

Chapter 2

Job Shop Scheduling - Formulation and Modeling

Machine scheduling problems arise in different structures and settings. In order to obtain a clear comprehension of the contents in this thesis, the introduction of the job shop scheduling problem with total weighted tardiness objective will be detailed later in this chapter. In the first section, the considered problem structure will be described. After that, the differentiation from other scheduling problems will be presented in Section 2.2. The Section 2.3 will then provide the mathematical model and basic information about the complexity of the problem. The modeling of the problem as a disjunctive graph and the resulting implications are important fundamentals for topics which will later be addressed. For this purpose, Section 2.4 and 2.5 contain the concept of the graph representation, the critical tree and the gained conclusions. Section 2.6 provides an illustration of the presented concepts with the help of the problem instance ft06.

2.1 Problem Structure

Following the introduction of French [47], the job shop scheduling problem (JSP) consists of a finite set of jobs $J = \{1, 2, \dots, n\}$ and a finite set of machines $M = \{1, 2, \dots, m\}$. More precisely, every job consists of a finite set of operations. The processing of an operation has to be performed on a preassigned machine, i. e. the i -th operation of job j , denoted by o_{ij} , is processed on machine $\mu_{ij} \in M$. The operation order of each job is fix, i. e. the technological machine sequence given for

every job has to be taken into account. The aim is to find a schedule for processing these n jobs on the m machines. Further conditions are as follows:

- The processing of the i -th operation of job j takes $p_{ij} > 0$ time units.
- Each operation has to be processed exactly once.
- Preemption is not allowed while operations are being processed.
- Processing operations may not overlap with one another.
- There is no machine-dependent or sequence-dependent setup time.
- Machines must always be available.

Beyond the described structure, the thesis focuses on the standard, dynamic version of JSP with job weights and due dates. In the standard JSP, every job has to be processed on every machine exactly once. This means that every job consists of m operations in which every pair of these operations is processed on different machines. This standardization helps to comprise a broad range of problem instances. If a job is not executed on machine $k \in M$ in a problem instance, the corresponding operation is simply neglected in the technological machine sequence. To be consistent with the assumption that every job passes every machine, one can alternatively set the processing time of these virtual operations to zero and add them at the first position of the technological sequence in order to avoid blockings.

One can furthermore distinguish between static and dynamic JSP. In the dynamic JSP, every job has an assigned release time $r_j \geq 0$ so that the first operation cannot start before r_j . In the static JSP, all jobs are available at the beginning of the planning horizon, i. e. $r_j = 0, \forall j \in J$. Note that the dynamic JSP covers the static JSP.

An additional attribute of a job j is its weight w_j which represents the job's relative importance in comparison to other jobs. Furthermore, every job has a due date $d_j \geq 0$ which should, but does not necessarily have to, be fulfilled in a schedule.

The quality of a solution is assessed by the obtained total weighted tardiness, defined as $TWT = \sum_{j=1}^n w_j \cdot t_j$, where $t_j = \max\{0, c_j - d_j\}$ is the resulting tardiness of job j in a schedule, and c_j its completion time. The resulting minimization problem is referred to as JSPTWT. The solution of a problem instance can be illustrated in a Gantt-Chart, as shown in Section 2.6.

The problem structure of the JSPTWT involves two questions: 1. Which job sequences should be determined on the machines? 2. When does the processing of each operation start? In general, both questions are answered at the same time. On the one hand, by fixing the start time of each operation, the job sequences can be derived from this outcome. On the other hand, given the job sequence on every machine, the start times of the operations are calculated by their earliest possible beginning with respect to the precedence relations given by their technological sequence.

In scheduling theory, there are three different classes of schedules [110]: semi-active, active and non-delay schedules. A schedule is called semi-active if and only if no operation could start earlier than its assigned start time without changing the corresponding job sequence on the machine. A schedule is called active if and only if there exist no earlier start of an operation without delaying the start of another operation. Finally, a schedule is called non-delay if and only if no machine is kept idle while an operation is schedulable on this machine at the same time. Based on these definitions, the following inclusions are observed: the class of semi-active schedule includes the set of active schedules, and the class of active schedules includes every non-delay schedule.

Since the total weighted tardiness objective is a regular measure, i. e. it is a non-decreasing function of the completion times, it is clear that at least one optimal solution corresponds to an active schedule, without necessarily having to be a non-delay schedule [47]. For this reason, the contents and the results in this thesis are applied to active schedules.

