


Article

Two-Machine Job-Shop Scheduling with Equal Processing Times on Each Machine

Evgeny Gafarov ^{1,*} and Frank Werner ² 

¹ V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia

² Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany; frank.werner@mathematik.uni-magdeburg.de

* Correspondence: axel73@mail.ru; Tel.: +7-925-809-0907

Received: 23 January 2019; Accepted: 19 March 2019; Published: 25 March 2019



Abstract: In this paper, we consider a two-machine job-shop scheduling problem of minimizing total completion time subject to n jobs with two operations and equal processing times on each machine. This problem occurs e.g., as a single-track railway scheduling problem with three stations and constant travel times between any two adjacent stations. We present a polynomial dynamic programming algorithm of the complexity $O(n^5)$ and a heuristic procedure of the complexity $O(n^3)$. This settles the complexity status of the problem under consideration which was open before and extends earlier work for the two-station single-track railway scheduling problem. We also present computational results of the comparison of both algorithms. For the 30,000 instances with up to 30 jobs considered, the average relative error of the heuristic is less than 1%. In our tests, the practical running time of the dynamic programming algorithm was even bounded by $O(n^4)$.

Keywords: scheduling; total completion time; job-shop

1. Introduction

We consider a two-machine job-shop scheduling problem. Each job $j \in N = \{1, 2, \dots, n\}$ consists of two operations, i.e., we have $n_j = 2$ according to [1]. The operation $O_{j,a}$ is processed on the machine M_a and its processing time is equal to a . The operation $O_{j,b}$ is processed on the machine M_b and its processing time is equal to b , where $a, b \in \mathbb{Z}^+$ and $a < b$. For simplicity of the subsequent consideration, we use both notations a and M_a , where a is a descriptor in M_a and a is the processing time of any job on this machine.

Let N_{ab} be the subset of jobs j , for which operation $O_{j,a}$ precedes operation $O_{j,b}$ and let N_{ba} be the subset of jobs j , for which operation $O_{j,b}$ precedes operation $O_{j,a}$. Moreover, denote $n_{ab} = |N_{ab}|$ and $n_{ba} = |N_{ba}|$. Thus, we have $n = n_{ab} + n_{ba}$. Please note that the parameter n_j is different from n_{ab} and n_{ba} . The parameter n_j is often used in publications on job-shop scheduling to denote the number of operations of job j , and we use n_{ab}, n_{ba} to denote the numbers of jobs with the two possible technological routes. A schedule Π is uniquely determined by two permutations π_{M_a} and π_{M_b} of the operations of the set $N_{ab} \cup N_{ba}$. Let $C_{j,x}(\Pi)$ be the completion time of operation $O_{j,x}$ and $S_{j,x}(\Pi) = C_{j,x}(\Pi) - x, x \in \{a, b\}$ be the starting time of the operation in the schedule Π .

For the two-machine job-shop scheduling problem of minimizing total completion time subject to given processing times, the objective is to find an optimal schedule Π^* that minimizes the total completion time, i.e.,

$$\sum_{j \in N_{ab}} C_{j,b} + \sum_{j \in N_{ba}} C_{j,a}. \quad (1)$$

We denote this problem by $J2|n_j = 2, p_{j1} = a, p_{j2} = b|\sum C_j$ according to the traditional three-field notation $\alpha|\beta|\gamma$ for scheduling problems proposed by Graham et al. [2], where α describes the machine environment, β gives the job characteristics and further constraints, and γ describes the objective function. Please note that, without loss of generality, we can restrict to the case $a < b$ since the case $a = b$ can be trivially solved in constant time.

Our motivation to deal with this problem with an open complexity status is as follows:

- it has a theoretical significance as a special case of the classical job-shop scheduling problem with two machines with another objective function than the makespan considered by Jackson in the well-known paper from 1956 [3];
- it has also practical significance as a particular sub-problem e.g., arising in railway scheduling.

Namely, the following single-track railway scheduling problem (STRSP) can be reduced to this problem. In the STRSP, there is a single track between the stations A and C and a middle station B between stations A and C . Trains go in both directions. Each of the sub-tracks AB and BC can process only one train at a time. At the station B , a train can pass other trains, and at all stations there are enough parallel tracks to deposit trains. A single-track network can be seen as a bottleneck portion for any type of railway network topology. Furthermore, almost all national railway networks have sections, where there is a single-track between some stations. For some countries (e.g., USA, Australia), a significant part of the network is single-track. For multi-track networks such a single-track segment can be considered as a bottleneck, in which the traffic capacity is restricted.

In this paper, we present a new polynomially solvable case for the two-machine job-shop problem with minimizing total completion time based on dynamic programming [4]. At the same time, this extends an existing polynomial algorithm for the two-station single-track railway scheduling problem from [5] to the case of three stations. In addition, we present a fast polynomial heuristic of lower complexity which is able to construct near-optimal solutions.

