Chapter 2
Makespan Minimization for Two-Machine Open Shops

2.1 Introduction

The two-machine open shop makespan minimization problem, $O2||C_{\max}$, is one of the most gracious scheduling problems. It has been studied in the seminal papers on open shop scheduling by Gonzalez and Sahni [11] and Pinedo and Schrage [16]. Detailed presentations of linear-time algorithms for the problem given in these two papers can be found in books by Pinedo [17], Błażewicz et al. [2], and Brucker [4], and in a book chapter by Gonzalez [10].

The two main observations that follow from those algorithms are that the problems $O2||C_{\max}$ and $O2|\text{pmtn}|C_{\max}$ have the same value of minimum makespan for any instance and that the value equals

$$C_{\max} = \max\{L, P\}. \tag{2.1}$$

That is, the minimum makespan equals either maximum machine workload or the length of the longest job, whichever is greater. This holds regardless of preemptions being allowed or not. de Werra [6] further observes that the general n-job two-machine open shop problem reduces to just 3-job problem in linear time. This leads to another linear-time algorithm for the problem. The algorithm will be presented in Sect. 2.2. Those features make the two-machine open shop makespan minimization problem quite unique in scheduling theory.

The importance of the problem has recently been further underlined by the operation of non-adjacent vertex cloning by which some real-life networks may evolve, in particular non-adjacent vertex cloning in wireless networks; see Chap. 6. Through such cloning a pre-existing vertex of degree 2 is cloned, and the clone becomes adjacent to both neighbors of the pre-existing vertex. The non-adjacent vertex cloning is shown in Fig. 2.1 where vertex v in Fig. 2.1a is cloned in vertices v_1, v_2, v_3, v_4, and v_5 in Fig. 2.1b.

W. Kubiak, *A Book of Open Shop Scheduling*, International Series in Operations Research & Management Science 325,
https://doi.org/10.1007/978-3-030-91025-9_2

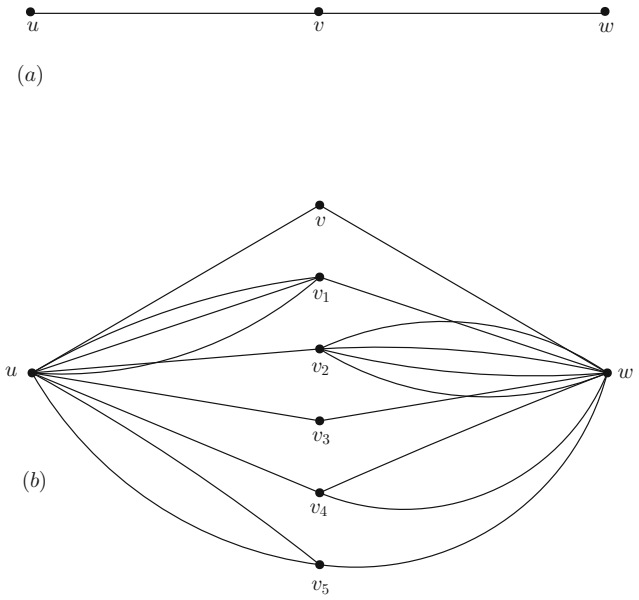

Fig. 2.1 (**a**) vertex v with $\deg(v) = 2$; (**b**) clones v_1, v_2, v_3, v_4, and v_5

The clones in Fig. 2.1b make up the following two-machine open shop with $n = 6$ jobs

$$\mathbb{P} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 4 \\ 1 & 1 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}.$$

It is to be expected that some of those desirable features of $O2||C_{\max}$ problem disappear when we extend it to include additional characteristics of operations. The main challenge of the extension is to add the characteristics so that they stay relevant enough for the open shop scheduling yet they preserve some key features of $O2||C_{\max}$ at the same time. Section 2.3 presents an extension of the problem $O2||C_{\max}$. The extension requires operations to be allocated a certain number of units of an additional renewable resource, besides a machine, in order to be processed. All the units are released once the operation is completed in order to be used by other operations. This generalized problem can be solved in $O(n^3)$ time, which is shown in Sect. 2.5. The main observation is that the generalization has the same value of minimum makespan regardless of whether preemptions are being allowed or not. This preserves a key feature of the optimal solutions to $O2||C_{\max}$.

2.2 A Linear-Time Algorithm for Two-Machine Open Shop

In this section we show that the problem with n jobs reduces to the problem with three jobs; de Werra [6]. The three jobs represent a partition of the set $\mathcal{J}$ of n jobs into three disjoint subsets. Thus once the optimal schedule for the three jobs has been found, the partition can be used to find the optimal schedule for the original n-job instance. We now present the details of this algorithm. Take any order of jobs $1, \ldots, n$, let

$$\alpha = \max\{L_1 = \sum_{j=1}^{n} p_{j,1}, L_2 = \sum_{j=1}^{n} p_{j,2}, \max_j\{P_j = p_{j,1} + p_{j,2}\}\}. \tag{2.2}$$

Determine the smallest i such that

$$\sum_{j=1}^{i}(p_{j,1} + p_{j,2}) \geq \alpha. \tag{2.3}$$

Since $\alpha \leq \sum_{j=1}^{n}(p_{j,1}+p_{j,2})$, such an i exists. Denote it by i^*. Let $A = \{1, \ldots, i^*-1\}$, $B = \{i^* + 1, \ldots, n\}$, and $C = \{i^*\}$. Define three jobs: A with processing time $p_{A,1} = \sum_{j=1}^{i^*-1} p_{j,1}$ and $p_{A,2} = \sum_{j=1}^{i^*-1} p_{j,2}$ on M_1 and M_2, respectively, B with processing time $p_{B,1} = \sum_{j=i^*+1}^{n} p_{j,1}$ and $p_{A,2} = \sum_{j=i^*+1}^{n} p_{j,2}$ on M_1 and M_2, respectively, and C with processing times $p_{C,1} = p_{i^*,1}$ and $p_{C,2} = p_{i^*,2}$ on M_1 and M_2, respectively. We use the same notation for the sets as for their corresponding jobs, but this should not cause any confusion. We show that a schedule that minimizes makespan for the three jobs can be easily converted into a schedule that minimizes makespan for the original n-job instance. We first show that

$$\alpha = \max\{p_{A,1} + p_{B,1} + p_{C,1}, p_{A,2} + p_{B,2} + p_{C,2}, p_{C,1} + p_{C,2}\}. \tag{2.4}$$

We observe that $L_1 = p_{A,1} + p_{B,1} + p_{C,1}$, and $L_2 = p_{A,2} + p_{B,2} + p_{C,2}$. Thus (2.4) holds, for $\max\{L_1, L_2\} \geq \max_j\{P_j\}$. Suppose that $\max\{L_1, L_2\} < P_{j^*} = \alpha$ for some j^*. If $j^* = i^*$, then (2.4) holds. Otherwise, $j^* > i^*$, which leads to contradiction because by (2.3) it implies

$$2\alpha \leq \sum_{j=1}^{i^*}(p_{j,1} + p_{j,2}) + (p_{j^*1} + p_{j^*2}) \leq L_1 + L_2 < 2\alpha.$$

Thus $i^* = j^*$ and (2.4) holds. Let us now find an optimal schedule for the jobs A, B, and C. For $\max\{L_1, L_2\} \leq p_{C,1} + p_{C,2} = \alpha$, the schedule in Fig. 2.2 is optimal.

