Journal of Theoretical and Applied Information Technology
31st May 2018. Vol.96. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SCHEDULING FOR JOB SHOP PROBLEMS WITH TRANSPORTATION AND BLOCKING NO-WAIT CONSTRAINTS

[1]SAAD LOUAQAD, [2]OULAID KAMACH, [3]ADIL IGUIDER

[1]Laboratoire des technologies innovantes, ENSAT, Abdelmalek Essadi University, Tanger, Maroc

[2] Laboratoire des technologies innovantes, ENSAT, Abdelmalek Essadi University, Tanger, Maroc

[3] Institut Nationale des Postes et Télécommunications, Rabat, Maroc

E-mail: [1] louaqad@gmail.com, [2] kamach@ensat.ac.ma, [3] adil.iguider@gmail.com,

ABSTRACT

The problem studied in this paper is the Transportation job shop problem with blocking and no wait constraints. This problem is an extension of the classical job shop problem that take into account transport operations and the absence of storage space between machines. We formulate the problem by means of a new disjunctive graph. We modelized blocking situations and described properties of partial schedule that lead to deadlock situations. The new disjunctive graph and deadlock situations properties served us to develop a greedy heuristics based on priority and avoiding blocking cycle rules. We also propose to resolve our problem by an exact method based on a Mixed Integer Linear Program (MILP). Computational results for a set of benchmarking tests are reported and the effectiveness of our methods are discussed.

Keywords: *Job Shop Scheduling problem, Transport, Blocking No Wait Constraint, Mixed Integer Linear Program, Heuristics*

1. INTRODUCTION

The classical job-shop problem is known as a standard problem in scheduling and has been widely investigated over the last few decades. However, in real world, several problems often cannot be modeled as a classical job-shop problem, due to additional features. This is especially the case in flexible manufacturing systems where the model set by researchers have to take into account some aspects such as material handling, storage space etc.

The jobs shop problems with transportation, blocking and no wait constraints (NWBT JSSP) are met for example in factories with robotic cells. A robotic cell is a flow-shop or job-shop scheduling cell in which the jobs are transported from machine to machine by one or more robots. %We have to assign the transport operations to the robots and to schedule both the machine and robot operations.

In some kind of production systems, there is no buffer between machines and jobs must be conducted from one machine to another one with no interruption, so we have to deal with blocking no wait constraints.

Applications related to the Job shop problems with blocking constraints (BJSSP) have been reported in the processing and logistics industries, such as scheduling for the manufacturing of concrete blocks by [12], steelmaking by [13], chemical batch production by [14], container handling at a port by [15] and railway networks by [16]. [3] describes several applications of machine scheduling with blocking and no wait in process and reviews the computational complexity of a variety of related problems.

Several researchers studied the BJSSP and the No Wait Job Shop Scheduling Problem (NWJSSP). [4] and [5] formulate these problems by means of alternative graphs. [7] develop a genetic algorithm for solving no-wait and Blocking Job Shop problems (NWB JSSP) and [8] and [9] introduce a local search approach for the generalized Blocking Job Shop problem with application in automated warehouses. [6] study a multi-resource job shop problem with blocking constraints. [10] propose a tabu search algorithm to solve the BJSSP for cyclical scheduling. [11] proposes a combination of a branch and bound algorithm with alternative graphs and develops two methods based on genetic algorithms to solve the BJSSP.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Several researchers have devoted to study job shop scheduling problems with transportation constraints (TJSSP) in various systems. However, the progress is limited as this kind of problem is difficult to solve even for simplified and small size cases. [19] developed a mixed integer programming (MIP)formulation raising this constraint on the vehicles. [29] used a mixed integer linear program (MILP) to find optimal solutions for the Flexible Manufacturing Systems Scheduling Problem with one vehicle. [31] studied coupled task problem and one-machine robotic cell problems. It reported new algorithmic procedure for this problem with or without tolerances on the distance. [32] applied a decomposition method where the master problem (scheduling) is modeled with constraint programming and the subproblem (conflict free routing) with mixed integer programming. [30] proposed a polynomial algorithm for finding the optimal cycle in a robotic cell with production of identical parts. [17] integrated transport constraints in the scheduling problem with one robot. [28] considered a cyclic hoist scheduling problem with a single hoist, but without assignment problem. [18] proposed a dynamic programming approach to construct optimal machine and vehicle schedules. [20] and [21] elaborated a genetic algorithm. [22] and [23] proposed, respectively, neural networks and tabu search approaches. [24] described a hybrid method composed of a genetic algorithm for the scheduling of machines and a heuristic for the scheduling of vehicles. [25] and [26] considered a job shop problem with several robots, with fixed operation times andfixed assignment of machine for each job's operation. [27] studied a two machines flow shop scheduling problem with intermediate transportation with a single transporter.