The rest of this paper is organized as follows. A brief literature review is given in Section 2. In Section 3, some properties of the problem are presented which are the base for the dynamic programming algorithm. Polynomial exact and heuristic solution procedures for this problem are presented in Section 4. Some results of numerical experiments are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. Literature Overview

The problem $J2|n_j = 2, p_{j1} = a, p_{j2} = b|C_{max}$ of minimizing the makespan (maximal completion time) can be solved in constant time by Jackson's algorithm [3]. In an optimal schedule, on the machine M_a , first all operations $O_{i,a}$, $i \in N_{ab}$, are processed and then all operations $O_{j,a}$, $j \in N_{ba}$. On the machine M_b , first all operations $O_{j,b}$, $j \in N_{ba}$, are processed and then all operations $O_{i,b}$, $i \in N_{ab}$. However, the problem $J2||C_{max}$ without the restriction to at most two operations per job and arbitrary processing times is already NP-hard [1].

Moreover, when minimizing total completion time, only very special unit-time problems can be polynomially solved (see e.g., [1]). Even the two-machine unit-time problems $J2|p_{jk} = 1, r_j \geq 0|\sum C_j$ with release dates r_j , $J2|p_{jk} = 1|\sum w_j C_j$ with job weights w_j or the three-machine problem $J3|p_{jk} = 1|\sum C_j$ are already NP-hard (see [1]). Two-machine job shop scheduling problems with unit processing times and $n_j > 2$ operations per job, where the even operations are processed on one machine and the odd operations on the other one are considered in [6,7]. The scheduling problem to minimize total completion time is considered in [8].

Some results on parallel machine and single machine scheduling problems with unit and equal processing times of the jobs are presented in [9,10]. Single machine problems are equivalent to the special case of a two-machine job shop scheduling problem with $n_j = 2, p_{j1} = a, p_{j2} = b$, where a is sufficiently small so that it can be disregarded. These problems without precedence relations are known to be polynomially solvable, except the problem $1|r_j \geq 0, p_j = p|\sum w_j T_j$ the complexity status

of which is open. An additional motivation of our research is the search for an NP-hard job scheduling problem with equal processing times which is most close to single machine job scheduling problems with equal processing times without precedence relations and preemptions.

As previously mentioned, the problem under consideration is closely related to a particular single-track railway scheduling problem. Often such problems are considered in the case of maintenance of one track of a double-track line. For example, the French railway company SNCF develops such models to produce a new transport schedule in the event of an incident on one of the double-track line sections [11]. The work on single-track railway scheduling problems (STRSP) goes back to the 1970s, with the initial publication [12]. A recent literature review on the single-track railway scheduling problem can be found, e.g., in [13]. A short survey on the STRSP with several stations, where trains are able to pass each other, is presented in [14]. In [5], a single-track railway scheduling problem with two stations and several segments of the track is considered. In [15], train scheduling problems are modeled as job-shop scheduling problems with blocking constraints. Four MIP formulations are developed for the problem of minimizing total tardiness, and a computational study is made on hard instances with up to 20 jobs (trains) and 11 machines (tracks or track sections). Blocking constraints make the job-shop scheduling problem very hard from a practical point of view. In [16], a complex neighborhood for the job-shop scheduling problem with blocking and total tardiness minimization has been developed and tested on benchmark instances from the literature. Further algorithms for general railway scheduling problems have been given for instance in [17–19] and for job-shop scheduling problems with blocking in [20,21]. The blocking job-shop with rail-bound transportation has also been discussed in [22]. Please note that for a small railway network with only a few stations and enough parallel tracks at each station, the blocking constraint can be skipped as in our three-station case.

In this paper, we deal with an exact dynamic programming approach. For some further recent general approaches for the solution of different types of single and multiple criteria scheduling problems, the interested reader is referred to [23–29] which highlight the importance of developing advanced scheduling approaches. This concerns both the identification of new polynomially solvable problems as well as new MILP models and metaheuristic or hybrid algorithms.

3. Properties of the Problem $J2|n_j = 2, p_{j1} = a, p_{j2} = b| \sum C_j$

In this section, we present and prove in Lemmas 1–3 some basic properties of the problem. While Lemma 1 characterizes the structure of partial solutions, Lemmas 2 and 3 are used in the proof of the subsequent Theorem 1 which is the foundation of the dynamic programming algorithm given in Section 4.

Without loss of optimality, we can restrict to schedules, where the operations $O_{j,a}$ are processed in the same order as the operations $O_{j,b}$, $j \in N_{ab}(N_{ba})$. Then we can schedule the jobs from each subset according to increasing numbers. To distinguish the jobs from the sets N_{ab} and N_{ba} , the jobs from the set N_{ba} are overlined, i.e., we have $N_{ab} = \{1, 2, \dots, n_{ab}\}$ and $N_{ba} = \{\bar{1}, \bar{2}, \dots, \bar{n}_{ba}\}$.

In an active schedule, a job cannot be started earlier without violating the feasibility. Without loss of optimality, we consider active schedules only.

It is obvious that there is only a single case when an idle time on the machine M_b arises. It can be immediately before time $C_{i,a} = S_{i,b}$, $i \in N_{ab}$, i.e., when for the job $i \in N_{ab}$, the completion time of the short operation (with processing time a) is equal to the starting time of the long one (with processing time b). The same holds for an idle time on the machine M_a . An idle time can be immediately before time $C_{\bar{j},b} = S_{\bar{j},a}$, $\bar{j} \in N_{ba}$.

