Preemptive Scheduling on Two Identical Parallel Machines with a Single Transporter

Hans Kellerer
Institut für Statistik und Operations Research, Universität Graz,
Universitätstraße 15, A-8010, Graz, Austria
e-mail: hans.kellerer@uni-graz.at

Alan J. Soper 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: {A.J.Soper,V.Strusevich}@greenwich.ac.uk

Abstract

We consider a scheduling problem on two identical parallel machines, in which the jobs are moved between the machines by an uncapacitated transporter. In the processing preemption is allowed. The objective is to minimize the time by which all completed jobs are collected together on board the transporter. We identify the structural patterns of an optimal schedule and design an algorithm that either solves the problem to optimality or in the worst case behaves as a fully polynomial-time approximation scheme.

Keywords: Scheduling with transportation · Parallel machines · Approximation scheme

1 Introduction

We consider a processing system that consists of two identical parallel machines. The jobs are delivered to the system by a single transporter, moved between the machines by that transporter, and on their completion are transported away. This paper is a complete version of that by Kellerer et al. (2010), where due to the format of that publication not all proofs were included.

Integrating scheduling and logistics decision-making into a single model can be seen as one of the current trends of scheduling theory. In these enhanced models it is required to combine typical scheduling decisions with various logistics decisions, normally related to inventory control, machine breakdowns, maintenance, and various transportation issues.

In the scheduling literature there are several approaches that address the issue of scheduling with transportation. Normally, transportation occurs between the processing stages, and therefore more often than not the processing system is a *multi-stage* or *shop* system, e.g., the *flow shop* and the *open shop*. Recall that for two machines, e.g., denoted by A and B, in the case of the flow shop each job is first processed on machine A and then on machine B, while for the open shop, the processing route of each job is not known in advance. In both shop models, each job is seen as consisting of two operations, and the operations of the same job are not allowed to overlap. As a rule, for the problems considered and reviewed in this paper the objective is to minimize the completion time of all jobs on all machines.

In our study, we focus on approximability issues, which have been a topic of considerable interest in the area. A polynomial-time algorithm that creates a schedule with an objective function value that is at most $\rho \geq 1$ times the optimal value is called a $\rho-$*approximation* algorithm; the value of ρ is called a *worst-case ratio bound.* A family of $\rho-$approximation algorithms is called a *fully polynomial-time approximation scheme (FPTAS)* if $\rho = 1 + \varepsilon$ for any $\varepsilon > 0$ and the running time is polynomial with respect to both the length of the problem input and $1/\varepsilon$.

Reviews of four known types of scheduling models with a transportation component can be found in the papers by Lee and Strusevich (2005) and Lushchakova et al. (2009). Two of these types (the robotic cells and the transportation networks) appear to be less relevant to this study and are not discussed below.

The model with *Transportation Lags* is the most studied among those that combine scheduling with transportation. Here it is assumed that there is a known time lag between the completion of an operation and the start of the same job on the machine that is next in the processing route. These lags can be interpreted as *transportation times* needed to move a job between the machines, provided that the transportation device is always available. A detailed review of the complexity results on open shop and flow shop scheduling with transportation lags is given by Brucker et al. (2004). For the general case with job-dependent transportation lags, the two-machine open shop problem is unary $NP-$hard even if for any job the durations of its operations are equal. A $\frac{3}{2}-$ approximation algorithm for this problem is due to Strusevich (1999). The two-machine flow shop problem is unary NP-hard even if all processing times are unit, see Yu et al. (2004); several 2-approximation algorithms are given by Dell'Amico (1996). A polynomial-time approximation scheme for the classical flow shop problem developed by Hall (1998) can be modified to handle the transportation lags.

The model that we study in this paper belongs to the class of *Models with Interstage Transporters.* For the two-machine flow shop the general model of this type is introduced by Lee and Chen (2001). Assume that there are v transporters each capable of carrying c jobs between the machines. The transportation time from A to B is equal to τ, while the travel time of an empty transporter from B to A is equal to σ. The problem with $c = 1$ and $\sigma = 0$ is shown to be unary NP-hard by Hurink and Khust (2001). The problem with $v = 1$ and $c \geq 3$ is unary NP-hard, while the case of $c = 2$ is open, see Lee and Chen (2001). The open shop version of this problem is addressed by Lee and Strusevich (2005) and Lushchakova et al. (2009).

There are no known approximation results for these models, apart from the two-machine flow shop and open shop with a single uncapacitated transporter, i.e., $v = 1$ and $c \geq n$. For both the flow shop and open shop models, it is assumed that the jobs are brought by the transporter to one of the machines, moved between the machines in batches, and when the processing is over, the transporter collects the jobs together and carries them away. For this model, the objective is to minimize the time by which all completed jobs are collected on board the transporter.

The classes of heuristic flow shop schedules in which the jobs are split in at most b batches on each machine are studied by Lee and Strusevich (2005) for $b = 2$ and by Soper and Strusevich (2007) for $b = 3$, and $\frac{b+1}{b}-$approximation algorithms are designed, these ratios being the best possible as long as a heuristic schedule contains at most b batches. For the open shop counterpart of the above problem with $\tau = \sigma$ a $\frac{7}{5}-$ approximation algorithm is developed by Lushchakova et al. (2009).

The problem that we study in this paper belongs to the same family, and its main features are as follows. There are two identical parallel machines, M_1 and M_2. The processing time of a job $j \in N = \{1, 2, \ldots, n\}$ is equal to p_j. At time zero, the jobs are brought to the system by a transporter. Each job is either processed without preemption on one of the machines or its processing is split into several portions. In the latter case, the total duration of the portions of a job j is still equal to p_j. For a job to be (partially) processed on a machine, it must be delivered there by the transporter. A move of the transporter between the machines takes τ time units, and the number of jobs transferred by a move can be arbitrary. Extending the notation adopted by Lee and Strusevich (2005), we call this problem $TP2|v = 1, c \geq n|K_{\max}$, where $K_{\max}$ is the time by which all completed jobs are collected together on board the transporter.

We are aware of only one other study that combines scheduling on parallel machines with transportation, namely the paper by Qi (2006). As in our case, Qi's model also involves two parallel identical machines. However, there are several points of difference between the two models. First, the type of transportation used by Qi is essentially a transportation lag. Second, there is no preemption allowed in Qi's model. Third, the jobs are known to be assigned to the machines in advance and are moved only to be processed on the other machine and returned to the originally assigned machine.

The remainder of this paper is organized as follows. In Section 2 we establish properties of schedules that are optimal for problem $TP2|v = 1, c \geq n|K_{\max}$, and derive lower bounds on the optimal length. Section 3 describes and analyzes an algorithm that creates a schedule with an even number (either two or four) of moves of the transporter, while the case of the schedules with three moves is considered in Section 4. Section 5 presents an approximation algorithm that outputs a non-preemptive schedule. Some concluding remarks are contained in Section 6.

In all NP-hardness proofs the following decision problem is used.

PARTITION: Given positive integers $e_1, \ldots, e_r$ and E such that $\sum_{j=1}^{r} e_j = 2E$, does there exist a partition of the index set $R = \{1, \ldots, r\}$ into two subsets R_1 and R_2 such that $\sum_{j \in R_1} e_j = \sum_{j \in R_2} e_j = E$?

It is well known that PARTITION is NP-complete in the ordinary sense.

2 Feasible Schedules: Properties, Structure and Lower Bounds

In this section, we describe properties of optimal schedules for problem $TP2|v = 1, c \geq n|K_{\max}$, identify possible shapes of their structures and establish lower bounds on the optimal value of the objective function.

In problem $TP2|v = 1, c \geq n|K_{\max}$ the jobs of a set $N = \{1, 2, \ldots, n\}$ have to be processed on any of two identical processing machines, M_1 and M_2. The processing time of a job $j \in N$ is equal to p_j. Since the machines are identical, we may assume that all jobs are brought by the transporter to machine M_1 at time zero, so that the first move of the transporter is made from machine M_1 to machine M_2. Sometimes we refer to machine M_1 as the *top* machine and to machine M_2 as *bottom* machine; also the transporter will be said to move *down* if it moves from M_1 to M_2, and to move *up* otherwise. On their arrival, some of the jobs will be left at M_1 to be processed and totally completed on that machine. Some other jobs will be moved to machine M_2 to be processed and totally completed on that machine. There may be jobs that are partly processed on M_1 and partly on M_2; however each such job has to be transported to the corresponding machine for (partial) processing. The transporter can move any number of jobs at a time, and the length of a move in either direction is equal to τ time units. While a job is being transported it cannot be processed on either machine; besides, it is not allowed to process a job on both machines simultaneously. The objective is to minimize the *length* of a schedule, i.e., the time $K_{\max}$ by which all completed jobs are collected together on board the transporter.

For problem $TP2|v = 1, c \geq n|K_{\max}$, let S^* denote an optimal schedule, i.e., $K_{\max}(S^*) \leq K_{\max}(S)$ for all feasible schedules S. There are three possible types of a feasible schedule S:

Type 0: all jobs are processed on one machine M_1;

Type 1: the number of moves of the transporter in S^* is odd (upon their completion the jobs are collected at machine M_2);

Type 2: the number of moves of the transporter in S^* is even (upon their completion the jobs are collected at machine M_1).

In what follows, we assume that a Type 0 schedule is not optimal; otherwise, the problem is trivial.

Given a schedule S, the set of jobs that are processed only on machine M_1 (or on machine M_2 only) is denoted by $W_1(S)$ ($W_2(S)$, respectively). Notice that set W_i, $i \in \{1, 2\}$, includes the jobs that are processed on machine M_i without preemption, as well as those jobs that are processed on M_i in several portions, but not on the other machine. Let $Q(S)$ denote the set of jobs processed with preemption, partly on machine M_1 and partly on machine M_2. If no confusion arises, we may drop the reference to a schedule and write W_1, W_2 and Q.

If in a schedule S some job j is fractional, i.e., $j \in Q(S)$, then we denote by x_j and y_j the total length of the time intervals during which job j is processed on machine M_1 and machine M_2, respectively. Obviously, $x_j + y_j = p_j$. For a non-empty set $H \subseteq N$ of jobs, denote $p(H) := \sum_{j \in H} p_j$ and for completeness define $p(\varnothing) := 0$. In a similar sense, we use the pieces of notation $x(H)$, $y(H)$, etc.

Definition 1 *For problem $TP2|v = 1, c \geq n|K_{\max}$, a class of feasible schedules in which (i) the first move of the transporter starts at time zero, and (ii) the last move does not transfer any jobs that need to be completed is called* Class $\mathcal{S}$.

The following two lemmas show that the search for an optimal schedule can be reduced to schedules of Class $\mathcal{S}$.

Lemma 1 *For problem $TP2|v = 1, c \geq n|K_{\max}$, the search for an optimal schedule can be limited to the schedules in which the first move of the transporter from M_1 to M_2 starts at time zero.*

Proof: If in a feasible schedule S each job is processed on one machine only (i.e., $Q(S) = \varnothing$), then without increasing the value of $K_{\max}$ schedule S can be transformed into another schedule in which (i) all jobs to be processed on machine M_2 are brought there by the first move of the transporter and (ii) that move starts at time zero.