2.2 Classification into the Scheduling Theory

In general, scheduling problems cover a broad range of practical problems. As mentioned in Chapter 1, the process of production planning and control in manufacturing systems is one motivation. The occurring scheduling problems are summarized in the class of machine scheduling. The considered JSPTWT is also classified as a machine scheduling problem. For this reason, this section will only provide a short introduction to machine scheduling and differentiate between JSPTWT and other related problems. There are, however, other classes and fields of applications in industrial and service settings, e. g. timetabling, sports scheduling, crew scheduling, scheduling in health care, and many more [109].

Since the practical applications are manifold, there are different types of machine scheduling problems (see e. g. [84, 110]). Graham et al. [58] have proposed a classification scheme based on a three field notation $\alpha \mid \beta \mid \gamma$. The α -field describes the machine environment of the considered problem. Processing restrictions and additional constraints are indicated in the β -field. The last field γ specifies the objective function, which should be fulfilled in the best possible way.

The simplest form of the machine environment describes a planning problem with a single machine [110]. Other problem structures involve e. g. identical machines or flexible structures in the technological sequence. Closely related problems to the job shop problem are the flow shop problem and the flexible job shop problem. The flow shop scheduling problem is a specialization of the job shop where all jobs have the same technological sequence. The flexible job shop scheduling problem is a generalization of the job shop. Instead of needing exactly one specific machine to process an operation, there is a set of identical machines available for carrying out this single task.

Further restrictions and constraints occur in e. g. sequence-dependent setup times or blocking constraints. The introduction of setup times describes a scenario in which the setup activities on certain machines strongly depend on the job order given in the schedule. A blocking constraint results from limited buffers in front of the machines, i. e. a job blocks the current machine if the buffer of the subsequent machine is full. More details as well as other additional characteristics can be found in [110]. Note that the consideration of due dates is usually not additionally specified in the β -field.

In the last 5 decades, job shop scheduling has mainly focused on the minimization of the makespan c_{max} . The corresponding optimization problem is also referred to as minimum makespan job shop scheduling problem. This objective function is motivated by the fact that minimizing the makespan directly corresponds to a high utilization of the machines, and thus, decreases production costs. As mentioned before, the considered total weighted tardiness objective leads to a minimization of penalty costs. This objective function belongs to the class of min-sum objectives such as the minimization of the total completion times ($\sum_{j=1}^n c_j$), the minimization of the total flow time ($\sum_{j=1}^n (c_j - r_j)$) or the number of tardy jobs ($\sum_{j=1}^n u_j$).

In this thesis, according to the classification of Graham et al. [58] and the described problem structure in Section 2.1, the considered optimization problem is as follows:

$$J \mid r_j \mid \sum w_j t_j$$

2.3 Mathematical Model and Complexity

Based on the notation introduced in Section 2.1, the mathematical model of the JSPTWT can be formulated as a disjunctive program [2]:

$$\min \sum_{j=1}^n w_j t_j \quad (2.1)$$

$$\text{s.t.} \quad s_{ij} + p_{ij} \leq s_{i+1,j} \quad \forall j \in J, \forall i \in M \setminus \{m\} \quad (2.2)$$

$$s_{ij} + p_{ij} \leq s_{k,l} \vee s_{kl} + p_{kl} \leq s_{i,j} \quad \forall j, l \in J, \forall i, k \in M \setminus \{m\} \quad (2.3)$$

$$\text{with } j \neq l, \text{ and } \mu_{ij} = \mu_{kl}$$

$$t_j \geq s_{mj} + p_{mj} - d_j \quad \forall j \in J \quad (2.4)$$

$$t_j \geq 0 \quad \forall j \in J \quad (2.5)$$

$$s_{1j} \geq r_j \quad \forall j \in J \quad (2.6)$$

The decision variable s_{ij} represents the start time for operation o_{ij} . The constraints (2.2) ensure the technological sequence given for every job. The disjunctive constraints (2.3) capture the sequencing problem on every machine. The constraints (2.4) and (2.5) are part of the program to measure the resulting tardiness of each job. Finally, constraints (2.6) ensure that a job cannot start before its release time, and thus, capture the non-negativity of the decision variables s_{ij} . In the objective function (2.1) the weighted sum of the tardiness has to be minimized.

The disjunctive constraints (2.3) can be reformulated as a set of linear constraints with the help of the M-method [37]. For this purpose, additional binary variables have to be introduced indicating the different precedence relations on the machines. For a problem instance with 10 jobs and 5 machines, this linearization leads to a mixed-integer program with 285 decision variables, of which 225 are binary variables, and 510 constraints.