Lemma 1. *In any active schedule, the starting times of the operations belong to the set*

$$\Theta = \{xa + yb | x, y \in Z, x, y < n_{ab} + n_{ba}\}. \tag{2}$$

Proof. We consider the possible starting time $S_{i,a}$ of operation $O_{i,a}$ in an active schedule. Let in the interval $[t_1, S_{i,a})$, x_1 operations be processed on the machine M_a without idle time and let there be an idle time immediately before t_1 . Then there is a job \bar{j}_1 for which $C_{\bar{j}_1,b} = S_{\bar{j}_1,a} = t_1$. Let in the interval $[t_2, C_{\bar{j}_1,b}]$, y_1 operations be processed on the machine M_b without an idle time and let there be an idle time immediately before t_2 . Then there is a job i_1 for which $C_{i_1,a} = S_{i_1,b} = t_2$. For an illustration, see Figure 1.

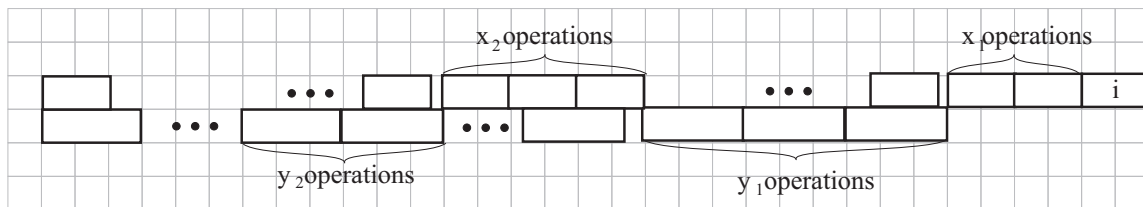


Figure 1. Illustration for the proof of Lemma 1.

By continuing this consideration, we have

$$S_{i,a} = (x_1 + x_2 + x_3 + \dots)a + (y_1 + y_2 + \dots)b. \tag{3}$$

□

Lemma 2. In any optimal schedule, we have $S_{1,a} = S_{\bar{1},b} = 0$, i.e., the starting times of the first operations of the first job from each subset are equal to 0.

Proof. It is obvious that in each active schedule $S_{1,a} = 0$ for $O_{1,a} \in N_{ab}$.

Next, we show that the lemma holds for the first job $\bar{1} \in N_{ba}$. Let in a schedule Π , this does not hold. The operation sequence for machine M_a is $(O_{i_1,a}, O_{i_2,a}, \pi_1, O_{\bar{1},a}, \pi_2)$, where in the partial sequence π_1 there are k_1 operations and $i_1, i_2 \in N_{ab}$. The operation sequence for machine M_b is $(\pi_3, O_{\bar{1},b}, \pi_4)$, where in the partial sequence π_3 there are $k_2 \geq 1$ operations (see Figure 2).

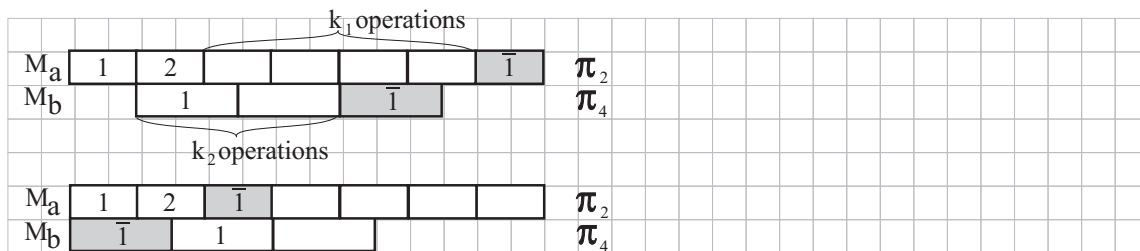


Figure 2. Illustration for the proof of Lemma 2.

Then for the schedule Π' with the operation sequence $(O_{i_1,a}, O_{i_2,a}, O_{\bar{1},b}, \pi_1, \pi_2)$ for machine M_a and the operation sequence $(O_{\bar{1},b}, \pi_3, \pi_4)$ for machine M_b , we increase the completion times for the operations $O_{i,b} \in \pi_3$ on the value $(b - a)$, and we decrease the completion time of the operation $O_{\bar{1},a}$ on the value

$$C_{\bar{1},a}(\Pi) - C_{\bar{1},a}(\Pi') = (\max\{a + (k_2 + 1)b, (2 + k_1)a\} + a) - 3a \geq (k_2 + 1)b - a. \tag{4}$$

Let $C_{i_2,a} = 2a > b$, then the completion times of the operations from π_2 and π_4 are not increased. Thus, the total completion time is decreased on a value greater than or equal to

$$(k_2 + 1)b - a - (b - a)k_2 \geq b. \tag{5}$$

An analogous proof can be presented for the case $2a \leq b$. The lemma is true for the first job from the set N_{ba} . □

Lemma 3. In any active schedule Π at each time t , where an operation $O_{i,a}, i \in N_{ab}$, is such that $t \leq C_{i,a}(\Pi)$, the number of short operations completed is greater than or equal to the number of long operations completed.

Proof. The proof is done by induction. We consider the completion times t of the first, \dots , $(k - 1)$ -th and k -th operations processed on the machine M_b . The lemma holds for t equal to the completion time of the first operation processed on the machine M_b , i.e., $t = C_{\bar{1},b} = b$.

Let the lemma hold for t equal to the completion time of the $(k - 1)$ -th operation processed on the machine M_b . Moreover, there are $l, l \geq k - 1$, operations completed on the machine M_a before t . Let $\tau, \tau \leq t$, be the completion time of the last operation completed on the machine M_a before time t .

