setminus \{p\})\}$.

Step 5. Call the resulting schedule S_p. Stop.

Lemma 1 *If an instance of the problem contains a job p that satisfies (7), then Algorithm P finds a schedule S_p for which (8) holds.*

Proof: It is easy to check that in schedule S_p there are no clashes, i.e., no job is processed on more than one machine at a time. Besides, either $F_A = a(N) \leq LB$ or $F_A = \Phi_{BC}(N \setminus \{p\}) + a(N \setminus \{p\}) \leq \Phi_{BC}(N \setminus \{p\}) + a(N) - a_p \leq (2-\lambda)\, LB$.

It follows that $F_B = \max\{a_p + b_p, b(N)\}$, so that

$$\begin{aligned} F_C &= \max\{a_p + b_p, b(N), \Phi_{BC}(N \setminus \{p\})\} + c_p \\ &\leq \max\{a_p + b_p + c_p, LB + c_p\} \leq (2-\lambda)\, LB, \end{aligned}$$

since $c_p \leq (a_p + b_p + c_p) - a_p \leq (1-\lambda)\, LB$. ∎

4.2 A Standard Schedule and Its Transformations

Assume that the jobs are renumbered in accordance with a permutation that corresponds to an optimal flow shop schedule on machines B and C. The following schedule plays an important role in our development of an approximation algorithm for problem $A\,(BC)\,||\,C_{\max}$. This is a flow shop schedule, in which (i) all three machines process the jobs in accordance with the permutation $(1, 2, \ldots, n)$ and (ii) each machine, once started does not have any intermediate idle time. The latter condition is known as "no-idle" [27]. In what follows, we call that schedule *standard* and denote it by S_0. Using the argument in [27], we derive

$$C_{\max}(S_0) = \max_{1\leq\mu\leq n}\left\{\sum_{j=1}^{\mu} a_j + \sum_{j=\mu}^{n} b_j\right\} + \max_{1\leq\nu\leq n}\left\{\sum_{j=1}^{\nu} b_j + \sum_{j=\nu}^{n} c_j\right\} - b(N). \tag{9}$$

For schedule S_0, let R_L denote the start time of uninterrupted processing on machine $L \in \{A, B, C\}$. Define three jobs that are structurally important in schedule S_0 :

Job u : This job is critical for machines A and B, i.e., u is the value of μ that delivers the maximum to the first term of the right-hand side of (9);

Job v : This job is critical for machines B and C, i.e., v is the value of ν that delivers the maximum to the second term of the right-hand side of (9);

Job w : This job is the first job to complete on machine A no earlier than time R_B; notice that $w \leq u$ in all cases.

ftbpFU5.4345in1.9761in0ptStructure of schedule S_0 : (a) $u \leq v$; (b) $u > v$FigS0uvFigure

Figure **??** shows two typical structures of schedule S_0, depending on mutual positions of the two critical jobs.

Let N_1 and N_2 denote the sets $\{1, 2, \ldots, w-1\}$ and $\{w+1, \ldots, n\}$, respectively. For operation $O_{A,w}$, denote its duration from its start time in S_0 till time R_B by a'_w; define $a''_w = a_w - a'_w$.

To complement the case studied in Section 4.1, in the remainder of this section we assume that for a given $\lambda,\ 0 < \lambda < 1$, the inequalities

$$a_j \leq \lambda \cdot LB \tag{10}$$

hold for each $j \in N$.

4.2.1 Early Completion of Job w on Machine C

Assume that in schedule S_0 job w completes early, i.e.,

$$F_{C,w} \leq a(N). \tag{11}$$

Notice that if $w < v \leq u$ (as in Figure **??**(b)), then $F_{C,w} \leq F_{B,v} \leq F_{A,u} \leq a(N)$. Alternatively, inequality (11) may hold for $w < u < v$ (as in Figure **??**(a)).

We show how to find a schedule S_1 such that

$$\frac{C_{\max}(S_1)}{C_{\max}(S^*)} \leq 1 + \frac{\lambda}{2}. \tag{12}$$

The following algorithm transforms schedule S_0 by moving the block of jobs N_1, either together or without job w, to the end of the processing sequence on machine A. The conditions of an early completion of job w in schedule S_0 guarantee that these moves create no idle time on machine A.

ftbpFU4.0101in2.0271in0pt(a) schedule S', (b) schedule S''FigS1Figure

Algorithm EarlyW

Step 1. Given schedule S_0 that satisfies (11), find schedule S' obtained from schedule S_0 by moving the set of jobs N_1 to start on A at time $a(N)$, followed by reducing the start time of all operations so that operation $O_{A,w}$ starts at time zero.

Step 2. Find schedule S'' obtained from schedule S_0 by moving the set of jobs $N_1 \cup \{w\}$ to start on A at time $a(N)$, followed by reducing the start time of all operations so that machine B starts its uninterrupted processing at time zero.

Step 3. Output the better of the two found schedules as schedule S_1. Stop.

Lemma 2 *If in an instance of the problem all jobs satisfy (10) and for schedule S_0 inequality (11) holds, then Algorithm EarlyW finds a schedule S_1 which satisfies (12).*

Proof: The condition (11) implies that in each schedule S' and S'' the jobs that are moved on machine A start after they are completed on machine C; therefore, the move does not produce any clashes. For schedule S' (see Figure **??**(a)) we have that

$$F_A = a(N);\ F_C = a'_w + \Phi_{BC}(N),$$

while for schedule S'' (see Figure **??**(b)) we have that

$$F_A = a(N) + a''_w;\ F_C = \Phi_{BC}(N).$$

Thus, $C_{\max}(S_1) = \min\{C_{\max}(S'), C_{\max}(S'')\} \leq LB + \min\{a'_w, a''_w\} \leq LB + \frac{1}{2}a_w \leq \left(1 + \frac{\lambda}{2}\right) LB$, as required. ∎

4.2.2 Late Completion of job on machine C

Assume that in schedule S_0 job w completes late, i.e.,

$$F_{C,w} > a(N). \tag{13}$$

As established in Section 4.2.1, (13) implies that in S_0 job u is sequenced no later than job v. We split our consideration into two cases:

Case 1: $w < u \leq v$, and

Case 2 $w = u \leq v$.

We show that in either case schedule S_0 can be transformed into a schedule S_2 such that

$$\frac{C_{\max}(S_2)}{C_{\max}(S^*)} \leq 1 + \lambda. \tag{14}$$

The algorithm below, similarly to Algorithm EarlyW, moves the block of jobs N_1 together with job w to the end of the schedule, but due to a late completion of job w in schedule S_0 this move leaves a gap on machine A. The algorithm performs additional actions aimed at reducing that gap by successively inserting jobs into it, which can be accomplished without clashes. The length of the gap is thereby reduced to less than the processing time of any $A-$operation, so that (14) is satisfied.

Algorithm LateW1

Step 1. Given schedule S_0 that satisfies the conditions of Case 1, find schedule $\hat{S}$ obtained from schedule S_0 by moving the set of jobs $N_1 \cup \{w\}$ to start on A as a block as early as possible after job n. Reduce the start time of all operations so that machine B starts its uninterrupted processing at time $R_B\left(\hat{S}\right) = 0$. In schedule $\hat{S}$, let γ denote the length of the gap on machine A, i.e., the idle period $[G_1, G_2]$, where $G_1 = F_A(N_2)$ is the completion time of the block of jobs N_2 and G_2 is the start time of the block of jobs $N_1 \cup \{w\}$.

Step 2 If $\gamma \leq a'_w$, go to Step 4; otherwise determine a job $q \in N_1 \cup \{w\}$ such that in schedule $\hat{S}$ operation $O_{A,q}$ starts exactly when operation $O_{C,q}$ completes, i.e., $G_2 = F_{A,q} - \sum_{j=1}^{q} a_j$. If $q = w$ then go to Step 4; otherwise, go to Step 3.

Step 3. Determine the set $\hat{N} = \{q+1, \ldots, w\}$. Scanning the jobs of this set in the sequence w, $w-1, \ldots$ (opposite to their numbering) move their $A-$operations one by one to fill the gap $[G_1, G_2]$, starting from $O_{A,w}$ to start at time G_1 until one of the following happens:

(i) the length of the remaining gap does not exceed a'_w;

(ii) all operations are moved into the gap;

(iii) job $p > q$ is found with a_p greater than the length of the remaining gap.

If either outcome (i) or outcome (ii) occurs, go to Step 4. Otherwise, start operation $O_{A,p}$ immediately after the completion of the previous operation $O_{A,p+1}$ (or at time G_1 for $p = w$), followed by the block of $A-$ operations of the jobs $\{1, \ldots, p-1\}$. If required, delay the block of $C-$ operations of jobs $\{p, p+1, \ldots, w, \ldots, n\}$ to start at the completion time of operation $O_{A,p}$ and go to Step 4.