The above formulated mathematical model of the JSPTWT can be easily converted for solving the minimum makespan JSP. In addition to the swap of the objective function toward minimizing the makespan c_{max} , the following set of constraints replaces constraints (2.4) and (2.5):

$$c_{max} \geq s_{mj} + p_{mj} \quad \forall j \in J \quad (2.7)$$

Although the mathematical models of the JSPTWT and the minimum makespan

JSP are very similar, the JSPTWT is more complex. If the JSPTWT is already restricted to 1 machine (i. e. it becomes a single machine scheduling problem) and the release times of the jobs are set to zero, than the scheduling of n jobs with TWT objective is an $\mathcal{NP}$ -complete problem [83, 86]. With the integration of release times $r_j \geq 0$ the complexity result is the same [110]. On the other hand, scheduling n jobs on a single machine without release times while focusing on the minimization of the makespan is easy to solve. Every arbitrary sequence that corresponds to an active schedule is an optimal solution.² The problem classification into classes of different complexity is also demonstrated by the following example: processing two jobs on m machines with release times and unit processing times is strongly $\mathcal{NP}$ -hard for the TWT objective [133], whereas this problem is solvable in polynomial time with regard to the minimization of the makespan [132]. Further complexity results are presented by Knust [74].

A standard JSP, as defined in Section 2.1, is $\mathcal{NP}$ -hard for the TWT objective as well as makespan objective [110]. Since the optimal schedule for the TWT objective depends on the completion time of all jobs, finding this solution and proving its optimality is usually quite hard and takes a considerable amount of time. The comparison of the needed computation time according to TWT objective and makespan objective is additionally demonstrated by solving six small problem instances by using CPLEX 12.2. In the instances la01-la03 10 jobs have to be processed on 5 machines, whereas in the instances abz05, abz06, ft10, 10 jobs have to be processed on 10 machines. Further details of the instances as well as their modification to JSPTWT instances are presented in Section 7.1.1. For these computations, it is necessary to mention that the due dates of the jobs are generated with the help of

Instance	n	m	TWT objective		c_{max} objective	
			Opt. value	Comp. time	Opt. value	Comp. time
la01	10	5	2.299	43,5 s	666	9,6 s
la02	10	5	1.762	74,5 s	655	44,3 s
la03	10	5	1.951	47,6 s	597	16,2 s
abz05	10	10	1.403	6.905,4 s	1.234	73,7 s
abz06	10	10	436	371,8 s	943	9,5 s
ft10	10	10	1.363	3.510,0 s	930	2.730,2 s

Tab. 2.1: Results of CPLEX 12.2 for several problem instances.

²Note that every semi-active schedule is also active in this special case.

due date factor $f = 1.3$. Tab. 2.1 shows the objective function value of the optimal solutions and the computation times for both objective functions. The reported computation times for the TWT objective are significantly higher than those of the makespan objective. Even though the problem instances are small in comparison to real world instances, the computation time is up to more than 100 minutes. These results motivate again to develop powerful heuristics and metaheuristics for tackling the JSPTWT.

2.4 The Disjunctive Graph Model

The disjunctive graph model is a fundamental representation form of the minimum makespan JSP [2, 18]. Efficient heuristic concepts are based on this network model. As a consequence, the related literature already provides a disjunctive graph model for the JSPTWT (see e. g. [76, 111]). However, the presented models do not enable to determine the solution quality in terms of the resulting tardiness of a job. For this reason, a proper transformation of the disjunctive graph model for the JSPTWT will be presented in this section.

In general, problem instances of the minimum makespan JSP can be represented by a disjunctive graph $G = (N, A, E)$, where the i -th operation of job j is denoted by node $(i/j) \in N$. The set of nodes is completed by two dummy nodes, namely the source node 0 and the sink node 1. The set of operations of job j is connected by directed arcs $(i/j) \rightarrow (i+1/j) \in A$ representing its technological machine sequence. For every job j , there is one directed arc connecting the source node with the first operation node, $0 \rightarrow (1/j) \in A$, and one further directed arc connecting the last operation node with the finishing node, $(m/j) \rightarrow 1 \in A$. A weight is given for every arc $(i/j) \rightarrow (i+1/j) \in A$ which represents the processing time of operation (i/j) . Arcs $0 \rightarrow (1/j)$ have a weight 0. Finally, arcs $(m/j) \rightarrow 1$ are weighted by

Job j	o_{1j}		o_{2j}		o_{3j}		w_j	r_j	d_j
	μ_{1j}	p_{1j}	μ_{2j}	p_{2j}	μ_{3j}	p_{3j}			
1	1	3	2	4	3	4	1	0	12
2	3	2	1	3	2	4	2	2	15
3	1	5	3	5	2	2	1	2	18

Tab. 2.2: Data to the example 3x3 instance.