Let $t', t' \geq t + b$, be the completion time of the k -th operation processed on the machine M_b . Then, in the interval $[\tau, t']$, at least one operation can be processed on the machine M_a and thus, at time t' , the number of short operations completed is greater than or equal to the number of long operations completed. \square

Theorem 1. In any optimal schedule, there is no idle time on the machine M_b before the last operation from the set N_{ba} , i.e., before the time $S_{\bar{n}_{ba},a}$.

Proof. The proof is done by induction. First, we show that the theorem holds for the first job from the set N_{ba} and then for the next jobs $\bar{j} \in N_{ba}$. If for an operation $O_{\bar{j},b}, \bar{j} \in N_{ba}$, in a schedule Π , there is an idle time before the time $S_{\bar{j},b}(\Pi)$ on the machine M_b , then we construct a modified schedule Π' , where the operations $O_{\bar{j},a}$ and $O_{\bar{j},b}$ are shifted to an earlier time so that the idle time is vanished and total completion time decreases.

If $a \leq \frac{1}{2}b$, then in any active schedule, there is no idle time on the machine M_b , since for any operation $O_{i,a}, i \in N_{ab}, i > 1$, we have $C_{i,a} < S_{i,b}$. Next, we consider the only remaining case $a > \frac{1}{2}b$.

According to Lemma 2, the theorem holds for the first job $\bar{1} \in N_{ba}$. Let the theorem hold for the job $\bar{j} - 1 \in N_{ba}$, and we consider the next job $\bar{j} \in N_{ba}$. Let there exist an idle time before the time $S_{\bar{j},a}$ in a schedule Π . We prove that this idle time is before the time $S_{\bar{j},b}$ as well. We do this by contradiction. Let there exist a schedule Π with $C_{\bar{j},b} < C_{i,a} = S_{i,b}$ and there is an idle time on the machine M_b in the interval $(t, S_{i,b})$. We have $C_{\bar{j},b} \leq t$ and $S_{\bar{j},a} \geq S_{i,b}$ (see Figure 3). In this case, at time t , the number $(i - 1 + \bar{j})$ of long operations completed is greater than the number $(i - 1 + \bar{j} - 1)$ of short operations completed which is a contradiction to Lemma 3. So, assume that we have a schedule Π , where the operation sequences are:

$$\begin{aligned} \text{for machine } M_a : & (\pi_1, O_{\bar{j}-1,a}, O_{i,a}, O_{i+1,a}, \pi_2, O_{\bar{j},a}, \pi_3), \\ \text{for machine } M_b : & (\pi_4, O_{i-1,b}, O_{i,b}, O_{i+1,b}, \pi_5, O_{\bar{j},b}, \pi_6), \end{aligned}$$

(see Figure 3). In the schedule Π , the last operation completed before the idle time is $O_{i-1,b} \in N_{ab}$ according to the assumption made above that there is no idle time before operation $O_{\bar{j}-1,a}$. Operation $O_{\bar{j}-1,a}$ is processed immediately before operation $O_{i,a}$ since $S_{i,a} > S_{i-1,b}$.

Assume that there are k_1 short operations processed between the operations $O_{i,a}$ and $O_{\bar{j},a}$ in Π and k_2 long operations processed between the times $S_{i,b}$ and $S_{\bar{j},b}$. It is easy to show that there is no idle time between the times $S_{i,b}$ and $S_{\bar{j},b}$.

Denote $\Delta_1 = S_{i-1,b} - S_{\bar{j}-1,a}$ and $\Delta_2 = C_{i-1,b} - S_{i,b}$ (see Figure 1). We have $2a = \Delta_1 + b + \Delta_2$. Then $a > \Delta_1 + \Delta_2$, otherwise

$$\Delta_1 + b + \Delta_2 \geq a + b,$$

i.e., $2a \geq a + b$ and $a \geq b$, which is false. As a consequence, we get

$$b > a > \Delta_1 + \Delta_2.$$

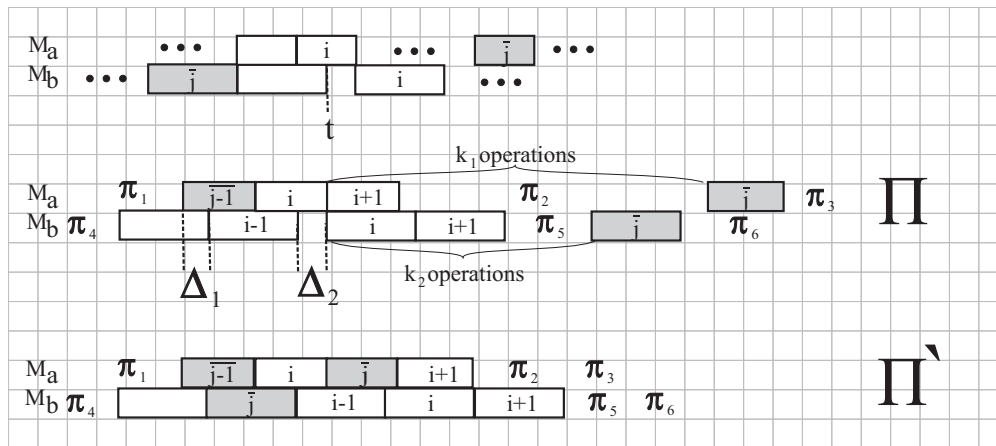


Figure 3. Illustration for the proof of Theorem 1.