Step 4. Output the last found schedule as schedule S_2. Stop.

fhFU5.4189in2.738in0pt(a) schedule $\hat{S}$; (b) schedule found in Step 3(iii)FigLateW1Figure

Lemma 3 *If in an instance of the problem all jobs satisfy (10) and for schedule S_0 the conditions (13) and $w < u \leq v$ hold, then Algorithm LateW1 finds a schedule S_2 which satisfies (14).*

Proof: Let $\hat{S}$ be the schedule found in Step 1 of Algorithm LateW1; see Figure **??**(a). It follows that

$$F_A\left(\hat{S}\right) = a''_w + a(N) + \gamma;\ F_C\left(\hat{S}\right) = \Phi_{BC}(N) \leq LB.$$

so that we only need to be concerned with the completion time on machine A.

Thus, if $\gamma \leq a'_w$ (as in Step 2) then $F_A\left(\hat{S}\right) \leq a''_w + a(N) + a'_w = a_w + a(N) \leq (1+\lambda)\cdot LB$, and (14) holds. Thus, assume that $\gamma > a'_w$.

Job q found in Step 2 can be seen as the critical job in the flow shop schedule for the jobs of set $N_1 \cup \{w\}$, in which each job follows the processing route (C, A). If $q = w$ then no further transformation is required and $F_A\left(\hat{S}\right) = \Phi_{BC}(N_1 \cup \{w\}) + a_w \leq (1+\lambda)\cdot LB$. Thus, assume that $q < w$ and consider schedule S_2 found as a result of the transformations in Step 3.

First, notice that moving the $A-$operations into the gap $[G_1, G_2]$ does not create any conflicts, since the jobs of set $\hat{N}$ are completed on B before time G_1 (since $F_{w,B} \leq F_{v,A} \leq G_1$) and start after time G_2 on machine C (by construction).

In the case of outcome (i), we have that $F_A(S_2) \leq a''_w + a(N) + a'_w \leq (1+\lambda)\cdot LB$. In the case of outcome (ii), we have that $F_A(S_2) = \Phi_{BC}(\{1,\ldots,q\}) + a_q \leq (1+\lambda)\cdot LB$. In the case of outcome (iii), we obtain a schedule shown in Figure **??**(b), in which machine A starts at time a''_w and has no idle time, while on machine C an extra idle time of at most a_p time units is created, i.e.

$$F_A(S_2) = a''_w + a(N);\ F_C(S_2) = \Phi_{BC}(N) + a_p.$$

Clearly, $\max\{F_A(S_2), F_C(S_2)\} \leq (1+\lambda)\cdot LB$. This proves the lemma. ∎

In the remainder of this section we consider Case 2, i.e., assume that $w = u \leq v$ in schedule S_0.

The algorithm below, similarly to Algorithm EarlyW, moves the block of jobs N_1 without job w to the end of the schedule. The obtained schedule is subject to preprocessing aimed at adjusting the start time on machine C to become a_w. The obtained schedule contains a gap on machine A. To reduce that gap, the operations are moved into it according to a procedure similar to that employed in Algorithm LateW1. If the gap reduction is insufficient, the algorithm performs the rescheduling of the jobs of set N_1 on machines A and C in the open shop manner.

Algorithm LateW2

Step 1. Given schedule S_0 that satisfies the conditions of Case 2, perform the following transformations. Remove the jobs of set N_1 on A and reduce the start times of all operations so that on machine A job w starts at time zero; call this schedule S'_0. If in S'_0 machine C starts processing earlier than time a_w, increase the start times of

all operations on that machine, so that the machine starts at time a_w; otherwise, decrease the start times of all jobs of set N_1 on C, so that the machine starts at time a_w. Start the block of jobs N_1 on machine A as early as possible. Call the resulting schedule $\tilde{S}$. See Figure **??** for illustration of this preprocessing step.

Step 2. In $\tilde{S}$, let $G_1 = F_A(N_2)$ be the completion time of the block of jobs N_2 and G_2 be the start time of the block of jobs N_1 on machine A. Define $\gamma = G_2 - G_1$. If $\gamma \leq \lambda \cdot LB$, go to Step 5; otherwise determine a job $q \in N_1$ such that in schedule $\tilde{S}$ operation $O_{A,q}$ starts exactly when operation $O_{C,q}$ completes, i.e., $G_2 = F_{A,q} - \sum_{j=1}^{q} a_j$. Go to Step 3.

Step 3. Determine the set $\tilde{N} = \{q+1, \ldots, w-1\}$. Scanning the jobs of this set in the sequence $w-1, w-2 \ldots$ move their $A-$operations one by one to fill the gap $[G_1, G_2]$, starting from $O_{A,w-1}$ to start at time G_1 until on of the following happens:

(i) the length of the remaining gap γ' does not exceed the the processing time of the next operation $O_{A,j}$, $j \in \tilde{N}$, to be moved;

(ii) all operations $O_{A,j}$, $j \in \tilde{N}$, of are moved into the gap;

If outcome (i) occurs, go to Step 5; otherwise, go to Step 4.

Step 4. If the length of the remaining gap γ' on machine A does not exceed $\lambda \cdot LB$, then go to Step 5. Otherwise, find an optimal open shop schedule $S_{AC}(N_1)$ for processing the jobs of set N_1 on machines A and C. In $\tilde{S}$, replace the processing of the jobs of set N_1 on machines A and C by schedule $S_{AC}(N_1)$ in the following way. If $C_{\max}(S_{AC}(N_1)) = a_h + c_h$ for some job $h \in N_1$, then process the block of jobs $(h, N_1 \setminus \{h\})$ on C starting from time a_w and the block of jobs $(N_1 \setminus \{h\}, h)$ on A so that job h starts at time $a_w + c_h$. If $C_{\max}(S_{AC}(N_1)) = c(N_1)$ then insert schedule $S_{AC}(N_1)$ in such a way that the the last job on each machine completes at time $a_w + c(N_1)$.

Step 5. Output the best found schedule as schedule S_2. Stop.

fhFU5.4172in3.7628in0pt(a) schedule S_0', machine C starts earlier than a_w, (b) schedule S_0', machine C starts later than a_w, (c) modified schedule $\tilde{S}$ for (a), (d) modified schedule $\tilde{S}$ for (b)FigS0PreProFigure

Lemma 4 *If in an instance of the problem all jobs satisfy (10) and for schedule S_0 the conditions (13) and $w = u \leq v$ hold, then Algorithm LateW2 finds a schedule S_2 which satisfies (14).*

Proof: Figure **??** shows the preprocessing of schedule S_0 performed in Step 1 with a purpose of obtaining a schedule in which machine C starts its processing at time a_w.

For schedule $\tilde{S}$ created in Step 1 of Algorithm LateW2 we have that

$$F_A\left(\tilde{S}\right) = a(N) + \gamma;\ F_C\left(\tilde{S}\right) \leq a_w + \max\{\Phi_{BC}(N), c(N)\} \leq (1+\lambda) \cdot LB,$$

so that the lemma holds for $\gamma \leq \lambda \cdot LB$. The subsequent transformations are aimed at reducing the completion time on machine A and do not affect the completion time on machine C.

Similarly to the proof of Lemma 3, job q found in Step 2 can be seen as the critical job in the flow shop schedule for the jobs of set N_1, in which each job follows the processing route (C, A). After the transformations in Step 3, we only need to consider outcome (ii) with the remaining gap γ' on machine A larger than $\lambda \cdot LB$; otherwise, $F_A\left(\tilde{S}\right) = a(N) + \gamma' \leq (1+\lambda) \cdot LB$, and the lemma holds.

In the remaining case, $\sum_{j=1}^{q} c_j = a(N_2) + a(N_1) - a_q + \gamma'$, which due to $\gamma' > \lambda \cdot LB \geq a_q$ implies that

$$c(N_1) \geq a(N_1) + a(N_2). \tag{15}$$

If in the open shop schedule $S_{AC}(N_1)$ created in Step 4 the makespan is defined by the total processing of job h, then $a_h + c_h > c(N_1)$, which due to (15) implies

$$c_h > c(N_1) - a_h \geq a(N_1) + a(N_2) - a_h,$$

and the resulting schedule S_2 is as shown in Figure **??**(a). We deduce

$$F_A(S_2) = a_w + c_h + a_h \leq a_w + LB \leq (1+\lambda) \cdot LB.$$

If the makespan in schedule $S_{AC}(N_1)$ is determined by the total processing time on one of the machines, then it follows from (15) that $C_{\max}(S_{AC}(N_1)) = c(N_1)$. There is a certain flexibility in the structure of schedule $S_{AC}(N_1)$. For instance, if this schedule is found by Gonzalez-Sahni algorithm [10] it can be guaranteed that in $S_{AC}(N_1)$ all jobs either start or complete at the same time. The resulting schedule S_2 is as shown in Figure **??**(b). Notice that due to (15) it is possible to process all jobs of set $N_1 \cup N_2$ on machine A while the jobs of set N_1 are processed on machine C, so that

$$F_A(S_2) = a_w + c(N_1) \leq a_w + LB \leq (1+\lambda) \cdot LB.$$

In any case, $F_C(S_2) = a_w + \Phi_{BC}(N)(1+\lambda) \cdot LB$. ∎

ftbpFU5.4172in1.868in0ptSchedule created in Step 4: (a) $C_{\max}(S_{AC}(N_1)) = a_h + c_h$; (b) $C_{\max}(S_{AC}(N_1)) = c(N_1)$FigS2OpenShopFigure

4.3 Implications

Combining the results of Sections 4.1 and 4.2, we deduce the following statement.

