Scheduling Problem of Job-Shop with Blocking: A Taboo Search Approach

Yazid Mati[*] *Nidhal Rezg*[*] *Xiaolan Xie*[*]

[*] INRIA/MACSI Project & LGIPM
ENIM-ILE DU SAULCY, 57045 Metz, France
Email: {ymati, nrezg, xie}@loria.fr

1 Introduction

In the job shop scheduling problem (JSP), a set of N jobs $\Im = \{J_1, J_2, \ldots, J_N\}$ is to be processed on a set of m machines $M = \{1, 2, \ldots, m\}$. Each job requires a sequence of operations that must be accomplished according to its manufacturing process $J_i = \{O_{i1}, O_{i2}, \ldots, O_{im}\}$. Operation O_{ij} requires a processing time of p_{ij} time units on machine $M_{ij} \in M$. Each machine can perform at most one operation at a time, and an operation must be accomplished without interruption. The job shop scheduling problem is concerned with the allocation of the set of machines in order to minimize the makespan. It assumes that the buffer space between any two machines is infinite. So, a machine will be immediately available for processing the next operation after the current operation is completed.

While researchers have concentrated on the resolution of the traditional JSP, new technologies in the context of automated manufacturing systems have been introduced. In fact, modern manufacturing companies tend to design production lines without buffers in order to limit the accumulation of inventory, which can be expensive. As a consequence of the lack of buffer space, a machine cannot release a completed part, and must hold it until the next machine on the routing becomes available. This constraint is known as the *blocking* or *hold while wait* constraint. It is well known that this constraint and the no preemption assumption lead to undesirable system deadlock if the machines are not adequately managed [2]. A system deadlock is a situation when the parts are assigned to various machines in a manufacturing system such that any further part move is inhibited [3].

A survey on the blocking and no-wait problems was presented in [5]. In this survey, applications of the blocking and no-wait problems are presented as well as results on the complexity of the problems under such constraints. In [8] a real-life automated manufacturing workstation with three machines and a robot to handle the parts between the machines was investigated. More related to our paper is the working paper of [6]. The authors proposed a general disjunctive graph model for modeling problems with more general constraints such as blocking and no-wait constraints. They proposed four dispatching rules to obtain feasible solutions, and implemented a branch and bound method based on the ideas of the classical job shop.

In this paper, we simultaneously deal with the scheduling and deadlock prevention problems of the job shop under the blocking constraint. The related model will be called the Job Shop Problem with Blocking or JSPB for short. For sake of convenience, a fictitious operation is added at the end of the manufacturing process of each job. Such operation will be used in Section 2 to construct the disjunctive graph with blocking. In Section 2, the disjunctive graph that allows modeling the JSPB is presented. Section 3 describes the neighborhood structure developed. In Section 4, we present a taboo search heuristic to explore the solution space. Finally, Section 5 reports the computational results.

2 Disjunctive graph model

In this section, the disjunctive graph model that we will use as a basis for developing the taboo search heuristic is described. It is an extension of the classical disjunctive graph model $G = (N, A, E)$[9] in order to take into account the blocking constraint.

In the classical representation for the job shop problem, each vertex $v \in N$ is associated with an operation. Further, two vertices 0 and $\star$, which are respectively a source vertex and a sink vertex and correspond to the beginning and the end of all operations are added. There are two types of arcs: conjunctive arcs and disjunctive arcs. The set A of conjunctive arcs consists of all arcs that connect any two consecutive operations of the same job and ensures the precedence constraints. Any two operations sharing common resources are connected by a disjunctive arc. Vertex 0 is connected via a conjunctive arc to all initial operation of each job and the final operation of each job is connected to the sink $\star$ via a conjunctive arc. Each arc is weighted by the processing time of the operation corresponding to its initial vertex. By convention, the fictitious operations 0 and $\star$ have zero process time. A feasible solution is determined by choosing a direction for each disjunctive arc such that the resulting graph is acyclic i.e. circuit-free. The makespan of the resulting solution is given by the length of the longest path from the source vertex 0 to the sink vertex $\star$, called the critical path.

To deal with the blocking constraint, each disjunctive arc of the classical representation is replaced by a disjunctive couple. More precisely, for each pair of operation O_{ik} and $O_{i'k'}$ sharing common machine, we introduce a disjunctive couple of two arcs: one from vertex $O_{i\,k+1}$ to $O_{i'\,k'}$ and another one from $O_{i'\,k'+1}$ to O_{ik}. The weight of the disjunctive couples is zero, which means that the machine held is released just after the progression to the next operation on the routing. A feasible solution is obtained by selecting an arc for each disjunctive couple such that the resulting graph is circuit-free. Figure 1 and Figure 2 show the disjunctive arcs for problems without and with blocking.