Now assume that $\alpha = \max\{L_1, L_2\} > p_{C,1} + p_{C,2}$. For $L_1 \geq L_2$, one of the schedules in Figs. 2.3, 2.4, and 2.5 is optimal if the following condition is met:

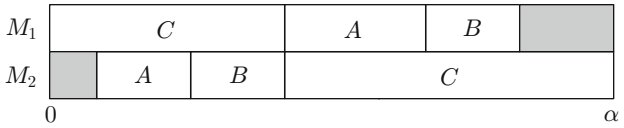

Fig. 2.2 An optimal schedule for $\max\{L_1, L_2\} \leq p_{C,1} + p_{C,2} = \alpha$

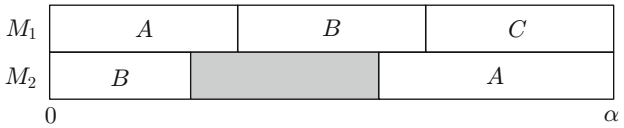

Fig. 2.3 An optimal schedule for $p_{A,1} \geq p_{B,2}$ and $p_{C,1} \leq p_{A,2}$. The job C can be scheduled anywhere in the shaded interval on M_2

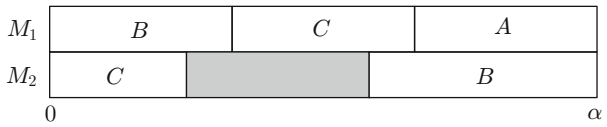

Fig. 2.4 An optimal schedule for $p_{B,1} \geq p_{C,2}$ and $p_{A,1} \leq p_{B,2}$. The job A can be scheduled anywhere in the shaded interval on M_2

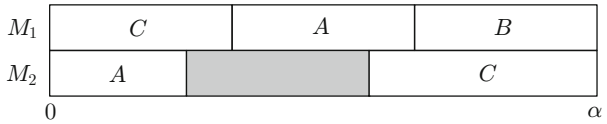

Fig. 2.5 An optimal schedule for $p_{C,1} \geq p_{A,2}$ and $p_{B,1} \leq p_{C,2}$. The job B can be scheduled anywhere in the shaded interval on M_2

($p_{A,1} \geq p_{B,2}$ and $p_{C,1} \leq p_{A,2}$) or ($p_{B,1} \geq p_{C,2}$ and $p_{A,1} \leq p_{B,2}$) or ($p_{C,1} \geq p_{A,2}$ and $p_{B,1} \leq p_{C,2}$). If this condition is not met, i.e., the following condition is met: ($p_{A,1} < p_{B,2}$ or $p_{C,1} > p_{A,2}$) and ($p_{B,1} < p_{C,2}$ or $p_{A,1} > p_{B,2}$) and ($p_{C,1} < p_{A,2}$ or $p_{B,1} > p_{C,2}$), then ($p_{A,1} < p_{B,2}$ and $p_{C,1} < p_{A,2}$ and $p_{B,1} < p_{C,2}$), which contradicts the assumption $L_1 \geq L_2$, or ($p_{C,1} > p_{A,2}$ and $p_{B,1} > p_{C,2}$ and $p_{A,1} > p_{B,2}$) in which case the optimal schedule is shown in Fig. 2.6. For $L_2 \geq L_1$, optimal schedules are obtained by the symmetry between M_1 and M_2.

To illustrate the algorithm run, let us consider the instance $\mathbb{Q}$ below:

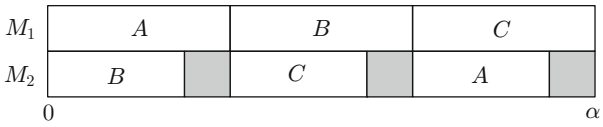

Fig. 2.6 An optimal schedule for $p_{C1} > p_{A2}$ and $p_{B1} > p_{C2}$ and $p_{A1} > p_{B2}$

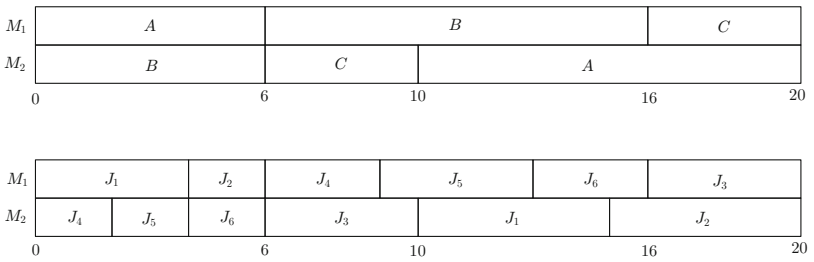

Fig. 2.7 An optimal schedule for the instance $\mathbb{Q}$. The schedule is obtained from the partition into jobs A, B, and C

$$\mathbb{Q} = \begin{bmatrix} 4 & 5 \\ 2 & 5 \\ 4 & 4 \\ 3 & 2 \\ 4 & 2 \\ 3 & 2 \end{bmatrix}.$$

We have $\alpha = 20$ for $\mathbb{Q}$. For the order $J_1, \ldots, J_6$ of jobs, we have $A = \{J_1, J_2\}$, $B = \{J_4, J_5, J_6\}$, and $C = \{J_3\}$. The processing times of jobs A, B, and C are equal: $p_{A,1} = 6$, $p_{A,2} = 10$, $p_{B,1} = 10$, $p_{B,2} = 6$, and $p_{C,1} = p_{C,2} = 4$. Thus the schedule in Fig. 2.3 is optimal for the three jobs A, B, and C; see Fig. 2.7. Observe that a reduction to two jobs is also possible for this instance. Take $D = \{J_1, J_5, J_6\}$ and $E = \{J_2, J_3, J_4\}$ for which the processing times are equal: $p_{D,1} = 11$, $p_{D,2} = 9$, $p_{E,1} = 9$, and $p_{E,2} = 11$. The optimal schedule is shown in Fig. 2.8.

