MULTIPLE EXCHANGES OF JOB ORDERS FOR LOCAL SEARCH OF NO-BUFFER JOB SCHEDULING PROBLEM

Takehiro Hayasaka and Rei Hino

Department of Mechanical Science and Engineering
Graduate School of Engineering
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan
Email: hino@mech.nagoya-u.ac.jp

Abstract

This paper describes a procedure for local modifications of a job shop schedule planned under a no-buffer constraint. The procedure is for correcting the infeasibility caused by exchanging two consecutive operations on the same machine. An extended disjunctive graph, which has reverse conjunctive arcs connecting two consecutive operating processes, is introduced to identify the operations preventing the feasibility of such a schedule. An example of the modifications for a given job shop schedule is shown to discuss the advantages of the proposed procedure.

Keywords: job shop schedule, no-buffer, local search, disjunctive graph

1. INTRODUCTION

Industrial scheduling performs an important role in manufacturing activities to produce a high variety of products with low material consumption and energy use. It is one of the reasons that scheduling is being studied by many researchers in a variety of fields; nevertheless, scheduling problems are difficult to optimize mathematically.

The fundamental scheduling problem can be described simply, but its adaptation to actual manufacturing processes introduces a number of different complications. Various extensions to scheduling problems are developed in response to particular situations (Muth et al., 1963; Tamaki, et al., 1995; Brucker, 2006). The consideration of buffer capacity is one of the typical extensions used to reduce the inventory of products and shorten their lead-time in a manufacturing company (Hall, et al., 1996; Brucker, et al., 2006).

Buffer capacity constraints complicate the problem such that planning even one feasible schedule becomes difficult (Pinedo, et al., 1989). An alternative graph is proposed to describe any schedule under no-buffer and no-wait constraints (Mascis, et al., 2002). A reference chain graph is another approach to judging the feasibility of a schedule when assigning each job to a machine (Yong, et al., 2001; Fu, et al., 2003). A general resource model is proposed from the point of view of role of resources in manufacturing, and an equivalent job shop scheduling problem with finite buffer capacity is derived (Hino, et al., 2005). Also a disjunctive graph with reverse conjunctive arcs is proposed in order to judge the feasibility of a planned schedule (Hino, et al., 2007).

In the present paper, a job shop scheduling problem under a no-buffer constraint is studied and a procedure is proposed to change the operating order of operations processed on the same machine. The procedure is to exchange multiple operating orders on multiple machines simultaneously when the operating order of two consecutive operations is exchanged to improve the performance of a planned schedule. According to the proposed procedure, any exchange of two consecutive operations becomes possible, and a local search for improvement of a schedule is performed with ease.

2. JOB SHOP SCHEDULING PROBLEM WITH NO-BUFFER CONSTRAINT

In this paper, a no-buffer constraint is taken into account in job shop scheduling. A machine cannot begin any other operation until a downstream machine receives the semi-finished product that it just processed (Brucker, 2006), which essentially means that the machine is blocked until it releases the product. This problem is described in the form of the following mixed-integer programming (Hino, 2008):

\[
\text{minimize} \quad C_{\text{max}}
\]

subject to

\[
C_{\text{max}} \geq f_{a,n_a}^w \quad (2)
\]

\[
s_{\alpha,i}^w \geq 0 \quad (3)
\]

\[
\zeta_{\alpha,i+1}^w \geq s_{\alpha,i}^w + p_{\alpha,i}^w
\]

\[
x_{\beta,j}^w \geq s_{\alpha,i+1}^w - M \cdot x_{\alpha,i+1,\beta,j} \quad (4)
\]

\[\forall \zeta, \psi, \zeta \in R \quad \forall \alpha, \beta \in E \quad \forall \psi \in Q_{a} \forall j \in Q_{\beta} \alpha \neq \beta
\]

\[x_{\alpha,i,\beta,j} \in \{0,1\}
\]
The symbols used in this paper are defined as follows.

\( j_{a,i} \): \( i \) th process for product \( a \).  
\( j_{a,i}^c \): blocking by \( j_{a,i} \).  
\( s, f, p \): starting time, finishing time and duration time of operation and blocking

In this paper, a blocking with swap is allowed, i.e. when two or more products are waiting for a machine that is blocked by another product, all the products must move cyclically and simultaneously to the next machine (Mascis, et al., 2002).

A pair of jobs is examined to calculate the time of a job, because the time depends on whether the other job is assigned to the same machine or a different one as shown in Fig. 1. In order to clearly state the relation between two jobs, the term “operation” is used when referring to jobs assigned to the same machine, whereas the term “process” is used when referring to jobs assigned to different machines for the same product.

The time of each job is determined according to the following steps when a set of feasible operating orders of jobs assigned to machines is prepared.