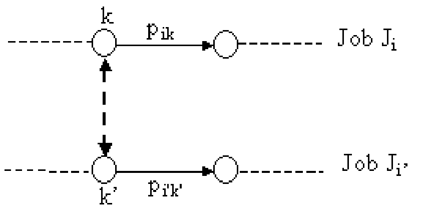

Figure 1. Classical representation for JSP

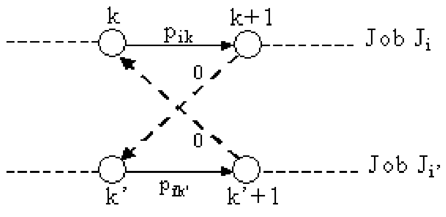

Figure 2. disjunctive graph for JSPB

3 Neighborhood structure

The definition of the neighborhood structure is a key element in the success of local search heuristic. A neighborhood must be not too large for making it simple to evaluate and must ensure that the connectivity property is verified. For the classical job shop problem, several connected neighborhoods were proposed. The most efficient of them are based on the classical permutation move on the critical path.

In contrast to the JSP case, in our model, the permutation that consists of dropping the first disjunctive arc and replacing it by the second one may not be feasible. Worse, there exist examples in which all permutations along the critical path lead to deadlock situations. In our application, the feasibility of the obtained solution is tested using an exploration in-depth starting from vertex O_{ik}. Three stopping criteria based on the notion of layer and longest path are used to avoid exploration of vertices that do not lead to vertex $O_{i'k'}$.

3.1 Methodology

To deal with the problem of the connectivity property, the following methodology is adopted. First, we apply the classical permutation move until no permutation on the critical path leads to a feasible solution. At this stage, a deadlock recovery move (O_{ik}, $O_{i'k'}$) that allows getting out from this situation is employed. The deadlock recovery move is a rescheduling algorithm, which uses an extension of the classical geometric approach [1] proposed in [7]. The main idea of the rescheduling algorithm is to reschedule job J_i without completely altering the structure of the current solution. This is realized by keeping for all jobs $\Im \setminus J_i$, the same input sequence into machines i.e. maintain the same selection of disjunctive couple for jobs $\Im \setminus J_i$, and next, scheduling operations of job J_i by enforcing constraint expressed by the permutation (O_{ik}, $O_{i'k'}$).

Having fixed the same disjunctive arcs for all other jobs $\Im \setminus J_i$, a composed job J_{com} can be constructed by combining operations of jobs $\Im \setminus J_i$ that may be executed at the same time into a same operation. As a result, the deadlock recovery move becomes the scheduling problem of two jobs J_{com} and J_i under the precedence constraint imposed by permutation, i.e. $O_{com,u} > O_{ik}$ where $O_{com,u}$ corresponds to the end of $O_{i'k'}$.

3.2 Steps of the recovery move

The steps of the deadlock recovery algorithm are explained on the following example:

$$\begin{aligned} J_1 &= (M_1,1), (M_2,3), (\phi,0) \\ J_2 &= (M_2,3), (M_3,2), (\phi,0) \\ J_3 &= (M_1,1), (M_3,2), (\phi,0) \end{aligned}$$

A feasible selection is represented in Figure 3. The bold path in Figure 3 gives the critical path, which contains two disjunctive arcs (O_{13}, O_{21}) and (O_{23}, O_{31}). The permutation of the first disjunctive arc consists in deleting arc (O_{13}, O_{21}) and replacing it by the second arc (O_{22}, O_{12}). Clearly, this move is not feasible since it leads to a circuit $O_{12}O_{22}O_{12}$ with a length equal to zero. This circuit corresponds to deadlock situation of state O_{11} and O_{21}. Similarly, the permutation of the second disjunctive arc on the critical path leads to an unfeasible solution. Therefore, the deadlock recovery move must be used.

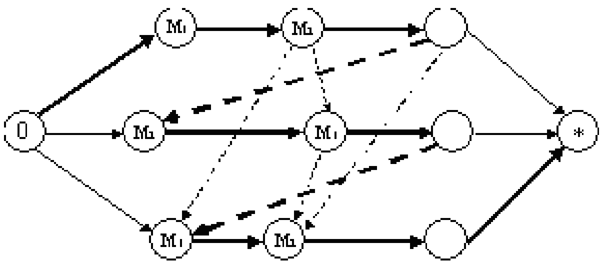

Figure 3. An initial selection of disjunctive couples

The first step is the construction of the combined job J_{com} of jobs J_2 and J_3. The combined job J_{com} is constructed in four steps:

1. Deriving disjunctive graph of jobs J_2 and J_3 (see Figure 4);
2. Determining the corresponding schedule using critical path approach. The Gantt diagram is given in Figure 5;