A natural question arises whether the number of jobs in the reduction can always be reduced to two. The answer unfortunately is negative since there may be instances of n jobs such that $\sum_{i=1}^{n}(p_{i1} + p_{i2}) = 2\alpha$ and such that for any subset A of the jobs either $\sum_{i\in A}(p_{i,1}+p_{i,2}) < \alpha$ or $\sum_{i\in A}(p_{i,1}+p_{i,2}) > \alpha$. Thus either $\sum_{i\in\mathcal{J}\setminus A}(p_{i,1}+p_{i,2}) > \alpha$ or $\sum_{i\in A}(p_{i,1} + p_{i,2}) > \alpha$ for any subset A of $\mathcal{J}$. Therefore either the job $\mathcal{J} \setminus A$ or job A would be too long to guarantee makespan α for the two-job instance. However α is an optimal makespan for $\mathcal{J}$. This proves that the reduction to a two-job instance is not always possible. The problem to decide whether there is a reduction to a two-job instance or not is shown NP-complete by Gribkovskaia et al. [13]. Soper [20] uses similar ideas in yet another linear-time algorithm for the $O2||C_{\max}$.

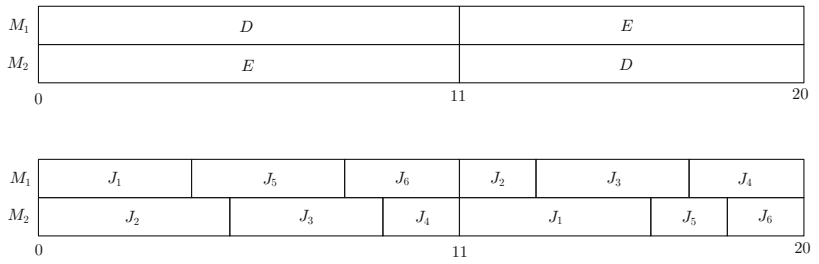

Fig. 2.8 An optimal schedule for the instance $\mathbb{Q}$. The schedule is obtained from the partition into jobs D and E

Van den Akker et al. [24], see also Shakhlevich and Strusevich [19], consider the two-machine open shop problem with bounds C_1 and C_2 on the completion time of machine M_1 and M_2, respectively. They show necessary and sufficient conditions for a feasible schedule that meets the bounds to exist. The conditions can be checked in linear time, and a feasible schedule, if any, can be computed in linear time.

2.3 Open Shop with Additional Renewable Resources

We now define the two-machine open shop scheduling problems with additional renewable resources we alluded to in the introduction. There are two machines M_1 and M_2 and l renewable resource types $R_1, \ldots, R_l$; $0 \leq s_\ell$ units of resource type R_ℓ are available at any time. The number s_ℓ is called resource capacity of resource type R_ℓ. Operation $O_{i,h}$ needs machine M_h and $r_\ell(O_{i,h}) \leq s_\ell$ units of resource type R_ℓ at any time during its execution. The number $r_\ell(O_{i,h})$ is called resource requirement of operation $O_{i,h}$. In a feasible schedule each job is processed by at most one machine at a time, each machine processes at most one operation at a time, and the total number of resource requirements of operations processed simultaneously does not exceed resource capacity for any resource type at any time. Błażewicz et al. [3] propose a three-dot notation, res . . . , to extend the notation of Graham et al. [12] to include scheduling problems with additional renewable resources. The first dot represents an arbitrary number of resource types, the second dot arbitrary resource capacities, and the third dot arbitrary resource requirements. The arbitrary here means that those numbers are part of the problem input. By replacing any of the dots by a positive integer, we make the corresponding part of the input constant for all instances. For example, the notation res1.. denotes all instances with a single additional resource type, and the other two dots mean that the capacity of that additional resource type is a part of the input and so are the resource requirements of operations. In Sect. 2.4 we consider the two-machine problem $O2|\text{res} \ldots, \text{pmtn}|C_{\max}$ with preemptions where the number of additional

resources, their capacities, and operation's resource requirements are all part of the input. Section 2.5 considers the problem $O2|\text{res}1..|C_{\max}$ with a single resource type.

The resource requirements of each operation O are represented by a vector $[r_1(O), \ldots, r_l(O)]$, where $0 \leq r_\ell(O) \leq s_\ell$ for $\ell = 1, \ldots, l$, and the resource capacities of resource types $R_1, \ldots, R_l$ by a vector $[s_1, \ldots, s_l]$. Though we will use the vector representation of resource requirements and capacities to express resource constraints in the following sections, it is worth mentioning that other equivalent representations, e.g., conflict graphs or agreement graphs, are possible for two-machine open shops in particular to express those constraints. Namely, the resource requirement vectors define an operation conflict graph $G = (V, E)$ for two-machine open shops with additional resources where V is the set of all operations, and $(O_{i,h}, O_{j,k}) \in E$ if and only if $i = j$ or there is ℓ such that $r_\ell(O_{i,h}) + r_\ell(O_{j,k}) > s_\ell$. On the other hand, for an operation conflict graph $G = (V, E)$ for operations in V there are $|V|$ resource types each with capacity 1. The resource type R_v corresponds to the vertex v of G. The vertex v has a 0–1 vector of resource requirements where the resource requirements are set to 1 for the resource types corresponding to the vertices in $N(v) \cup \{v\}$, where $N(v)$ is the neighborhood of v, and they are set to 0 for the resources corresponding to the vertices in $V \setminus (N(v) \cup \{v\})$. Thus operations u and v can be processed simultaneously if and only if they are not neighbors (they are not adjacent) in G. If they were, u would request one unit of resource R_u and one unit of resource R_v, and v would request one unit of resource R_u and one unit of resource R_v. Thus the total request for both R_u and R_v would equal 2, which exceeds resource capacity 1. The two representations of resource constraints are equivalent. Tellache and Boudhar [22] consider a similar model of constraints. They study a job conflict graph $G = (V, E)$ (rather than the operation conflict graph), where V is the set of jobs and $(J_i, J_j) \in E$ if and only if jobs J_i and J_i cannot be processed simultaneously. For a job conflict graph $G = (V, E)$, Tellache et al. [23] define a job agreement graph that is simply $\bar{G} = (V, \bar{E})$. Therefore clearly the translations between different representations of the resource constraints can easily be done in polynomial time, which allows for translations between the complexity results as well.

2.4 A Network Flow Algorithm for $O2|\text{res}\ldots,\text{pmtn}|C_{\max}$

Following Jurisch and Kubiak [14], we show a polynomial-time algorithm for the $O2|\text{res}\ldots,\text{pmtn}|C_{\max}$ problem in this section. For a given instance of the problem, we define a network $G = (V, A)$ as follows:

- The set of vertices, V, consists of a source s, vertices representing the operations $O_{1,2}, \ldots, O_{n,2}$ on machine M_2, vertices representing the operations $O_{1,1}, \ldots, O_{n,1}$ on machine M_1, and sink t.
- The set of directed arcs with capacities, A, consists of arcs (s, O_{i2}) with capacities $p_{i,2}$, $i = 1, \ldots, n$, arcs $(O_{i,2}, O_{j,1})$, $i, j = 1, \ldots, n$, with unlimited

capacity ∞ if $O_{i,2}$ and $O_{i,1}$ can be processed in parallel, i.e., if $i \neq j$ and $r_\ell(O_{i,2}) + r_\ell(O_{j,1} \leq s_\ell$ for all $\ell = 1, \ldots, l$, and arcs $(O_{j,1}, t)$ with capacities $p_{j,1}$, $j = 1, \ldots, n$.