The starting time of a job \( s_{a,i}^w \) is calculated from the finishing time of blocking by the preceding operation \( f_{a,i-1}^w \) and the finishing time of the preceding process \( f_{a,i-1} \):

\[
s_{a,i}^w = \max (f_{a,i-1}^w, f_{a,i-1}^c) + p_{a,i}^w
\]

Here, \( \max() \) takes the later time of the two in the parenthesis. The finishing time of a job \( f_{a,i}^w \) is delayed for the processing time \( p_{a,i}^w \),

\[
f_{a,i}^w = s_{a,i}^w + p_{a,i}^w
\]

The starting time of blocking by a job \( s_{a,i}^c \) is equal to the finishing time of the same job \( f_{a,i}^w \), and the job \( f_{a,i}^w \) blocks any other operation until the next machine becomes available for the next process \( f_{a,i+1}^w \). This means that the finishing time of blocking \( f_{a,i}^w \) is equal to the starting time of the next process \( f_{a,i+1} \):

\[
s_{a,i}^c = f_{a,i}^w = s_{a,i+1}^w
\]

The duration time of blocking is identified after these calculations by the following equation:

\[
f_{a,i}^w = f_{a,i}^w - s_{a,i}^w
\]

A Gantt chart is used to evaluate a planned schedule in practical situations. On the other hand, a disjunctive graph is used to analyze the logical structure of a scheduling problem. We introduce a disjunctive graph with additional conjunctive arcs to cope with the blocking status due to the no-buffer constraint.

In this type of graph, the conjunctive arcs represent the processing orders of a product, and the disjunctive arcs represent the operating orders of a machine. When considering the blocking of jobs, another conjunctive arc is set in the reverse direction of the one mentioned before is added between every two processes. This is because the finishing time of the blocking of a certain process is determined by the starting time of the succeeding process.

If the order of two jobs is exchanged, there is the possibility of obtaining an infeasible schedule. An infeasible schedule is derived because the finishing time of the blocking of a certain job cannot be determined. In this case, the common practice is to judge the feasibility of a schedule using a graph representation. The Gantt chart and the graph of a schedule are shown in Fig. 2.

When considering the blocking of jobs, not all of the operating orders are able to be determined as parts of a semi-active schedule. This means that there are infeasible schedules. For example, Fig. 3 shows an infeasible exchange of
a schedule. The schedule is a 2-product by 2-machine flow shop schedule, and the number \( m-n \) in the rectangle representing the job means the \( n \)-th process of the \( m \)-th product. Also, the blocking of a job is represented as a shaded rectangle, in which the length of the rectangle is equal to the duration time of the blocking of a job.

In the schedule shown in Fig. 3, it is not an option to exchange either pair of jobs. There are only 2 machines, so machine 1 is only able to work on the second product after it has released the first product. But if the exchange shown in Fig. 3 occurs, machine 1 will work on the second product before it has released the first product. Therefore, the exchange is not allowed because the finishing time of blocking by job 1-1 cannot be determined in such a case.

As mentioned before, it is easier to judge the feasibility of a schedule by using a graph. Figure 4(a) shows the graph of the initial schedule, which is the same schedule as in Fig. 3. The disjunctive arc that will be exchanged is shown by the dot-dashed line. After the exchange is executed, the graph shown in Fig. 4(b) is obtained. When an exchange that causes an infeasible schedule occurs, a closed loop can be identified on the graph. The closed loop is shown in dashed lines. When a schedule is infeasible, the process of determining the time of jobs circulates, resulting in a closed loop on the graph. Although some closed loops are accepted due to swapping (Hino, et al., 2007), usually the existence of a closed loop can be used to determine that a schedule is infeasible. In this research, the graph is used to judge the feasibility of a schedule.

3. PROCEDURE TO GUARANTEE FEASIBILITY OF SCHEDULE

The idea of the proposed procedure is to exchange several pairs of jobs at once, in order to guarantee a feasible schedule. For example, Figure 5 shows the exchanges of two pairs of operations of a schedule for obtaining a feasible schedule. Figure 5(a) shows the same schedule as in Fig. 3, and Fig. 5(b) is the schedule after the exchange. Exchanging only one of the pairs results in an infeasible schedule, but when two exchanges occur simultaneously with one synchronizing with the other, a feasible schedule is obtained. The improvement procedure has been proposed using this idea.

Generally, an improvement of a schedule is made by exchanging the order of two consecutive operations. The problem is that when executing the exchange, there is a certain probability of obtaining an infeasible schedule. In this case, the selected two operations cannot be exchanged, and a different pair of operations must be selected instead. But when thinking about further objectives, there must be a way to exchange the two operations that are not able to be exchanged. Therefore, the proposed procedure is necessary to avoid a situation such as that just mentioned.