Theorem 1 *For problem $A(BC)\,|\,|\,C_{\max}$ a schedule S_H can be found such that*

$$\frac{C_{\max}(S_H)}{C_{\max}(S^*)} \leq \min\{2-\lambda, 1+\lambda\}.$$

The theorem immediately follows from the observation that for a given λ we either deal with an instance of the problem with a long $A-$operation, so that Lemma 1 applies, or without long $A-$operations, so that the results of Section 4.2 hold.

Applying this theorem with $\lambda = \frac{1}{2}$, we deduce the following

Corollary 1 *Problem $A(BC)\,|\,|\,C_{\max}$ admits a $\frac{3}{2}-$approximation algorithm.*

Recall that the ultimate goal of this paper is to develop a heuristic algorithm that for problem $A\,(BC)\,|\,|\,C_{\max}$ delivers a schedule S_H such that

$$\frac{C_{\max}\,(S_H)}{C_{\max}\,(S^*)} \leq \frac{7}{5}. \tag{16}$$

Let us identify which instances of the problem require additional consideration. First, we may assume that

$$a_j \leq \frac{3}{5}LB,\ j \in N;$$

otherwise, we can apply Algorithm P from Section 4.1 with $\lambda = \frac{3}{5}$. On the other hand, we assume that there exists a job j with $a_j > \frac{2}{5}LB$; otherwise, we can apply the algorithms presented in Section 4.2 with $\lambda = \frac{2}{5}$.

In the subsequent sections, we only consider the instances of problem $A\,(BC)\,|\,|\,C_{\max}$ that satisfy these conditions. The consideration is split into three parts in accordance with the following three possible types of instances:

Type 1: There exists a job $p \in N$ such that

$$\frac{2}{5}LB < a_p < \frac{3}{5}LB,\ c_p > \frac{2}{5}LB.$$

Type 2: There exist two jobs $p \in N$ and $q \in N$ such that

$$\frac{2}{5}LB < a_p < \frac{3}{5}LB,\ \frac{2}{5}LB < a_q < \frac{3}{5}LB,\ c_p \leq \frac{2}{5}LB,\ c_q \leq \frac{2}{5}LB.$$

Type 3: There exists a unique job $p \in N$ such that

$$\frac{2}{5}LB < a_p < \frac{3}{5}LB,\ c_p \leq \frac{2}{5}LB,$$

while $a_j \leq \frac{2}{5}LB$ for all other jobs.

5 Instances of Type 1

In this section, we consider Type 1 instances of problem $A\,(BC)\,|\,|\,C_{\max}$. In the description of the corresponding algorithm and all algorithms in the remaining sections, we distinguish between the blocks of *fixed* jobs and blocks of *movable* jobs. The fixed jobs are prescheduled on each machine, typically in the beginning of the schedule, while movable jobs start on the corresponding machine after the fixed jobs as early as possible.

In this and the remaining sections, in the Gantt charts that are used for illustration of the algorithms, the blocks of movable jobs are indicated by a left block arrow, which stresses that these blocks can be shifted to the left on the corresponding machine to start as early as possible.

As above, assume that the jobs are numbered as in sequence that defines schedule $S_{BC}\,(N)$. Let schedule $S_{BC}\,(N \setminus \{p\})$ be obtained from schedule $S_{BC}\,(N)$ by the removal of job p. The jobs of set $N \setminus \{p\}$ are kept in the order of their numbering.

This algorithm below promotes (i.e., shifts to the left) job p to the start of the schedule on machines B and C, and relegates (i.e., shifts to the right) that job to the end of the

schedule on machine A. A job u may become critical in a schedule on machines A and B, and u may also need either to be promoted on A or relegated on B if its processing time is sufficiently large. The net impact of these modifications to an optimal flow shop schedule $S_{BC}(N \setminus \{p\})$ is small enough to ensure that result (16) holds.

Algorithm Type1

Step 1. Create schedule $S_1^{(1)}$ in which the block of fixed jobs $N \setminus \{p\}$ on A starts at time zero, while job p is fixed to start on B at time zero and on C at time b_p; the remaining jobs are movable. Let $R_B(N \setminus \{p\})$ be the start time of uninterrupted processing of the block of jobs $N \setminus \{p\}$ on machine B. If $R_B(N \setminus \{p\}) \leq \frac{2}{5}LB$, then define $S_H^{(1)} = S_1^{(1)}$ and go to Step 6; otherwise identify job u, which is critical in the flow shop schedule of the jobs of set $N \setminus \{p\}$ on machines A and B, split the jobs of set $N \setminus \{p\}$ into two subsets, N_u and N'_u consisting of all jobs before job u and after job u, respectively. If $a_u \leq \frac{1}{5}LB$, go to Step 2, otherwise go to Step 3.

Step 2. Create schedule $S_2^{(1)}$ in which the blocks of fixed jobs are (u, N'_u) and (p, N_u) to start at time zero on A and B, respectively, and job p to start on C at time b_p, while the blocks of movable jobs are (N_u, p) on A, (u, N'_u) on B and $N \setminus \{p\}$ on C. Define $S_H^{(1)} = S_2^{(1)}$ and go to Step 6.

Step 3. Create schedule $S'_{BC}(N)$ from schedule $S_{BC}(N)$ by moving job p into the first position and job u into the last position. Create schedule $S_3^{(1)}$ in which job u starts on A at time zero, while the block of movable jobs on that machine is $(p, N \setminus \{p, u\})$. On machines B and C, the jobs are processed from time zero in accordance with schedule $S'_{BC}(N)$. If $b_u \leq c_u$, go to Step 4; otherwise, define $S_H^{(1)} = S_3^{(1)}$ and go to Step 6.

Step 4. Create schedule $S''_{BC}(N)$ from schedule $S_{BC}(N)$ by moving job p into the first position on both machines, followed by moving job u into the second position on machine B only. Create schedule $S_4^{(1)}$, in which the block $N \setminus \{p, u\}$ of fixed jobs starts on A at time zero, while the block of movable jobs on that machine is (u, p). On machines B and C, the jobs are processed from time zero in accordance with schedule $S''_{BC}(N)$, provided that the block of jobs $N \setminus \{p, u\}$ on machine B is treated as movable. If $c_u \leq \frac{2}{5}LB$, go to Step 5; otherwise define $S_H^{(1)} = S_4^{(1)}$ and go to Step 6.

Step 5. Modify schedule $S_4^{(1)}$ by moving job u on machine C to the last position; call the resulting schedule $S_5^{(1)}$, define $S_H^{(1)} = S_5^{(1)}$ and go to Step 6.

Step 6. Output schedule $S_H^{(1)}$. Stop.

Theorem 2 *For Type 1 instances of problem $A(BC) \mid \mid C_{\max}$ Algorithm Type1 creates a schedule $S_H = S_H^{(1)}$ such that the bound (16) holds.*

Proof: By the Type 1 conditions,

$$b_p < \frac{1}{5}LB \tag{17}$$

due to $\min\{a_p, c_p\} > \frac{2}{5}LB$. For schedule $S_1^{(1)}$ found in Step 1 of Algorithm Type1 we have that

$$\begin{aligned} F_A &= \max\{a(N), b_p + c_p + a_p\} \leq LB, \\ F_C &= \max\{b_p + c(N), R_B(N \setminus \{p\}) + \Phi_{BC}(N \setminus \{p\})\} \leq \frac{7}{5}LB, \end{aligned}$$

provided that $R_B(N \setminus \{p\}) \leq \frac{2}{5}LB$; see Figure **??**(a).

If $R_B(N \setminus \{p\}) > \frac{2}{5}LB$ in schedule $S_1^{(1)}$, then from $R_B(N \setminus \{p\}) = a(N_u) + a_u - b(N_u) > \frac{2}{5}LB$ and $a_p > \frac{2}{5}LB$, we deduce that $b(N_u) \leq \frac{1}{5}LB$. Besides, $b_p \leq \frac{1}{5}LB$. Using these inequalities, we derive that for schedule $S_2^{(1)}$ found in Step 2 of Algorithm Type1 the inequality

$$F_A \leq \max\{a(N), b_p + b(N_u) + a(N_u) + a_p, b_p + c_p + a_p\} \leq \frac{7}{5}LB$$

holds; see Figure **??**(a). Additionally, if in schedule $S_2^{(1)}$ there is no idle time on B, then $F_C = b_p + \Phi_{BC}(N \setminus \{p\}) \leq LB$, due to (2) with $k = p$. We may also exclude the case that $F_C = b_p + c(N) \leq \frac{6}{5}LB$.