Now, for schedule S, suppose that $Q(S) \neq \varnothing$ and that the first move of the transporter starts at time t_1. Let $N(t_1)$ be the set of jobs such that for each job $j \in N(t_1)$ either its full processing on machine M_1 or the processing of its portion on M_1 is completed before time t_1. In set $N(t_1)$, distinguish between the jobs of set $W_1(S)$ and those of set $Q(S)$ by defining

$W(t_1) := W_1(S) \cap N(t_1)$ and $Q(t_1) := Q(S) \cap N(t_1)$. In turn, set $Q(t_1)$ can be partitioned as $Q(t_1) = Q'(t_1) \cup Q''(t_1)$, where set $Q'(t_1)$ consists of all jobs of $Q(t_1)$ that are transported to M_2 at time t_1 by the first move of the transporter. Denote the duration of the processing of job $j \in Q'(t_1)$ on machine M_1 before time t_1 by x'_j. Since the jobs of set $W(t_1)$ do not have to be moved between the machines and the jobs of set $Q''(t_1)$ are not moved to M_2 at time t_1, it follows that, without increasing the length of the schedule, we can

(i) interchange the jobs and portions of jobs in such a way that machine M_1 first processes the jobs of set $Q'(t_1)$ (each job as one portion), followed by the remaining jobs (or portions of jobs) of set $Q(t_1)$;

(ii) start the first move of the transporter at time $x'(Q'(t_1))$, on completion of all portions of jobs of set $Q'(t_1)$ on M_1.

In the resulting schedule S', the processing on machine M_2 starts upon arrival of the transporter at time $x'(Q'(t_1)) + \tau$. Let k be a job of set $Q'(t_1)$ whose portion completes on machine M_1 at time $x'(Q'(t_1))$ in schedule S'. Without changing the length of the schedule $K_{\max}(S')$, we can start the first move of the transporter at time $x'(Q'(t_1)) - x'_k$ and move the processing of the portion of job k to machine M_2 to start at time $x'(Q'(t_1)) - x'_k + \tau$, so that the processing of the other jobs and portions of jobs on M_2 can still start at time $x'(Q'(t_1)) + \tau$. Repeating this process, we obtain a schedule S'' such that $K_{\max}(S'') = K_{\max}(S') \leq K_{\max}(S)$ and in S'' the first move of the transporter starts at time zero. ∎

Lemma 2 *For problem $TP2|v = 1, c \geq n|K_{\max}$, the search for an optimal schedule can be limited to the schedules in which the last move of the transporter does not deliver to a machine $M \in \{M_1, M_2\}$ the jobs of set $Q(S)$ which still have to be processed on M.*

Proof: We present the proof for the case that the last move of the transporter is made from machine M_2 to machine M_1, i.e., the initial schedule S is a Type 2 schedule; the case of schedule S being a Type 1 schedule is symmetric.

Suppose that in S the transporter starts its last move at time t to arrive at M_1 at time $t+\tau$. Assume that among the jobs brought by this move, there is a non-empty set $Q' \subseteq Q(S)$ of fractional jobs, each of which still has to be processed on M_1. Denote the start time of the first job (or a portion of a job) on machine M_1 after time $t + \tau$ by t^0 and denote the duration of the processing of job $j \in Q'$ on machine M_1 after time t^0 by x'_j. Interchange the jobs and portions of jobs processed on M_1 after time t^0 in such a way that machine M_1 starting at time t^0 processes the jobs of set Q' (each job as one portion), followed by the remaining jobs, if any. Call the resulting schedule S'. Let l be the job of set Q' that starts on machine M_1 at time t^0. Without changing the length of the schedule, we can start the last move of the transporter at time $t^0 + x'_l - \tau$ and move the processing of the portion of job l to machine M_2 to start at time $t^0 - \tau$, so that the processing of the other jobs and portions of jobs on M_1 can still start at time $t^0 + x'_l$. Repeating this process, we obtain a schedule S'' such that $K_{\max}(S'') \leq K_{\max}(S') = K_{\max}(S)$ and in S'' the last move of the transporter does not deliver any jobs of set $Q(S)$ that need further processing on M_1. ∎

We now derive several lower bounds on the length of a feasible schedule.

Define

$$T = \frac{p(N)}{2} + \tau. \tag{1}$$

Lemma 3 *For any schedule S that is feasible for problem $TP2|v=1, c \geq n|K_{\max}$, the lower bound*

$$K_{\max}(S^*) \geq T \tag{2}$$

holds.

Proof: For schedule S, let X denote the total load on machine M_1, i.e., the sum of the processing times of the jobs and portions of jobs processed on M_1. If S is a Type 1 schedule then $K_{\max}(S^*) \geq X + \tau$ and $K_{\max}(S^*) \geq p(N) - X + \tau$. Otherwise, if S is a Type 2 schedule then $K_{\max}(S^*) \geq X$ and $K_{\max}(S^*) \geq p(N) - X + 2\tau$. In either case, the required lower bound (2) follows immediately. ■

It is not possible that there is exactly one move in an optimal schedule, since the length of such a schedule would be equal to $p(N) + \tau$, which is larger than $p(N)$, the length of a Type 0 schedule. Thus, in each Type 1 schedule there are at least three moves, so that the first move and the last move of the transporter are made from the top machine M_1, and the schedule terminates when all completed jobs are collected on board of the transporter at the bottom machine M_2. For any Type 1 schedule S the lower bound

$$K_{\max}(S) \geq p_j + \tau \tag{3}$$

holds for each job $j \in N$. Indeed, if a job p_j is completed on the top machine M_1, it has to be moved to machine M_2, where the schedule terminates. If it is completed on machine M_2, it has to be brought there before its (possibly, partial) processing may start on that machine. This implies the bound (3).

In each Type 2 schedule the first move of the transporter is made from the top machine M_1, the last move is made from the bottom machine M_2, and the schedule terminates when all completed jobs are collected on board of the transporter at the top machine M_1.

It is clear that for any Type 2 schedule S a lower bound

$$K_{\max}(S) \geq p_j \tag{4}$$

holds for each job $j \in N$. Besides, if in a Type 2 schedule S job j or its part is processed on machine M_2 then

$$K_{\max}(S^*) \geq p_j + 2\tau. \tag{5}$$

To see this, notice that job j must be brought to M_2 and returned to M_1, and while it is processed on M_2 or being moved it cannot be processed on M_1.

Definition 2 *A job $j \in N$ is said to have rank r, $r \in \{0, 1, 2, 3\}$ if*

$$p_j + r\tau \geq T. \tag{6}$$

It is clear that a job of rank $r - 1$ is also a job of rank r, $r \in \{1, 2, 3\}$. The rank of a job plays an important role in the proof of the main statement of this section, Theorem 1, and in several auxiliary statements.

In the proofs and algorithms presented in this paper, we will often use the following splitting procedure.

Procedure Split(X, Y, γ)

Input: Two sets X and $Y = \{\sigma(1), \ldots, \sigma(y)\}$ of jobs and a bound γ such that $|X| \geq 0$, $|Y| = y > 1$, and $p(X) < \gamma$, $p(X) + p(Y) > \gamma$

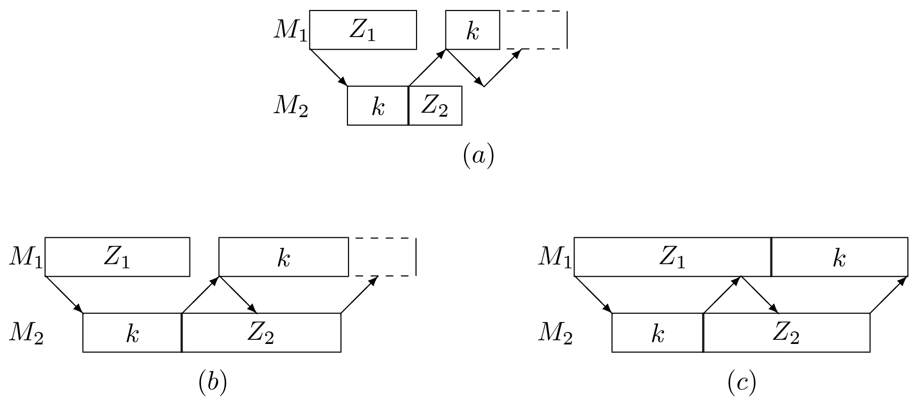

Figure 1: A preemptive schedule $S_4(k)$ with four moves

Output: A set $Y' \subset Y$ and a job $u \in Y$ that is split in two portions $p_u = x_u + y_u$, so that $p(X) + p(Y') + x_u = \gamma$

Step 1: Considering the jobs of set Y in the order given by the list σ, find the position v, where $1 \leq v \leq y$ such that

$$p(X) + \sum_{j=1}^{v-1} p_{\sigma(j)} < \gamma,\ p(X) + \sum_{j=1}^{v} p_{\sigma(j)} \geq \gamma.$$

Step 2: Output $Y' := \{\{\sigma(1), \ldots, \sigma(v-1)\}\}$, $u := \sigma(v)$, $x_u := \gamma - p(X) - p(Y')$, $y_u := p_u - x_u$.

It is clear that the running time of Procedure Split is linear in $|X \cup Y|$.

Given a schedule S, the set of whole jobs that are processed on machine M_1 (or on machine M_2 only) is denoted by $Z_1(S)$ ($Z_2(S)$, respectively). If no confusion arises, we may drop the reference to a schedule and write Z_1 and Z_2.

Lemma 4 *Let A and B be disjoint sets of jobs such that $p(A) \leq T$ and $p(A) + p(B) > T$. Suppose that Procedure Split(A, B, T) is run and a set $B' \subset B$ and a job $k \in B \backslash B'$ are found such that $p(A) + p(B') + x_k = T$, where $0 < x_k < p_k$. Define $Z_1 := A \cup B'$ and $Z_2 := N \backslash (Z_1 \cup \{k\})$. Let $S_4(k)$ be a Type 2 schedule shown in Figure 1. Then either*

- *job k is a rank 2 job,*
- *or $K_{\max}(S_4(k)) = T$, provided that $p(Z_2) \geq 2\tau$.*

Proof: In schedule $S_4(k)$ only job k is fractional, and $p(Z_1) + x_k = p(Z_2) + y_k + 2\tau = T$. If $y_k + 2\tau > T - x_k = p(Z_1)$, then $p_k + 2\tau > T$, i.e., job k is a rank 2 job, and the structure of schedule $S_4(k)$ is as shown in Figure 1(a) or (b). If $y_k + 2\tau \leq T - x_k = p(Z_1)$ then job k is delivered to machine M_1 before all jobs of set Z_1 are completed on that machine. Furthermore, due to the condition $p(Z_2) \geq 2\tau$, the next downward move of the transporter is finished no later than all jobs are completed on machine M_2. Thus, the structure of schedule $S_4(k)$ is as shown in Figure 1(c) and $K_{\max}(S_4(k)) = T$. ■

Lemma 5 *Let A and B be disjoint sets of jobs such that $p(A) \leq T - \tau$ and $p(A) + p(B) > T - \tau$. Suppose that Procedure Split$(A, B, T - \tau)$ is run and a set $B' \subset B$ and a job $k \in B \backslash B'$ are found such that $p(A) + p(B') + x_k = T - \tau$, where $0 < x_k < p_k$. Define $Z_1 := A \cup B'$ and $Z_2 := N \backslash (Z_1 \cup \{k\})$. Let $S_3(k)$ be a Type 1 schedule shown in Figure 2. Then either*

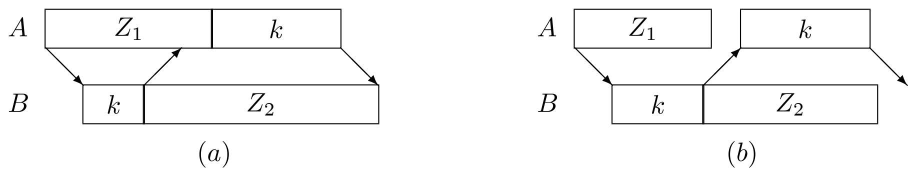

Figure 2: A preemptive schedule $S_3(k)$ with three moves

- $K_{\max}(S_3(k)) = T$
- *or job k is a rank 3 job and $K_{\max}(S_3(k)) = p_k + 3\tau$.*