To the best of our Knowledge there is no research that addressed the problem of job shop scheduling that take into account transfer time between machine performed by a limited number of robots and the absence of buffers between machines that lead to blocking and no wait constraints.

Two common approaches to tackle the Job Shop scheduling problems with additional constraints are the utilization of exact methods and heuristic approaches [1] and [33]. The exact methods allow us to find optimal solutions but are not suitable for large instances. Heuristic approaches allow us to find a good quality approached solutions for a large size instances in a reasonable calculating time.

In this paper, we propose a Mixed Integer Linear program based on the model of [34] to find exact solutions and a construction heuristics based on priority rules and preventing deadlock situations during the construction of partial schedules. Scheduling problems with blocking constraints appear more difficult to solve by heuristics than the classical job shop. This is due, to the fact that a feasible partial schedule for BJSSP cannot always be extended to a feasible complete schedule [4] and [5]. As a consequence, any heuristic that incrementally builds up a solution (e.g. based on priority rules) risks the chance of running into infeasibility. Therefore, the effectiveness of any greedy heuristic is evaluated based on its capacity to find good quality solution and to reach feasible schedules.

This paper is structured as follows. In the next section, we will define our problem and the notations associated to its formulation. After, in Section 3, we present the disjunctive graph representation for the NWBT JSSP. In Section 4, we describe our Mixed Integer Linear Program for the NWBT JSSP. In section 5, we describe and illustrate blocking cycles and present our two developed heuristics.

Section 6 discusses a series of experimental tests and computational results. Conclusion follows in section 7.

2. PROBLEM DEFINITION

We consider a job shop problem with several transport robots and no buffers. In this problem, a set of n jobs $\{J_1, J_2, \ldots, J_n\}$ are processed on a set of m machines $\{M_1, M_2, \ldots, M_m\}$ and transported by a set of k $\{r_1, r_2, \ldots, r_k\}$. Transportation times are robot-independent. Every job J_i require an operation order, $\{J_i=\{O_{i1}, O_{i2} \ldots, O_{in_i}\}\}$, that must be executed according to its manufacture process. Operation O_{ij} of the job J_i requires the exclusive use of $M_l(1 \in \{1,2,\ldots m\})$ for an uninterrupted duration p_{ij}, its processing time; the preemption is not allowed; each machine can process only one job at a time; the machine which execute the operation O_{ij} is denoted as M_{ij}. In addition, we consider transportation operations between two machines. Consider two successive operations of the same job O_{ij} and O_{ij+1} to execute in two machines M_{ij} and M_{ij+1}. T_{ij} is used to denote transport operation of job J_i from machine M_{ij} to machine M_{ij+1}. Each robot can handle at most one job at one time. Loaded transfer times do not depend on the job transported, but only on the travel routes and the robot which perform the transportation operation. These times are given by
C^r_{pl} where r represents the robot and p,l represents the route between machine M_p and M_l. It is assumed that the triangle inequality is satisfied:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$\forall \{p, l, h\} \in \{1,2,..,m\}$ machine indexes.
$\forall r \in \{r_1, r_2, .., r_k\}$

$$C^r_{ph} + C^r_{hl} \geq C^r_{pl} \quad (1)$$

(1) means that the direct way between two machines is at least as short as the detour through a third machine. Otherwise, the robot always takes the shorter way through the third machine.

Note that a sequence of loaded transport operations indirectly induces necessary empty moves. Empty transfer time from machine M_p to M_l is denoted V^r_{pl}. It is assumed that:
$\forall \{p, l\} \in \{1,2,..,m\}$ machine indexes.
$\forall \{r, r'\} \in \{r_1, r_2, .., r_k\}$

$$\begin{cases} V^r_{pp} = 0 \\ V^r_{ph} + V^r_{hl} \geq V^r_{pl} \\ C^r_{pl} \geq V^r_{pl} \end{cases} \quad (2)$$

