The Ninth International Symposium on Operations Research and Its Applications (ISORA'10)
Chengdu-Jiuzhaigou, China, August 19–23, 2010

Structural Properties for Job Shop Scheduling with Setups in a Local Search Environment

Udo Buscher* Liji Shen†

Chair of Industrial Management, Department of Business Administration and Economics,
Dresden University of Technology, 01062 Dresden, Germany

Abstract In this paper, we address the feasibility issue in the job shop scheduling problem involving sequence dependent setup times and reentrancy. Feasibility properties are first developed for metaheuristic search methods. In addition, we present a modified version of the standard labelling algorithm. An alternative algorithm is also developed which requires negligible computing time and can directly exclude infeasible moves.

Keywords Job shop scheduling; Sequence dependency; Metaheuristics.

1 Introduction

In this study, we focus on exploring feasibility properties for metaheuristic search methods in the job shop scheduling system, where sequence dependent setup times are involved. Applications of sequence dependent scheduling are encountered frequently in process industry operations, where a facility has to be shut down after completing a particular job, in order to adjust the facility to a desired state for processing the next job. This changeover delay and its magnitude depend on the similarity in technological requirements for the two consecutive jobs. In terms of Laguna [8], sequence dependency is, in fact, one of the most difficult aspects in scheduling area. Although the research on sequence-dependent scheduling is initiated decades ago [7], literature considering such problems in a job shop environment is rather limited. In respect to their strong practical relevance, job shop problems with sequence dependent setups thus need to be addressed adequately.

Regarding solution procedures for the job shop problem with sequence dependent setup times and makespan objective, [4] proposed a hybrid algorithm combining a genetic algorithm and heuristic rules. Computational analysis confirms that their hybrid algorithm is superior to methods developed earlier. [3] presented a mixed integer programming model and a local search scheme for the same problem. The latter utilizes a property to reduce computing time. By using benchmark data, [3] showed that the scheme significantly enhances the performance of several greedy-based dispatching rules. A fast tabu search heuristic for the underlying problem was first proposed by [1]. [2] presented

*buscher@rcs.urz.tu-dresden.de
†liji.shen@tu-dresden.de

a synthesis of the methods described in their earlier studies by integrating tabu search in multi-pass sampling heuristics.

In the subsequent section, we first introduce the symbols used in this paper and some important definitions. Properties and algorithms are then developed to exclude infeasible solutions.

2 Notations

Assume that a set of operations O is given. For each operation v, there is a job $j(v)$ to which it belongs, a machine $m(v)$ on which it must be processed, the corresponding processing time p_v and sequence dependent setup time s_{vw}, if v directly precedes w.

Furthermore, let $PJ(v)$ represent an operation to be processed prior to operation v in the job precedence relations, denoted by A. Similarly, $SJ(v)$ is a successor of operation v in the routing of job $j(v)$. For simplicity, $PJ(v)$ and $SJ(v)$ are referred to as *job predecessor* and *job successor* of operation v. More specifically, $pj(v)$ and $sj(v)$ are respectively the *immediate* job predecessor and successor of operation v. Symbol $pj^2(v)$ is equivalent to $pj(pj(v))$ and represents the immediate job predecessor of $pj(v)$. Symbol $sj^2(v)$ is defined analogously. In addition, let $pm(v)$ and $sm(v)$ denote the operations scheduled directly before and after v on machine $m(v)$.

Moreover, each operation v is associated with a release date (head) r_v indicating the earliest time to start its processing, and a delivery time (tail) q_v during which period the corresponding successors must remain in the system. Next, relevant definitions are introduced as follows [5, 6, 9].

Definition 2.1 (Adjacent operations)**.**
Two operations v and w are adjacent if the following conditions are satisfied:

$$\begin{aligned} r_w &= r_v + p_v + s_{vw} \\ q_v &= q_w + p_w + s_{vw}. \end{aligned}$$

According to the definition, every pair of consecutive operations on a critical path is adjacent.

Definition 2.2 (Block)**.**
A block consists of a maximum sequence of adjacent critical operations that are processed on the same machine.