If there is idle time on B in $S_2^{(1)}$,then $F_C \leq a_u + a(N_u') + \Phi_{BC}(N \setminus \{p\})$. Notice that $a_u + a(N_u') \leq \frac{2}{5}LB$, since otherwise, $(a_u + a(N_u')) + (a(N_u) + a_u) = a(N \setminus \{p\}) + a_u > \frac{4}{5}LB$, a contradiction to $a_u \leq \frac{1}{5}LB$ and $a(N \setminus \{p\}) \leq \frac{3}{5}LB$.

We come to Step 3 with

$$a_u > \frac{1}{5}LB, \tag{18}$$

which implies that $a(N \setminus \{p, u\}) \leq \frac{2}{5}LB$. Let Φ' denote the makespan of schedule $S'_{BC}(N)$. In schedule $S_3^{(1)}$, we have that $F_C = \Phi'$; see Figure **??**(c). Since $b_u > c_u$, it follows from $b_p < \frac{1}{5}LB < \frac{2}{5}LB < c_p$ and from (2) with $k = p$ and $k = u$ that

$$\Phi' \leq b_p + \Phi_{BC}(N \setminus \{p, u\}) + c_u = \Phi_{BC}(N),$$

so that $\Phi' \leq LB$. Besides, for schedule $S_3^{(1)}$ we have that

$$F_A = \max\{a(N), b_p + c_p + a_p + a(N \setminus \{p, u\}), \Phi' - c_u + a(N \setminus \{p, u\})\} \leq \frac{7}{5}LB.$$

We come to Step 4 with $b_u \leq c_u$ and $c_u > \frac{2}{5}LB$. Let Φ'' denote the makespan of schedule $S''_{BC}(N)$. Due to (17), we deduce

$$\Phi'' \leq b_p + \max\{c(N), b_u + \Phi_{BC}(N \setminus \{p, u\})\} \leq \max\{b_p + c(N), \Phi_{BC}(N)\} \leq \frac{6}{5}LB,$$

so that for schedule $S_4^{(1)}$ we have that either $F_C \leq \max\{\Phi'', a(N \setminus \{p, u\}) + \Phi_{BC}(N \setminus \{u\})\} \leq \frac{7}{5}LB$ or $F_C = b_p + b_u + a_u + c_u + c(N_u')$; Figure **??**(d). In the latter case, since $c_u + c_p > \frac{4}{5}LB$ we obtain that $c(N_u') \leq \frac{1}{5}LB$, so that again $F_C \leq \frac{7}{5}LB$ due to (17). Besides, in schedule $S_4^{(1)}$ we have that

$$F_A = \max\{a(N), b_p + c_p + a_p, b_p + b_u + a_u + a_p\} \leq \max\{LB, 2LB - (c_u + c_p)\} \leq \frac{6}{5}LB.$$

For schedule $S_5^{(1)}$ found in Step 5, we need a different analysis of the situation $F_A = b_p + b_u + a_u + a_p$. Suppose that $a_u + b_u > \frac{4}{5}LB$. Then $c_u \leq \frac{1}{5}LB$, but since $b_u \leq c_u$ and $a_u \leq \frac{3}{5}LB$, we get a contradiction. Thus, $F_A \leq \frac{7}{5}LB$, since $a_u + b_u \leq \frac{4}{5}LB$ and $a_p + b_p \leq \frac{3}{5}LB$. On machine C we have

$$\begin{aligned} F_C &\leq b_p + \max\{c(N), \Phi_{BC}N \setminus \{p, u\} + c_u, b_u + a_u + c_u\} \\ &\leq \max\{b_p + c(N), \Phi_{BC}N \setminus \{u\} + c_u, b_p + b_u + a_u + c_u\} \leq \frac{7}{5}LB. \end{aligned}$$

This proves the theorem. ∎

ftbpFU4.8992in5.6818in0ptSchedules found by Algorithm Type1: (a) $S_1^{(1)}$; (b) $S_2^{(1)}$; (c) $S_3^{(1)}$;(d) $S_4^{(1)}$; (e) $S_5^{(1)}$FigType1FigureIf follows from the results obtained in this section that from now on we only need to consider the instances of problem $A\left(BC\right)|\,|C_{\max}$, in which for every job j the inequality $a_j > \frac{2}{5}LB$ implies that $c_j < \frac{2}{5}LB$. An instance may contain either two such jobs (Type 2) or exactly one (Type 3). Such instances are handled in the forthcoming sections.

6 Instances of Type 2

Let jobs p and q satisfy the conditions of a Type 2 instance. In this section, the following schedule is of a special importance. Let $S'_{BC}\left(N\right)$ be a flow shop schedule obtained from schedule $S_{BC}\left(N\right)$ by moving job p into the first position and job q into the last position.

Lemma 5 *For a Type 2 instance of problem $A\left(BC\right)|\,|C_{\max}$, let Φ' denote the makespan of schedule $S'_{BC}\left(N\right)$. Then*

$$\Phi' \leq \frac{7}{5}LB$$

if either

(i) $b_p \leq c_p$, or

(ii) $b_p > c_p,\ b_q > c_q$ and $b_p \leq \frac{2}{5}LB$.

Proof: If follows that

$$\Phi' = \max\left\{b_p + c\left(N\right), b_p + \Phi_{BC}\left(N \setminus \{p,q\}\right) + c_q, b(N) + c_q\right\}.$$

Recall that $c_q \leq \frac{2}{5}LB$. Under conditions (ii), we are given $b_p \leq \frac{2}{5}LB$. Under condition (i), we have that $b_p = \min\left\{b_p, c_p\right\} \leq \frac{1}{2}\left(b_p + c_p\right) \leq \frac{3}{10}LB < \frac{2}{5}LB$, since $a_p > \frac{2}{5}LB$. Thus, $\max\left\{b_p + c\left(N\right), b(N) + c_q\right\} \leq \frac{7}{5}LB$.

Under condition (i), $b_p + \Phi_{BC}\left(N \setminus \{p,q\}\right) + c_q = \Phi_{BC}\left(N \setminus \{q\}\right) + c_q$, while under conditions (ii) $b_p + \Phi_{BC}\left(N \setminus \{p,q\}\right) + c_q = b_p + \Phi_{BC}\left(N \setminus \{p\}\right)$. This proves the lemma. ∎

The following algorithm is presented under the assumption that either $b_p \leq c_p$ holds or both $b_p > c_p$ and $b_q > c_q$ hold. If $b_p > c_p$ and $b_q \leq c_q$, then the roles of jobs p and q can be swapped. The positions of jobs p and q are suitably adjusted on all machines to avoid possible clashes; the exact decisions depend on the relative processing times a_p, a_q, c_p and c_q. Lemma 5 guarantees that any loss of optimality is small enough to ensure that (16) holds.

Algorithm Type2

Step 1. Create the following schedule $S_1^{(2)}$. On machine A job q is the fixed job to start at time zero, while the block of movable jobs on A is $(p, N \setminus \{p,q\})$. On machines B and C, job p is fixed to start at zero and at b_p, respectively. The jobs of set $N \setminus \{p,q\}$ are processed on B and C as in schedule $S_{BC}\left(N \setminus \{p,q\}\right)$, starting from time b_p, while job q is a movable job on each of these two machines. If $b_p > \frac{1}{5}LB$, go to Step 2; otherwise define $S_H^{(2)} = S_1^{(2)}$ and go to Step 4.

Step 2. If $b_p > c_p$ and $b_q > c_q$ hold and additionally $\min\{b_p, b_q\} > \frac{2}{5}LB$, then go to Step 3; otherwise perform Step 2, provided that in the case that $b_p > c_p$, $b_q > c_q$ and $b_p > \frac{2}{5}LB \geq b_q$ hold, the roles of jobs p and q are swapped. Change schedule $S_1^{(2)}$ into schedule $S_2^{(2)}$, by altering the order on machine A, where the block of fixed jobs $(N \setminus \{p,q\}, q)$ starts at time zero, while job p is the movable job. Define $S_H^{(2)} = S_2^{(2)}$ and go to Step 4.

Step 3. Create schedule $S_3^{(1)}$, in which job q starts on A at time zero, while the block of movable jobs on that machine is $(p, N \setminus \{p,q\})$. On machines B and C, the jobs of set $N \setminus \{p,q\}$ are processed from time zero in accordance with schedule $S_{BC}(N \setminus \{p,q\})$. Job p starts on B at time $b(N \setminus \{p,q\})$ and job q is movable, while the block of jobs (p,q) is movable on C. Define $S_H^{(2)} = S_3^{(2)}$ and go to Step 4.