The first assumption means that no empty transfer time is considered if a robot waits at the same machine the next transportation operation. The second one is the triangular inequality for empty moves. The third one means that empty transfers between two machines by a robot r take less time than loaded transfers between this two machines by another robot r'. (It is also valid if $r = r'$). In the other hand, we consider the blocking constraint because there is no machine buffer. This means that after finishing its processing on the machine, a job has to stay there until it is unloaded by the robot. During this stay, the machine is blocked and not available for processing any other job. We also consider the no wait constraint that means if the robot transporting the job J_i reaches machine M_{ij}, the operation O_{ij} must start immediately without any interruption.
We distinguish two cases of blocking: blocking with swap allowed and blocking no swap [37]. In our case, we consider that the job can move independently and therefore the swaps are allowed. It means that whenever there is a job J_i in Machine M_l and a Job $J_{i'}$ in a Robot r_s, each one waiting for another to be freed as the job J_i has to be transferred from Machine M_l to robot r_s and job $J_{i'}$ from robot r_s to machine M_l , Job J_i and $J_{i'}$ could move simultaneously.
The scheduling problem objectives are:

- To determine the starting time d_{ij} for each machine operation O_{ij}. (the completion time is denoted f_{ij}).
- To assign a handling robot to each transport operation T_{ij} and to determine its starting time d'_{ij}. (the completion time is denoted f'_{ij})
- To minimize the Makespan denoted C_{max} = max (C_i) where C_i denotes the completion time of the last operation of job J_i.

All data p_{ij}, C^r_{pl} , V^r_{pl}, d_{ij}, d'_{ij}, f_{ij}, f'_{ij}, and C_{max} are assumed to be non-negative integers.

3. THE DISJUNCTIVE GRAPH MODEL

In this section, the disjunctive graph model that we will use as a basis for developing our two Heuristics is described. It is an extension of the classical disjunctive graph model $G = (N, A, E)$ [35] in order to take into account transportation and the blocking no wait constraint. [17] extend the disjunctive graph model for classical job shop to correspond to $G = (V, C, D_M, D_R)$ to describe the classical job shop problem with transportation operation performed by a single robot. [25] extend it to $G = (V_M, V_T, C, D_M, D_R)$ to encompass several robots. The disjunctive graph $G = (V_M, V_T, C, D_M, D_R)$ dedicated to job shop problem with transportation by several robot consists of: a set of vertices V_M containing all machine operations; a set of vertices V_T representing the set of transport operations obtained by an assignment of one robot to each transport operation ; two dummy nodes $\{0\}$ and $\{*\}$. The graph consists also of a set of conjunctive arcs C representing precedence constraints between operations of the same job, a set D_M of disjunctive arcs connecting the operations to be processed by the same machine and a set D_R of disjunctive arcs connecting the transport operations to be processed by the same robot. [11] introduced Alternative graph formulation to model the blocking constraints in the classical job shop.
Since the graph of [25] already deals with all conflicts regarding job-shop machines and transport operations we will use this graph to incorporate blocking and no wait constraints.

To deal with the blocking constraint in the job shop with transportation, each disjunctive arc in the set D_M is replaced by a disjunctive couple. More precisely, for each pair of operation O_{ij} and $O_{i'j'}$ sharing common machine, we introduce a disjunctive couple of two arcs: one from vertex T_{ij} to $O_{i'j'}$ and another one from $T_{i'j'}$ to O_{ij}. Since the swap is allowed, the weight of the disjunctive couples is zero.

To deal with no wait constraints, for each two consecutive transport and machine operations T_{ij} et O_{ij+1}, we add to the Set C a negative arc between

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

these two operations, The weight of the negative arc is $-t_{ij}$. This negative arc assume that a machine operation must start processing immediately after the completion of transport operation.

The Figure 1 represents the non-oriented disjunctive graph of the NWBT JSSP *(P1)* of size (*3* jobs × *3* machines × *4* robots) defined as follows:

- Machine operations:
  J_1 [M_1:8 ; M_2:10 ; M_3:6] ;
  J_2 [M_2:14 ; M_3:10 ; M_1:10] ;
  J_3 [M_1:14 ; M_3:10 ; M_2:8] ;

- On load transfer times (Identical robots):
  M_1 [M_1:0 ; M_2:2 ; M_3:4] ;
  M_2 [M_1:2 ; M_2:0 ; M_3:2] ;
  M_3 [M_1:4 ; M_2:2 ; M_3:0]
  ;
- empty transfer times:
  M_1 [M_1:0 ; M_2:1 ; M_3:2] ;
  M_2 [M_1:1 ; M_2:0 ; M_3:1] ;
  M_3 [M_1:2 ; M_2:1 ; M_3:0] ;

To solve the scheduling problem it is necessary to select one arc from each couple of the Set D_M to assign one robot to each transport operation and to turn all undirected arcs between robots into directed ones. We suppose that *(S1)* defined by table 1 is the solution of the problem *(P1)*.. This solution *(S1)* is represented by the conjunctive Graph of the figure 2.