3 Feasibility properties

While constructing a neighbourhood structure for job shops, the first concern is the possibly induced infeasible solution. It has been proved that for the standard job shop scheduling problem, swapping adjacent operations of the same block guarantees the feasibility of the new resulting solution. However, this property does not apply to the case including sequence dependent setup times [1].

Lemma 3.1.
Starting from a feasible solution, a permutation of two adjacent operations v and w of the same block may lead to an infeasible solution.

This can be proved by a counterexample as shown in figure 1. If setup times are not considered, the critical path contains all four operations, of which operations 1 and 4 belong to two independent blocks and operations 2 and 3 form one block. Moreover, operations 1 and 4 are not adjacent. As a result, the swap of these two operations is not performed. Conversely, as illustrated in the lower part of figure 1, operations 1 and 4 are grouped into the same block, and are adjacent in the case with sequence dependent setup times. Permuting operations 1 and 4 would create a cycle $(4,1,2,3,4)$, which implies an infeasible solution.

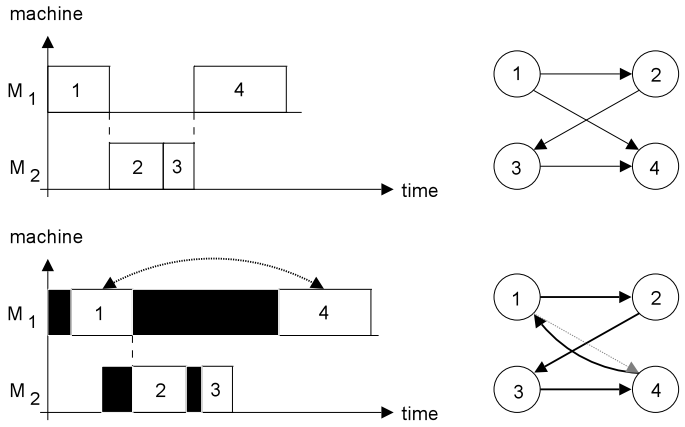

Figure 1: Example for illustrating lemma 3.1

Property 3.1.
Starting from a feasible solution, a permutation of two adjacent operations v and w of the same block leads to a feasible solution if

$$s_{vw} < p_{sj(v)} + p_{pj(w)} + \min\{s_{kl} | k \in O, l \in O\}. \tag{1}$$

Property 3.2.
Starting from a feasible solution, a permutation of two adjacent operations v and w of the same block leads to a feasible solution if

$$r_{pj(w)} \leq r_{sj(v)} + p_{sj(v)}. \tag{2}$$

Proof The permutation of v and w indicates removing arcs $(pm(v),v)$, (v,w), $(w,sm(w))$ and adding arcs $(pm(v),w),(w,v),(v,sm(w))$. Note that the current graph is acyclic. Thus, if the new graph contains cycles, they must be created by the newly added arcs. In other words, these arcs must be on the resulting cycles. As illustrated on the left side of figure 2, if a path from $sm(w)$ to v exists, it must also exist in the graph before the permutation, which results in a cycle (for example, $(v,w,sm(w),v)$). This obviously contradicts the assumption that the permutation starts from a feasible solution. The same argument holds for a path between w and $pm(v)$ which would imply a cycle in the current graph (for example, $(pm(v),v,w,pm(v))$).

Therefore, the only possibility for creating a cycle is that a path from v to w exists. In the current graph, as shown on the right side of figure 2, such a path must connect conjunctive arcs $(v, sj(v))$ and $(pj(w), w)$. Otherwise, there would be operations scheduled between v and w indicating they are not adjacent.

Regarding the new graph after reversing (v, w), note that each node has at most two outgoing arcs and the arc connecting v and w is removed. As illustrated by the dotted arcs on the right side of figure 2, the path cannot pass through the newly added arc $(v, sm(w))$, since this would lead to a cycle $(sm(w), pj(w), w, sm(w))$ in the current graph (In this context, the incoming arc to w can only pass through $pj(w)$ since arc (v, w) is removed and a path from $sm(w)$ to $pm(v)$ would induce a cycle $(sm(w), pm(v), v, w, sm(w))$ in the current graph). Hence, the path must connect v and $sj(v)$.