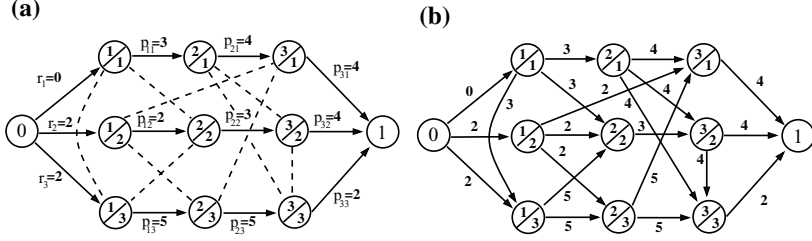


Fig. 2.1: Minimum makespan JSP: (a) Disjunctive graph G (dashed line = pair of disjunctive arcs), (b) feasible solution as directed graph G' .

the processing time of operation (m/j) . The precedence relation between operations $(i/j), (k/l) \in N$ of different jobs being processed on the same machine are represented by pairs of disjunctive arcs $\{(i/j) \rightarrow (k/l), (k/l) \rightarrow (i/j)\} \in E$. Again, a weight is assigned to each arc $(i/j) \rightarrow (k/l)$ of the set E that represents the processing time of operation (i/j) . By choosing suitable subsets of E , all possible job sequences on the machines can be displayed. Furthermore, by selecting one arc of each pair of disjunctive arcs, a subset E' of E is obtained, thereby describing a feasible schedule if and only if the corresponding graph $G' = (N, A, E')$ is acyclic [2].

Fig. 2.1(a) shows the disjunctive graph for an instance with 3 jobs and 3 machines and data summarized in the first 7 columns of Tab. 2.2. A solution of this problem is shown in Fig. 2.1(b). The makespan of this schedule corresponds to the length of one longest path from source 0 to sink 1. Note that there can be more than one longest path in the directed graph. For the example in Fig. 2.1(b), a longest path is $0 \rightarrow (1/1) \rightarrow (1/3) \rightarrow (2/3) \rightarrow (3/1) \rightarrow 1$; another one is $0 \rightarrow (1/1) \rightarrow (1/3) \rightarrow (2/2) \rightarrow (3/2) \rightarrow (3/3) \rightarrow 1$. Both have length 17.

For modeling the JSPTWT in an appropriate manner, the described disjunctive graph is modified in the following way. For every job j , an individual completion node B_j and an individual finishing node F_j are introduced (instead of the common sink node 1)³. Further arcs $0 \rightarrow F_j$ and $B_j \rightarrow F_j$ are added to the set A . Weights 0 and $-d_j$ are assigned to these arcs to compute the tardiness of job j as the non-negative distance between its completion time and due date. Additionally, the weight of arc $0 \rightarrow (1/j) \in A$ represents the release time of the job j . Fig. 2.2(a) shows the modified disjunctive graph $G = (N, A, E)$ for the considered example above with the additional data of weights, release times and due dates (see the right

³The introduction of the completion nodes B_j has already been proposed by Pinedo and Singer [111]. In their disjunctive graph, these nodes are denoted as sink nodes V_j .

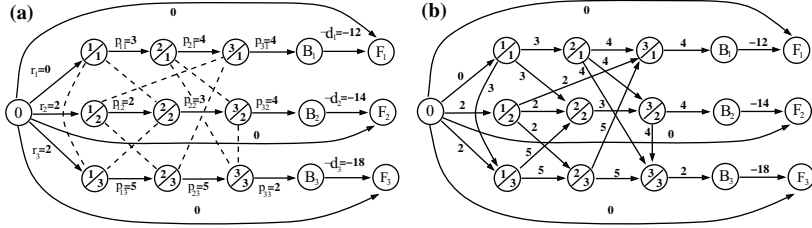


Fig. 2.2: JSPTWT: (a) Disjunctive graph G (dashed line = pair of disjunctive arcs), (b) feasible solution as directed graph G' .