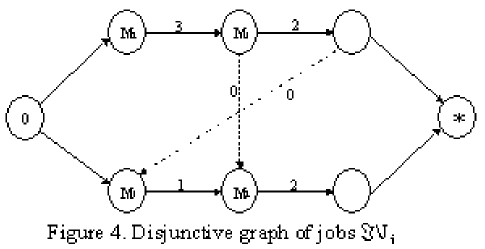

Figure 4. Disjunctive graph of jobs ℑVᵢ

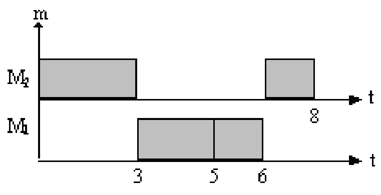

Figure 5. Gantt diagram of jobs in ℑVᵢ

3. Decomposing the time axis into sub-intervals such that no operation starts/terminates inside a sub-interval. For the illustrative example, four sub-intervals are obtained: $[0, 3], [3, 5], [5, 6], [6, 8]$;
4. Associating with each sub-interval an operation of the composed job with the machines occupied during the sub-interval as the machine requirement of the operation. For the illustrative example, $J_{com} = \{(M_2,3), (M_1,2), (M_1,1), (M_2,2)\}$.

The rescheduling problem consists in scheduling two jobs J_{com} and $J_1 = \{(M_1,1), (M_2,2), (\phi,2)\}$ under the additional precedence constraint imposed by permutation: $O_{com} = O_{22} \succ O_{12}$. This problem is solved using an extension of the classical geometric approach [7] in order to take into account the precedence constraint, the characteristics of the combined job and the blocking constraint. The extended approach is summarized as follows:

1. Represent the jobs on a plane (see Figure 6). Each job is represented along an axe according to its manufacturing process and processing time. J_{com} is put on axe X and J_1 is put on axe Y;
2. Create an obstacle for each rectangle ($O_{com,k'}$, $O_{ik'}$) such that O_{ik} and $O_{com,k'}$ share common resources. Obstacles ($O_{com,2}$,O_{11}), ($O_{com,3}$,O_{11}), ($O_{com,1}$,O_{12}) and ($O_{com,4}$,O_{12}) are created for the illustrative example;
3. Create additional obstacle (O_{ik},$O_{com,k'}$) if starting with operations O_{ik} and $O_{com,k'}$ leads to deadlock whatever the schedule of remaining operations. For the illustrative example, obstacle ($O_{com,1}$, O_{11}) is created. Clearly, if job J_{com} is on the first operation, it cannot release machine M_2 until machine M_1 becomes available. On the other hand, J_1 cannot release M_1 until M_2 becomes available. These two operations are embedded in a circular-wait phenomenon. The state ($O_{com,1}$, O_{12}) defines a deadlock state, and it must be forbidden;
4. To impose precedence relation $O_{com,u} \succ O_{ik}$, create a horizontal line obstacle along the bottom line of operation O_{ik} starting from $X = 0^-$ and terminating at right boundary of operation $O_{com,u}$. For the illustrative example, the new horizontal line obstacle is represented by a thick line in Figure 6.

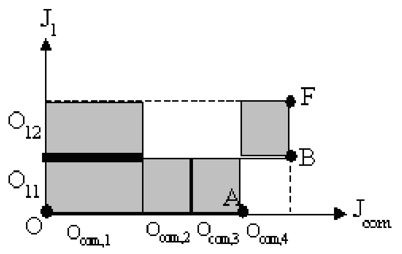

Figure 6. The geometric representation

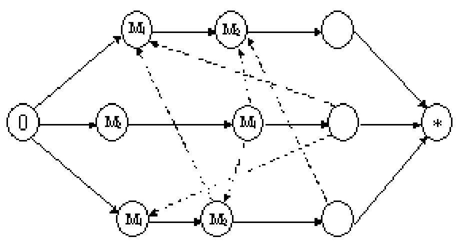

Figure 7. Result of deadlock recovery

Having obtained the geometric representation of the rescheduling problem, the two-job problem becomes a shortest path problem from the origin O of the plane to point F by going either horizontally, vertically or diagonally while avoiding obstacles. Different from geometric approach for the classical job-shops, in our case, the upper boundary and the right boundary of any obstacle is forbidden as a result of *hold while wait* constraint. The shortest path problem on a plane can be transformed into a shortest path problem of a graph. Detailed of the method can be found in [7]. For the illustrative example, the only feasible path is OABF. The final schedule is represented by the disjunctive graph of Figure 7.

4 Taboo search heuristic

We assume that the reader is familiar with the taboo search heuristic. Otherwise, we refer to [4] for a detailed description of the approach.