Table 1: Solution (*S1)* of the NWBT JSSP (*P1)*

Machine 1	$\{O_{11}, O_{31}, O_{23}\}$
Machine 2	$\{O_{21}, O_{12}, O_{33}\}$
Machine 3	$\{O_{22}, O_{32}, O_{13}\}$
Robot 1	$\{T_{11}, T_{22}, T_{32}\}$
Robot 2	T_{12}
Robot 3	T_{21}
Robot 4	T_{31}

4. MIXED INTEGER LINEAR PROGRAM

This section presents the MILP model to formulate NWBT JSSP. Our following formulation is based on the model of [34]. We used their ideas to model the NWB JSSP.

To model NWB JSSP, the following notations are used.

List of parameters:

- p_{ij} the processing time of operation O_{ij}
- C_{pl}^{r}: on load transfer time of robot r between machine p and machine l.
- V_{pl}^{r}: Empty transfer time of robot r between machine p and machine l.
- H a large number

List of variables:

- d_{ij}: the start time of machine operation O_{ij}
- f_{ij}: the completion time of machine operation O_{ij}. (Time when operation O_{ij} leave machine Mij
- d'_{ij}: the start time of machine operation T_{ij}
- f'_{ij}: the completion time of machine operation T_{ij}.

List of decision variables:

- $\alpha_{ij;lq}$: Binary variable that takes value *1* if O_{ij}. is processed before O_{lq}. and *0* otherwise.
- $\beta_{ij;rs}$: Binary variable that takes value 1 if T_{ij}.require processing on robot r_s, and 0 otherwise.
- $\delta_{ij;lq}$: Binary variable that takes value *1* if T_{ij} is processed before T_{lq} and *0* otherwise.
- $\gamma_{ij;lq}$: Binary variable that takes value *1* if T_{ij} and T_{lq} are processed by the same robot and *0* otherwise.
- $\sigma_{ij,lq}^{r}$: Binary variable that takes value *1* if T_{ij} and T_{lq} are processed by the same robot and *0* otherwise.

The problem is formulated as follows :

Minimize C_{max}

Subject to :

- Finish time of machine operations:
  For $i \in [1,n]; j \in [1,m]$
  $$f_{ij} \geq d_{ij} + p_{ij} \quad (3)$$
- Precedence Constraints between transport operations and machine operations:
  For $i \in [1,n]; j \in [1,m-1]$
  $$\begin{cases} f'_{ij} = d_{ij+1} \\ d_{ij+1} = d'_{ij+1} \sum_{s=1}^{k} \beta_{ij,r_s} C_{M_{ij}M_{ij+1}}^{r_s} \end{cases} \quad (4)$$
- Precedence constraints between machine operations and transport operations:
  For $i \in [1,n]; j \in [1,m-1]$
  $$d'_{ij} = f_{ij} \quad (5)$$

Journal of Theoretical and Applied Information Technology
31st May 2018. Vol.96. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

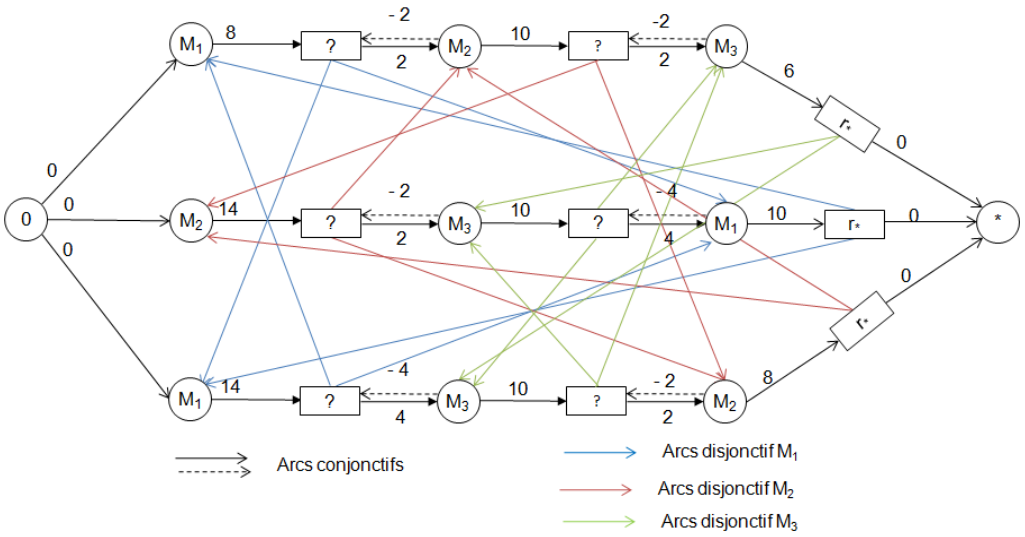

Figure 1: The Disjunctive Graph representing the problem (P1)