Let $f(i, j)$ be the flow through $(i, j) \in A$ in a feasible solution to the max-flow problem defined by G. We define a feasible preemptive schedule to the corresponding open shop problem as follows:

- Schedule the operations $O_{1,1}, \ldots, O_{n,1}$ on machine M_1 in any order.
- Schedule $f(O_{i,2}, O_{j,1})$ time units of operation O_{i2} in parallel with $O_{j,1}$, $i, j = 1, \ldots, n$.
- If there are time units of an operation $O_{i,2}$ left, i.e., if $\sum_{j=1}^{n} f(O_{i,2}, O_{j1}) < p_{i2}$, then schedule $p_{i,2} - \sum_{j=1}^{n} f(O_{i,2}, O_{j,1})$ time units of $O_{i,2}$ on M_2 after the completion of the last M_1 operation.

This procedure converts any feasible solution to the network max-flow problem G into a feasible solution to $O2|\text{res} \ldots, \text{pmtn}|C_{\max}$ with the makespan

$$C_{\max} = \sum_{i=1}^{n} p_{i,1} + \sum_{i=1}^{n} [p_{i,2} - \sum_{j=1}^{n} f(O_{i,2}, O_{j,1})]. \tag{2.5}$$

Thus when $\sum_{i=1}^{n} \sum_{j=1}^{n} f(O_{i,2}, O_{j,1})$ is maximized in G, $C_{\max}$ of the open shop schedule is minimized.

Lemma 2.1 *An optimal schedule for $O2|res \ldots, pmtn|C_{\max}$ can be found in $O(n^3)$ time. The number of preemptions in the schedule does not exceed n^2.*

2.5 An Algorithm for $O2|\text{res } 1 \ldots|C_{\max}$

We now focus on the non-preemptive case of the problem with a single resource, $O2|\text{res } 1 \ldots|C_{\max}$. We show that any preemptive schedule obtained in Sect. 2.4 can be turned into a non-preemptive schedule with the same makespan, and the conversion can be done in polynomial time. To streamline the presentation of the conversion and the algorithm for the non-preemptive case, let us simplify the notation introduced in Sect. 2.3 and make simple observations about the preemptive schedules first. For simplicity, we denote the resource by R and assume that s units of R are available at any time. Operation $O_{i,h}$, $i = 1, \ldots, n$; $h = 1, 2$, requires $r(O_{i,h})$ units of R all the time during it execution. Without loss of generality, we let $r(O_{1,1}) \geq r(O_{2,1}) \geq \cdots \geq r(O_{n,1})$. We shall solve this two-machine open shop problem with a single resource, i.e., $O2|\text{res } 1 \ldots|C_{\max}$, in two steps:

Step 1: Solve $O2|\text{res } 1 \ldots, \text{pmtn}|C_{\max}$ with a max-flow algorithm (Lemma 2.1) to obtain schedule S.

Step 2: Convert the resulting schedule S into a schedule of $O2|\text{res } 1\,..|C_{\max}$ with the same makespan.

To simplify the presentation of Step 2, we make two assumptions about schedule S. First, without loss of generality, we may assume that neither machine is idle in S. We can easily meet this condition by adding dummy operations on either machine, if necessary. Second, again without loss of generality, we may assume that S meets the following conditions:

- No operation on M_1 is preempted.

(2.6)

- The operations on M_1 are scheduled in descending order of their resource requirements.

(2.7)

Thus, schedule S determines time slots and their orders. The time slots correspond to time interval occupied by exactly one operation on M_1. By the ith time slot, or simply slot I_i of S, we mean the time interval $[\sum_{j=1}^{i-1} p_{j,1}, \sum_{j=1}^{i} p_{j,1}]$. The following theorem gives the details of Step 2.

Theorem 2.1 *Any feasible schedule for $O2|res\ 1\,.., pmtn|C_{\max}$ can be converted into a feasible schedule for $O2|res\ 1\,..|C_{\max}$ with the same makespan.*

Proof The proof consists of Lemmas 2.2 and 2.3 and a recursive procedure Process-Time-Slot. Lemma 2.2 shows how to reduce the number of preempted M_2-operations scheduled both in slot I_i and in some slots I_j, $j > i$ to two. It also gives a good characterization of the case with exactly two such operations in I_i. The lemma is as follows. □

Lemma 2.2 *Let S be a schedule for which both (2.6) and (2.7) hold. The S can be converted into a schedule S' for which both (2.6) and (2.7) still hold; moreover,*

- $C_{\max}(S') = C_{\max}(S)$.

- *For any i, there are at most two M_2-operations scheduled in slot I_i and in some slots I_j with $j > i$.*

(2.8)

- *If there are exactly two operations $O_{a,2}$, $O_{b,2}$, $a < b$, scheduled both in slot I_i and in some slots I_j with $j > i$, then $b > i + 1$ and $O_{b,2}$ is continued in slots $I_{i+1}, \ldots, I_{b-1}$ only, and $O_{b,2}$ is the only operation scheduled on M_2 in these slots.*

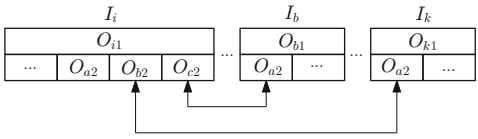

Fig. 2.9 Proof of Lemma 2.2: reduction of the number of preempted M_2-operations in I_i

(2.9)

Proof The proof is by induction. Let S be a schedule that meets conditions (2.6) and (2.7), and let i be the smallest index such that (2.8) or (2.9) is not met for slot I_i. We construct a schedule S' that meets (2.6)–(2.9) for I_i without making any change to I_j with $j < i$. First, assume that there are at least three M_2-operations $O_{a,2}$, $O_{b,2}$, and $O_{c,2}$ scheduled both in slot I_i and in some slots I_j with $j > i$. We show that it is then possible to exchange parts of $O_{b,2}$ and $O_{c,2}$ from I_i with parts of $O_{a,2}$ from slots I_j, $j > i$ until either:

(1) $O_{a,2}$ is no longer in slots I_j with $j > i$; or
(2) $O_{b,2}$ or $O_{c,2}$ is no longer in I_i.