It can be proved in a similar manner that the path must go through $pj(w)$ instead of $pm(v)$.

Next, it is to show that the conditions stated in propositions 3.1 and 3.2 prevent such paths.

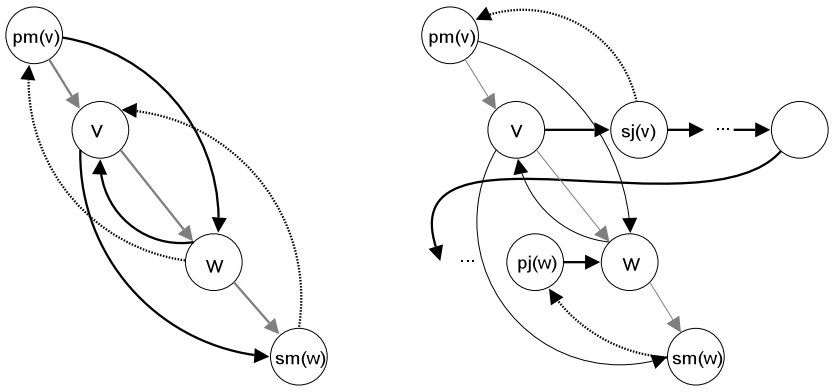

Figure 2: Illustration of propositions 3.1 and 3.2

1. Note that a resulting cycle contains at least two conjunctive arcs. We know that the conjunctive arc from v connects v and $sj(v)$, and the conjunctive arc terminating at w originates from $pj(w)$. If inequality (1) in proposition 3.1 holds, which requires that the length of s_{vw} must be smaller than at least the sum of $p_{sj(v)}$, $p_{pj(w)}$ and the smallest setup time, an alternate path from v, through $sj(v)$ and $pj(w)$ to w does not exist. Otherwise there would be a path $(0, \ldots, v, sj(v), \ldots, pj(w), w, \ldots, *)$ longer than $(0, \ldots, v, w, \ldots, *)$, indicating that v and w are neither critical nor adjacent. This again contradicts the assumption.
2. A path from $sj(v)$ to $pj(w)$ indicates that

$$r_{sj(v)} + p_{sj(v)} < r_{sj(v)} + p_{sj(v)} + s \leq r_{pj(w)},$$

where s denotes an unspecified setup time. If operations $sj(v)$ and $pj(w)$ are to be processed on the same machine, then $s \neq 0$. On the other hand, the path connecting

$sj(v)$ and $pj(w)$ must pass through other job successor(s) of v or job predecessor(s) of w. Obviously, in either situation, setup times on the path connecting $sj(v)$ and $pj(w)$ are greater than 0. Therefore, $r_{sj(v)} + p_{sj(v)} < r_{sj(v)} + p_{sj(v)} + s \leq r_{pj(w)}$ is valid. This is a contradiction of inequality (2) given in proposition 3.2.■

Recall the example introduced to illustrate lemma 3.1 where an infeasible solution would be induced by swapping operations 1 and 4. As shown on the left side of figure 3, job predecessor of operation 4 (operation 3) must be processed before operation 2 in the current schedule if the setup time satisfies inequality (1). Otherwise, operations 4 and 1 would not be adjacent. Swapping these two operations thus leads to a feasible schedule. Concerning inequality (2), no infeasible solution occurs after the permutation as long as operation 3 is released prior to operation 2. In order to ensure feasibility, proposition 3.1

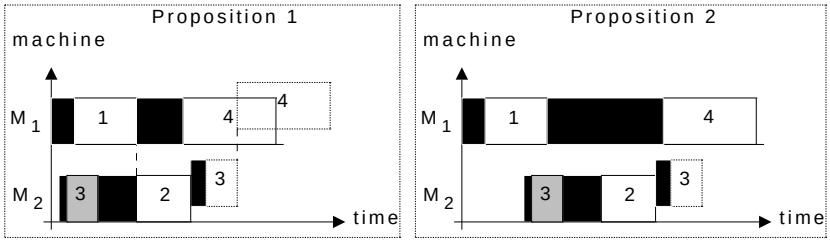

Figure 3: Example for illustrating propositions 3.1 and 3.2

specifies the value of setup times. Proposition 3.2 provides another important condition, according to which, a permutation may still be feasible even if the setup time exceeds the value determined by (1). Therefore, proposition 3.2 complements the first one, and it actually represents another perspective to identify infeasible solutions.