Step 4. Output schedule $S_H^{(2)}$. Stop.

fhFU5.418in2.949in0ptSchedules found by Algorithm Type2: (a) $S_1^{(2)}$; (b) $S_2^{(2)}$; (c) $S_3^{(2)}$FigType2Figure

Theorem 3 *For Type 2 instances of problem $A(BC)||C_{\max}$ Algorithm Type2 creates a schedule $S_H = S_H^{(2)}$ such that the bound (16) holds.*

Proof: If in schedule $S_1^{(2)}$ there is idle time on machine B before job q, then $F_C = a_q + b_q + c_q \leq LB$. Otherwise, the jobs are processed on machines B and C as in schedule $S'_{BC}(N)$, i.e., $F_C = \Phi'$; see Figure **??**(a). Notice that Φ' does not exceed $\frac{7}{5}LB$, since in Step 1 of the algorithm the conditions of Lemma 5 hold. In schedule $S_1^{(2)}$ on machine A we have that

$$\begin{aligned} F_A &= \max\{a(N), b_p + c_p + a_p + a(N \setminus \{p,q\}), \\ &\quad b_p + \max\{\Phi_{BC}(N \setminus \{p,q\}), c(N) - c_q\} + a(N \setminus \{p,q\})\}. \end{aligned}$$

For a Type 2 instance the inequality

$$a(N \setminus \{p,q\}) \leq \frac{1}{5}LB, \tag{19}$$

holds, so that $b_p + c_p + a_p + a(N \setminus \{p,q\}) \leq \frac{6}{5}LB$. Besides, since $b_p \leq \frac{1}{5}LB$ holds by the conditions of Step 1, it follows that $b_p + \max\{\Phi_{BC}(N \setminus \{p,q\}), c(N) - c_q\} + a(N \setminus \{p,q\}) \leq \frac{7}{5}LB$.

For schedule $S_2^{(2)}$, notice that $b_p > \frac{1}{5}LB$, and from (19) we deduce that the block of jobs $N \setminus \{p,q\}$ starts on B at time b_p; see Figure **??**(b). It follows that $F_A = \max\{a(N), b_p + c_p + a_p\} \leq LB$, while on machine C we have that $F_C = \max\{\Phi', a(N \setminus \{p,q\}) + a_q + b_q + c_q\}$. Due to (19), $a(N \setminus \{p,q\}) + a_q + b_q + c_q \leq \frac{6}{5}LB$. The conditions of Lemma 5 hold, so that $\Phi' \leq \frac{7}{5}LB$.

We arrive at Step 3 if the inequalities $b_p > c_p$, $b_q > c_q$, $b_p > \frac{2}{5}LB$, and $b_q > \frac{2}{5}LB$ hold simultaneously. These conditions imply that $\max\{c_p, c_q, b(N \setminus \{p,q\})\} \leq \frac{1}{5}LB$.

For schedule $S_3^{(2)}$ illustrated in Figure **??**(c), we have that

$$\begin{aligned} F_A &= \max\{a(N), b(N) - b_q + a(N) - a_q, \Phi_{BC}(N \setminus \{p,q\}) + a(N \setminus \{p,q\})\} \\ &\leq \max\left\{LB, 2LB - \frac{4}{5}LB, LB + \frac{1}{5}LB\right\} = \frac{6}{5}LB. \end{aligned}$$

On machine C, we have that

$$\begin{aligned} F_C &= \max\{\Phi_{BC}(N\setminus\{p,q\})+c_p+c_q, \max\{a_q+b_q, b(N)\}+c_q, \\ &\quad \max\{a_q, b(N\setminus\{p,q\})+b_p\}+a_p+c_p+c_q\}. \end{aligned}$$

Since $b_p > c_p$, $b_q > c_q$ we have that $\Phi_{BC}(N\setminus\{p,q\})+c_p+c_q = \Phi_{BC}(N\setminus\{q\})+c_q = \Phi_{BC}(N) \le LB$. Besides, $\max\{a_q+b_q, b(N)\}+c_q \le \frac{6}{5}LB$, $a_q+b_p+c_p+c_q \le \frac{6}{5}LB$ and $b(N\setminus\{p,q\})+b_p+a_p+c_p+c_q \le \frac{7}{5}LB$.

This proves the lemma. ∎

7 Instances of Type 3

Let job p be the only job that satisfies $a_p > \frac{2}{5}LB$. The conditions of a Type 3 instance of problem $A(BC)||C_{\max}$ imply that $c_p \le \frac{2}{5}LB$.

In order to design an approximation algorithm for the Type 3 instances we need a special procedure that splits the set of jobs $Q = N\setminus\{p\}$ into two subsets.

Procedure Split
INPUT: Jobs of set Q of a Type 3 instance, renumbered by the integers $1, 2, \ldots, n-1$ taken in the order associated with schedule $S_{BC}(Q)$, such that $\frac{2}{5}LB \le a(Q) \le \frac{3}{5}LB$ and $a_j \le \frac{2}{5}LB$ for all $j \in Q$
OUTPUT: Partition of set Q into two subsets Q_1 and Q_2 such that $\max\{a(Q_1), a(Q_2)\} \le \frac{2}{5}LB$ and $a(Q_1) > \frac{1}{5}LB$

Step 1. If there exists a job $u \in Q$ such that $a_u > \frac{1}{5}LB$, then output the sets

$$Q_1 = \{u\},\ Q_2 = Q\setminus\{u\};$$

otherwise go to Step 2.

Step 2. Scanning the jobs of set Q in the order opposite to their numbering, find a job u, $1 \le u \le n-1$, such that

$$\sum_{j=u+1}^{n-1} a_j \le \frac{1}{5}LB,\ \sum_{j=u}^{n-1} a_j > \frac{1}{5}LB.$$

Output the sets

$$Q_1 = \{1, \ldots, u-1,\},\ Q_2 = \{u, u+1, \ldots, n-1\}.$$

Procedure Split requires $O(n)$ time.

Lemma 6 *Procedure Split finds the required partition of set Q. Moreover, if $|Q_1| > 1$, then $a_j \le \frac{1}{5}LB$ for all $j \in Q_1$.*

Proof: If there exists a job u with $a_u > \frac{1}{5}LB$, then for the partition found in Step 1, we have that $a(Q_2) = a_u \le \frac{2}{5}LB$ and $a(Q_1) = a(Q) - a_u \le \frac{3}{5}LB - \frac{1}{5}LB = \frac{2}{5}LB$.

In Step 2 we deal with instances for which $a_j \leq \frac{1}{5}LB$ for all $j \in Q$. For the found partition, we have that $a(Q_2) > \frac{1}{5}LB$ and

$$\begin{aligned} a(Q_2) &= \sum_{j=u+1}^{n} a_j + a_u \leq \frac{1}{5}LB + \frac{1}{5}LB = \frac{2}{5}LB; \\ a(Q_1) &= a(Q) - a(Q_2) \leq \frac{3}{5}LB - \frac{1}{5}LB = \frac{2}{5}LB. \end{aligned}$$

This proves the lemma. ∎

The general framework of an approximation algorithm that handles the Type 3 instances of problem $A(BC) \mid \mid C_{\max}$ is as follows:

- If $a(Q) \leq \frac{2}{5}LB$, run Algorithm Type3.1; otherwise run Procedure Split.
- If Procedure Split outputs the sets such that $|Q_2| = 1$, then run Algorithm Type3.2; otherwise run Algorithm Type3.3.

We use Procedure Split to partition the set $N \setminus \{p\}$ into two subsets to be scheduled as blocks to get a large degree of overlap between processing on the three machines. Lemma 6 implies that the largest possible idle time on any machine can be limited to the processing time of one partition. We describe and analyze the corresponding algorithms separately. The first of these algorithms presented below fixes some operations to start at time zero and then start the remaining (movable) operations as early as possible.

Algorithm Type3.1

Step 1. Create schedule $S_1^{(3)}$, in which at time zero job p is fixed on machine A and and schedule $S_{BC}(Q)$ is run on machines B and C. Set Q is movable on machine A, while job p is movable on machines B and C.

Step 2. Output schedule $S_H^{(3)} = S_1^{(3)}$ and stop.

Lemma 7 *For Type 3 instances of problem $A(BC) \mid \mid C_{\max}$ with $a(Q) \leq \frac{2}{5}LB$, Algorithm Type3.1 creates a schedule $S_H = S_H^{(3)}$ such that the bound (16) holds.*

Proof: It follows from $\max\{a(Q), c_p\} \leq \frac{2}{5}LB$ that for schedule $S_1^{(3)}$ we have

$$\begin{aligned} F_A &= \max\{a(N), \Phi_{BC}(Q) + a(Q)\} \leq LB + \frac{2}{5}LB; \\ F_C &= \max\{a_p + b_p + c_p, \max\{b(N), \Phi_{BC}(Q)\} + c_p\} \leq LB + \frac{2}{5}LB. \end{aligned}$$

This proves the lemma. ∎

From now on, we assume that $a(Q) > \frac{2}{5}LB$, and set Q is partitioned into two subsets Q_1 and Q_2 in accordance with Procedure Split. Algorithm Type3.2 applies if $Q_2 = \{u\}$, where $a_u > \frac{1}{5}LB$, while Algorithm Type3.3 applies if $|Q_2| > 1$.