In either case, the number of M_2-operations that occur both in I_i and in I_j with $j > i$ decreases by at least 1. Assume that a part of $O_{a,2}$ is scheduled in slot I_k with $k > i$. Denote by:

- q_b the length of the piece of $O_{b,2}$ in I_i,
- q_c the length of the piece of $O_{c,2}$ in I_i, and
- q_a the length of the piece of $O_{a,2}$ in I_k.

We proceed as follows:

- If $k = b$, then exchange $\min\{q_c, q_a\}$ time units of $O_{c,2}$ from I_i with the same number of time units of $O_{a,2}$ in I_k. If $q_a \geq q_c$, then (2) is met.
- If $k \neq b$, then exchange $\min\{q_b, q_a\}$ time units of $O_{b,2}$ from I_i with the same number of time units of $O_{a,2}$ in I_k. If $q_a \geq q_b$, then (2) is met.

This procedure is shown in Fig. 2.9. Note that the exchanges are feasible since we have $r(O_{1,1}) \geq \ldots \geq r(O_{n,1})$. We apply this procedure to any part of $O_{a,2}$ in slots I_j with $j > i$ until eventually either (1) or (2) is met. Then, the number of M_2-operations scheduled in both I_i and I_j with $j > i$ decreases by at least 1. The procedure is repeated for I_i until the number of such M_2-operations is reduced to at most 2.

Now, assume that there are exactly two M_2-operations $O_{a,2}$, $O_{b,2}$, $a < b$, scheduled both in slot I_i and in slots I_j with $j > i$. We show that it is possible to exchange parts of M_2-operations between slots $I_i, \ldots, I_n$ until one of the following holds:

(3) Either $O_{a,2}$ is not processed in any slot I_j with $j > i$ or $O_{b,2}$ is no longer processed in I_i.

(4) Either $O_{b,2}$ is not processed in any slot I_j with $j > i$ or $O_{a,2}$ is no longer processed in I_i.
(5) Each of the two operations $O_{a,2}$ and $O_{b,2}$ is processed in I_i as well as in I_j with $j > i$, but in $I_{i+1}, \ldots, I_{b-1}$ only $O_{b,2}$ is processed on M_2, and $O_{b,2}$ is not processed in any slot I_j with $j \geq b$.

In cases (3) and (4), the number of operations processed in I_i and in I_j with $j > i$ is reduced to 1 or 0. In all three cases (2.8) and (2.9) hold for slot I_i. In case (5), (2.8) and (2.9) also hold for $I_{i+1}, \ldots, I_{b-1}$. The exchange works as follows.

If $b < i$ (or $a < i$), then we can exchange parts of $O_{b,2}$ (or $O_{a,2}$) in I_i with parts of $O_{a,2}$ (or $O_{b,2}$) in slots I_j with $j > i$ until (3) (or (4)) holds.

Now, assume that $i < a < b$. We show that then we can modify slots $I_i, \ldots, I_n$ step by step until either (3) or (4) or (5) is met. First we attempt to extend $O_{a,2}$ in I_i in such a way that (3) is met. This is done as follows:

First, exchange parts of $O_{b,2}$ scheduled in I_i with parts of $O_{a,2}$ scheduled in slots I_j with $j > i$, $j \neq b$ (see Fig. 2.10a) until either:

- (3) is met and consequently (2.8) and (2.9) hold for I_i or
- no more parts of $O_{a,2}$ are scheduled in slots I_j with $j > i$, $j \neq b$. Thus, a part of $O_{a,2}$ is scheduled in I_b.

Again, these exchanges are feasible since $r(O_{1,1}) \geq \ldots \geq r(O_{n,1})$. Now, assume that an operation $O_{x,2}$ with $x \neq b$ is processed in a slot I_j with $i < j < b$. In this case we shift parts of:

- $O_{b,2}$ from I_i to I_j,
- $O_{x,2}$ from I_j to I_b,
- $O_{a,2}$ from I_b to I_i,

as shown in Fig. 2.10b until either:

- (3) is met and consequently (2.8) and (2.9) hold for I_i or
- there are no operations $O_{x,2}$ with $x \neq b$ scheduled in I_j with $i < j < b$.

Again, it is easy to see that these shifts are feasible. Now, assume that parts of $O_{b,2}$ are processed in slots I_j with $j > b$ (if they were not, then (5) would be met). In this case, we attempt to extend $O_{b,2}$ in I_i until (4) is met. This is done as follows:

We exchange parts of $O_{a,2}$ scheduled in I_i with parts of $O_{b,2}$ scheduled in slots I_j with $j > b$ (see Fig. 2.10c) until either:

- (4) is met and consequently (2.8) and (2.9) hold for I_i or
- no more parts of $O_{b,2}$ are scheduled in slots I_j with $j > b$. In this case, (5) is met (see Fig. 2.10d). Thus, (2.8) and (2.9) hold for slots $I_i, \ldots, I_{b-1}$. □

Lemma 2.3 shows when and how a slot and a feasible partial schedule without preemptions and idle time can be combined into another feasible schedule without preemptions and idle time. The lemma is as follows.

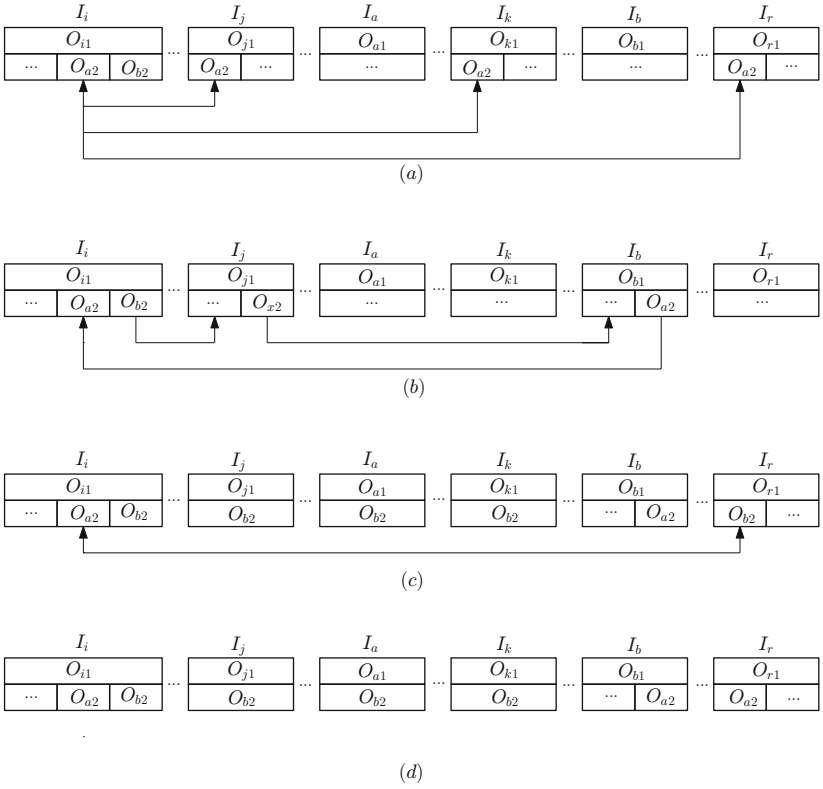

Fig. 2.10 Proof of Lemma 2.2: the case of exactly two preempted M_2-operations in I_i

