The Job-Shop Problem: Old and New Challenges

Peter Brucker
Universität Osnabrück, Albrechtstr. 28a, 49069 Osnabrück, Germany,
pbrucker@uni-osnabrueck.de

The job-shop problem is one of the most difficult classical scheduling problems. An instance with ten jobs to be processed on ten machines formulated in 1963 was open for more than 25 years. It was finally solved by a branch-and-bound algorithm. Very simple special cases of the job-shop problem are already strongly NP-hard.

After a short review of these old challenges we consider practical applications like problems in flexible manufacturing, multiprocessor task scheduling, robotic cell scheduling, railway scheduling, air traffic control which all have an underlying job-shop structure. Methods to solve these problems and new challenges in connection with them are indicated.

Keywords: Job-shop scheduling problem, complexity, branch-and-bound algorithm, local search, flexible manufacturing, multi-processor task scheduling, robotic cell, railway scheduling, air traffic control.

1 Introduction

The job-shop problem can be formulated as follows. Given are m machines $M_1, M_2, ..., M_m$ and n jobs $J_1, J_2, ..., J_n$. Job J_j consists of n_j operations $O_{ij} (i = 1, ..., n_j)$ which have to be processed in the order $O_{1j}, O_{2j}, ..., O_{n_j j}$. It is convenient to enumerate all operations of all jobs by $k = 1, ..., N$ where $N = \sum_{j=1}^{n} n_j$. For each operation $k = 1, ..., N$ we have a processing time $p_k > 0$ and a dedicated machine $M(k)$. k must be processed for p_k time units without preemptions on $M(k)$. Additionally a dummy starting operation 0 and a dummy finishing operation $N + 1$, each with zero processing time, are introduced. We assume that for two succeeding operations $k = O_{ij}$ and $s(k) = O_{i+1,j}$ of the same job $M(k) \neq M(s(k))$ holds. Let S_k be the starting time of operation k. Then $C_k = S_k + p_k$ is the finishing time of k and (S_k) defines a schedule. A schedule (S_k) is feasible if for any succeeding operations k and $s(k)$ of the same job $S_k + p_k \leq S_{s(k)}$ holds and for two operations k and h with $M(k) = M(h)$either $S_k + p_k \leq S_h$ or $S_h + p_h \leq S_k$. One has to find a feasible schedule (S_k) which minimizes the makespan $\max_{k=1}^{N} C_k$.

The flow-shop scheduling problem is a special case of the job-shop scheduling problem in which each job has m operations and the $i-th$ operation of each job has to be processed on machine M_i.

The paper is organized into four sections. Section 2 describes how to present solutions for the job shop scheduling problem and discusses some old challenges. New challenges arise in connection with applications. Section 3 describes some of these applications and shows how the classical job-shop scheduling problem needs to be extended in connection with these applications. Also corresponding solution methods are indicated. The last section contains some conclusion.

2 The job-shop problem: Old challenges

2.1 Solution representation

To solve the job-shop scheduling problem on each machine M_l one has to sequence all operations to be processed on M_l. Given all these machine sequences a corresponding left shifted schedule (S_k) (if it exists) can be constructed as follows: Let (V, A, d) be the directed network where:

1. V is the set of all operations $0, 1, ..., N, N+1$.
2. $(i, j) \epsilon A$ if and only if one of the following conditions is satisfied:
   - i and j belong to the same job and j is an immediate successor of i,
   - $i = 0$ and j is the first operation of a job or i is the last operation of a job and $j = N+1$,
   - i and j are processed on the same machine and j is sequenced immediately after i.

     The first two types of arcs are called conjunctive arcs, the last type of arcs are called fixed disjunctions.
3. $d_{ij} = p_i$ for all $(i, j) \epsilon A$.

The machine sequences define a feasible schedule if and only if (V, A, d) contains no positive cycle. In this case the lengths S_k of the longest $0-k$ paths define a feasible schedule (S_k). A longest path from 0 to $N+1$ is called critical. The length of a critical path is equal to the makespan of the schedule (S_k).