In addition, the next lemma follows immediately.

Lemma 3.2.
Proposition 3.2 applies to job shop problems with reentrancy.

Job shop problems with reentracy indicate that a job may pass through the same machine more than once. In this context, proposition 3.2 actually prevents all alternative paths from v to w that contain operations assigned to different reentrant levels. Therefore, this proposition is also valid for reentrant job shop settings.

4 Algorithm for excluding infeasibilities

Propositions 3.1 and 3.2 can be directly utilized to ensure feasibility, which requires only constant and negligible computing time. However, it should also be noted that they are sufficient but not necessary conditions. Feasible solutions may therefore be discarded as well. For example, $pj^2(w)$ and $sj^2(v)$ are processed on the same machine (connected by a disjunctive arc) in figure 4. Thus, no cycles would be created by reversing arc (v,w) as long as $r_{pj^2(w)} \leq r_{sj^2(v)} + p_{sj^2(v)}$ holds. In comparison to inequality (2), we have

$$r_{pj^2(w)} < r_{pj(w)} \leq r_{sj(v)} + p_{sj(v)} < r_{sj^2(v)} + p_{sj^2(v)}.$$

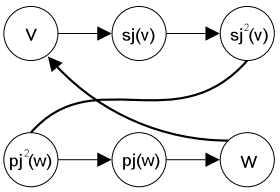

Figure 4: Example ($pj^2(w)$ and $sj^2(v)$ connected by a disjunctive arc)

Obviously, proposition 3.2 represents rather strict conditions.

Generally, a so-called *labelling algorithm* can be applied to identify infeasible schedules in addition to determining makespan. First, we present a modified version of the standard labelling algorithm in [10], where sequence dependent setup times are present. More specifically, we focus on anticipatory setup indicating that setup can be carried out in the absence of the corresponding operation/job. Hence, $r(u)$ and $q(u)$ can be calculated according to the following equations:

$$r(u) = \max\{r(pj(u)+p(pj(u)); r(pm(u))+p(pm(u))+s(pm(u),u)\}, \quad (3)$$

$$q(u) = \max\{q(sj(u)+p(sj(u)); q(sm(u))+p(sm(u))+s(u,sm(u))\}. \quad (4)$$

It is obvious that the processing of operation u can only start upon the completion of both $pj(u)$ and $pm(u)$. Analogously, $q(u)$ is determined by its job and machine successors. In comparison to the conventional determination, sequence dependent setup times are incorporated. Regarding anticipatory setup times, setup of an operation may directly follow the associated preceding operation on the same machine. Therefore, the setup calculation is included in the term with $r(pm(u))$. In other words, $pj(u), u$ and $sj(u)$ may be processed without interruption. Consequently, the length of a longest path containing operation u is written as (denoted by $\mathscr{L}_u^*$)

$$\mathscr{L}_u^* = r(u)+p(u)+q(u). \quad (5)$$

Furthermore, the makespan of a given schedule is determined by

$$C_{max} = \max_u\{\mathscr{L}_u^*\}. \quad (6)$$

The procedure of the modified labelling algorithm AMLAB is presented below:

Step 1.	Group all operations that have no job and machine predecessor into set $\mathscr{R}$.
Step 2.	Select an operation $u \in \mathscr{R}$, calculate $r(u)$ according to equation (3).
Step 3.	If operation $pm(sj(u))$ is labelled or does not exist, then $\mathscr{R} = \mathscr{R} \cup sj(u)$.
Step 4.	If operation $pj(sm(u))$ is labelled or does not exist, then $\mathscr{R} = \mathscr{R} \cup sm(u)$.
Step 5.	Label operation u and remove it from set $\mathscr{R}$.
Step 6.	If $\mathscr{R} = \oslash$ then Stop. Otherwise goto Step 2.