Lemma 2.3 *Let S be a feasible partial schedule without preemptions and idle time. Let I_i be a slot not in S that meets the following conditions:*

- *For each operation $O_{k,2}$ scheduled in I_i and each operation O_{l1} scheduled in S, we have $r(O_{k,2}) + r(O_{l,1}) \leq s$.*
- *There is at most one operation $O_{a,2}$ processed both in I_i and in S.*

Then I_i and S can be merged into a feasible schedule S' without preemptions and idle time. □

Proof First, assume that, in S, the operation $O_{a,2}$ starts or finishes together with an operation on M_2. Then, we can easily modify S in such a way that $O_{a,2}$ starts the whole schedule S. By scheduling $O_{a,2}$ at the end of I_i, we obtain a feasible schedule S' without preemptions and idle time as shown in Fig. 2.11a.

Now, we assume that, in S, operation $O_{a,2}$ is scheduled in parallel with only one operation $O_{x,1}$ on M_1. If $O_{a,1}$ is scheduled after $O_{a,2}$ in S, then we can move $O_{a,2}$ to the left by successively exchanging it with its preceding operations until it either

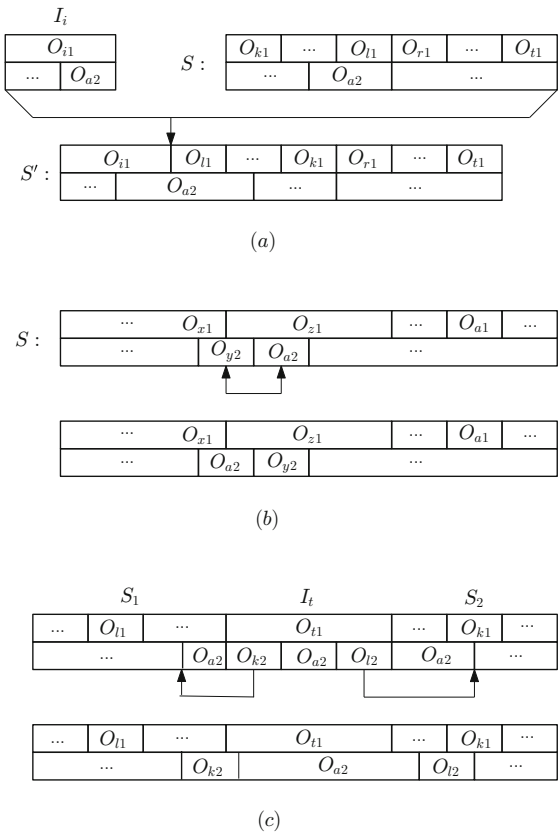

Fig. 2.11 Proof of Lemma 2.3, insertion of slot I_i in schedule S

starts or finishes together with an operation on M_1, or it is scheduled in parallel with at least two operations on M_1. Observer that $O_{a,2}$ can potentially be scheduled in parallel with all M_1-operations in S except $O_{a,1}$ (see Fig. 2.11b). If O_{a1} is not scheduled after $O_{a,2}$ in S, then move $O_{a,2}$ to the right. Thus, we may assume that $O_{a,2}$ is scheduled in parallel with at least two operations $O_{x,1}$, $O_{z,1}$ in S and that it neither starts nor finishes together with an operation on M_1. We obtain a feasible schedule by merging S and I_i as follows. Let S_1 be the part of S that finishes with $O_{x,1}$ on M_1, and let S_2 be the part of S that starts with $O_{z,1}$ on M_1 (observe that $O_{a,2}$ finishes S_1 and starts S_2 on M_2). We insert I_i between S_1 and S_2 and apply the following procedure to all operations $O_{y,2} \neq O_{a,2}$ scheduled in I_i (Fig. 2.11c):

- If $O_{y,1}$ is scheduled in S_1, then we insert $O_{y,2}$ into S_2 immediately after $O_{a,2}$ and shift the intermediate operations on M_2 to the left.
- Otherwise, we insert $O_{y,2}$ into S_1 immediately before $O_{a,2}$ and shift the intermediate operations on M_2 to the right.

Note that these shifts are feasible due to the assumption that $r(O_{k,2}) + r(O_{l,1}) \leq s$ for all $O_{k,2}$ scheduled in I_i and all $O_{l,1}$ scheduled in S. After moving all operations

$O_{y,2} \neq O_{a,2}$ either to the left or to the right, we obtain a feasible schedule S' without preemptions and idle time as required. □

We use Lemmas 2.2 and 2.3 in the following recursive procedure for converting a preemptive schedule into a non-preemptive schedule with the same makespan. It returns a feasible non-preemptive schedule made of the time slots $I_j, \ldots, I_n$ if called with $i = j$. Observe that we call the procedure recursively with index $i + 1$ if we reduce the number of M_2-operations that are processed both in slot I_i and in some slots I_j with $j > i$ to 1 or 0. Otherwise, due to Lemma 2.3 the time slots $I_{j+1}, \ldots, I_{b-1}$ have a special structure: only operation $O_{b,2}$ is processed on M_2, and moreover $O_{b,2}$ is not processed in any slot I_j with $j \geq b$. In this case, we first insert I_i into the non-preemptive schedule made of the slots $I_b, \ldots, I_n$, and then we insert the slots $I_{i+1}, \ldots, I_{b-1}$ one after another. The insertion can be done easily because the properties given in Lemma 2.3 are met. □

The following theorem is a direct consequence of Lemma 2.1 and Theorem 2.1.