Proof: In schedule $S_3(k)$ only job k is fractional, and the load of each machine is equal to $T - \tau$, i.e., $p(Z_1) + x_k = p(Z_2) + y_k = T - \tau$. If $y_k + 2\tau \leq (T - \tau) - x_k = p(Z_1)$ then job k is delivered to machine M_1 before all jobs of set Z_1 are completed on that machine, the structure of schedule $S_3(k)$ is as shown in Figure 2(a), and $K_{\max}(S_3(k)) = T$. If $y_k + 2\tau > (T - \tau) - x_k$, then $p_k + 3\tau > T$, i.e., job k is a rank 3 job. Moreover, job k can only start on M_1 at time $y_k + 2\tau$, so that $K_{\max}(S_3(k)) = p_k + 3\tau$ and the structure of schedule $S_3(k)$ is as shown in Figure 2(b). ■

Lemma 6 *For problem $TP2|v = 1, c \geq n|K_{\max}$, suppose that S is a schedule with at least five moves. If for some job $j \in N$ the inequality*

$$p_j + 3\tau > K_{\max}(S) \tag{7}$$

holds, then

$$p_j > 2\tau \tag{8}$$

and

$$p_j > p(N)/4. \tag{9}$$

Proof: Since there are at least five moves in schedule S, it follows that a lower bound

$$K_{\max}(S) \geq 5\tau \tag{10}$$

holds. Then (7) implies (8). Due to (2), we deduce from (7) that $p_j + 3\tau > T = p(N)/2 + \tau$, which together with (8) yields (9). ■

Theorem 1 *For problem $TP2|v = 1, c \geq n|K_{\max}$, the search for an optimal schedule can be limited to schedules with at most 4 moves of the transporter.*

Proof: Suppose that S^0 is an optimal schedule with at least five moves, and there is no optimal schedule with less than five moves. For schedule $S = S^0$ inequality (10) holds. To prove the theorem, we will show that we can always construct a schedule S^* with $2, 3$ or 4 moves and such that $K_{\max}(S^*) \leq K_{\max}(S^0)$.

If necessary, renumber the jobs in such a way that

$$p_1 \geq p_2 \geq p_3 \geq \max\{p_j | j \in N \backslash \{1, 2, 3\}\}. \tag{11}$$

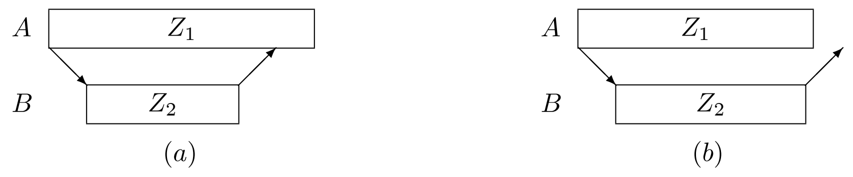

Figure 3: A schedule with two moves

Rank 0 job. Assume that job 1 has rank 0. Construct the following schedule $S_2(1)$ with two moves, in which machine M_1 processes job 1 alone and the rest of the jobs are processed on machine M_2. The first move of the transporter (from M_1 to M_2) starts at time zero, and the second move (from M_2 to M_1) starts at $\tau + p(N) - p_1$ to complete at $2\tau + p(N) - p_1$. See Figure 3(a) with $Z_1 = \{1\}$, $Z_2 = N \backslash \{1\}$. We use (6) with $r = 0$ to deduce $2\tau + p(N) - p_1 \leq 2T - T = T \leq p_1$, so that $K_{\max}(S_2(1)) = p_1$, and this schedule is optimal due to (4).

In the rest of this proof, it is assumed that there is no job of rank 0.

Rank 1 job. Suppose that job 1 has rank 1, which is equivalent to $p_1 \geq p(N)/2$, i.e., there are no other rank 1 jobs.

Construct a Type 1 schedule $S_3(1)$ in which machine M_1 processes only job 1 while machine M_2 processes the rest of the jobs. At time zero, the transporter leaves job 1 on machine M_1, moves to M_2 to deliver the jobs of set $N \backslash \{1\}$, immediately returns to machine M_1, and then moves the completed job 1 from M_1 to M_2.

If $K_{\max}(S_3(1)) = 3\tau$, which happens if $p_1 < 2\tau$, then $K_{\max}(S_3(1)) < 5\tau \leq K_{\max}(S^0)$. Otherwise, since $p_1 \geq p(N)/2 \geq p(N) - p_1$, it follows that $K_{\max}(S_3(1)) = p_1 + \tau$. If there exists an optimal Type 1 schedule then $S_3(1)$ is optimal due to (3).

Thus, in our further analysis of the rank 1 job, we assume that an optimal schedule S^0 is a Type 2 schedule, i.e., in this schedule there are at least 6 moves, and therefore $K_{\max}(S^0) \geq 6\tau$ and additionally $K_{\max}(S^0) < K_{\max}(S_3(1)) = p_1 + \tau$. Notice that this implies that in schedule S^0 job 1 is fully processed on machine M_2, since otherwise we would have $K_{\max}(S^0) \geq p_1 + 2\tau$ due to (5).

Consider the situation that

$$p_2 + 2\tau < T.$$

Run Procedure Split$(\{1\}, N \backslash \{1\}, T)$ to find a set $H_1 \subseteq N \backslash \{1\}$ and a job k such that $p_1 + p(H_1) + x_k = T$. If $x_k = 0$, then we create a schedule with two moves in which machine M_1 processes the jobs of set $H_1 \cup \{1\}$, while machine M_2 processes the remaining jobs; the length of this schedule is T; see Figure 3 with $Z_1 = H_1 \cup \{1\}$, $Z_2 = N \backslash Z_1$. If $0 < x_k < p_k$, then notice that $p_k + 2\tau \leq p_2 + 2\tau < T$. If for set $Z_2 = N \backslash (H_1 \cup \{1, k\})$, the inequality $p(Z_2) \geq 2\tau$ holds, then due to Lemma 4 applied to $A = \{1\}$ and $B = N \backslash \{1\}$, the structure of schedule $S_4(k)$ with four moves is as shown in Figure 1(c), and $K_{\max}(S_4(k)) = T$, so that this schedule is optimal due to (2). On the other hand, if $p(Z_2) < 2\tau$, then we derive $p_1 + p(H_1) + p(Z_2) = p(N) - p_k > T$, and since $p_1 + p(H_1) < T$, we can run Procedure Split$(\{1\} \cup H_1, Z_2, T)$ to find a set $H_1' \subseteq Z_2$ and job $h \in Z_2$ such that $p_1 + p(H_1) + p(H_1') + x_h = T$. If $x_h = 0$, then we create a schedule with two moves in which machine M_1 processes the jobs of set $H_1 \cup H_1' \cup \{1\}$, while machine M_2 processes the remaining jobs; the length of this schedule is T, see Figure 3 with $Z_1 = H_1 \cup H_1' \cup \{1\}$, $Z_2 = N \backslash Z_1$. If $0 < x_h < p_h$, then notice that $p_h + 2\tau \leq p_2 + 2\tau < T$.

Define $Z'_1 := H_1 \cup H'_1 \cup \{1\}$ and $Z'_2 := N \setminus (Z'_1 \cup \{h\})$. If the inequality $p(Z'_2) \geq 2\tau$ holds, then due to Lemma 4 applied to $A = \{1\} \cup H_1$ and $B = Z_2$, the structure of schedule $S_4(h)$ with four moves is as shown in Figure 1(c) with $k = h$, so that $K_{\max}(S_4(h)) = T$ and this schedule is optimal due to (2). On the other hand, if $p(Z'_2) < 2\tau$, then $K_{\max}(S_4(h)) = \tau + y_h + 3\tau$ and since $y_h < p(Z_2) < 2\tau$, we conclude that $K_{\max}(S_4(h)) = 4\tau + y_h < 6\tau \leq K_{\max}(S^0)$.

Assume now that

$$p_2 + 2\tau \geq T,$$

i.e., job 2 is a rank 2 job. Notice that $p(N) - p_2 = 2T - 2\tau - p_2 \leq T$.

If in schedule S^0 job 2 or its part is processed on machine M_2, then (5) holds for $j = 2$. We can create a schedule $S_2(2)$ with two moves, in which machine M_2 processes job 2 alone and the rest of the jobs are processed on machine M_1; see Figure 3 with $Z_1 = N \setminus \{2\}$, $Z_2 = \{2\}$. The first move of the transporter (from M_1 to M_2) starts at time zero, and the second move (from M_2 to M_1) starts at $\tau + p_2$ to complete at $2\tau + p_2 \geq p(N) - p_2$. Thus, $K_{\max}(S_2(2)) = p_2 + 2\tau$, and this schedule is optimal due to (5) with $j = 2$.

If in schedule S^0 job 2 is fully processed on machine M_1, then $K_{\max}(S^0) \geq p_1 + p_2$. However, $K_{\max}(S^0) < p_1 + \tau$ implies that $p_2 < \tau$, which together with $p_2 + 2\tau \geq T$ yields $3\tau > T$. We use $K_{\max}(S^0) \geq 6\tau$ to derive $K_{\max}(S^0) > 2T > p(N)$, i.e., S^0 is worse than a Type 0 schedule, which is not possible.

No Rank 1 jobs. From now on, we assume that there is no job of rank 1, so that $p_j < T - \tau$ for each $j \in N$.

In the remaining part of this proof, a job $j \in N$ is called *special*, if (i) job j is a rank 3 job, and (ii) $p_j + 3\tau > K_{\max}(S^0)$. Let g denote the number of special jobs. As follows from Lemma 6, for each special job j the inequalities (8) and (9) hold, so that $g \in \{0, 1, 2, 3\}$.

If $g \in \{0, 1\}$, then run Procedure Split$(\{1\}, N \setminus \{1\}, T - \tau)$ to find a set $H_1 \subset N \setminus \{1\}$ and a job $k \in N \setminus \{1\}$ such that $p_1 + p(H_1) + x_k = T - \tau$. If $x_k = 0$, we construct a non-preemptive schedule with three moves of length T, in which machine M_1 processes the set of jobs $H_1 \cup \{1\}$, and machine M_2 processes the rest of the jobs. If $x_k > 0$, then notice that job k is not special since $p_k \leq p_1$, i.e., either job k is not a rank 3 job and $p_k + 3\tau < T$, or it is a rank 3 job but $p_k + 3\tau \leq K_{\max}(S^0)$. Applying Lemma 5 with $A = \{1\}$ and $B = N \setminus \{1\}$, we derive that for schedule $S_3(k)$ either $K_{\max}(S_3(k)) = T$ or $K_{\max}(S_3(k)) = p_k + 3\tau \leq K_{\max}(S^0)$ holds.

If $g = 2$, then we have two special jobs, job 1 and job 2. Run Procedure Split$(\{1\}, N \setminus \{1, 2\}, T - \tau)$ to find a set $H_1 \subset N \setminus \{1, 2\}$ and a job $k \in N \setminus \{1, 2\}$ such that $p_1 + p(H_1) + x_k = T - \tau$. As above, if $x_k = 0$, we construct a non-preemptive schedule with three moves of length T. If $x_k > 0$, notice that job k is not special since $p_k < p_2 \leq p_1$, i.e., job k either is not a rank 3 job and $p_k + 3\tau < T$, or it is a rank 3 job but $p_k + 3\tau \leq K_{\max}(S^0)$. Applying Lemma 5 with $A = \{1\}$ and $B = N \setminus \{1, 2\}$, we derive that a Type 1 schedule $S_3(k)$ is of structure shown in Figure 2(a) and $K_{\max}(S_3(k)) = T$.

Finally, assume that $g = 3$, i.e., we have three special jobs: job 1, job 2 and job 3. Let R denote the set of all other jobs; none of these jobs is special.