Let us consider a schedule Π' , where the operation sequences are

$$\text{for machine } M_a : (\pi_1, O_{\bar{j}-1,a}, O_{i,a}, O_{\bar{j},a}, O_{i+1,a}, \pi_2, \pi_3),$$

$$\text{for machine } M_b : (\pi_4, O_{\bar{j},b}, O_{i-1,b}, O_{i,b}, O_{i+1,b}, \pi_5, \pi_6),$$

(see Figure 1).

If in the schedule Π' , there is an idle time Δ_3 between the times $C_{i,b}$ and $S_{i+1,b}$, then

$$\begin{aligned} \Delta_3 &= C_{i+1,a}(\Pi') - C_{i,b}(\Pi') \\ &= (C_{\bar{j}-1,a}(\Pi) + 4a) - (C_{\bar{j}-1,a}(\Pi) + \Delta_1 + 3b) \\ &= 4a - (\Delta_1 + 3b) \\ &= 2(b + \Delta_1 + \Delta_2) - (\Delta_1 + 3b) \\ &= \Delta_1 + 2\Delta_2 - b < \Delta_2. \end{aligned}$$

It is easy to show that there is no idle time on the machine M_b between the time $C_{i+1,b}$ and the first operation in π_5 in the schedule Π' . Then for all operations in the sequences π_3 and π_6 , the completion times are not increased.

Now, we increase the completion time:

$$\begin{aligned} &\text{on } b && \text{for operation } O_{i-1,b} \\ &\text{on } b - \Delta_2 && \text{for operation } O_{i,b} \\ &\text{on } b - \Delta_2 + \Delta_3 && \text{for the } k_2 - 1 \text{ operations } \{O_{i+1,b}\} \cup \pi_5 \end{aligned}$$

We decrease the completion time of the operation $O_{\bar{j},a}$ on the value

$$C_{\bar{j},a}(\Pi) - C_{\bar{j},a}(\Pi') = (C_{\bar{j}-1,a}(\Pi) + 2a + \max\{k_1 a, (k_2 + 1)b\} + a) - (C_{\bar{j}-1,a}(\Pi) + 3a) \geq (k_2 + 1)b.$$

Thus, we decreased the total completion time on a value greater than or equal to

$$(k_2 + 1)b - (b + b - \Delta_2 + (b - \Delta_2 + \Delta_3)(k_2 - 1)) = k_2 \Delta_2 - (k_2 - 1)\Delta_3 > 0.$$

So, the theorem holds for job \bar{j} . \square

4. Solution Algorithms for the Problems $J2|n_j = 2, p_{j1} = a, p_{j2} = b| \sum C_j$

In this section, we first present a fast polynomial heuristic and then a polynomial dynamic programming algorithm which is based on Theorem 1. The problem under consideration can be solved approximately by the following polynomial heuristic, which includes 3 major steps.

Heuristic H:

1. Construct a schedule Π according to Jackson’s algorithm.
2. Consider one by one the operations $O_{\bar{j},a}$ for $\bar{j} = \bar{1}, \bar{2}, \dots, \bar{n}_{ba}$. Shift the operation $O_{\bar{j},a}$ to the earliest position in the sequence, where the total completion time is not increased in comparison to the currently best schedule obtained.
3. Consider one by one the operations $O_{i,b}$ for $i = 1, 2, \dots, n_{ab}$. Shift the operation $O_{i,b}$ to the earliest position in the sequence, where the total completion time is not increased in comparison to the currently best schedule obtained.

Each of the steps 2 and 3 needs $O(n^3)$ operations since for each operation, we consider $O(n)$ positions and $O(n)$ operations are needed to compute the total completion time of the modified schedule. So, the running rime of the algorithm is $O(n^3)$.

Next, we present an idea of an exact dynamic programming algorithm (DP). In the first step of Algorithm DP, we construct an active schedule that contains only jobs from the set N_{ab} . Then, in each stage $\bar{j} = \bar{1}, \bar{2}, \dots, \bar{n}_{ba}$, we compute all possible states $t = (i_a, i_b, C_{\bar{j},b}, C_{\bar{j},a})$, where

- operation $O_{\bar{j},a}, \bar{j} \in N_{ba}$, is processed between the operations $O_{i_a,a}$ and $O_{i_a+1,a}, i_a, i_a + 1 \in N_{ab}$ and after all operations $O_{j',a}, j' < \bar{j}, j' \in N_{ba}$,
- operation $O_{\bar{j},b}, \bar{j} \in N_{ba}$, is processed between the operations $O_{i_b,b}$ and $O_{i_b+1,b}, i_b, i_b + 1 \in N_{ab}$ and after all operations $O_{j',b}, j' < \bar{j}, j' \in N_{ba}$.

For each state t , the total completion time TCT_t of the operations $O_{i,b}, i \in \{1, 2, \dots, i_b\}$, and the operations $O_{j',a}, j' \in \{\bar{1}, \bar{2}, \dots, \bar{j}\}$, is saved.