Theorem 2.2 *The problem $O2|res\ 1\ldots|C_{\max}$ can be solved in $O(n^3)$ time.*

Proof The max-flow algorithm computes an optimal preemptive schedule in $O(n^3)$ time. The number of operations processed both in slot I_i and in slots I_j with $j > i$ can be reduced to at most 2 in time $O(n^2)$. It takes $O(n)$ time to change the schedule so that it meets (2.8) and (2.9) for a given slot I_i. Finally, the insertion of one time slot into a feasible partial schedule takes $O(n)$ time. Since the number of slots is n, an overall complexity is $O(n^3)$. □

Procedure Process-time-slot(i)

```
begin
    Exchange operations on M2 between time slots I_i, I_{i+1}, ..., I_n such that the number P
    of M2-operations processed both in I_i and in some slots I_j with j > i is minimal (Proof
    of Lemma 2.2);
    if P ≤ 1 then
        S:=Process-Time-Slot (i + 1);
        S:= Insert I_i into S (Proof of Lemma 2.3);
    else
        (P = 2; let O_{b,2} with b > i be the operation scheduled in I_i, I_{i+1}, ..., I_{b-1}(Proof
        of Lemma 2.2))
        S:=Process-Time-Slot(b);
        FOR j := i TO b − 1 DO
        S:= Insert I_j into S (Proof of Lemma 2.3);
    end
    return(S);
end
```

To illustrate the working of the procedure Process-Time-Slot, let us consider an instance shown in Table 2.1 with $s = 10$. Figure 2.12a shows a preemptive schedule S for the instance that meets conditions (2.8) and (2.9). The time slots of S are shown

Table 2.1 An instance of $O2|\text{res}1..|C_{\max}$

	Processing times		Resource requirements	
Job	M_1	M_2	M_1	M_2
1	4	5	10	3
2	2	5	8	0
3	4	4	7	0
4	3	2	5	2
5	4	2	2	9
6	3	2	0	5

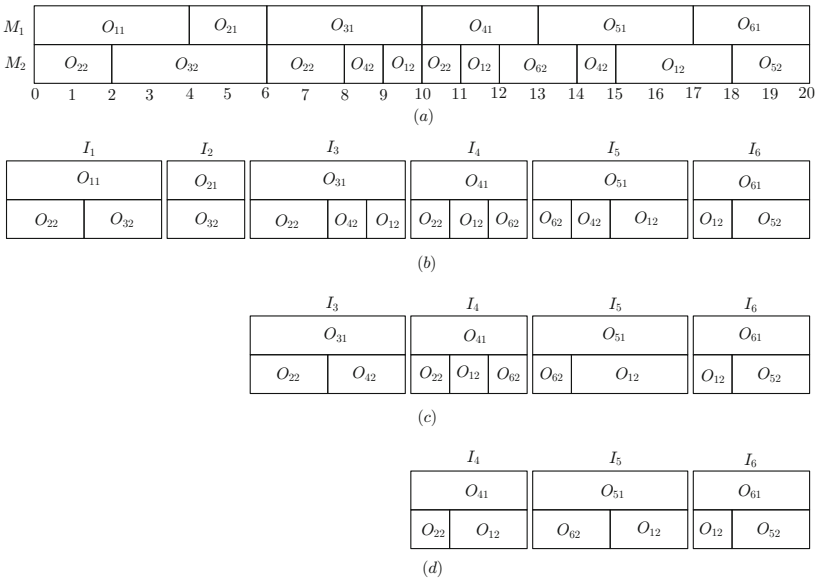

Fig. 2.12 Reduction of the number of preempted M_2-operations in time slots in the example

in Fig. 2.12b. We cannot reduce the number of preempted M_2-operations processed in slot I_1 to 1, but the condition (2.9) is satisfied for I_1. Thus, we skip the slot I_2 and continue with I_3. We extend $O_{4,2}$ by moving one unit of $O_{1,2}$ to slot I_5; see Fig. 2.12c. We continue with slot I_4 and extend $O_{1,2}$ there; see Fig. 2.12d. After this step, all slots meet the conditions in Lemma 2.2 and we can start to build up the non-preemptive schedule recursively. Figure 2.13a shows the schedule obtained by combining the slots I_5 and I_6. Next, slots I_4 and I_3 are inserted; see Fig. 2.13b and c. Now we have to skip I_2 and insert I_1; see Fig. 2.13d. Finally, we obtain a feasible non-preemptive schedule by inserting slot I_2; see Fig. 2.13e. Figure 2.14 shows how the demand for the resource changes over time in the schedule from Fig. 2.13e.

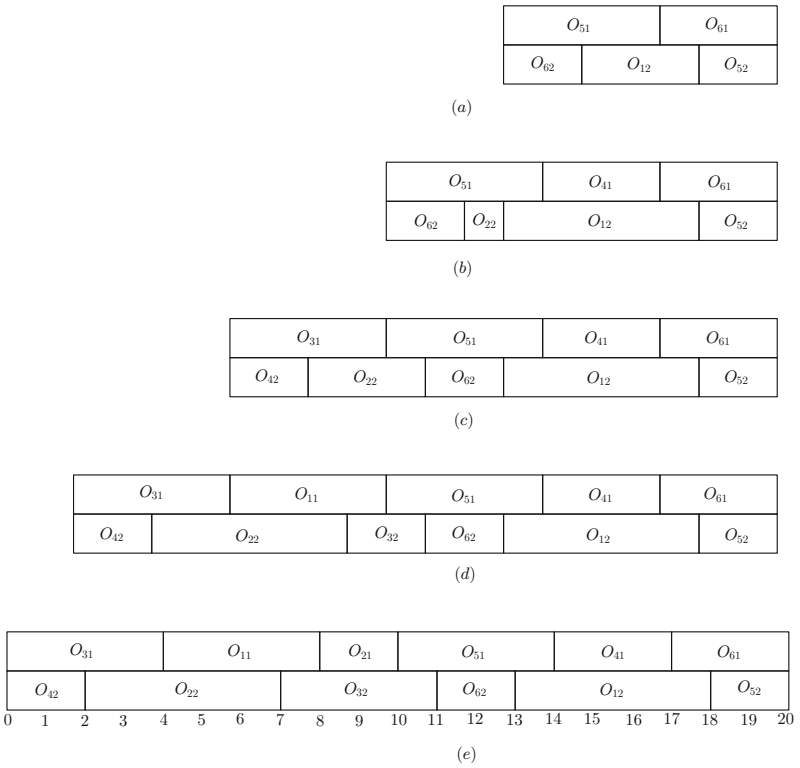

Fig. 2.13 Recursive buildup of the schedule in the example

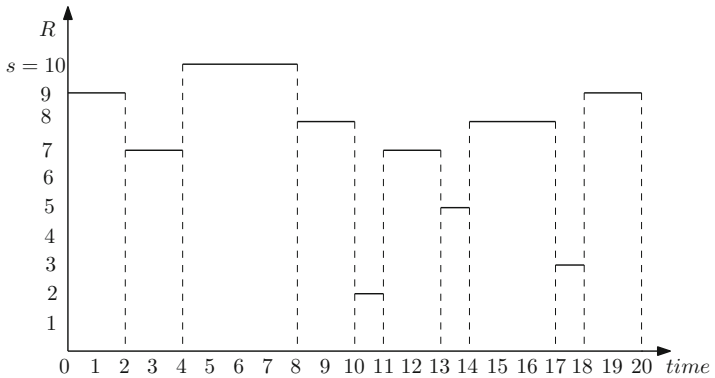

Fig. 2.14 The number of units of the additional resource required by the optimal schedule in Fig. 2.13e

2.6 Open Problems

Let $\mathcal{R}_1$ be the class of operation conflict graphs $G = (V, E)$ corresponding to the instances of $O2|\text{res}1..|C_{\max}$. We show in Sect. 2.5 that for any $G \in \mathcal{R}_1$ optimal preemptive and non-preemptive solutions have the same makespan. The vertex set V of $G \in \mathcal{R}_1$ can be partitioned into two disjoint sets $V_1 = \{O : R(O) \leq \frac{s}{2}\}$ and $V_2 = \{O : R(O) > \frac{s}{2}\}$. The set V_2 is a clique in G, and the set V_1, however, is *not* an independent set since it may include both operations of the same job, which creates a conflict (an edge in G). In addition, edges may exist between the vertices in V_1 and V_2. The following question remains open.