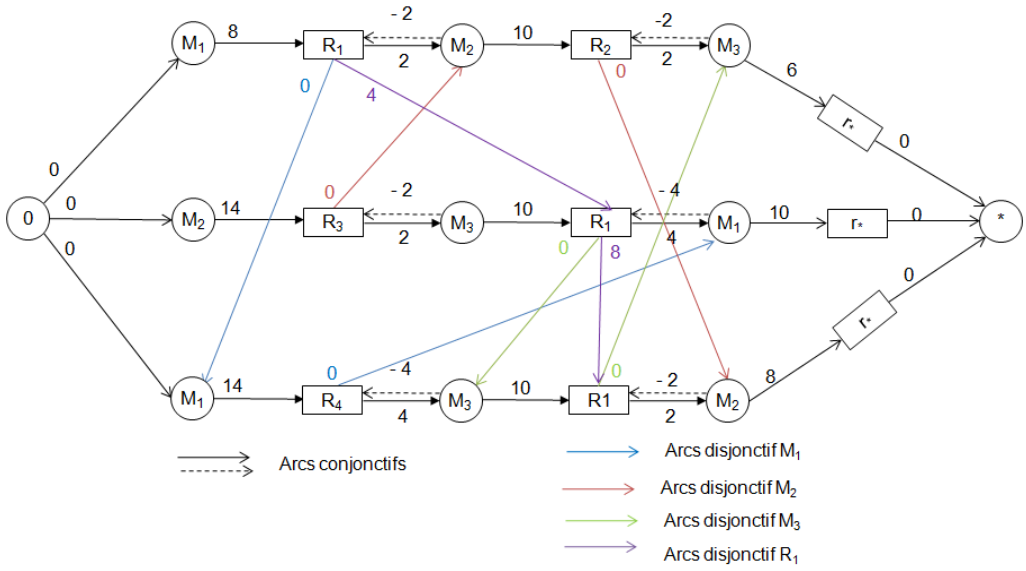

Figure 2: The conjunctive Graph representing the solution (S1) of the problem (P1)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Disjunctive constraints between machine operations:

For i,l∈[1,n]; j,q ∈ [1,m]/ $M_{ij}=M_{lq}$

$$\begin{cases} d_{ij} \geq f_{lq} - H * \alpha_{ij;lq} \\ d_{lq} \geq f_{ij} - H * (1 - \alpha_{ij;lq}) \\ \alpha_{ij;lq} + \alpha_{lq;ij} \end{cases} \quad (6)$$

- Robot assignment constraints:

For i∈[1,n]; j ∈ [1,m-1] ; s ∈ [1,k]

$$\sum_{s=1}^{k} \beta_{ij,rs} = 1 \quad (7)$$

- Disjunctive constraints between robot:

For i,l∈[1,n]; j,q ∈ [1,m-1] ; s ∈ [1,k]

$$\begin{cases} \sigma_{ij;lq}^{rs} \geq (1 - \beta_{ij,rs})H - (1 - \beta_{lq,rs})H \\ \sigma_{ij;lq}^{rs} \leq \beta_{ij,rs} \\ \sigma_{ij;lq}^{rs} \leq \beta_{lq,rs} \\ \lambda_{ij;lq}^{rs} = \sum_{s=1}^{k} \sigma_{ij;lq}^{rs} \\ d'_{ij} \geq d'_{lq} + \sum_{s=1}^{k} \beta_{lq,rs} \left(C_{M_{lq}M_{lq+1}}^{rs} + V_{M_{lq+1};M_{ij}}^{rs} \right) + (\lambda_{ij,lq} - 1)H - \delta_{ij,lq}H \\ d'_{lq} \geq d'_{ij} + \sum_{s=1}^{k} \beta_{ij,rs} \left(C_{M_{ij}M_{ij+1}}^{rs} + V_{M_{ij+1};M_{lq}}^{rs} \right) + (\lambda_{ij,lq} - 1)H - (\delta_{ij,lq} - 1) \\ \delta_{ij,lq} + \delta_{lq,ij} = 1 \end{cases} \quad (8)$$