Similarly to Section 6, let $S''_{BC}(N)$ be a flow shop schedule obtained from schedule $S_{BC}(N)$ by moving job u into the first position and job p into the last position. Let Φ'' denote the makespan of schedule $S''_{BC}(N)$. Notice that

$$\Phi'' = \max\{b_u + c(N), b_u + \Phi_{BC}(Q_1) + c_p, b(N) + c_p\}. \tag{20}$$

In all schedules created by the algorithm below, the jobs of block $Q_1 = N \setminus \{p, u\}$ are kept in accordance with the sequence associated with schedule $S''_{BC}(N)$. The actions of this algorithm resemble those taken by Algorithm Type2.

Algorithm Type3.2

Step 1. If $b_u > c_u$, go to Step 2. Otherwise, create schedule $\hat{S}$, in which at time zero machine A processes job p and machine B runs the block of jobs (u, Q_1), while machine C processes job u starting at time b_u. The jobs of block (u, Q_1) are movable on A, and job p is movable on B and C, and the jobs of set Q_1 are processed on C as in schedule $S''_{BC}(N)$. Additionally, create schedule $\check{S}$ by modifying schedule $\hat{S}$ by taking the processing sequence on A to be (Q_1, p, u). In schedule $\check{S}$, machine A processes the block of jobs (Q_1, p) from time zero, while job u is movable. Compared to $\hat{S}$, in schedule $\check{S}$ the block of jobs Q_1 may be delayed on both machines B and C by $\max\{a(Q_1) - b_u, 0\}$. Job p remains movable on B and C. Define $S_2^{(3)}$ to be the better of the two schedules $\hat{S}$ and $\check{S}$. Go to Step 4.

Step 2. If $b_p > c_p$, go to Step 3; otherwise, swap the roles of jobs p and u and create schedule $S_2^{(3)}$, as described in Step 1. Go to Step 4.

Step 3. Create schedule $S_2^{(3)}$, in which from time zero machine A processes the block (u, p), while machines B and C run the block of jobs Q_1 as in schedule $S''_{BC}(N)$. The block Q_1 is movable on A, while the block (u, p) is movable on B and C.

Step 4. Output schedule $S_H^{(3)} = S_2^{(3)}$ and stop.

ftbpFU5.124in2.8833in0ptSchedules found by Algorithm Type3.2 (a) $\hat{S}$; (b) $\check{S}$; (c) schedule found in Step 3.FigType3.2Figure

Lemma 8 *For Type 3 instances of problem $A(BC) \mid \mid C_{\max}$ with $a(Q) > \frac{2}{5}LB$ and $Q_2 = \{u\}$, Algorithm Type3.2 creates a schedule $S_H^{(3)}$ such that for $S_H = S_H^{(3)}$ the bound (16) holds.*

Proof: In Step 1, we have that $b_u \leq c_u$. This implies that $b_u = \min\{b_u, c_u\} \leq \frac{1}{2}(b_u + c_u) \leq \frac{2}{5}LB$, since $a_u > \frac{1}{5}LB$. The three main inequalities used in analyzing schedules created in Step 1 are

$$b_u \leq \frac{2}{5}LB,\ c_p \leq \frac{2}{5}LB,\ a(Q_1) \leq \frac{2}{5}LB. \tag{21}$$

If in schedule $\hat{S}$ found in Step 1 there is idle time before job p on machine C, then $F_C = \max\{a_p + b_p + c_p, b(N) + c_p\} \leq \frac{7}{5}LB$; see Figure **??**(a). If in $\hat{S}$ there is no idle time before job p on machine C, then the jobs are processed on machines B and C as in schedule $S''_{BC}(N)$, so that $F_C = \Phi''$.

Notice that for schedule $S''_{BC}(N)$, if the makespan $\Phi'' = \max\{b_u + c(N), b(N) + c_u\}$, then due to (21) we deduce that $\Phi'' \leq \frac{7}{5}LB$. On the other hand, if $\Phi'' = b_u + \Phi_{BC}(Q_1) + c_p$, then due to $b_u \leq c_u$, we derive that $b_u + \Phi_{BC}(Q_1) = \Phi_{BC}(N \setminus \{p\}) \leq LB$, so that again $\Phi'' \leq \frac{7}{5}LB$.

If in schedule $\hat{S}$, we have that $F_A = b_u + c(N \setminus \{p\}) + a(Q_1)$, then further analysis is required, while otherwise

$$F_A = \max\{a(N), b_u + c_u + c_u + a(Q_1), b_u + \Phi_{BC}(Q_1) + a(Q_1)\} \leq LB + \frac{2}{5}LB.$$

In schedule $\check{S}$, we have that $F_A = \max\{a(Q_1) + a_p, b_u + c_u\} + a_u \leq LB$; see Figure **??**(b). If in schedule $\check{S}$ there no idle time on C before job p starts, then for $F_C = a(Q_1) + \Phi_{BC}(Q_1) + c_p$ further analysis is required; otherwise $F_C = \max\{a(Q_1), b_u\} + c(N) \leq \frac{7}{5}LB$. If there is idle time on C before job p and $F_C = a(Q_1) + b(Q_1) + b_p + c_p$ then further analysis is required; otherwise, $F_C = \max\{a(Q_1) + a_p + b_p, b(N) + c_p\} \leq \frac{7}{5}LB$.

We are left to consider the case that $C_{\max}\left(\hat{S}\right) = b_u + c(N \setminus \{p\}) + a(Q_1)$ and $C_{\max}\left(\check{S}\right) = a(Q_1) + \max\{\Phi_{BC}(Q_1), b(N \setminus \{u\})\} + c_p$. We deduce that

$$\begin{aligned} C_{\max}\left(S_2^{(3)}\right) &= \min\left\{C_{\max}\left(\hat{S}\right), C_{\max}\left(\check{S}\right)\right\} \leq \frac{1}{2}\left(C_{\max}\left(\hat{S}\right) + C_{\max}\left(\check{S}\right)\right) \\ &\leq \frac{1}{2}\left(2a(Q_1) + c(N) + \max\{\Phi_{BC}(N \setminus \{p\}), b(N)\}\right) \leq LB + a(Q_1) \leq \frac{7}{5}LB. \end{aligned}$$

In Step 2, after the roles of jobs u and p are swapped, the condition (21) holds, together with $b_u < c_u$.

For schedule $S_2^{(3)}$ found in Step 3, we have that $F_A = \max\{a(N), \Phi_{BC}(Q_1) + a(Q_1)\} \leq \frac{7}{5}LB$, since $a(Q_1) \leq \frac{2}{5}LB$; Figure **??**(c). If $F_C = \max\{a_u + a_p + b_p + c_p, a_u + b_u + c_u + c_p\}$ then $F_C \leq \frac{7}{5}LB$ due to $\max\{a_u, c_p\} \leq \frac{2}{5}LB$. The inequality $b_u > c_u$ implies that $\Phi_{BC}(Q_1) + c_u \leq \Phi_{BC}(N \setminus \{p\}) \leq LB$, so that for $F_C = \Phi_{BC}(Q_1) + c_u + c_p$ the inequality $F_C \leq \frac{7}{5}LB$ holds. Finally, we need to consider the case that $F_C = b(Q_1) + b_u + c_u + c_p = b(N) + G$, where $G := F_C - b(N)$. Since $b_p > c_p$, we deduce $G = c_u + c_p - b_p \leq c_u$. The inequalities $a_u > \frac{1}{5}LB$ and $b_u > c_u$ imply that $c_u \leq \frac{2}{5}LB$, which means that in the case under consideration $F_C \leq \frac{7}{5}LB$.

This proves the lemma. ∎

In the remainder of this section, we assume that for the partition of set Q by Procedure Split the inequality $|Q_1| > 1$ holds. In all schedules created by the algorithm below the jobs of each block Q_1 and Q_2 are kept in accordance with the sequence associated with schedule $S_{BC}(N \setminus \{p\})$. If either $b_p \leq \frac{1}{5}LB$ or $a(Q_2) \leq b(Q_1) + b_p$, the algorithm fixes certain operations to start at time zero and appropriately promotes the movable operations; see Steps 2-4. Under the conditions of Step 5, the algorithm fills the gap on machine A in a similar style that is used in Algorithms Late W1 and Late W2.

Algorithm Type3.3

Step 1. If $b_p > \frac{1}{5}LB$, go to Step 2; otherwise, go to Step 3.

Step 2. If $\Phi_{BC}(Q_1) \leq \frac{4}{5}LB$, create schedule $\hat{S}$, in which at time zero machine A processes job p, while machines B and C run the block of jobs (Q_1, Q_2) as in schedule $S_{BC}(N \setminus \{p\})$. On machine A, the block of jobs Q_1 starts at time $\max\{a_p, \Phi_{BC}(Q_1)\}$, while the block Q_2 is movable. Job p is movable on B and C. If $\Phi_{BC}(Q_1) > \frac{4}{5}LB$ create schedule $\check{S}$ by modifying schedule $\hat{S}$ by changing the order of the blocks Q_1 and Q_2 on A, and delaying block Q_2 to start on A at time $\max\{a_p, b(N \setminus \{p\})\}$ and on C immediately after its completion on A. Define $S_3^{(3)}$ to be the better of the two schedules $\hat{S}$ and $\check{S}$. Go to Step 6.