The steps of the procedure are as follows.

Step 1 Select two operations that are consecutive and exchange the operating order of the two and go on to Step 2.

Step 2 Search for a closed loop in the graph. If a closed loop cannot be identified, end the procedure. If a closed loop is identified, go on to Step 3.

Step 3 Beginning from the disjunctive arc that has caused the closed loop, exchange the last pair of operations in the closed loop. Return to Step 2.

The important part of the procedure is how to choose the pair of operations that needs to be exchanged in Step 3. Fig. 6 shows how the pair is chosen. Fig. 6(a) is an infeasible schedule represented by a graph. In this graph, the closed
loop causing the infeasibility is shown in a dashed line, and the disjunctive arc that has caused the closed loop is shown in a double-dashed line. Beginning from this disjunctive arc, the last pair of operations should be chosen to be exchanged, so the disjunctive arc connecting job 1-1 and job 2-1, which is shown in a dot-dashed line, is chosen. After the exchange is executed, the schedule shown in Fig. 6(b) is obtained, which is the same schedule as that shown in Fig. 5(b).

In this example, the closed loop consists of only two disjunctive arcs, so it may seem easy to choose the pair. Still, no matter how many disjunctive arcs are in the closed loop, the procedure to choose the pair to be exchanged is the same.

One point that needs to be minded is that there are two prohibitions to this rule, and they are listed as follows.

**Prohibition 1**
Selection of a disjunctive arc belonging to the same resource as the disjunctive arc selected just before.

**Prohibition 2**
Selection of a disjunctive arc that was selected before in the same steps.

Prohibition 1 must be satisfied; on the other hand, if there are no other disjunctive arcs that can be chosen, prohibition 2 is specially permitted. If a selection that applies to either of the prohibitions is made, then the next disjunctive arc is selected, i.e., the last disjunctive arc in the closed loop excluding the disjunctive arc that was just selected.

The proposed procedure allows the order of any two consecutive jobs to be exchanged, resulting in a feasible schedule. The steps of the procedure are shown as a Gantt chart and a graph in Fig. 7. Figure 7(a) is the initial schedule, and the first disjunctive arc to be exchanged is shown in a dot-dashed line. After the exchange, the schedule shown in Fig. 7(b) is obtained. But this schedule is infeasible because a closed loop can be identified on the graph, which is shown in dashed lines. In order to cancel the closed loop, the disjunctive arc shown in the dot-dashed line is selected newly, and the exchange is executed. After the exchange, the schedule shown in Fig. 7(c) is obtained. Still, a closed loop can be identified on the graph, so the disjunctive arc shown in the dot-dashed line is selected and exchanged. After the exchange is executed, the final schedule, shown in Fig. 7(d), is obtained as a feasible schedule.

As the example shows, even if the first exchange causes an infeasible schedule, the proposed procedure ultimately allows a feasible schedule to be obtained.

**Table 1** 5-product by 5-machine scheduling problem

<table>
<thead>
<tr>
<th>No</th>
<th>Machine (Processing time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4(4) 3(8) 2(6) 5(6) 1(6)</td>
</tr>
<tr>
<td>2</td>
<td>3(4) 4(2) 2(10) 1(8) 5(4)</td>
</tr>
<tr>
<td>3</td>
<td>3(8) 4(2) 1(2) 5(2) 2(6)</td>
</tr>
<tr>
<td>4</td>
<td>2(6) 3(10) 1(2) 4(6) 5(6)</td>
</tr>
<tr>
<td>5</td>
<td>2(4) 5(2) 4(4) 1(8) 3(2)</td>
</tr>
</tbody>
</table>

To have a better idea, a concrete example of an improvement of a schedule is shown in Fig. 8. A 5-product by 5-machine job shop scheduling problem has been generated randomly as shown in Table 1, and for the initial schedule the first-come-first-served rule has been applied as shown in Fig. 8(a).

Fig. 8(b) shows the schedule improved by the proposed procedure. The operating order of jobs of the schedule has been exchanged randomly so that the makespan, i.e., the finishing time of the last job, decreases. As a reference, the optimal schedule solved by a commercially available optimization software according to equations (1) - (4) is shown in Fig. 8(c). Although the optimal schedule is not obtained using the proposed procedure, a relatively suitable schedule can be obtained.
References


4. CONCLUSION

In this paper, a procedure was proposed to exchange several operating orders simultaneously, in which the exchanges are synchronizing with each other. A disjunctive graph with reverse conjunctive arcs was introduced to judge the feasibility of a schedule. According to the proposed procedure, the exchange of any two consecutive operations can be worked into a feasible schedule.