2.2 Complexity

The job-shop problem is one of the hardest scheduling problems. Only a few special cases are polynomially solvable, e.g.

- the flow-shop scheduling problem with two machines,
- the job-shop scheduling problems with two jobs,
- the job-shop scheduling problem with two machines and $p_k = 1$ for all operations k

Slightly generalized versions of these problems like

- the flow-shop scheduling problem with three machines,
- the job-shop scheduling problem with three machines and three jobs,
- the job-shop scheduling problem with three machines and $p_k = 1$ for all operations k

are already strongly NP-hard.

More detailed lists containing the hardest job-shop scheduling problems which are polynomially solvable and the easiest problems which are already NP-hard together with corresponding references can be found under http://www.mathematik.uni-osnabrueck.de/research/OR/class.

2.3 Old challenges

Besides questions concerning the complexity of special cases of the job-shop scheduling problem one of the old challenges was to find an optimal solution of a benchmark problem with 10 jobs and 10 machines. This benchmark problem was introduced in the book of Muth and Thompson [13] and could be solved only 25 years later by Carlier and Pinson [7]. Carlier and Pinson developed a clever branch-and-bound algorithm which uses constraint propagation techniques. Since then other enumerative methods have been developed, cf. the surveys of Pinson [15], Blazewicz et al. [2], and Jain & Meran [8]. Later Brinkkötter & Brucker [3] have solved 15×15 instances. It seems to be hopeless to solve larger instances by exact methods. One has to use heuristics. The best known heuristics for the job-shop scheduling problem are tabu search heuristics (cf. Nowicki & Smutnicki [14]).

3 Applications and new challenges

The job-shop problem is a basic scheduling model. Real world applications usually lead to more complex situations. We will introduce some of these applications and describe the extensions needed to cover them. Methods to solve these problems and corresponding new challenges are indicated.

3.1 Applications

Next we describe some applications which lead to extended job-shop scheduling models.

3.1.1 Flexible manufacturing

In a flexible manufacturing system the machines are equipped with tools. An operation can be processed on machines only which are equipped with the tools needed to process the operation. In other words, associated with each operation i is a set $\mu(i)$ of machines which can process i. One has to assign to each operation a machine in $\mu(i)$ and to solve the resulting job-shop problem.

3.1.2 Multiprocessor task scheduling

Tasks (operations) are instructions of a computer program (job). We assume that the tasks are to be performed one after the other. Associated with each task i is a set $\mu(i)$ of processors which are simultaneously needed to perform the task, i.e. two tasks i and j cannot be processed at the same time if $\mu(i) \cap \mu(j)$ is non-empty. If we want to schedule the instructions of a loop with a large number of repetitions we have to solve a cyclic scheduling problem.

3.1.3 Robotic cell scheduling

A robotic cell is a flow-shop or job-shop scheduling problem in which the jobs must be transported from machine to machine. Transports are performed by one or more robots. We have to assign the transport operations to the robots and to schedule both the machine and robot operations. Notice that empty moves of a robot may be necessary, e.g. if a robot moves a job from machine A to B and then another job from machine C to D it has to move empty from machine B to C before the second transport operation can be performed. Usually there is no buffer or buffers with only limited capacity to store the jobs outside the machines (or the robots).

If there are only few jobs to be processed in large numbers then periodic schedules are of interest. This leads to cyclic scheduling problems.

3.1.4 Railway scheduling

A railway scheduling problem can be modeled by a job-shop scheduling problem by dividing the railway network into track segments. A job corresponds with a train going from some origin to a destination. The operations of a job correspond with the track segments on the route of the train. The processing time of an operation is the time needed to pass the track segment. Only one train can enter a track segment. Thus, the next track segment on the route of the train must be empty before the train can enter. Otherwise the train must wait blocking the current track segment. Stopping or no-stopping has an influence on the processing time. On the other hand processing times have an influence on stopping. To resolve this problem simulation is used.