part of Tab. 2.2). Like in the disjunctive graph representation of the minimum makespan JSP, a feasible solution is obtained by selecting one arc of each pair of disjunctive arcs from the set E (see Fig. 2.2(b) for the corresponding directed graph $G' = (N, A, E')$). In order to assess the solution quality of the underlying schedule, a longest path for every job j has to be computed, i. e. starting from 0 and ending in its finishing node F_j . It can be observed that the value of the longest path leading from 0 to B_j represents the completion time of job j , and the length of the longest path from 0 to F_j corresponds to the tardiness of job j . For example, considering job 1 in the solution of Fig. 2.2(b), the longest path leading from 0 to B_1 is $0 \rightarrow (1/1) \rightarrow (1/3) \rightarrow (2/3) \rightarrow (3/1) \rightarrow B_1$ and is of length 17. This value expresses the completion time of the third operation of job 1, and thus, equals to the completion of the whole job. The longest path leading from 0 to F_1 is $0 \rightarrow (1/1) \rightarrow (1/3) \rightarrow (2/3) \rightarrow (3/1) \rightarrow B_1 \rightarrow F_1$ and is of length 5. Accordingly, the tardiness of job 1 is given by $t_1 = \max\{0, c_1 - d_1\} = \max\{0, 17 - 12\} = 5$.

In contrast to other disjunctive graph models for the JSPTWT which have been proposed in the literature (see e. g. [76, 107]), the presented model incorporates the information whether a job is tardy or not.

2.5 The Concept of the Critical Tree

The disjunctive graph representation of the problem structure and the derivation of feasible schedules help to identify important direct precedence relations on machines that determine the solution quality. These kinds of precedence relations are represented by arcs on the longest path, also known as critical arcs.

Following the concept of Nowicki and Smutnicki [101], a longest path can be

decomposed into critical arcs and critical blocks. An arc that (i) belongs to a longest path and (ii) connects two operations that are processed consecutively on a machine is referred to as a critical arc. A critical block is defined as the sequence of operations connected by a maximum chain of critical arcs according to one longest path. The identification of critical arcs is used for the definition of local search based neighborhood operators (see Chapter 4), because the reversal of one critical arc generates a new schedule that does not violate any technological machine sequence of a job. Furthermore, only the reversal of critical arcs can improve the solution. The reversal of arcs that do not belong to any longest path as well as represent precedence relations on machines, do not influence any origin longest path, and thus, cannot reduce their length.

While assessing the total weighted tardiness measure of the schedule represented in G' , all its critical arcs and blocks are identified during the construction of the n longest paths. The composition of all longest paths results in a tree graph with root 0 and n leaves. This tree is called the critical tree, and is formally defined as follows:

Definition 1 (Critical Tree). *Given the directed graph G' of a feasible schedule and a weight w_j for each job j . The composition of all longest paths $\mathcal{LP}(0, F_j)$, each leading from 0 to F_j , is called **critical tree** $\mathcal{CT}(0, F)$. The length of the tree is defined as:*

$$L(\mathcal{CT}(0, F)) = \sum_{j=1}^n w_j \cdot L(\mathcal{LP}(0, F_j))$$

The critical tree $\mathcal{CT}(0, F)$ derived for solution G' of the example instance (see Fig. 2.2(b)) is shown in Fig. 2.3. Since there are three jobs, it is composed of three longest paths. For job $j = 1$, path $\mathcal{LP}(0, F_1) = 0 \rightarrow (1/1) \rightarrow (1/3) \rightarrow (2/3) \rightarrow (3/1) \rightarrow B_1 \rightarrow F_1$ is determined which is of length $L(\mathcal{LP}(0, F_1)) = 3 + 5 + 5 + 4 - 12 = 5$. Consequently, job 1 is tardy by 5 time units. Job 2 is also tardy, as indicated by

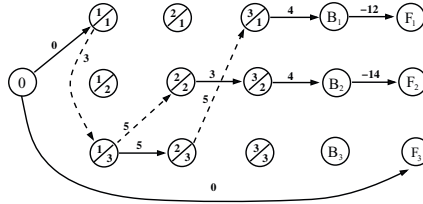


Fig. 2.3: Graph structure of all longest paths resulting in the critical tree $\mathcal{CT}(0, F)$ (dashed arc = critical arc).

<http://www.springer.com/978-3-658-10291-3>

Job Shop Scheduling with Consideration of Due Dates
Potentials of Local Search Based Solution Techniques
Kuhpfahl, J.

2016, XXV, 190 p. 37 illus., Softcover

ISBN: 978-3-658-10291-3