Step 3. If $a(Q_2) > b(Q_1)$, go to Step 4. Otherwise, create schedule $S_3^{(3)}$, in which from time zero machine A processes the block (Q_2, p), while machines B and C run the block of jobs (Q_1, Q_2) as in schedule $S_{BC}(N \setminus \{p\})$. The block Q_1 is movable on A, while the job p is movable on B and C. Go to Step 6.

Step 4. If $a(Q_2) > b(Q_1) + b_p$, go to Step 5. Otherwise, create schedule $S_3^{(3)}$, in which from time zero machine A processes the block Q_2, while machines B and C run the block Q_1 of jobs as in schedule $S_{BC}(N \setminus \{p\})$. The block of jobs (p, Q_2) starts on B at time $b(Q_1)$, and job p starts on A at time $b(Q_1) + b_p$. The block Q_1 is movable on A, while the block (Q_2, p) is movable on C. Go to Step 6.

Step 5. Create schedule $S_3^{(3)}$, in which machine A processes the block Q_2 from time zero, while machines B and C process the block (p, Q_1) with job p to start at time zero. On machines A and C the jobs of block (p, Q_1) are processed in this order as in a flow shop schedule, in which each job has the processing route (C, A), and the block starts on on machine A as early as possible. The block Q_2 starts on B at time $a(Q_2)$ and on C as early as possible. Let in the resulting schedule job $q \in Q_1$ be critical, i.e., it starts on A exactly when it finishes on C. Identify the sets of jobs Q_1' and Q_1'', sequenced before and after job q, respectively. Let γ be the idle time of A before job p starts its processing. If $\gamma > \frac{2}{5}LB$, move the block of jobs Q_1'' to start on A at time $a(Q_2)$. Go to Step 6.

Step 6. Output schedule $S_H^{(3)} = S_3^{(3)}$ and stop.

ftbpFU5.4163in1.9631in0ptSchedules found in Step 2 of Algorithm Type3.3: (a) $\hat{S}$; (b) $\check{S}$FigType3.3Step2Figure

ftbpFU5.4172in2.303in0ptSchedule $S_3^{(3)}$ found by Algorithm Type3.3: (a) in Step 3; (b) in Step 4FigType3.3Steps34FigureftbpFU5.4803in2.559in0ptSchedule $S_3^{(3)}$ found by Algorithm Type3.3 in Step 5: (a) $\gamma \leq \frac{2}{5}LB$; (b) $\gamma > \frac{2}{5}LB$FigType3.3Step5Figure

Lemma 9 *For Type 3 instances of problem $A(BC) \mid \mid C_{\max}$ with $a(Q) > \frac{2}{5}LB$ and $|Q_2| > 1$, Algorithm Type3.3 creates a schedule $S_H^{(3)}$ such that for $S_H = S_H^{(3)}$ the bound (16) holds.*

Proof: The main conditions used in the analysis of Algorithm Type3.3 can be summarized as

$$a(Q_1) \leq \frac{2}{5}LB,\ \frac{1}{5}LB < a(Q_2) \leq \frac{2}{5}, a_p > \frac{2}{5}LB,\ c_p \leq \frac{2}{5}LB. \tag{22}$$

For schedule $\hat{S}$ found in Step 2 if $\Phi_{BC}(Q_1) \leq \frac{4}{5}LB$, we have that

$$\begin{aligned} F_A &\leq \max\{a(N), \Phi_{BC}(Q_1) + a(N \setminus \{p\}), \Phi_{BC}(N \setminus \{p\}) + a(Q_2)\} \\ &\leq \max\left\{LB, \frac{4}{5}LB + \left(LB - \frac{2}{5}LB\right), LB + \frac{2}{5}LB\right\} \leq \frac{7}{5}LB \end{aligned}$$

and $F_C = \max\{b(N), \Phi_{BC}(N \setminus \{p\})\} + c_p \leq \frac{7}{5}LB$; see Figure **??**(a).

Let us analyze schedule $\check{S}$ that is created in Step 2 if $\Phi_{BC}(Q_1) > \frac{4}{5}LB$; see Figure **??**(b). Due to $b_p > \frac{1}{5}LB$, we have that

$$\begin{aligned} F_A &= \max\{b(N \setminus \{p\}) + a(N \setminus \{p\}), \Phi_{BC}(Q_1) + a(Q_1)\} \\ &\leq \max\left\{2LB - \frac{2}{5}LB - \frac{1}{5}LB, LB + \frac{2}{5}LB\right\} \leq \frac{7}{5}LB. \end{aligned}$$

If in schedule $\check{S}$ we have that $F_C = b(N) + c_p$, then, as above $F_C \leq \frac{7}{5}LB$. Otherwise, $F_C = b(N \setminus \{p\}) + a(Q_2) + c(Q_2) + c_p \leq \frac{4}{5}LB + \frac{2}{3}LB + c(Q_2) + c_p$. Due to $b_p > \frac{1}{5}LB$ the inequality $c_p \geq b_p$ is impossible, since $\Phi_{BC}(Q_1) \leq \Phi_{BC}(N) - \min\{b_p, c_p\}$. Thus, $c_p \leq b_p$,

so that $\Phi_{BC}(N \setminus \{p\}) + c_p = \Phi_{BC}(N)$. Also, $\Phi_{BC}(Q_1) > \frac{4}{5}LB \geq b(N \setminus \{p\})$, so that $\Phi_{BC}(N \setminus \{p\}) = \Phi_{BC}(Q_1) + c(Q_2)$. Therefore, $\Phi_{BC}(N) = \Phi_{BC}(Q_1) + c(Q_2) + c_p$, which yields $c(Q_2) + c_p \leq \frac{1}{5}LB$, leading to $F_C \leq \frac{7}{5}LB$.

For schedule $S_3^{(3)}$ found in Step 3, notice that the block Q_2 starts on B after it is completed on A. Thus, $F_A \leq \max\{a(N), \Phi_{BC}(Q_1) + a(Q_1)\} \leq \frac{7}{5}LB$ and $F_C = \max\{b(N), \Phi_{BC}(N \setminus \{p\}), a(Q_2) + a_p + b_p\} + c_p \leq \frac{7}{5}LB$; see Figure **??**(a).

For schedule $S_3^{(3)}$ found in Step 4, notice that the block Q_2 starts on B after it is completed on A; Figure **??**(b). Since

$$b(Q_1) < a(Q_2) \leq \frac{2}{5}LB,\ b_p \leq \frac{1}{5}LB,\ a(Q_2) > \frac{1}{5}LB,$$

we deduce that

$$\begin{aligned} F_A &\leq \max\{b(Q_1) + b_p + a_p + a(Q_1), \Phi_{BC}(Q_1) + a(Q_1)\} \\ &\leq \max\left\{\frac{2}{5}LB + \frac{1}{5}LB + \frac{4}{5}LB, LB + \frac{2}{5}LB\right\} \leq \frac{7}{5}LB. \end{aligned}$$

If there is an idle time on C before job p starts its processing then $F_C = b(Q_1) + b_p + a_p + c_p \leq \frac{7}{5}LB$. Otherwise, notice that the schedule on machines B and C can be obtained from schedule $S_{BC}(N \setminus \{p\})$ by inserting job p after the block Q_1 on machine B and after all jobs on machine C, so that $F_C \leq \Phi_{BC}(N \setminus \{p\}) + b_p + c_p \leq \Phi_{BC}(N) + \max\{b_p, c_p\} \leq \frac{7}{5}LB$.

We are left to analyze schedule $S_3^{(3)}$ found in Step 5. If there is no idle time on machine C before processing the block of jobs Q_2, then $F_C = b_p + \max\{c(N), \Phi_{BC}(N \setminus \{p\})\} \leq \frac{6}{5}LB$. Otherwise, the schedule of jobs of set $N \setminus \{p\}$ on machines B and C can be obtained from schedule $S_{BC}(N \setminus \{p\})$ by delaying the start time of each job of set Q_2 by at most $a(Q_2)$ time units, i.e., $F_C \leq a(Q_2) + \Phi_{BC}(N \setminus \{p\}) \leq \frac{7}{5}LB$.

On machine A, if $\gamma \leq \frac{2}{5}LB$, then $F_A = a(N) + G \leq \frac{7}{5}LB$, so that no transformation is needed; see Figure **??**(a). Otherwise, $\gamma > \frac{2}{5}LB \geq a(Q_1'')$, i.e., in the modified schedule the moved jobs can be processed on A from time $a(Q_2)$ and complete before job p start, see Figure **??**(b). As a result of this transformation, $F_A = b_p + c_p + c(Q_1') + c_q + a_q \leq b_p + c(N) + a_q$. Lemma 6 implies that $a_j \leq \frac{1}{5}LB$ for all $j \in Q_1$. This and the inequality $b_p \leq \frac{1}{5}LB$ guarantee that $F_A \leq \frac{7}{5}LB$.