3.1.5 Air traffic control

The air space is divided into Air Traffic Control Sectors (ATCS). Each ATCS has a given capacity. Furthermore, there are n planes $j = 1, ..., n$. For each plane we have a route given by a sequence of ATCS´s with flight times for passing the ATCS´s. There is also given a time window $[a_j, b_j]$ in which a plane j must start and a best starting time t_j in $[a_j, b_j]$. For each starting time $t \in [a_j, b_j]$ the value $f_j(t)$ defines the cost for starting at time t. f_j is a function which increases with the distance between t_j and t. The planes correspond to jobs and the sequence of ATCS´s correspond to the operations of the job. However no-wait constraints must be satisfied, i.e when an operation finishes the next operation must start immediately. Also the makespan objective function has to be replaced by the function $\sum_{j=1}^{n} f_j(S_j)$ where $S_j \in [a_j, b_j]$ is the departure time of plane j. One has to fix the departure times of all planes such that the ATCS capacities are not violated and the total costs are minimized.

3.2 Extensions of the model

In the previous section we have introduced some applications which lead to extensions of the job-shop problem. In this section we will describe such extensions.

3.2.1 Assignment and scheduling

To solve complex job-shop scheduling problems we often have to assign (transportation) operations to machines (robots) and to solve the resulting scheduling problem by sequencing all operations to be processed on the same machine (robot). This is done in two stages or simultaneously (cf. [11]).

3.2.2 Positive and negative time-lags

The relation $S_i + p_i \leq S_j$ may be replaced by $S_i + d_{ij} \leq S_j$. Depending on the sign of d_{ij} this relation has different meanings: if $d_{ij} > 0$ then d_{ij} is a minimal distance between S_i and S_j (positive time-lag). If $d_{ij} < 0$ then then we have $S_i \leq S_j + |d_{ij}|$ (negative time-lag). Thus, if additionally $S_i \geq S_j$ then $|d_{ij}|$ is a maximal distance between S_j and S_i. Positive or negative or zero time-lags may be used to model

- release times and deadlines,
- exact relative timing, especially no-wait constraints,
- setup times,
- machine unavailability's,

- maximum lateness objectives by makespan objectives.

3.2.3 Blocking operations

An operation i is called a blocking operation if there is no buffer to store i after finishing on $M(i)$. Thus, if the next machine on which the successor operation $s(i)$ of i must be processed is occupied by another job then i must stay on $M(i)$ until the other job leaves the next machine. During this stay $M(i)$ is blocked for other jobs. Let i and j be two operations to be processed on the same machine and assume that j is scheduled after i. If i is a blocking operation then j cannot start before the successor operation $s(i)$ of i is started. Therefore we have the constraint $S_{s(i)} \leq S_j$. If i is non-blocking then we have $S_i + p_i \leq S_j$. The last operation of a job is usually a non-blocking operation. These concepts have be discussed in [10].

3.2.4 General objective function

Let C_j be the finishing time of the last operation of job j. Instead of minimizing the makespan $\max C_j$ one may be interested in minimizing an objective function $f(C_1, ..., C_n)$. The following cases are of special interest (cf.[6]):

- f is monotone non-decreasing,
- $f(C_1, ..., C_n) = f_1(C_1) + ... + f_n(C_n)$ where the f_j are arbitrary cost functions.

3.2.5 Cyclic scheduling

In a cyclic machine scheduling problem each operation must be repeated infinitely often. By $< i, k >$ we denote the k-th occurrence of operation i. A schedule is defined by the starting times $t(i, k)$. It is called periodic with cycle time α if $t(i, k) = t(i, 0) + \alpha k$ for all operations i and integers k. There are generalized precedence constraints between pairs (i, j) of operations defined by

$$t(i,k) + d_{ij} \leq t(j, k + h_{ij}) \qquad \text{for all integers } k.$$

d_{ij} is called delay and h_{ij} is called height. All occurrences of operation i must be processed on a dedicated machine $M(i)$. For all operations i, j with $M(i) = M(j)$ and all integers k, l the machine constraints

$$t(i,k) + p_i \leq t(j,l) \text{ or } t(j,l) + p_j \leq t(i,k)$$

must be satisfied. One has to find a cyclic schedule satisfying the generalized precedence constraints and machine constraints which minimizes the cycle time.