If for two states $t = (i_a, i_b, C_{\bar{j},b}, C_{\bar{j},a})$ and $\check{t} = (i_a, i_b, \check{C}_{\bar{j},b}, \check{C}_{\bar{j},a})$, we have

$$C_{\bar{j},b} \leq \check{C}_{\bar{j},b}, \quad C_{\bar{j},a} \leq \check{C}_{\bar{j},a} \quad \text{and} \quad TCT_t \leq TCT_{\check{t}},$$

then the state \check{t} can be excluded from the further considerations.

According to Theorem 1, we have $C_{\bar{j},b} = (i_b + \bar{j})b$. Thus, the state is uniquely defined by the vector $(i_a, i_b, C_{\bar{j},a})$. Please note that only states with $i_a \geq i_b$ are considered. The states obtained at stage j are used to compute the states in the next stage. After the last stage, for each state $t = (i_a, i_b, C_{\bar{j},a})$, we schedule the remaining operations $O_{i_a,a}, i = i_a + 1, \dots, n_{ab}$, and $O_{i_b,b}, i = i_b + 1, \dots, n_{ab}$, and add to the value TCT_t the value $\sum_{i=i_b+1}^{n_{ab}} C_{i,b}$. Then we choose the best solution.

The value $C_{\bar{j},a}$ and TCT_t can be computed in constant time. For that, in the previous stage $\bar{j} - \bar{1}$, for a state $\hat{t} = (\hat{i}_a, \hat{i}_b, \hat{C}_{\bar{j}-1,a})$, we saved the value $\hat{C}_{\hat{i}_b+1,a}$. Let us compute the state $t = (i_a, i_b, C_{\bar{j},a})$ from the state \hat{t} . For the state t , the values $C_{\hat{i}_a+1,a}$ and $C_{i_a,a}$ are computed in constant time. Then the value $\sum_{i=\hat{i}_b+1}^{i_b} C_{i,b}$ can be computed in constant time according to the values $\hat{C}_{\bar{j}-1,a}$ and $\hat{C}_{\hat{i}_b+1,a}$. There can be only an idle time on the machine M_b before operation $O_{\hat{i}_b+1,b}$ but such a state can be excluded from consideration according to Theorem 1. Then, according to $C_{i_b,b}$, the value $C_{\bar{j},b}$ can be computed in constant time. Finally, the value $C_{\bar{j},a}$ according to the times $C_{\bar{j},b}$ and $C_{i_a,a}$ and the value $C_{i_b+1,a}$ can be computed in constant time.

Theorem 2. The problem $J2|n_j = 2, p_{j1} = a, p_{j2} = b| \sum C_j$ can be solved by Algorithm DP in $O(n^5)$ time.

Proof. According to Lemma 1, there are no more than $O(n^2)$ possible values $C_{\bar{j},a}$. Then there are no more than $O(n^4)$ possible states $t = (i_a, i_b, C_{\bar{j},a})$ at the stage \bar{j} , and each state is computed in constant time. Since there are $O(n)$ stages, the running time of Algorithm DP is $O(n^5)$. \square

5. Computational Results

In this section, we present some results of a numerical experiment, where we investigate the relative error of the heuristic algorithm H and the number of states considered in Algorithm DP . We generated the instances as follows. For each $n \in \{5, 10, 15, 20, 25, 30\}$, we generated 5000 instances with $b \in [3, 50], a \in [1, b - 1]$, yielding in total 30000 instances. For each instance, both the exact and the heuristic algorithms were used.

In Table 1, we present the results for 30 randomly selected instances, namely five for each value $n \in \{5, 10, 15, 20, 25, 30\}$, where the main goal is to report the relative error of the heuristic presented in this paper. In columns 1–4, we present the job numbers n_{ab} and n_{ba} as well as the processing times a and b . In column 5, we give the optimal total completion time value $TCT-DP$ obtained by Algorithm DP . In column 6, we present the maximal number MN of states remaining in the list at a stage. In column 7, we present the number $DIF = n_{ab}n_{ba}(n_{ab} + n_{ba}) - MN$, which is given to show that the maximal number of states remaining in the list is less than n^3 . In column 8, we give the total completion time value $TCT-H$ obtained by the heuristic. Finally, column 9 displays the percentage deviation PD of the heuristic from the optimal function value.

Table 1. Detailed results with Algorithm DP and Heuristic H for 30 randomly selected instances.

n_{ab}	n_{ba}	a	b	$TCT-DP$	MN	DIF	$TCT-H$	PD
1	2	3	4	5	6	7	8	9
instance data				DP			H	
2	3	13	15	264	6	24	275	4.2
2	3	34	48	822	5	25	842	2.4
3	2	9	11	188	11	19	197	4.8
3	2	24	49	783	5	25	783	0.0
1	4	16	25	439	2	18	439	0.0
5	5	10	17	985	16	234	997	1.2
2	8	39	41	2567	11	149	2604	1.4
3	7	12	19	1129	7	203	1139	0.9
8	2	40	44	2624	75	85	2752	4.9
1	9	1	24	1329	2	88	1329	0.0
4	11	17	25	3187	16	644	3214	0.8
9	6	41	43	5499	222	588	5718	4.0
12	3	31	37	4553	155	385	4808	5.6
9	6	14	17	2132	134	676	2212	3.8
13	2	3	5	606	59	331	618	2.0
2	18	6	21	4518	3	717	4518	0.0
18	2	11	16	3383	156	564	3484	3.0
1	19	2	4	878	2	378	878	0.0
12	8	26	28	6140	491	1429	6352	3.5
3	17	10	27	5840	5	1015	5840	0.0
6	19	14	16	5466	138	2712	5526	1.1
21	4	29	40	13,123	490	1610	13,476	2.7
21	4	17	20	6581	597	1503	6848	4.1
18	7	22	25	8302	798	2352	8602	3.6
12	13	39	42	14,157	671	3229	14,553	2.8
5	25	38	47	22,805	59	3691	22,921	0.5
16	14	13	21	9947	219	6501	10,022	0.8
21	9	26	30	14,208	1097	4573	14,624	2.9
4	26	19	21	10,259	82	3038	10,310	0.5
10	20	7	12	5720	59	5941	5738	0.3