We prove that in the case under consideration job 2 cannot have rank 2. Assuming the opposite, we derive

$$p_2 + 2\tau \geq T = \frac{p(N)}{2} + \tau \geq \frac{p_1}{2} + \frac{p_2}{2} + \frac{p_3}{2} + \tau,$$

which reduces to

$$\frac{p_2}{2} + \tau \geq \frac{p_1}{2} + \frac{p_3}{2}.$$

However the latter inequality is impossible due to $p_1 \geq p_2$ and (8) with $j = 3$.

Thus, job 2 is not a rank 2 job, i.e., $p_2 + 2\tau < T$. It follows that $p_1 + p_3 + p(R) = (2T - 2\tau) - p_2 > T$. On the other hand, $p_1 + p(R) = p(N) - p_2 - p_3 < p(N) - p(N)/4 - p(N)/4 = p(N)/2 < T$. Run Procedure Split$(\{1, R\}, \{3\}, T)$ to find the value x_3 such that $p_1 + p(R) + x_3 = T$. Notice that job 3 is not a rank 2 job since $p_3 \leq p_2 < T - 2\tau$. Applying Lemma 4 with $A = \{1, R\}$, $B = \{3\}$ and $k = 3$, we derive that a Type 2 schedule $S_4(3)$ with four moves is of structure shown in Figure 1(c) and $K_{\max}(S_4(3)) = T$.

This completes the proof of the theorem.

An immediate implication of Lemmas 1 and 2 and of Theorem 1 is that an optimal schedule will be sought for among the schedules of Class $\mathcal{S}$ with two, three or four moves.

Lemma 7 *For problem $TP2|v = 1, c \geq n|K_{\max}$, the search for the best schedule with two moves can be limited to the non-preemptive schedules.*

Proof: Suppose for a schedule $S \in \mathcal{S}$ the set $Q(S)$ of fractional jobs is not empty. By Lemma 1 each job $j \in Q(S)$ is brought to machine M_2 by the first move of the transporter that starts at time zero, and has to be completed subsequently on M_1. However, by Lemma 2, the second (last) move of the transporter cannot deliver any unfinished jobs. ■

Lemma 8 *For problem $TP2|v = 1, c \geq n|K_{\max}$, let $S_3 \in \mathcal{S}$ be a schedule with three moves in which some jobs are fractional. Then without increasing the value of the objective function, schedule S_3 can be transformed into either a non-preemptive schedule or into a schedule $S_3(k)$ in which only some job $k \in Q(S_3)$ remains fractional. In the latter case, the lower bound*

$$K_{\max}(S_3(k)) \geq p_k + 3\tau \tag{12}$$

holds.

Proof: Suppose that in schedule $S_3 \in \mathcal{S}$ a job $j \in Q(S_3)$ starts its processing on machine M_1. Since the first move of the transporter starts at time zero, it follows that job j is brought to machine M_2 by the third (last) move. However, upon arrival at M_2 job j requires further processing, which contradicts Lemma 2.

Thus, any job of set $Q(S_3)$ starts its processing on machine M_2 and is delivered there at time τ. After their processing on machine M_2 is finished, all these jobs together are delivered to M_1 to be completed on that machine.

Suppose that $|Q(S_3)| = \ell \geq 2$, and renumber (if required) the jobs of this set so that $Q(S_3) = \{1, 2, \ldots, \ell\}$. Without increasing the objective function, we can transform schedule S_3 into a new schedule S'_3 in which:

- the jobs of set $Q(S_3)$ are processed on machine M_2 in the time interval $[\tau, \tau + y(Q(S_3))]$ and are followed by an arbitrary sequence of the whole jobs;
- the second move of the transporter (from M_2 to M_1) starts immediately after all portions of the fractional jobs are completed on M_2, i.e., at time $\tau + y(Q(S_3))$;
- the jobs of set $Q(S_3)$ are processed on machine M_1 as a block after all whole jobs on that machine.

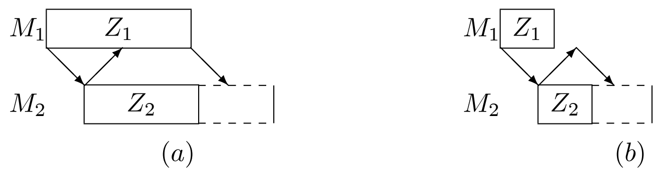

Figure 4: A non-preemptive schedule with three moves

Thus, in schedule S'_3, the set of jobs is partitioned into the set of fractional jobs $Q(S'_3) = Q(S_3)$ and the sets $Z_1(S'_3)$ and $Z_2(S'_3)$ of the whole jobs assigned to machine M_1 and machine M_2, respectively. It follows that

$$\begin{aligned} K_{\max}(S_3) \quad \geq \quad & K_{\max}(S'_3) = \max\left\{p(Z_1(S'_3)) + x(Q(S_3)) + \tau, \tau + y(Q(S_3)) + p(Z_2(S'_3)),\right. \\ & \left.\tau + y(Q(S_3)) + \tau + x(Q(S_3)) + \tau\right\}. \end{aligned}$$

Run Procedure Split$(\varnothing, Q(S_3), y(Q(S_3))$, which will scan the jobs of set $Q(S_3)$ in the order of their numbering and find a job $k \leq \ell$ such that

$$\sum_{j=1}^{k-1} p_j < y(Q(S_3)), \ \sum_{j=1}^{k} p_j \geq y(Q(S_3)).$$

If $\sum_{j=1}^{k} p_j = y(Q(S_3))$ then schedule S'_3 can be transformed into a non-preemptive schedule S''_3 shown in Figure 4 with $Z_1 = Z_1(S'_3) \cup \{k+1, \ldots, \ell\}$ and $Z_2 = Z_2(S'_3) \cup \{1, \ldots, k\}$. It is clear that

$$K_{\max}(S''_3) = \max\left\{p(Z_1(S'_3)) + x(Q(S_3)) + \tau, \tau + y(Q(S_3)) + p(Z_2(S'_3)), 3\tau\right\} \leq K_{\max}(S'_3).$$

Alternatively, create a schedule $S_3(k)$ shown in Figure 2 with $Z_1 = Z_1(S'_3) \cup \{k+1, \ldots, \ell\}$ and $Z_2 = Z_2(S'_3) \cup \{1, \ldots, k-1\}$ and the fractional job k split by $y_k = y(Q_3(S_3)) - \sum_{j=1}^{k-1} p_j$, $x_k = p_k - y_k$, so that

$$y(Q_3(S_3)) = y_k + \sum_{j=1}^{k-1} p_j, \ x(Q_3(S_3)) = x_k + \sum_{j=k+1}^{\ell} p_j.$$

Thus, we obtain that

$$\begin{aligned} K_{\max}(S_3) \quad \geq \quad & K_{\max}(S_3(k)) = \max\left\{p(Z_1(S'_3)) + x(Q(S_3)) + \tau, \tau + y(Q(S_3)) + p(Z_2(S'_3)),\right. \\ & \left.\tau + y_k + \tau + x_k + \tau\right\}, \end{aligned}$$

which in particular implies the required lower bound (12). ■

Lemma 9 *For problem $TP2|v = 1, c \geq n|K_{\max}$, the search for the best schedule with four moves can be limited to the schedules in which some jobs are assigned to be processed preemptively on both machines.*

Proof: Suppose S^*_4 is the best schedule with four moves in which no job is processed on both machines. Let Z_1 and Z_2 denote the sets of jobs assigned to machine M_1 and to machine M_2, respectively. It follows that $K_{\max}(S^*_4) \geq \max\{p(Z_1), p(Z_2) + 2\tau\}$. But there exists a two-move schedule of length $\max\{p(Z_1), p(Z_2) + 2\tau\}$, as shown in Figure 3. ■

Lemma 10 *For problem $TP2|v=1, c \geq n|K_{\max}$, let S_4 be a schedule with four moves. Then*

$$K_{\max}(S_4) \geq 4\tau + y(Q), \tag{13}$$

where Q is the set of jobs that are processed with preemption on both machines.

Proof: In schedule $S_4 \in \mathcal{S}$, any job of set $Q = Q(S_4)$ is delivered to M_2 either by the first move of the transporter (and each of these jobs is brought back to be completed on M_1 by the second move) or by the third move of the transporter (and each of these jobs is completed on M_2 before it is brought back to M_1 by the last move of the transporter). This implies that no job of set Q is processed on machine M_2 between the second and the third moves of the transporter. The desired lower bound (13) follows immediately. ■

Theorem 2 *For problem $TP2|v=1, c \geq n|K_{\max}$, let W_2^* be a set of jobs such that $p(W_2^*) \leq T - 2\tau$ and $p(W_2^*) \geq p(W)$ for any set $W \subseteq N$ with $p(W) \leq T - 2\tau$. Let S_4^* be the best schedule with four moves. Then*

$$K_{\max}(S_4^*) \geq y^* + 4\tau, \tag{14}$$

where

$$y^* = \frac{p(N) - p(W_2^*)}{2} - 2\tau. \tag{15}$$

Proof: Suppose that in schedule S_4^* machine M_2 processes a set W_2 of jobs that are processed on this machine only and a set Q of the jobs that are processed preemptively on both machines.

It is clear that the length of schedule S_4^* cannot be shorter than the total processing of the jobs assigned to machine M_1, i.e.,

$$K_{\max}(S_4^*) \geq p(N) - p(W_2) - y(Q),$$

Due to Lemma 10, we have that

$$K_{\max}(S_4^*) \geq y(Q) + 4\tau.$$

For a fixed set W_2, let y be the root of the equation

$$p(N) - p(W_2) - y(Q) = y(Q) + 4\tau,$$

i.e.,

$$y = \frac{p(N) - p(W_2)}{2} - 2\tau.$$

Then $K_{\max}(S_4^*) \geq y + 4\tau$, and by definition of set W_2^* the lower bound (14) holds. ■

The examples below exhibit instances of problem $TP2|v=1, c \geq n|K_{\max}$ for which an optimal schedule includes two, three or four moves of the transporter. The examples also demonstrate that all established lower bounds are tight.

In all five listed instances the transportation time τ is equal to 4. Optimal schedules for Instances 1 and 2 have no preemption; in the other schedules exactly one job is fractional. For Instances 1-4 the value of T is a tight lower bound, possibly together with another bound. For Instance 5, the optimal length of the schedule is strictly larger than T. Here $T = 16$, so that either $W_2^* = \{2\}$ or $W_2^* = \{3\}$ (as in Figure 7(b)), and $p(W_2^*) = 7 < T - 2\tau = 8$, see Theorem 2. Thus, the value of $y^* = 0.5$ is computed in accordance with (15).

Instance	p_1	p_2	p_3	T	$K_{\max}(S)$	Tight lower bound	Moves in S^*	Chart of S^*
1	10	7	5	15	15	(2,5)	2	Figure 5
2	10	7	3	14	14	(2)	3	Figure 6(a)
3	9	2	9	14	14	(2,12)	3	Figure 6(b)
4	10	8	8	17	17	(2)	4	Figure 7(a)
5	10	7	7	16	16.5	(14)	4	Figure 7(b)

Table 1: Instances of the problem

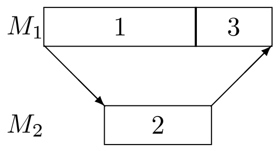

Figure 5: An optimal schedule with two moves for Instance 1

Lemmas 1 and 2, Theorem 1 and the presented examples imply that we can search for an optimal schedule among the schedules of class $\mathcal{S}$, which include either an even (two or four) number of moves or three moves. We split our further consideration accordingly.

In this paper, in various algorithms we use the following statement; see Lemma 4.6.1 in Kellerer et al. (2004).