A cyclic job-shop scheduling problem is as special cyclic machine scheduling problem. It can be defined by specializing the generalized precedence constraints in the following way:

- Each conjunctive arc (i, j) induces a constraint $t(i, k) + p_i \leq t(j, k)$, i.e. a generalized precedence constraint with delay $d_{ij} = p_i$ and height $h_{ij} = 0$.
- Additionally a return arc $(N+1, 0)$ is introduced with a corresponding generalized precedence constraint $t(N+1, k) \leq t(j, k+h_{N+1,0})$, i.e. with delay 0 and height $h_{N+1,0}$ which is considered as a parameter.

Depending on the height $h_{N+1,0}$ optimal cyclic schedules with different cycle times are provided. The optimal cycle time is a non-increasing function of $h_{N+1,0}$. The problem is equivalent to the job-shop problem with makespan objective function if $h_{N+1,0} = 1$. Increasing $h_{N+1,0}$ provides more compact cyclic schedules usually with smaller cycle times.

There are other possibilities to introduce return arcs (cf. Brucker & Kampmeyer[5]).

3.3 Solution methods

The following techniques have been applied to solve the job-shop problem and its generalizations:

- mixed integer linear programming,
- branch-and-bound,
- heuristics.

The first two methods may provide optimal solutions. They are successful if applied to small instances. In most applications we have medium or large sized instances. This implies that one has to apply heuristics. The most successful heuristics are local search heuristics, especially tabu search (cf.[14]). In the next section we will discuss such heuristics in more detail.

3.4 Local search methods

Local search is an iterative procedure which, starting with some initial solution, moves from one solution to the next until a stopping condition is satisfied. The overall goal of this search is to improve the solutions. We have

- a search space S, and
- a neighborhood structure $N : S \to 2^S$.

For each $s \in S$ the set $N(s) \subseteq S$ describes the subset of solutions which can be reached from s in an iteration.

3.4.1 Solution representation and the search space

A solution can be defined by

- an assignment of operations to machines, and
- for each machine M_i a sequence π_i of all operations to be precessed on M_i.

Given the assignment and sequences $\pi = (\pi_i)$ a corresponding optimal schedule (if it exists) can be calculated by solving

- a longest path problem for monotone objective functions $f(C_1, ..., C_n)$ (as indicated in Section 2.1), or
- a min-cut problem for objective functions of the form $f_1(C_1) + ... + f_n(C_n)$ (cf. [12]), or
- parametric longest path problems with the cycle time as parameter for a cyclic scheduling problem ([1], [9]).

3.4.2 Moves

Moves are performed

- by changing the assignment (e.g. by moving an operation from one machine to another), or
- by changing a sequence (e.g. by shifting an operation or swapping two operations).

This can be done hierarchically or simultaneously. In the following we assume that we want to minimize the makespan. Furthermore assume that the assignments and sequences have been fixed such that a corresponding feasible schedule exists. Let CP be a longest (critical) path in the corresponding network. Then it can be shown that changing the sequence on one machine by reversing an arc in CP (if such an arc exists) will provide again a feasible schedule. A neighborhood which is restricted to such moves is opt-connected, i. e. it is possible to reach an optimal solution from any feasible solution by applying a finite sequence of moves of this type. Unfortunately, many moves which change the sequence on one machine by reversing an arc in a critical path do not change the critical path length. For these reasons a more powerful neighborhood based on blocks has been invented. A block is a sequence $i_1, i_2, ..., i_k$ of at least $k > 1$ successive operations of a critical path assigned to the same machines where k is maximal. A more powerful neighborhood is provided by the following theorem.