In Table 2, we present some results for all 30,000 considered instances. In column 1, we give the number n of jobs. In columns 2 and 3, we present the average values MN and DIF , respectively, for Algorithm DP . For Heuristic H , we present in column 4 the average values PD and column 5 displays the percentage of instances PO solved by the heuristic optimally. We also emphasize that the average relative error over all 30,000 instances is 0.85%.

Table 2. Average results with Algorithm *DP* and Heuristic *H* for the 30,000 instances.

<i>n</i>	Average <i>MN</i>	Average <i>DIF</i>	Average <i>PD</i>	<i>PO</i>
1	2	3	4	5
<i>DP</i>			<i>H</i>	
5	4.33	23.17	1.05	67.3
10	15.43	183.57	1.04	63.44
15	37.61	523.39	0.87	59.66
20	87.52	1158.48	0.85	58.14
25	156.83	2843.17	0.74	56.1
30	202.54	4834.46	0.53	58.12
Aver	84.04	1594.37	0.85	60.46

Moreover, we can state that the maximal relative error of Heuristic *H* among all 30,000 instances is 6.9% which has been obtained for an instance with $n_{ab} = 3, n_{ba} = 2, a = 17, b = 23$. For this instance, the optimal objective function value is 384 and the total completion time computed by the heuristic is 401. Moreover, the maximal number of states saved to the state list in a stage is 32,811 which has been obtained for an instance with $n_{ab} = 20, n_{ba} = 10, a = 47, b = 49$. In addition, if a state $\bar{t} = (\bar{i}_a, \bar{i}_b, \bar{C}_{j,a})$ has been written to the list and later a state $t = (i_a, i_b, C_{j,a})$ is computed, where $C_{j,a} \leq \bar{C}_{j,a}$ and $TCT_t \leq TCT_{\bar{t}}$, then the state \bar{t} is deleted from the list. The maximal number of states in the list left after considering all states is 1743. So, there is a large difference between the number of states considered and the number of states remaining in the list.

According to the previous results, we can also present the following conjecture.

Conjecture 1. *There are only $O(n^3)$ states that have to be considered at each stage.*

As a consequence of the above conjecture, the running time of an advanced DP algorithm could be reduced to $O(n^4)$.

6. Concluding Remarks

In this paper, some properties of the scheduling problem $J2|n_j = 2, p_{j1} = a, p_{j2} = b|\sum C_j$ were considered which arises for instance in a single-track railway scheduling problem with three stations. A polynomial time solution algorithm of the complexity $O(n^5)$ and a heuristic algorithm of the complexity $O(n^3)$ were presented. In the numerical experiments with the 30000 instances, the running time of the dynamic programming algorithm was even bounded by the order $O(n^4)$. Moreover, in our tests, the average relative error of the polynomial heuristic was only 0.85%.

The two-machine job-shop problem of minimizing the makespan was considered in the pioneering work by Jackson. This result is now considered as a classical one in the scheduling theory. An interesting open question is whether there exists an NP-hard job-shop scheduling problem with equal processing times on each machine and other objective functions without precedence relations and preemptions, or whether such problems are also polynomially solvable.

Author Contributions: Investigation, E.G. and F.W.; software, E.G.; writing—original draft preparation, E.G. and F.W.

Funding: This work was funded by DAAD (Deutscher Akademischer Austauschdienst): 91695276 and by RFBR (Russian Foundation for Basic Research): 18-07-00656