For our application, the taboo search is adopted with the following parameters. The initial solution is derived using a constructive algorithm. The algorithm begins by grouping the set of jobs into pairs $P_1, P_2, \ldots, P_{nbr}$. Each pair is then solved optimally using the polynomial algorithm proposed in [7]. Based on the obtained schedules, the disjunctive couples are fixed for each pair. The remaining disjunctive couples between pairs are fixed in a naive manner by scheduling all operations of a pair P_i before operations of pair P_j, $i < j$, on common machines. This construction makes the disjunctive graph circuit-free and hence the initial solution is feasible.

The neighborhood structure is based on the critical paths. Given a solution, a critical path is obtained, and all the permutations of the disjunctive couples on this path are performed. If there is no feasible permutation, the deadlock recovery move that allows getting out from the current solution is used. The deadlock recovery move is evaluated using the rescheduling algorithm proposed in Section 3.

Two taboo restrictions are defined; each of themes corresponds to a move. For the permutation move $(O_{ik}, O_{i'k'})$, the reverse disjunctive arc $(O_{i,k'+1}, O_{i',k'})$ is added to the list, forbidding it to be candidate for selection in the next T_p iterations. For recovery move related to permutation $(O_{ik}, O_{i',k'})$, which imposes precedence relation $O_{i',k'} \succ O_{ik}$, (i, i') is added to the taboo list. As a result, it forbids any recovery move $(O_{i',k'}, O_{ik})$, which would impose precedence relation $O_{il} \succ O_{i',l'}$ during the next T_r iterations. The taboo list sizes T_p and T_r are generated randomly in $[N/2, N]$ and $[N/3,\ 2N/3]$, respectively. These sizes are changed every $2N$ iterations. Finally, the well known global aspiration criterion that overrides the taboo status of a permutation $(O_{ik}, O_{i',k'})$ whenever moving it result in a new best solution is adopted.

To overcome the problem of the connectivity property, an intensification strategy is employed. This strategy restarts the taboo search algorithm from the K best solutions recorded in order to obtain new search directions. The value of K is fixed experimentally to 20. Before restarting from a recorded solution the taboo lists are emptied and the sizes are updated.

5 Computational results

Extensive computational tests were performed on a set of benchmarks, which generalize the well-known instances for the classical job shop. These benchmarks are obtained by dropping the infinite buffer space constraint, and replacing it by zero buffer space. Our results are compared with the dispatching rules and the branch and bound approach developed in [6], which are to our knowledge the only existing approaches that are tested on such problems.

The first interesting observation is that compared to the optimal solutions, the standard deviation of our solutions are not very large and varies in the range [3%,9%]. The second observation is that the scheduling algorithm improves solutions given by the best dispatching rule in lower computation time that note exceed 1 CPU time for all problems.

The results obtained by our heuristic on these benchmarks reveal its efficiency to obtain good schedules in relatively short execution time, and confirm the efficiency of the taboo search parameters, in particular the neighborhood structure. Furthermore, the success of using the disjunctive graph and local search heuristics for the classical shop problem is extended for more general case of the JSBP.

References

[1] Brucker, P. An Efficient Algorithm for the Job-Shop Problem with Two Jobs. *Computing*, 40:353–359, 1988.

[2] Coffman, E., G., M. Elphiek, and A. Shoshani. System Deadlocks. *Computing surveys*, 3:67–78, 1971.

[3] Chu, F. and X.L. Xie. Deadlock analysis of Petri nets using siphons and mathematical programming. *IEEE Transactions on Robotics and Automation*, 13:793–804, 1997.

[4] Glover, F. Tabu Search Fundamentals and Uses. *Revised and Expanded, Technical Report, Graduate School of Business, University of Colorado, Bolder, CO*, 1995.

[5] Hall, N.G. and C. Sriskandarajah. A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Oper. Res*, 44:510–525, 1996.

[6] Mascis ,A., and D. Pacciarelli. Machine Scheduling via Alternative Graphs. *Research Report, RT-DIA-46-2000, Italy*, 2000.

[7] Mati, Y., N. Rezg and X.L. Xie. Geometric Approach and Taboo Search for Scheduling Flexible Manufacturing Systems. *submitted to IEEE Transactions on Robotics and Automation. Also available as a Research Report, INRIA, France*, 2000.

[8] Ramaswamy, S.E. and S.B. Joshi. Deadlock-free schedules for automated manufacturing workstations. *IEEE Transactions on Robotics and Automation*, 12, 391–400, 1996.

[9] Roy, B., and B. Sussman. Les problèmes d'ordonnacement avec contraintes disjonctives. *Note DS N° 9 bis, SEMA, Montrouge*, 1964.