Theorem 3 *Consider the subset-sum problem of the form*

$$\begin{array}{ll} \max & \sum_{j\in H} p_j x_j \\ & \sum_{j\in H} p_j x_j \le c \\ & x_j \in \{0,1\},\ j \in H \subseteq N, \end{array} \tag{16}$$

This problem admits an FPTAS that for a given positive ε either finds an optimal solution $x_j^* \in \{0,1\}$, $j \in H$ such that

$$\sum_{j\in H} p_j x_j^* < (1-\varepsilon)c$$

or finds an approximate solution $x_j^\varepsilon \in \{0,1\}$, $j \in H$, such that

$$(1-\varepsilon)c \le \sum_{j\in H} p_j x_j^\varepsilon \le c.$$

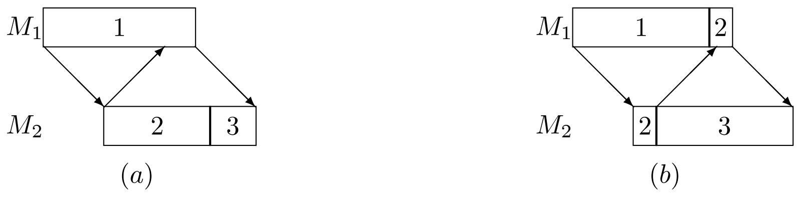

Figure 6: An optimal schedule with three moves for Instances 2 and 3

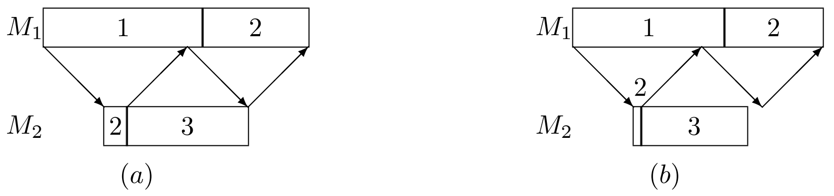

Figure 7: An optimal schedule with four moves for Instances 4 and 5

Such an FPTAS requires no more than $O(n/\varepsilon)$ time.

3 Finding the Best Schedule with an Even Number of Moves

In this section, we show that finding the best schedule with two moves is an NP-hard problem. We also give an algorithm that behaves as an FPTAS, provided that the number of moves in an optimal schedule is either 2 or 4.

Theorem 4 *For problem $TP2|v = 1, c \geq n|K_{\max}$, finding the best schedule with two moves is NP-hard in the ordinary sense.*

Proof: To prove the theorem, we provide a reduction of an arbitrary instance of Partition to the decision version of problem $TP2|v = 1, c \geq n|K_{\max}$ with $n = r + 1$ jobs.

Define $\tau := 2E$ and $p_j := e_j,\ j = 1, 2, \ldots, r$, while $p_{r+1} := 4E$. We show that for the constructed instance a schedule S with two moves such that $K_{\max}(S) \leq 5E$ exists if and only if Partition has a solution.

Suppose that Partition has a solution, and R_1 and R_2 are the required subsets with $e(R_1) = e(R_2) = E$. Define a schedule S with two moves of the transporter, in which machine M_1 processes the jobs of the set R_1 and job $r + 1$ in the time interval $[0, 5E]$, while machine M_2 processes the jobs of set R_2 in the time interval $[2E, 3E]$. The transporter brings the jobs of set R_2 to machine M_2 at time $2E$, waits till the jobs are completed and brings them back to M_1. It is easy to verify that $K_{\max}(S) = 5E$.

Suppose that schedule S with two moves and of length at most $5E$ exists. Due to Lemma 7, there is no fractional job in S. Job $r + 1$ is processed on machine M_1; otherwise, $K_{\max}(S) \geq \tau + p_{r+1} + \tau = 6E$. Let the total processing time on machine M_2 be denoted by X. If $X < E$ then $K_{\max}(S) \geq p_{r+1} + (2E - X) > 5E$, which is impossible. If $X > E$, then $K_{\max}(S) \geq 2\tau + X > 5E$. Therefore, $X = E$. This implies that the total processing time on machine M_1 is equal to $5E$. If for $i \in \{1, 2\}$ we denote the subset of jobs of set $\{1, 2, \ldots, r\}$ assigned to machine M_i by R_i, then the sets R_1 and R_2 form a solution to Partition. This proves the theorem. ■

Theorem 5 *For problem $TP2|v = 1, c \geq n|K_{\max}$ with $p(N) \geq 4\tau$, finding the best preemptive schedule with four moves is NP-hard in the ordinary sense.*

Proof: As in the previous NP-hardness proof, we reduce Partition to the decision version of problem $TP2|v = 1, c \geq n|K_{\max}$. Without loss of generality, in this proof we assume that in Partition the value of E is at least 2.

Define $n := r+2$, $\tau := E^2+1$, $p_j := 2e_jE,\ j = 1,2,\ldots,r$, $p_{r+1} := 2E^2+4$ and $p_{r+2} := 3$. We show that for the constructed instance a preemptive schedule S with four moves such that $K_{\max}(S) \le 4E^2+5.5$ exists if and only if Partition has a solution. Notice that $p(N) = 6E^2+7 > 4\tau$, as required.

Suppose that Partition has a solution, and R_1 and R_2 are the required subsets with $e(R_1) = e(R_2) = E$. Define a schedule $S_4(k)$ as in Figure 8 with $Z_1 = R_1 \cup \{r+1\}$, $Z_2 = R_2$ and $k = r+2$ split by $x_k = y_k = 1.5$. It is easy to verify that $K_{\max}(S_4(k)) = y_{r+2} + 4\tau = 4E^2+5.5$.

Now let a schedule S with four moves and with $K_{\max}(S) \le 4E^2+5.5$ exist with Z_1 and Z_2 being the sets of the whole jobs assigned to machine M_1 and M_2, respectively. In this schedule job $r+1$ cannot be fractional, since otherwise due to (5), $K_{\max}(S) > p_{r+1} + 2\tau = 4E^2+6$. Thus, job $r+1$ belongs to Z_1. Due to (13), the total length Y of all fractional jobs on machine M_2 is at most 1.5 time units.

For schedule S, suppose that $p(Z_2) > 2\tau$. Since $K_{\max}(S) \ge \tau + Y + p(Z_2) + \tau$, we deduce that $Y + p(Z_2) \le 2E^2 + 3.5$. If $Y \ge 1$, then since $p(Z_2)$ is integer, we have that actually $p(Z_2) \le 2E^2+2 = 2\tau$, a contradiction. On the other hand, if $0 < Y < 1$, then due to integrality of $p(Z_2)$, we derive that $p(Z_2) = 2E^2+3$. If job $r+2$ does not belong to Z_2, then there exists a non-negative integer x such that $p(Z_2) = 2E(E+x)$. However, the equality $2E(E+x) = 2E^2+3$ implies that $x = 3/(2E) < 1$, which is impossible. Otherwise, if job $r+2$ belongs to Z_2, we deduce that $Z_2 = \{r+2\} \cup R_2$, where $p(R_2) = 2E^2$, i.e., the sets R_2 and $R_1 = \{1,2,\ldots,r\} \backslash R_2$ form a solution to Partition.

From now on, assume that $p(Z_2) \le 2E^2+2 = 2\tau$. If $p(Z_2) < 2E^2$ then the total load on machine M_1 becomes $p(N) - Y - p(Z_2) > 6E^2 + 7 - 1.5 - 2E^2 = 4E^2 + 5.5$. Thus, $2E^2 \le p(Z_2) \le 2E^2+2$.

If set Z_2 contains job $r+2$, then, since the processing time of any job except $r+2$ is an even number, if follows that $p(Z_2)$ is odd, so that in fact $p(Z_2) = 2E^2+1$. This means that for some positive integer x we have $p(Z_2) = 2E(E-x)+3 = 2E^2+1$, which yields $x = \frac{1}{E} < 1$, a contradiction.

Now, if set Z_2 does not contain job $r+2$, then for some non-negative integer x we have $p(Z_2) = 2E(E+x) \le 2E^2+2$, which yields $x \le \frac{1}{E}$, and since x is integer we deduce that $x = 0$. As above, we can define $R_2 := Z_2$ and $R_1 := \{1,2,\ldots,r\} \backslash R_2$ and obtain a solution to Partition. This proves the theorem. ■

The following algorithm uses the FPTAS for the subset-sum problem for finding a solution to problem $TP2|v=1, c \ge n|K_{\max}$ that is as close to the optimum as desired.

Algorithm MoveEven

Step 1. Compute T in accordance with (1).

Step 2. Given an $\varepsilon > 0$, run an FPTAS for the subset-sum problem of the following structure

$$\begin{array}{ll} \max & \sum_{j\in N} p_j x_j \\ & \sum_{j\in N} p_j x_j \le T \\ & x_j \in \{0,1\},\ j \in N. \end{array} \tag{17}$$

For a found solution of problem (17), determine $Z_1' := \left\{ j \in N | x_j^{\varepsilon} = 1 \right\}$ and $Z_2' := \left\{ j \in N | x_j^{\varepsilon} = 0 \right\}$ and create schedule S_2' shown in Figure 3 with $Z_1 = Z_1'$ and $Z_2 = Z_2'$. If $p(Z_1') \ge (1-\varepsilon)T$, then go to Step 7.

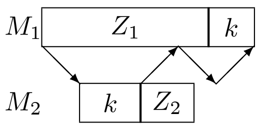

Figure 8: Schedule S''_4 with four moves and $y_k = y^*$

Step 3. Given an $\varepsilon > 0$, run an FPTAS for the subset-sum problem of the following structure

$$\begin{array}{ll} \max & \sum_{j\in N} p_j x_j \\ & \sum_{j\in N} p_j x_j \le T - 2\tau \\ & x_j \in \{0,1\},\ j \in N, \end{array} \tag{18}$$

For a found solution of problem (18), determine $Z''_2 := \left\{j \in N | x^\varepsilon_j = 1\right\}$ and $Z''_1 := \left\{j \in N | x^\varepsilon_j = 0\right\}$. Create schedule S''_2 shown in Figure 3 with $Z_1 = Z''_1$ and $Z_2 = Z''_2$. If $p(Z''_2) \ge (1-\varepsilon)(T - 2\tau)$, then go to Step 7.

Step 4. Find set $B_2 := \{j \in N | p_j + 2\tau > T\}$ of the jobs of rank 2. If either $p(Z''_1) \le 4\tau$ or $p(B_2) \ge T$, then go to Step 7; otherwise go to Step 5.

Step 5. If $p(Z''_2) \ge 2\tau$, select an arbitrary job $k \in Z''_1 \backslash B_2$, define $y_k := (T - 2\tau) - p(Z''_2)$ and $x_k := p_k - y_k$, make a schedule S'_4 with four moves as shown in Figure 1(c) with $Z_1 = Z''_1 \backslash \{k\}$, $Z_2 = Z''_2$ and go to Step 7; otherwise go to Step 6.

Step 6 Compute y^* by formula (15) with $W^*_2 = Z''_2$. Select an arbitrary job $k \in Z''_1 \backslash B_2$, define $y_k := y^*$, and $x_k := p_k - y_k$, make a schedule S''_4 with four moves as shown in Figure 8 with $Z_1 = Z''_1 \backslash \{k\}$, $Z_2 = Z''_2$ and go to Step 7.

Step 7. Output the best of the found schedules as schedule S^ε.

It is clear that Algorithm MoveEven requires $O(n/\varepsilon)$ time. Below we analyze its performance.

Theorem 6 *For problem $TP2|v = 1, c \ge n|K_{\max}$, Algorithm MoveEven behaves as an FPTAS, provided that there exists an optimal schedule S^* either with two or with four moves.*

Proof: Suppose first that in an optimal schedule S^* the transporter makes two moves.