Values of q are analogously determined starting with operations without machine or job successors. Infeasibilities occur if set $\mathscr{R} = \oslash$ and not all operations are labelled. Obviously, the modified labelling algorithm requires $O(mn)$ time, where m and n are respectively the total number of machines and jobs.

In order to preserve feasible solutions, we propose an alternative algorithm AFEA as follows:

Step 1.	Arrange all job successors $SJ(v)$ according to the corresponding machine index.
Step 2.	Arrange all job predecessors $PJ(w)$ according to the corresponding machine index.
Step 3.	Set machine $k = 1$.
Step 4.	If on machine k, the condition $r_{PJ(w)} > r_{SJ(v)} + p_{SJ(v)}$ is valid, then goto step 6, otherwise set $k = k+1$. In the case of reentrancy, divide all operations processed on the same machine into two groups and compare the values of $\min\limits_{PJ(w)} \left\{ r_{PJ(w)} \right\}$ and $\max\limits_{SJ(v)} \left\{ r_{SJ(v)} + p_{SJ(v)} \right\}$.
Step 5.	If $k \leq M$, then goto step 4, otherwise goto step 7.
Step 6.	Print: Infeasible move and Stop.
Step 7.	Print: Feasible move and Stop.

Based on proposition 3.2, this algorithm examines the sequence of operations relating to v and w on each individual machine. Consequently, a move is labelled 'infeasible' as long as an alternative path from v to w exists. Thus, this algorithm conducts a preliminary feasibility test before a move is actually performed.

Generally, it is essential to alleviate computational burden in a complex metaheuristic implementation. In this respect, excluding infeasible moves actually enables the investigation of a broader solution space, without redirecting the search process. If, on the other hand, a transformation algorithm potentially interrupts intensified examination in promising spaces, search process can hardly be controlled using integrated components such as tabu list or fitness value. This will, however, reduce the effectiveness of metaheuristics. The algorithms proposed here not only distinguishes feasible moves in advance, but also preserves the original structure of metaheuristics by avoiding erratic transformations. Compared to algorithm AMLAB, AFEA is solvable in only $O(M)$ time, where M denotes the total number of machines.

5 Conclusions

In this paper, we investigate the feasibility issue in a job shop environment involving sequence dependent setup times and reentrancy. The properties presented are especially applicable when constructing local search algorithms, such as tabu search and simulated annealing. In order to preserve feasible solutions, a modified version of the standard labelling algorithm is first presented. We also develop an algorithm which requires negligible computing time and can directly exclude infeasible moves.

References

[1] C. Artigues and F. Buscaylet. A fast tabu search method for the job-shop problem with sequence-dependent setup times. *Proceedings of Metaheuristic International Conference*, Kyoto, Japan, 2003.

[2] C. Artigues and P. Lopez and P. Ayache. Scheduling generation schemes for the job-shop problem with sequence-dependent setup times: Dominance properties and computational analysis. *Annals of Operations Research*, 138: 21–52, 2005.

[3] I. Choi and D. Choi. A local search algorithm for jobs scheduling problems with alternative operations and sequence-dependent setups. *Computers and Industrial Engineering*, 42: 43–58, 2002.

[4] W. Cheung, and H. Zhou. Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. *Annals of Operations Research*, 107: 65–81, 2001.

[5] S. Dauzère-Pérès and J. Paulli. An integrated approach for modelling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70: 281–306, 1997.

[6] S. Dauzère-Pérès and W. Roux and J. Lasserre. Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, 107: 289–305, 1998.

[7] J. Gupta. Economic aspects of scheduling theory. Ph.D thesis, Texas Tech University, Lubbock, Texas, 1969.

[8] M. Laguna. A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. *IIE Transactions*, 31: 125–134,1999.

[9] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6): 797–813, 1996.

[10] E. Taillard. Parallel taboo search techniques for the job shop scheduling problem *ORSA Journal on Computing*, 6: 108–117,1994.