This proves the lemma. ∎

8 Main Algorithm and Tightness

In this section, we give a formal description of the overall $\frac{7}{5}-$approximation algorithm and demonstrate that $\frac{7}{5}$ is a tight bound.

Algorithm Main
INPUT: An instance of problem set Q of problem $A(BC) \mid \mid C_{\max}$
OUTPUT: A schedule S_H such that (16) holds

Step 1. Find schedule $S_{BC}(N)$ and compute the lower bound LB by (6). If there exists a job p that satisfies (7) with $\lambda = \frac{3}{5}$, then find schedule S_p by running Algorithm P from Section 4.1 and go to Step 6; otherwise go to Step 2.

Step 2. If there exists a job j that satisfies $a_j > \frac{2}{5}LB$ go to Step 4; otherwise, find schedule S_0, identify jobs u, v and w and go to Step 3.

j	a_j	b_j	c_j
1	$3W+1$	$2W$	1
2	1	1	$5W$
3	$2W$	$3W$	1

Table 1: Processing times for the tightness example

Step 3. If in schedule S_0 job w completes early, i.e., if (11) with $\lambda = \frac{2}{5}$ holds, then find schedule S_1 by running Algorithm EarlyW from Section 4.2.1 and go to Step 6. Otherwise, find schedule S_2 by running either Algorithm LateW1 from Section 4.2.2 for a Case 1 instance ($w < u \leq v$) or Algorithm LateW2 from Section 4.2.2 for a Case 2 instance ($w = u \leq v$) and go to Step 6.

Step 4. Identify the type of the instance: Type 1, Type 2 or Type 3, introduced in Section 4.3. For a Type 1 instance, find schedule $S_H^{(1)}$ by running Algorithm Type1 from Section 5 and go to Step 6. For a Type 2 instance, find schedule $S_H^{(2)}$ by running Algorithm Type2 from Section 6 and go to Step 6. For a Type 3 instance, go to Step 5.

Step 5. Identify job p that satisfies $a_p > \frac{2}{5}LB$ and find set $Q = N \setminus \{p\}$. If $a(Q) \leq \frac{2}{5}LB$, then find schedule $S_1^{(3)}$ by running Algorithm Type3.1 from Section 7 and go to Step 6. Otherwise, find the sets Q_1 and Q_2 by running Procedure Split. If $|Q_2| = 1$ then find schedule $S_2^{(3)}$ by running Algorithm Type3.2 from Section 7 and go to Step 6. If $|Q_2| > 1$ then find schedule $S_3^{(3)}$ by running Algorithm Type3.3 from Section 7 and go to Step 6.

Step 6. Out put the found schedule as schedule S_H. Stop.

The following statement holds.

Theorem 4 *For problem $A(BC)\,||\,C_{\max}$, Algorithm Main in $O(n\log n)$ time finds a schedule S_H such that (16) holds and the bound of $\frac{7}{5}$ is tight.*

Proof: Finding each schedule $S_{BC}(N)$ and S_0 requires $O(n\log n)$ time. It is easy to verify that other actions of the algorithm, including Procedure Split, require $O(n)$ time. Thus, the overall running time of Algorithm Main is $O(n\log n)$.

The bound of $\frac{7}{5}$ has been proved in the corresponding statements of the previous sections. To see that $\frac{7}{5}$ is a tight ratio guaranteed by Algorithm Main, consider the instance of problem $A(BC)\,||\,C_{\max}$ with three jobs and the processing times shown in Table 1; here W denotes a large positive number, $W \gg 1$.

The lower bound LB is equal to $\Phi_{BC}(N) = 5W+3$, obtained if the jobs are processed on machines B and C in the sequence $(2,1,3)$. There exists a optimal schedule S^* with $C_{\max}(S^*) = 5W+3$, which meets the lower bound; see Figure **??**(a).

ftbpFU6.1817in2.3877in0pt(a) optimal schedule S^*; (b) schedule S_H found by Algorithm Main FigTightFigure

In the instance under consideration, we have that $\frac{2}{5}LB < a_1 < \frac{3}{5}LB$, so that the conditions of Step 4 of Algorithm Main hold. Moreover, the instance under consideration is a Type 1 instance, so that Algorithm Main outputs schedule S_H shown in Figure **??**(b). It follows that $C_{\max}(S_H) = 7W+2$, so that the ratio $C_{\max}(S_H)/C_{\max}(S^*)$ approaches $\frac{7}{5}$ as W goes to infinity. ∎

9 Conclusion

The paper studies a version of the three-machine shop problem scheduling problem, in which processing routes for all jobs are defined by the same precedence graph with three nodes and one directed arc. The problem is NP-hard, and we design and analyze a 7/5-approximation algorithm. We demonstrate that a performance guarantee of 7/5 is achievable and that this bound is tight. The obtained bound compares favourably with known bounds for basic three-machine problems, including the classical open shop and flow shop.

References

[1] Brucker P. Scheduling algorithms. 5th ed. Berlin: Springer; 2007.

[2] Leung JY-T (ed.) Handbook of Scheduling: Algorithms, Models and Performance Analysis. London: Chapman & Hall/CRC; 2004.

[3] Pinedo M. Scheduling: Theory, Algorithms and Systems. 4th ed. Berlin: Springer; 2012.

[4] Chen B, Potts CN, Woeginger GJ. A review of machine scheduling: complexity, algorithms and approximability. In: Du D-Z, Pardalos PM, editors. Handbooks of Combinatorial Optimization, Dordrecht: Kluwer Academic Publishers; 1998, p. 21–169.

[5] Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB. Sequencing and scheduling: algorithms and complexity. In: Graves SC, Rinnooy Kan ANG, Zipkin PH, editors. Handbooks in Operations Research and Management Science, Vol. 4, Logistics of Production and Inventory, Amsterdam: North-Holland; 1993, p. 455–522.

[6] Masuda T, Ishii H, Nishida T. The mixed shop scheduling problem. Discr Appl Math 1985; 11: 158–73.

[7] Strusevich VA., Two machine super shop scheduling problem, J Oper Res Soc 1991; 42: 479–92.

[8] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Nav Res Log Quart 1954; 1: 61–8.

[9] Jackson JR. An extension of Johnson's result on job lot scheduling. Nav Res Log Quart 1956; 3: 201–3.

[10] Gonzalez T, Sahni S. Open shop scheduling to minimize finish time. J ACM 1976; 23: 665–79.

[11] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Math Oper Res 1976; 1: 117–29.

[12] Neumytov YD, Sevastianov SV. An approximation algorithm with an exact bound for the three-machine problem with the opposite routes. Upravlyaemye Sistemy 1993; 31: 53–65 (in Russian).

[13] Williamson DP, Hall LA, Hoogeveen JA, Hurkens CAJ, Lenstra JK, Sevastianov SV, Shmoys DB. Short shop schedules. Oper Res 1997; 43: 288–94.

[14] Hall LA. Approximability of flow shop scheduling. Math Progr 1998; 82: 175–90.

[15] Sevastianov SV, Woeginger GJ. Makespan minimization in open shops: a polynomial time approximation scheme. Math Progr 1998; 82: 191–8.

[16] Jansen K, Solis-Oba R, Sviridenko M. Makespan minimization in job shops: a linear time approximation scheme. SIAM J Discr Math 2003; 16: 288–300.

[17] Aksjonov VA. A polynomial-time algorithm of approximate solution of a scheduling problem. Upravlyaemye Systemy 1988; 28: 8–11 (in Russian).

[18] Gonzalez T, Sahni S. Flow shop and job shop schedules: complexity and approximation. Oper Res 1978; 26: 36–52.

[19] Feige U, Scheideler G. Improved bounds for acyclic job shop scheduling. Combinatorica 2002; 22: 361–99.

[20] Shmoys DB, Stein C, Wein J. Improved approximation algorithms for shop scheduling problems. SIAM J Comput 1994; 23: 617–32.

[21] Chen B, Glass CA, Potts CN, Strusevich VA. A new heuristic for three-machine flow shop scheduling. Oper Res 1996; 44: 891–8.

[22] Chen B, Strusevich VA. Approximation algorithms for three machine open shop scheduling. ORSA J Comput 1993; 5: 321–6.

[23] Strusevich VA. A greedy open shop heuristic with job priorities. Ann Oper Res 1998; 83: 253–70.

[24] Drobouchevitch IG, Strusevich VA. Heuristics for short route job shop scheduling problems. Math Meth Oper Res 1998; 48: 359–75.

[25] Strusevich VA, Drobouchevitch IG, Shakhlevich NV. Three-machine shop scheduling with partially ordered processing routes. J Oper Res Soc 2002; 53: 574-82.

[26] Shafransky YM, Strusevich VA. The open shop scheduling problem with a given sequence on one machine. Naval Res Log 1998; 45: 705–31.

[27] Adiri I, Pohoryles D. Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times. Naval Res Log Quart 1982; 29: 495–504.