Consider schedule S'_2 found in Step 2. Notice that $p(Z'_1) \le T$. We have that $K_{\max}(S'_2) = \max\{p(Z'_1), p(Z'_2) + 2\tau\}$. If $p(Z'_1) \ge (1-\varepsilon)T$, then we derive $p(Z'_2) + 2\tau \le 2T - (1-\varepsilon)T = (1+\varepsilon)T$. Thus, due to (2) we obtain that $K_{\max}(S'_2) \le (1+\varepsilon)T \le (1+\varepsilon)K_{\max}(S^*)$.

Similarly, for schedule S''_2 found in Step 3, notice that $p(Z''_2) + 2\tau \le T$. We have that $K_{\max}(S''_2) = \max\{p(Z''_1), p(Z''_2) + 2\tau\}$. If $p(Z''_2) \ge (1-\varepsilon)(T - 2\tau)$, then we derive $p(Z''_1) = p(N) - p(Z''_2) = 2(T-\tau) - p(Z''_2) \le 2(T-\tau) - (1-\varepsilon)(T-2\tau) = (1+\varepsilon)T - 2\varepsilon\tau$. Thus, due to (2) we obtain that $K_{\max}(S''_2) \le (1+\varepsilon)T \le (1+\varepsilon)K_{\max}(S^*)$.

If for schedule S'_2 found in Step 2 the inequality $p(Z'_1) < (1-\varepsilon)T$ holds, then due to Theorem 3, the solution found by the FPTAS is an optimal solution to problem (17). This means that the value $p(Z_1')$ cannot be enlarged and the value $p(Z_2') + 2\tau$ cannot be reduced, as long as we require $p(Z_1') \le T$, i.e., $K_{\max}(S^*) \ge p(Z_2') + 2\tau$, provided that there are two moves in schedule S^*.

Similarly, if for schedule S_2'' found in Step 3 the inequality $p(Z_2'') < (1-\varepsilon)(T - 2\tau)$ holds, then due to Theorem 3, the solution found by the FPTAS is an optimal solution to problem (18). This means that the value $p(Z_2'')+2\tau$ cannot be enlarged and the value $p(Z_1'')$ cannot be reduced, as long as we require $p(Z_2'') \le T-2\tau$, i.e., $K_{\max}(S^*) \ge p(Z_1'')$. The algorithm outputs schedule S^ε such that $K_{\max}(S^\varepsilon) \le \min\{K_{\max}(S_2'), K_{\max}(S_2'')\} = \min\{p(Z_2') + 2\tau, p(Z_1'')\}$, which is optimal, provided that there are two moves in schedule S^*.

Thus, if there are two moves in an optimal schedule, Algorithm MoveEven outputs a schedule of a length that is at most $1+\varepsilon$ times the optimum.

The conditions of Step 4 describe situations in which the algorithm still outputs the best of the two-move schedules found so far, and such a schedule is optimal. If $p(Z_1'') \le 4\tau$ then no schedule with four moves can be shorter than $K_{\max}(S_2'') = p(Z_1'')$, i.e., for an optimal schedule with an even number of moves the equality $K_{\max}(S^*) = \min\{K_{\max}(S_2'), K_{\max}(S_2'')\}$ holds. For the set B_2 of rank 2 jobs found in Step 4 suppose that $p(B_2) \ge T$. If in schedule S^* each job of set B_2 is processed only on machine M_1, then $K_{\max}(S^*) \ge p(B_2)$. In this case, schedule S_2'' found in Step 3 is optimal, since no job of rank 2 may belong to set Z_2'', so that $Z_2'' = N\backslash B_2$, $Z_1'' = B_2$ and $K_{\max}(S_2'') = p(B_2)$. If in S^* a job of rank 2 is processed, even partly, on machine M_2 then due to (5) we deduce that $K_{\max}(S^*) \ge p_u + 2\tau$, where $u \in B_2$ is the shortest rank 2 job. In this case, schedule S_2' found in Step 2 is optimal, since $Z_1' = N\backslash\{u\}$, $Z_2' = \{u\}$ and $K_{\max}(S_2') = p_u + 2\tau$.

In the rest of the proof, it is assumed that there are four moves in an optimal schedule S^*. We arrive at Step 5 if $p(Z_1'') > 4\tau$ and $p(B_2) < T$. Consider the sets Z_1'' and Z_2'' found in Step 3. Since no rank 2 job belongs to set Z_2'', we deduce that $B_2 \subset Z_1''$ and there exists a job $k \in Z_1''\backslash B_2$ such that $p_k + 2\tau < T$.

If $p(Z_2'') > 2\tau$, then due to Lemma 4, we can create schedule S_4' shown in Figure 1(c) with $K_{\max}(S_4') = T$, which means this schedule is optimal; see Step 5.

On the other hand, if $p(Z_2'') \le 2\tau$, then $p(Z_1'') = p(N) - p(Z_2'') \ge (2T - 2\tau) - 2\tau = 2T - 4\tau$. Recall that the FPTAS in Step 3 solves the corresponding subset-sum problem optimally, so that for the value y^* found by formula (15) with $W_2^* = Z_2''$ the lower bound $K_{\max}(S^*) \ge y^* + 4\tau$ holds. Notice that

$$y^* = \frac{p(Z_1'')}{2} - 2\tau.$$

so that it follows from $p(Z_1'') > 4\tau$ that $y^* > 0$. Since the solution found by the FPTAS is an optimal solution to problem (18), it follows that for any job $k \in Z_1''$ the inequality $p(Z_2'') + p_k > T - 2\tau$ holds, which is equivalent to $p(Z_1'') - p_k < T$. Thus, we derive from $p(Z_1'') \ge 2T - 4\tau$ that

$$p_k > p(Z_1'') - T = y^* + \frac{p(Z_1'')}{2} + 2\tau - T \ge y^*.$$

In Step 6, the algorithm selects a job $k \in Z_1''\backslash B_2$ and assigns it to be processed on machine M_2 for y^* time units, where as proved above, $0 < y^* < p_k$. For job $k \in Z_1''\backslash B_2$, we derive

$$p(Z_1'') - p_k = \frac{p(Z_1'')}{2} + \frac{p(Z_1'')}{2} - p_k \ge \frac{p(Z_1'')}{2} + (T - 2\tau) - (T - 2\tau) = \frac{p(Z_1'')}{2}.$$

In schedule S_4'' job k is delivered back to machine M_1 at time $2\tau + y^* = \frac{p(Z_1'')}{2} < p(Z_1'') - p_k$, so that the structure of schedule S_4'' is as shown in Figure 8 with $Z_1 = Z_1''\backslash\{k\}$. Since

$K_{\max}(S_4'') = y^* + 4\tau = p(Z_1'') - y^* = \frac{p(Z_1'')}{2} + 2\tau$, this schedule is optimal due to (14).

■

Notice that if an instance of the problem contains a job of rank 0, Algorithm MoveEven outputs schedule S_2'' as an optimal schedule. Indeed, for a rank 0 job k we have that $p_k \geq T$, so that $p(N\backslash\{k\}) \leq T - 2\tau$ and in Step 3 we will have $Z_1'' = \{k\}$, $Z_2'' = p(N\backslash\{k\})$. Such a schedule is shown in Figure 3(a). This observation allows us to exclude from further consideration the instances of the problem with a rank 0 job, since for the purpose of finding a global optimal solution there is no need to create schedules with more than two moves.

4 Finding the Best Schedule with Three Moves

In this section, we show that for problem $TP2|v = 1, c \geq n|K_{\max}$ finding the best schedule with three moves is an NP-hard problem and give an algorithm that for many instances finds an exact solution, but in general behaves as an FPTAS.

In this section we refer to a rank 3 job as *long*; otherwise, if a job does not satisfy (6) for $r = 3$, it is called *short.* As seen from the consideration below, the presence of the short jobs is crucial for fast finding an exact solution to the problem.

As proved in Lemma 8, the search for the best schedule with three moves can be limited to (i) non-preemptive schedules with the closest possible loads on the machines (see Figure 4), or (ii) a preemptive schedules with a single fractional job (see Figure 2), for which the value of the objective function meets a lower bound.

First, we prove that under some conditions finding such a schedule is an NP-hard problem.

Theorem 7 *For problem $TP2|v = 1, c \geq n|K_{\max}$ with no short jobs and $p(N) \geq 4\tau$, finding the best non-preemptive schedule with three moves is NP-hard in the ordinary sense.*

Proof: As in the previous NP-hardness proofs, we reduce Partition to the decision version of problem $TP2|v = 1, c \geq n|K_{\max}$.

Define $n = r$, $\tau = E$ and $p_j = 2e_j$, $j = 1, 2, \ldots, n$. We show that for the constructed instance a non-preemptive schedule S with three moves and such that $K_{\max}(S) \leq 3E$ exists if and only if Partition has a solution. Notice that $p(N) = 4E = 4\tau$, as required.

Suppose that Partition has a solution, and R_1 and R_2 are the required subsets with $e(R_1) = e(R_2) = E$. Define a schedule S as in Figure 4(a) with $Z_1 = R_1$. It is easy to verify that $K_{\max}(S_3) = 3E$.

Suppose that a non-preemptive schedule S with three moves and with $K_{\max}(S) \leq 3E$ exists. The first (downward) move starts at time zero, the second (upward) move starts at time $\tau = E$, and the final (downward) move starts at $2E$ and completes at $3E$. All jobs assigned to machine M_1 are processed in the time interval $[0, 2E]$, while those assigned to machine M_2 are processed in the interval $[E, 3E]$. If R_i denotes the set of the jobs of set $\{1, 2, \ldots, n\}$ assigned to machine M_i, for $i \in \{1, 2\}$, then the sets R_1 and R_2 form a solution to Partition. This proves the theorem. ■

Theorem 8 *For problem $TP2|v = 1, c \geq n|K_{\max}$ with no short jobs and $p(N) \geq 4\tau$, finding the best preemptive schedule with three moves is NP-hard in the ordinary sense.*

Proof: As in the previous NP-hardness proofs, we reduce Partition to the decision version of problem $TP2|v=1, c \geq n|K_{\max}$. Without loss of generality, in this proof we assume that in Partition the value of E is at least 3.

Define $n = r+3$, $\tau = E^2+1$, $p_j = 2e_jE$, $j = 1, 2, \ldots, r$, $p_{r+1} = p_{r+2} = p_{r+3} = 3$. We show that for the constructed instance a preemptive schedule S with three moves such that $K_{\max}(S) \leq 3E^2+6$ exists if and only if Partition has a solution.

Notice that $p(N) = 4E^2+9 > 4\tau$, as required. Besides, there is no short job, since $p_j \geq 3$ for all $j \in N$, so that $p_j + 3\tau \geq 3 + 3\tau = 3E^2+6 > T = \left(4E^2+9\right)/2 + E^2 + 1 = 3E^2+5.5$.

Suppose that Partition has a solution, and R_1 and R_2 are the required subsets with $e(R_1) = e(R_2) = E$. Define a schedule $S_3(k)$ as in Figure 2(b) with $Z_1 = R_1 \cup \{r+1\}$, $Z_2 = R_2 \cup \{r+2\}$ and $k = r+3$ being the fractional job with $x_{r+3} = y_{r+3} = 1.5$. Since $y_{r+3} + 2\tau > p(Z_1)$ and $x_{r+3} + 2\tau > p(Z_2)$, we deduce that $K_{\max}(S_3(k)) = p_{r+3} + 3\tau = 3E^2+6$.