Problem 2.1 Is there a class $\mathcal{R}$, $\mathcal{R}_1 \subset \mathcal{R}$, of operation conflict graphs $G = (V, E)$ for which optimal preemptive and non-preemptive solutions to the two-machine open shop have the same makespan?

Problem 2.2 Characterize the class of all operation conflict graphs $G = (V, E)$ for which optimal preemptive and non-preemptive solutions to the two-machine open shop have the same makespan.

The two-machine open shop problem with *two* or more resource types benefits from preemptions. The optimal schedules with preemptions can be *shorter* than those without preemptions. Jurisch and Kubiak [14] show that the problem $O2|\text{res}211|C_{\max}$ with *two* resource types of capacity 1 each is NP-hard, and the problem $O2|\text{res}.11|C_{\max}$ with *arbitrary* number of resource types of capacity 1 each is NP-hard in the strong sense. The open shop problem with additional resources has been studied by Błażewicz et al. [1], Cochand et al. [5], de Werra et al. [9], and de Werra and Błażewicz [7, 8]. A review of some of those results can be found in Kubiak et al. [15]. Recent complexity results for two-machine open shop with additional resources can be found in Shabtay and Kaspi [18], Tellache and Boudhar [22], and Tellache et al. [23].

Problems

2.1 The network flow algorithm for multiple resources works in $O(n^3)$ time. Can the time be reduced to n^2 for the network for a single resource type?

2.2 Prove that the problem to decide whether there is a reduction to two jobs in the algorithm in Sect. 2.5 is NP-complete.

2.3 Define conflict and agreement graphs for the instance in Table 2.1.

2.4 Prove that the problem $O2|\text{res}211|C_{\max}$ is NP-hard.

2.5 Prove that the problem $O2|\text{res}.11|C_{\max}$ is NP-hard in the strong sense.

References

1. J. Błażewicz, W. Cellary, R. Słowiński, J. Węglarz, *Scheduling Under Resource Constraints—Deterministic Models* (J. C. Baltzer AG, Basel, Switzerland, 1986)
2. J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt, J. Węglarz, *Handbook on Scheduling: From Theory to Applications*. International Handbooks on Information Systems. (Springer, Berlin, 2007)
3. J. Blazewicz, J.K. Lenstra, A.H.G. Rinnooy Kan, Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)
4. P. Brucker, *Scheduling Algorithms* (Springer, Berlin, 1995)
5. D. Cochand, D. de Werra, R. Słowiński, Preemptive scheduling with staircase and piecewise linear resource availability. Z. Opns. Res. **33**, 297–313 (1989)
6. D. de Werra, Graph-theoretical models for preemptive scheduling, in *Advances in Project Scheduling*, ed. by R. Słowiński, J. Węglarz (Elsevier, Amsterdam, 1989), pp. 171–185
7. D. de Werra, J. Błażewicz, Some preemptive open shop problems with a renewable or a nonrenewable resources. Discrete Appl. Math. **35**, 205–219 (1992)
8. D. de Werra, J. Błażewicz, Addendum: some preemptive open shop scheduling problems with a renewable or a nonrenewable resources. Discrete Appl. Math. **35**, 103–104 (1993)
9. D. de Werra, J. Błażewicz, W. Kubiak, A preemptive open shop scheduling problem with one resource. Oper. Res. Letts. **10**, 9–15 (1991)
10. T. Gonzalez, Open shop scheduling, in *Handbook on Scheduling: Algorithms, Models, and Performance Analysis*, ed. by J.Y.-T. Leung (Chapman and Hall/CRC, 2004), pp. 6-1–6-14
11. T. Gonzalez, S. Sahni, Open shop scheduling to minimize finish time. J. ACM **23**, 665–679 (1976)
12. R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discr. Math. **5**, 287–326 (1979)
13. I.V. Gribkovskaia, C.-Y. Lee, V.A. Strusevich, D. de Werra, Three is easy, two is hard: open shop sum-bath scheduling problem refined. Oper. Res. Lett. **34**, 456–464 (2006)
14. B. Jurisch, W. Kubiak, Two-machine open shops with renewable resources. Opns. Res. **45**, 544–552 (1997)
15. W. Kubiak, C. Sriskandarajah, K. Zaras, A note on the complexity of openshop scheduling problems. INFOR **29** (1991)
16. M. Pinedo, L. Schrage, Stochastic shop scheduling: a survey, in *Deterministic and Stochastic Scheduling*, ed. by M.A.H. Dempster, J.K. Lenstra, A.H.G. Rinnooy Kan (Reidel: Dordrecht, 1982), pp. 181–196
17. M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th edn. (Springer, Berlin, 2016)
18. D. Shabtay, M. Kaspi, Minimizing the makespan in open-shop scheduling problems with a convex resource consumption function. Naval Res. Logist. **53**, 204–216 (2006)
19. N.V. Shakhlevich, V.A. Strusevich, Two machine open shop scheduling problem to minimize an arbitrary regular penalty function. Eur. J. Oper. Res. **70**, 391–404 (1993)
20. A.J. Soper, A cyclical search for the two machine flow shop and open shop to minimize finish time. J. Sched. **18**, 311–314 (2015)
21. V.S. Tanaev, Y.N. Sotskov, V.A. Strusevich, *Scheduling Theory: Multi-Stage Systems* (Kluwer Academic Publishers, Dordrecht, 1994)
22. N.E.H. Tellache, M. Boudhar, Open shop scheduling problems with conflict graphs. Discrete Appl. Math. **227**, 103–120 (2017)
23. N.E.H. Tellache, M. Boudhar, F. Yalaoui, Two-machine open shop problem with agreement graph. Theor. Comput. Sci. **796**, 154 –169 (2019)
24. M. van den Akker, H. Hoogeveen, G.J. Woeginger, The two-machine open shop problem: to fit or not to fit, that is the question. Oper. Res. Lett. **31**, 219–224 (2003)
25. G.J. Woeginger, The open shop scheduling problem, in *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, ed. by R. Niedermeier, B. Vallée Leibniz. International Proceedings in Informatics (Dagstuhl Publishing, 2018), pp. 4:1–4:12