Blocktheorem*: To improve the makespan, at least one operation of a block B must be shifted in the corresponding machine sequence before the first or the last operation in B.*

The neighborhood defined by moves described in the Blocktheorem provides good results. However, it is a long-standing open question wether such a neighborhood is opt-connected. The Blocktheorem can be generalized to job-shop problems with blocking operation [4] and to cyclic scheduling problems with or without blocking operations [9].

3.5 New challenges

There is much room to improve existing heuristics and to develop new heuristics for complex job-shop scheduling problems. For example the following areas and their combinations are of interest:

- job-shop scheduling problems with buffers of limited capacity,
- cyclic job-shop scheduling problems, and
- routing and sequencing problems.

When applying local search heuristics an important issue is to avoid deadlock when moving from one solution to the next. Repair mechanisms have been developed which can be applied when a move turns a feasible solution into a deadlock situation. However, often the quality of solutions provided by such repair mechanisms have a poor quality.

Another issue is to improve local search methods for combined assignment and sequencing problems. In [4] buffer slot assignments are replaced by sequences which allow to calculate the assignments in polynomial time. It is an interesting question wether similar ideas can be applied to other combined assignment and sequencing problems.

4 Conclusion

The classical job-shop scheduling problem is an optimization problem which has some basic structure. Practical problems are usually more complex. We have presented models for complex job-shop scheduling problems. Based on these models heuristics to solve such complex scheduling problems have been developed. Most promising heuristics are local search algorithms (especially tabu search). Structural properties of the problem should be exploited when developing these heuristics.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin (1993), *Network Flows*, Prentice Hall, Englewood Cliffs.

[2] J. Błażewicz, W. Domschke, E. Pesch (1996), The job shop scheduling problem: conventional and new solution techniques, *European Journal of Operational Research* **93**, 1 – 33.

[3] W. Brinkkötter, P. Brucker (2001), Solving open benchmark instances for the job shop problem by parallel head-tail adjustments, *Journal of Scheduling*, 53 – 64.

[4] P. Brucker, S. Heitmann, J. Hurink, T. Nieberg (2006), Job-shop scheduling with limited capacity buffers, *OR Spectrum* **25**, 151 – 176.

[5] P. Brucker, T. Kampmeyer (2007), Cyclic Scheduling Problems and their Applications, submitted to *Discrete Applied Mathematics*.

[6] P. Brucker, S. Knust (2006), *Complex Scheduling*, Springer, Heidelberg.

[7] J. Carlier, E. Pinson (1989), An algorithm for solving the job-shop problem, *Management Science* **35**, 164 – 176.

[8] A.S. Jain, S. Meeran (1999), Deterministic job-shop scheduling: past, present and future, *European Journal of Operational Research* **113**, 390 – 434.

[9] T. Kampmeyer (2006), *Cyclic scheduling problems*, PhD-Thesis, Fachbereich Mathematik/Informatik, University of Osnabrück.

[10] A. Mascis, D. Pacciarelli (2000), Job-shop scheduling with blocking and no-wait constraints, *European Journal of Operational Research* **143**, 498 – 517.

[11] M. Mastrolilli, L.M. Gambardella (2000), Effective neighborhood functions for the flexible job shop problem, *Journal of Scheduling* **3**, 3 – 20.

[12] R.H. Möhring, A.S. Schulz, F. Stork, M. Uetz (2003), Solving project scheduling problems by minimum cut computations, *Management Science* **49**, 330 – 350.

[13] J.F. Muth, G.L. Thompson (1963), Industrial Scheduling, Prentice Hall, Englewood Cliffs.

[14] E. Nowicki, C. Smutnicki (2005), An advanced tabu search algorithm for the job-shop problem, *Journal of Scheduling* **8**, 145 – 159.

[15] E. Pinson (1995), The job-shop scheduling problem: a concise survey and some recent developments, in: P. Chretienne, E. G. Coffman, J.K. Lenstra, Z. Liu (eds.), *Scheduling Theory and its Applications*, John Wiley, 277 – 293.