Now let a preemptive schedule S with three moves and with $K_{\max}(S) \leq 3E^2+6$ exist. Due to Lemma 8 exactly one job k is fractional and $K_{\max}(S) \geq p_k + 3\tau$. This implies that the fractional job is one of the jobs of the set $\{r+1, r+2, r+3\}$. Without loss of generality, assume that job $k = r+3$ is fractional, and it is processed on machine M_1 during x_k time units and on machine M_2 during $y_k = 3 - x_k$ time units. Additionally assume that $p(Z_2) \leq p(Z_1)$. If either $p(Z_1) > 2\tau + y_k$ or $p(Z_2) > 2\tau + x_k$, then we would have $K_{\max}(S) \geq \max\{p(Z_1)+x_k, p(Z_2)+y_k\} + \tau > 3\tau + p_{r+3} = 3E^2+6$, which is impossible. Thus, $p(Z_2) \leq \frac{1}{2}\left(p(Z_1)+p(Z_2)\right) \leq \frac{1}{2}\left(4\tau + y_k + x_k\right) = 2\tau + 1.5 = 2E^2+3.5$. Since for each job of set Z_2 the processing time is integer, we obtain that actually $p(Z_2) \leq 2E^2+3$. Set Z_2 contains l jobs of length 3, where $l \in \{0,1,2\}$ (i.e., some or none of jobs $r+1$ and $r+2$). For $l \in \{0,1\}$ we have that for some non-negative integer x the inequality $p(Z_2) = 2E(E+x) + 3l \leq 2E^2+3$ holds, which yields $x \leq \frac{3-3l}{2E} < 1$. For $l = 2$ we have that $p(Z_2) = 2E(E-x)+6$ for some non-negative integer x, and $p(Z_1) = p(N) - p_k - p(Z_2) = 2E(E+x)$. Since $K_{\max}(S) \geq p(Z_1) + \tau$, we deduce $2E(E+x) \leq 2E^2+5$, so that $x \leq \frac{5}{2E} < 1$. Since x must be integer, we conclude that $x = 0$. If the set R_i denotes the set of the jobs of set $\{1, 2, \ldots, r\}$ assigned to machine M_i, for $i \in \{1,2\}$, then the sets R_1 and R_2 form a solution to Partition. This proves the theorem. ■

Algorithm Move3 presented below finds the best schedule with three moves for problem $TP2|v=1, c \geq n|K_{\max}$.

Algorithm Move3Try

Step 1. Compute T in accordance with (1).

Step 2. Take an $\varepsilon > 0$ and run an FPTAS for the subset-sum problem of the following structure

$$\begin{array}{ll} \max & \sum_{j \in H} p_j x_j \\ & \sum_{j \in H} p_j x_j \leq T - \tau \\ & x_j \in \{0,1\}, \ j \in H, \end{array} \tag{19}$$

where $H = N$. For a found solution of problem (**??**), define $H^{(1)} = \left\{ j \in H | x_j^{\varepsilon} = 1 \right\}$.

Step 3. Create a non-preemptive schedule S_3 shown in Figure 4 with $Z_1 = H^{(1)}$. If $p(H^{(1)}) < (1-\varepsilon)(T-\tau)$, then go to Step 4.

Step 4. Determine set $B := \{j \in N | p_j \geq 2\varepsilon (T-\tau)\}$. Let $|B| = h$. Renumber the jobs of set B so that $p_1 \leq p_2 \leq \ldots \leq p_h$. Define $\underline{k} := 1$ and $\overline{k} := h$.

(a) Compute $k = \left\lceil \left(\overline{k} + \underline{k}\right)/2 \right\rceil$. If $\underline{k} = \overline{k}$, then go to Step 5.

(b) Define $N_k = N \backslash \{k\}$. Compute $\varepsilon_k = p_k/(T-\tau)$, take $\varepsilon = \varepsilon_k$ and apply an FPTAS to the subset-sum problem of the form (**??**) with $H = N_k$. For a found solution of problem (**??**), define $H^{(1)} = \left\{ j \in H | x_j^{\varepsilon} = 1 \right\}$.

(c) If $p(H^{(1)}) \geq (1-\varepsilon_k)(T-\tau)$, then define $\overline{k} := k$; otherwise define $\underline{k} := k$. Return to Step 7(a).

Step 5. For the current value of k, define $Z_1 := H^{(1)}$ and $Z_2 := N \backslash (Z_1 \cup \{k\})$. Create a schedule $S_3(k)$ shown in Figure 2(b).

The statements below analyze the performance of Algorithm Move3.

Theorem 9 *For, problem $TP2|v=1, c \geq n|K_{\max}$ for finding the best candidate schedule with three moves Algorithm Move3 behaves as an FPTAS that requires at most $O((n \log n)/\varepsilon)$ time.*

Proof: If in Step 3, $p(H^{(1)}) \geq (1-\varepsilon)(T-\tau)$, then $p(H^{(1)}) \leq T - \tau \leq p(N \backslash Z_1) \leq (1+\varepsilon)(T-\tau)$ and $K_{\max}(S_3) = \max\{\tau + p(N\backslash Z_1), 3\tau\}$, i.e., schedule S_3 is either optimal with $K_{\max}(S_3) = 3\tau$ or $K_{\max}(S_3) = \tau + p(N \backslash Z_1) \leq (1+\varepsilon) T - \varepsilon\tau \leq (1+\varepsilon) K_{\max}(S^*)$.

If
$$p\left(H^{(1)}\right) < (1-\varepsilon)(T-\tau) \tag{20}$$
then due to Theorem 3, the value $p\left(H^{(1)}\right)$ is optimal for the corresponding subset-sum problem of the form (**??**), i.e., $p\left(H^{(1)}\right) \geq p(X)$ for any set $X \subset H$ such that $p(X) \leq T - \tau$. Thus, in this case schedule S_3 is the best non-preemptive schedule with three moves.

A possibly better schedule can be found among the preemptive schedules. Recall that in such a preemptive schedule $S_3(k)$ there is exactly one fractional job k; see Figure 2. Below we show that under the condition (20), the duration p_k of the fractional job cannot be smaller than $2\varepsilon(T-\tau)$.

Indeed, suppose that for schedule $S_3(k)$, the sets of the whole jobs on machine M_i is denoted by Z_i, $i \in \{1,2\}$. The fractional job is split between the machines so that $p_k = x_k + y_k$, where
$$= p(Z_2) + y_k = T - \tau. \tag{21}$$

The latter equation implies that $\max\{p(Z_1), p(Z_2)\} < T - \tau$. Since $p\left(H^{(1)}\right) \geq \max\{p(Z_1), p(Z_2)\}$, it follows from (20) that
$$\max\{p(Z_1), p(Z_2)\} < (1-\varepsilon)(T-\tau),$$
which leads to $p_k > 2\varepsilon(T-\tau)$.

In a preemptive schedule $S_3(k)$ with the fractional job k either $K_{\max}(S_3(k)) = T$ or, if job k is a Rank 3 job, the length $K_{\max}(S_3(k))$ is given by $p_k + 3\tau$. Thus, we need to find the shortest k with $p_k \geq 2\varepsilon(T-\tau)$ that can be fractional in schedule $S_3(k)$. This is done by the binary search procedure that is outlined in Step 4. Notice only that job k can be fractional, for which $p(H^{(1)}) + p_k \geq (T-\tau)$ for the corresponding set $H^{(1)}$.

To justify the binary search to be performed in Step 4, we apply the following reasoning. If in some iteration of the search procedure for the current job k we find the set $H^{(1)} = H_k^{(1)}$ such that $p\left(H_k^{(1)}\right) < (1-\varepsilon_k)(T-\tau)$, then due to the choice of ε_k the inequality $p\left(H_k^{(1)}\right) + p_k <$

$T-\tau$ holds, so that job k cannot be fractional. In this case, we continue our search among those jobs that are longer than p_k. To show that the required fractional job cannot be found among the jobs with indices less than k, assume that for some $k' < k$ the set $H^{(1)} = H^{(1)}_{k'}$ is found such that $p\left(H^{(1)}_{k'}\right) \geq (1-\varepsilon_{k'})(T-\tau)$, which implies that $p\left(H^{(1)}_{k'}\right) + p_{k'} \geq T-\tau$, so that job k' can be fractional. Since $p_k \geq p_{k'}$, we derive that $p\left(H^{(1)}_{k'}\right) > p\left(H^{(1)}_{k}\right)$. The latter inequality is impossible if $k \notin H^{(1)}_{k'}$, since $p\left(H^{(1)}_{k}\right) \geq p(X)$ for any set $X \subseteq N \backslash \{k\}$. On the other hand, if $k \in H^{(1)}_{k'}$ then we derive that $p\left(H^{(1)}_{k}\right) < p\left(\left(H^{(1)}_{k'} \cup \{k'\} \backslash \{k\}\right)\right)$, again a contradiction.

If for the current job k, the set $H^{(1)}$ is such that $p(H^{(1)}) \geq (1-\varepsilon_k)(T-\tau)$, then job k can be fractional and we need to consider only those jobs that are shorter than p_k. Eventually in at most $O(\log h)$ iterations the shortest possible fractional job will be found.

For the found job k, in Step 5 schedule $S_3(k)$ is found with the split $p_k = x_k + y_k$ of job k defined by (**??**). If job k is a Rank 3 job, i.e., $p_k \geq T - 3\tau$, we deduce from $p(Z_1) + x_k = T - \tau$ that $p(Z_1) + p_k = T - \tau + y_k$, and therefore $2\tau + y_k > p(Z_1)$; similarly, $2\tau + x_k > p(Z_2)$. Thus, in this case, schedule $S_3(k)$ is as shown in Figure 2(b), and $K_{\max}(S_3(k)) = p_k + 3\tau$. This schedule is optimal due to the choice of job k. If job k has a smaller rank, i.e., $p_k < T - 3\tau$, then both inequalities $2\tau + y_k \leq p(Z_1)$ and $2\tau + x_k \leq p(Z_2)$ hold, i.e., schedule $S_3(k)$ is as shown in Figure 2(a), and $K_{\max}(S_3(k)) = T$.

The best of all found schedules is the best candidate schedule with three moves. Step 4 involves sorting which requires $O(h \log h)$ time and $O(\log h)$ applications of an FPTAS. Since each time the FPTAS is run with $\varepsilon_k \geq 2\varepsilon$, the running time for each run does not exceed $O(n/\varepsilon)$. Thus, the overall running time of running Algorithm Move3 does not exceed $O((n \log n)/\varepsilon)$. ■

Thus, Algorithm Move3 behaves as an FPTAS, provided that there exists an optimal schedule with three moves, and finds an optimal schedule in some cases.

The general algorithm for handling problem $TP2|v=1, c \geq n|K_{\max}$ involves the following stages:

1. Create a schedule S_0 in which all jobs are processed on machine M_1.
2. Run Algorithm MoveEven.
3. Run Algorithm Move3.
4. Output the best of all found schedules.

It follows from the results of Sections 3 and 4 that for some instances of the problem the final algorithm will find an optimal schedule, while for others it will behave as an FPTAS. The running time of the algorithm does not exceed $O(n \log n + n/\varepsilon)$.

5 Non-Preemptive Schedules

In this section, for problem $TP2|v=1, c \geq n|K_{\max}$ we study the quality of the non-preemptive schedule compared with that of the global optimal schedule. We exhibit an instance of the problem for which the length of a non-preemptive schedule cannot be less than $\frac{4}{3}$ times the optimal length and present an algorithm that delivers the best possible non-preemptive schedule. In Step 3 the algorithm outputs a schedule with two moves and in Step 4 it outputs a schedule with three moves. The length of the best of all found schedules is at most $\frac{4}{3}$ times the optimum length.

Algorithm NoPmtn

Step 1. Create a schedule S_0 in which all jobs are processed on machine M_1.

Step 2. Compute T in accordance with (1).

Step 3. If there exists a job $k \in N$ such that

$$p_k \geq T,$$

perform Step 3(a), otherwise perform Step 3(b).

(a) Create a schedule S_2 shown in Figure 3(a) with $Z_1 = \{k\}$ and $Z_2 = N \backslash \{k\}$.