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brucker, P. *Scheduling Algorithms*, 5th ed.; Springer: Berlin, Germany, 2007.
2. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Optimization and Approximation in Deterministic Machine Scheduling: A Survey. *Ann. Discr. Math.* **1979**, *5*, 287–326.
3. Jackson, J.R. An Extension of Johnson's Results on Job Lot Scheduling. *Naval Res. Logist.* **1956**, *3*, 201–203. [[CrossRef](#)]
4. Bellman, R.E. *Dynamic Programming*; Princeton University Press: Princeton, NJ, USA, 1954.
5. Gafarov, E.R.; Dolgui, A.; Lazarev, A.A. Two-Station Single-Track Railway Scheduling Problem With Trains of Equal Speed. *Comput. Ind. Eng.* **2015**, *85*, 260–267. [[CrossRef](#)]
6. Timkovsky, V. On the Complexity of Scheduling an Arbitrary System. *Sov. J. Comput. Syst. Sci.* **1985**, *23*, 46–52.
7. Timkovsky, V. Is a Unit-Time Job Shop not Easier than Identical Parallel Machines? *Discr. Appl. Math.* **1998**, *85*, 149–162. [[CrossRef](#)]
8. Kubiak, W.; Timkovsky, V. Total Completion Time Minimization in Two-Machine Job Shops with Unit-Time Operations. *Eur. J. Oper. Res.* **1996**, *94*, 310–320. [[CrossRef](#)]
9. Baptiste, P. Scheduling Equal-Length Jobs on Identical Parallel Machines. *Discr. Appl. Math.* **2000**, *103*, 21–32. [[CrossRef](#)]
10. Kravchenko, S.A.; Werner, F. Parallel Machine Problems with Equal Processing Times: A Survey. *J. Sched.* **2011**, *14*, 435–444. [[CrossRef](#)]
11. Sourd, F. A New Tool for Reducing Delays, *Avancees*, SNCF, 1 October 2009. Available online: <http://www.avancees.eu/01/index.htm> (accessed on 30 November 2018).
12. Szpigel, B. Train Scheduling on a Single Track Railway. In *Proceedings of the IFORS Conference on Operational Research '72*, Dublin, Ireland, 21–25 August 1972; pp. 343–352.
13. de Oliveira, E.S. Solving Single Track Railway Scheduling Problem Using Constraint Programming. Ph.D. Thesis, School of Computing, The University of Leeds, Leeds, UK, 2001.
14. Sotskov, Y.N.; Gholami, O. Shifting Bottleneck Algorithm for Train Scheduling on a Single-Track Railway. In *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing*, Bucharest, Romania, 23–25 May 2012; Borangiu, T., Dumitrache, I., Dolgui, A., Filip, F., Eds.; Elsevier Science: Amsterdam, The Netherlands, 2012; pp. 87–92, ISSN 1474-6670.
15. Lange, J.; Werner, F. Approaches to Modeling Train Scheduling Problems as Job Shops with Blocking Constraints. *J. Sched.* **2018**, *21*, 191–207. [[CrossRef](#)]
16. Lange, J.; Werner, F. A Permutation-Based Neighborhood for the Blocking Job-Shop Problem with Total Tardiness Minimization. In *Operations Research Proceedings*; Kliewer, N.; Ehmke, J.F.; Borndörfer, R. (eds.); Springer: Cham, Switzerland, 2018; pp. 581–586.
17. Burdett, R.L.; Kozan, E. A Disjunctive Graph Model and Framework for Constructing New Train Schedules. *Eur. J. Oper. Res.* **2010**, *200*, 85–98. [[CrossRef](#)]
18. D'Ariona, A.; Paciarelli, D.; Pranzo, M. A Branch and Bound Algorithm for Scheduling Trains in a Railway Network. *Eur. J. Oper. Res.* **2007**, *183*, 643–657. [[CrossRef](#)]
19. Zhou, X.; Zhong, M. Single-Track Train Timetabling with Guaranteed Optimality: Branch-and-Bound Algorithms with Enhanced Lower Bounds. *Transp. Res. Part B Methodol.* **2007**, *41*, 320–341. [[CrossRef](#)]
20. Liu, S.Q.; Kozan, E. Scheduling Trains as a Blocking Parallel-Machine Job-Shop Scheduling Model. *Comp. Oper. Res.* **2009**, *36*, 2840–2852. [[CrossRef](#)]
21. Liu, S.Q.; Kozan, E. Scheduling Trains with Priorities. *Transp. Sci.* **2011**, *45*, 175–198. [[CrossRef](#)]
22. Bürgy, R.; Gröflin, H. The Blocking Job-Shop with Rail-Bound Transportation. *J. Comb. Optim.* **2014**, *31*, 152–181. [[CrossRef](#)]
23. Canales-Bustos, L.; Santibanez-Gonzalez, E.; Candia-Vejar, A. A Multi-objective Optimization Model for the Design of an Effective Decarbonized Supply Chain in Mining. *Int. J. Prod. Econ.* **2017**, *193*, 449–464. [[CrossRef](#)]
24. Dulebenets, M.A. A Comprehensive Evaluation of Weak and Strong Mutation Mechanisms in Evolutionary Algorithms for Truck Scheduling at Cross-Docking Terminals. *IEEE Access* **2018**, *6*, 65635–65650. [[CrossRef](#)]
25. Peres, I.T.; Repolho, H.M.; Martinelli, R.; Monteiro, N.J. Optimization in Inventory-routing Problem with Planned Transshipment: A Case Study in the Retail Industry. *Int. J. Prod. Econ.* **2017**, *193*, 748–756. [[CrossRef](#)]

26. Dulebenets, M.A. Application of Evolutionary Computation for Berth Scheduling at Marine Container Terminals: Parameter Tuning versus Parameter Control. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 25–37. [[CrossRef](#)]
27. Fonseca, G.B.; Nogueira, T.H.; Ravetti, M.G. A hybrid Lagrangian metaheuristic for the cross-docking flow shop scheduling problem. *Eur. J. Oper. Res.* **2019**, *275*, 139–154. [[CrossRef](#)]
28. Dulebenets, M.A. A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping. *Int. J. Prod. Econ.* **2018**, *196*, 293–318. [[CrossRef](#)]
29. Zulj, I.; Kramer, S.; Schneider, M. A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *Eur. J. Oper. Res.* **2018**, *264*, 653–664. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).