(b) Take $\varepsilon = \frac{1}{3}$ and apply an FPTAS to the subset-sum problem (17). For a found solution, define $N_1 = \left\{ j \in N | x_j^{\varepsilon} = 1 \right\}$ and $N_2 = N \backslash N_1$.

(b.1) Create a schedule S_2' shown in Figure 3(b) with $Z_1 = N_1$ and $Z_2 = N_2$.

(b.2) If $|N_2| \geq 2$, take an arbitrary job $q \in N_2$ and create a schedule S_2'' shown in Figure 3(a) with $Z_1 = N_1 \cup \{q\}$ and $Z_2 = N \backslash Z_1$.

Step 4. If there exists a job $k \in N$ such that

$$p_k + \tau \geq T,$$

perform Step 4(a), otherwise perform Step 4(b).

(a) Find a schedule S_3 with three moves shown in Figure 4 with $Z_1 = \{k\}$ and $Z_2 = N \backslash \{k\}$.

(b) Take $\varepsilon = \frac{1}{3}$ and apply an FPTAS to the subset-sum problem (**??**) with $H = N$. For a found solution, define $N_1 = \left\{ j \in N | x_j^{\varepsilon} = 1 \right\}$ and $N_2 = N \backslash N_1$. Create a schedule S_3' with three moves shown in Figure 4 with $Z_1 = N_1$ and $Z_2 = N_2$.

Step 5. Output the best of all created schedules as a heuristic schedule S^H.

The running time of Algorithm NoPmtn does not exceed $O(n)$. Below we analyze its worst-case performance.

Theorem 10 *For problem $TP2|v = 1, c \geq n|K_{\max}$, Algorithm NoPmtn outputs a non-preemptive schedule S^H such that*

$$\frac{K_{\max}(S^H)}{K_{\max}(S^*)} \leq \frac{4}{3}. \tag{22}$$

For any $\varepsilon > 0$ there exists an instance of problem $TP2|v = 1, c \geq n|K_{\max}$ for which the length of the best non-preemptive schedule S is no less than $\left(\frac{4}{3} - \varepsilon\right) K_{\max}(S^*)$.

Proof: We start with proving the bound (22) and then prove that this bound is the best possible as long as the search for a heuristic solution is limited to the class of non-preemptive schedules.

We split the proof into several parts, based on the assumption regarding the type of an optimal schedule.

Case 0. If there exists an optimal schedule in which all jobs are assigned to one machine, then schedule S_0 is optimal.

Case 1. Assume that an optimal schedule S^* is a Type 2 schedule and analyze the schedules created in Step 3.

Suppose that there exists a rank 0 job $k \in N$. This implies that

$$p_k \geq \frac{p(N)}{2} + \tau,$$

and

$$p_k \geq p(N) - p_k + 2\tau.$$

The latter inequality guarantees that

$$K_{\max}(S_2) = \max\{p_k, p(N) - p_k + 2\tau\} = p_k,$$

so that S_2 is an optimal schedule due to (4).

In Step 3(b) for each job $j \in N$ the inequality

$$p_j < T$$

holds. Analyzing schedule S_2', we derive that

$$K_{\max}(S_2') = \max\{p(N_1), p(N) - p(N_1) + 2\tau\}.$$

Since $p(N_1) \leq T$, we only need to consider the situation that $K_{\max}(S_2') = p(N) - p(N_1) + 2\tau$.

A possible outcome of the FPTAS is that a solution x_j^ε, $j \in N$, will be found such that

$$(1 - \frac{1}{3})T \leq \sum_{j=1}^{n} p_j x_j^\varepsilon = p(N_1) \leq T,$$

so that

$$\begin{aligned} K_{\max}(S_2') &= p(N) - p(N_1) + 2\tau \leq p(N) - \frac{2}{3}T + 2\tau \\ &= 2(T - \tau) - \frac{2}{3}T + 2\tau = \frac{4}{3}T. \end{aligned}$$

An alternative outcome of the application of an FPTAS with $\varepsilon = \frac{1}{3}$ to problem (17) is that the problem will be solved optimally and for the found solution x_j^ε, $j \in N$, the inequality

$$(1 - \frac{1}{3})T > \sum_{j=1}^{n} p_j x_j^\varepsilon = p(N_1)$$

holds. Notice that $p(N_1) < \frac{2}{3}T = \frac{1}{3}p(N) + \frac{2}{3}\tau$ and $p(N_2) > \frac{2}{3}p(N) - \frac{2}{3}\tau$. For schedule S_2'' we have that

$$K_{\max}(S_2'') = \max\{p(N_1) + p_q, p(N) - p(N_1) - p_q + 2\tau\}.$$

For any job $q \in N_2$, it follows that that $p_q \leq p(N_1) < \frac{2}{3}T$ and $p(N_1) + p_q > T$, so that $p_q + p(N_1) \leq \frac{4}{3}T$. If $|N_2| = 1$, then the latter inequality implies that the bound (22) holds for $S^H = S_0$. Otherwise, if $|N_2| \geq 2$, then

$$p(N) - p(N_1) - p_q + 2\tau \leq p(N) - T + 2\tau = 2(T - \tau) - T + 2\tau = T,$$

and for $S^H = S_2''$ the bound (22) holds.

Case 2. Assume that an optimal schedule S^* is a schedule with three moves and analyze the schedules created in Step 4.

Suppose that there exists a rank 1 job $k \in N$. We deduce

$$p_k \geq \frac{p(N)}{2},$$

which is equivalent to

$$p_k \geq p(N) - p_k.$$

The latter inequality implies that

$$K_{\max}(S_3) = \max\{\tau + p_k, p(N) - p_k + \tau, 3\tau\} = \max\{\tau + p_k, 3\tau\},$$

so that schedule S_3 is in fact optimal due to (3).

In Step 4(b) for each job $j \in N$ the inequality

$$p_j + \tau < T$$

holds, which means that

$$p_j < \frac{p(N)}{2}. \tag{23}$$

We show that the FPTAS will always find a solution x_j^{ε}, $j \in N$, such that

$$(1 - \frac{1}{3})(T - \tau) \leq \sum_{j=1}^{n} p_j x_j^{\varepsilon} \leq T - \tau, \tag{24}$$

so that

$$\frac{2}{3}(T - \tau) \leq p(N_1) \leq (T - \tau).$$

If (24) were not true, then the instance would not contain a job k such that

$$p_k > \frac{2}{3}(T - \tau),$$

otherwise, the FPTAS would assign $x_k^{\varepsilon} = 1$ and $k \in N_1$.

If the outcome of the FPTAS is such that the inequality

$$(1 - \frac{1}{3})(T - \tau) > \sum_{j=1}^{n} p_j x_j^{\varepsilon} = p(N_1)$$

holds, it follows that any job $k \in N_2$ is such that $p(N_1) + p_k > T - \tau$. However, since $p_k \leq \frac{2}{3}(T - \tau)$, we derive that $p(N_1) + p_k < \frac{4}{3}(T - \tau)$. This implies that

$$\frac{2}{3}(T - \tau) \leq p(N_2 \backslash \{k\}) < T - \tau,$$

and the existence of such a set $N_2 \backslash \{k\}$ should have been identified by the FPTAS due to Theorem 3. Due to the derived contradiction, under the condition (23) the FPTAS will find a solution that satisfies (24).

Analyzing schedule S_3' we derive that

$$K_{\max}(S_3') = \max\{p(N_1) + \tau, p(N) - p(N_1) + \tau, 3\tau\}.$$

If the maximum in the right-hand side of the above equality is equal to 3τ, then S_3' is optimal. Otherwise,

$$p(N_1) + \tau \leq (T - \tau) + \tau = T$$

and

$$\begin{aligned} p(N) - p(N_1) + \tau &\leq p(N) - \frac{2}{3}(T - \tau) + \tau \\ &= 2(T - \tau) - \frac{2}{3}(T - \tau) + \tau \\ &= \frac{4}{3}(T - \tau) + \tau \leq \frac{4}{3}T. \end{aligned}$$

Thus, for $S^H = S_3'$ the bound (22) holds.

We have proved that for the best of the schedules found by Algorithm NoPmtn the bound (22) holds.

To see that the bound (22) cannot be improved for a non-preemptive schedule, consider the following instance of problem $TP2|v = 1, c \geq n|K_{\max}$. There are three jobs with $p_1 = p_2 = p_3 = 2$ and $\tau = \varepsilon$. The best non-preemptive schedule S^N is a schedule with two moves in which two jobs are processed on machine M_1 and one job on machine M_2, so that $K_{\max}(S^N) = 4$. The best preemptive schedule S^* is a schedule with three moves in which one of the jobs is split between the machines, with 1 unit of processing on each machine. We have that $K_{\max}(S^*) = 3 + \varepsilon$ and

$$\frac{K_{\max}(S^N)}{K_{\max}(S^*)} = \frac{4}{3 + \varepsilon} > \frac{4}{3} - \varepsilon.$$

This completes the proof of the theorem. ∎

6 Conclusion

The paper presents an algorithm for the problem on two identical parallel machines to minimize the length of a schedule, i.e., the time by which all completed jobs are collected together on board the transporter that brings all jobs to one of the machines and moves the jobs between the machines. The algorithm either solves the problem optimally, or behaves as an FPTAS for those instances that can be proved NP-hard. An important feature of the algorithm is the use of an FPTAS for the subset-sum problem that is known to delver an exact optimal solution under certain conditions. The paper continues the study of scheduling models with a single uncapacitated transporter, see Soper and Strusevich (2007) and Lushchakova et al. (2009).

It would be interesting to extend this approach to uniform machines, i.e., machines of different speeds. An ultimate research goal in this area will be the study of general models with more than two machines and several transporters of different capacities.

Acknowledgement

The first and third authors were partly supported by the EPSRC funded project EP/I018441/1 “Quadratic and Linear Knapsack Problems with Scheduling Applications”.

References

Brucker P, Knust S, Cheng TCE, Shakhlevich NV (2004) Complexity results for flow-shop and open-shop scheduling problems with transportation delays. Ann Oper Res 129: 81–106

Dell’Amico M (1996) Shop problems with two machines and time lags. Oper Res 44: 777–787

Hall LA (1998) Approximability of flow shop scheduling. Math Progr B 82: 175-190

Hurink J, Knust S (2001) Makespan minimization for flow-shop problems with transportation times and a single robot, Discrete Appl Math 112: 199–216

Kellerer H, Mansini R, Pferschy U, Speranza MG. (2003) An efficient fully polynomial approximation scheme for the Subset-Sum Problem. J Comput Syst Sci 66: 349-370.

Kellerer H, Pferschy U, Pisinger D (2004) Knapsack problems, Springer, Berlin

Kellerer H, Soper AJ, Strusevich VA (2010) Transporting jobs through a processing center with two parallel machines. In: Wu W, Daescu O (eds) COCOA 2010, Part I, Lect Notes Comput Sci 6508, pp. 408–422

Lee C-Y, Chen Z-L (2001) Machine scheduling with transportation times. J Schedul 4: 3-24

Lee C-Y, Strusevich VA (2005) Two-machine shop scheduling with an uncapacitated interstage transporter. IIE Trans 37: 725–736

Lushchakova IN, Soper AJ, Strusevich VA (2009) Transporting jobs through a two-machine open shop. Naval Res Log 56: 1–18

Qi X (2006) A logistic scheduling model: scheduling and transshipment for two processing centers. IIE Trans 38: 609–618

Soper AJ, Strusevich VA (2007) An improved approximation algorithm for the two-machine flow shop scheduling problem with an interstage transporter, Int J Found Comput Sci 18: 565–591.

Strusevich VA (1999) A heuristic for the two-machine open-shop scheduling problem with transportation times. Discrete Appl Math 1999: 287–304.

Yu W, Hoogeveen H, Lenstra JK (2004) Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. J Schedul 7: 333–348.