(3) Ensures that each operation is processed at least for its process duration.
(4) Ensures that each machine operation starts immediately after the finish of the transport operation that precede it.
(5) As there is no buffer in machines, transport operation starts immediately when the job leaves a machine.
(6) Ensures that each machine process one operation at a time.
(7) Ensures that each transport operation is performed by only one Robot.
(8) Ensures that each robot process one operation at a time. No two transport operations are performed by the same robot at any time

5. HEURISTICS

5.1 Blocking Situations cycles - Graph of last scheduled operations

In this section, we will look for situations that could lead to blocking states in order to avoid them during the construction of our algorithm. For this purpose, we will identify blocking situations by using a graph $G_s = (M,J)$ we will call the graph of last scheduled operations. This graph is defined as follows: Consider the graph of last scheduled operations $G_s = (M,J)$. A set of vertices M represents machines; A set of arcs J represents the last scheduled operations O_{ij} of job J_i. The starting point of the last scheduled operation O_{ij} of job J_i is the machine M_{ij} and its end point is the machine M_{ij+1}.

Blocking condition (C1) can be formulated as follows: "Systems may confront a blocking situation if the graph of the last scheduled operations $G_s = (M,J)$ contains a cycle of length $p \geq 2$.

Consider for example a problem with 5 machines $\{M_1, M_2, ..., M_5\}$ and 4 jobs $\{J_1, J_2, ..., J_4\}$. The last scheduled operations are:

J_1: M_1--> M_2 ; J_2: M_2--> M_3; J_3: M_3--> M_1, J_4: M_4--> M_5 .

The associated graph G_s is modeled as follows:

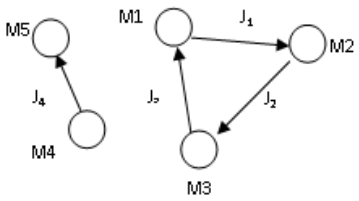

Figure 3: Associated graph of last scheduled operations

Following the topology of $G_s = (M,J)$, the graph contains a cycle of length $p=3$ (p equals the number of jobs that forms the cycle). Under this cycle, the necessary condition of blocking *(C1)* is satisfied because job J_1 that is processed on the machine M_1 cannot pass to machine M_2 occupied by the job J_2, as well as for job J_2 that is processed on machine M_2 cannot pass to machine M_3 occupied by job J_3 as well as for job J_3 that is processed on machine M_3 cannot pass to machine M_1 occupied by the job J_1.

Thereafter, we will check if the condition *(C1)* is a sufficient condition to blocking situation in BNWT JSSP. Consider the graph G_s which a cycle of length $p \geq 2$.

Journal of Theoretical and Applied Information Technology
31st May 2018. Vol.96. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

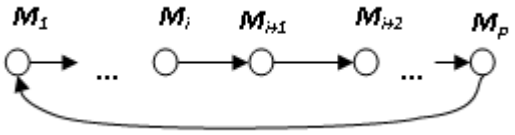

Figure 4: Graph with a cycle of length p ≥2

We suppose that the processing of all the jobs J_i has finished. Each job J_i remains blocked on machine M_i until its transportation operation T_i from M_i to M_{i+1} start. We assume that a robot r_1 is available and will be assigned to transportation operation T_i. This operation can start and liberate the machine M_i. No wait condition imply that the processing of job J_i on machine M_{i+1} must start immediately after the termination of transport operation T_i otherwise J_i cannot be transported by r_1 and then machine M_i remains blocked. On the other hand, M_{i+1} is blocked by J_{i+1} and will be liberated at the latest when the transport operation T_i ends. Therefore The transport operation T_{i+1} of J_{i+1} starts and so on until the blocking cycle is totally liberated. In the case of SWAP allowed (Our case) transport operations can be performed by the same robot r_1 (jobs between machines and robots can be done simultaneously) or an other robot r_s if available. If swap is not allowed M_{i+1} must be liberated before the transport operation T_i ends, therefore we must have a second robot to perform transport operation T_{i+1} .

Proposition :

- In the case of NWBT JSSP with Swap allowed: If the graph of last scheduled operations $G_s = (M,J)$ contains a cycle of length $p \geq 2$, the system is partially blocked (every blocking situation can be liberated).
- In the case of NWBT JSSP with Swap allowed, If the graph of last scheduled operations $G_s = (M,J)$ contains a cycle of length $p \geq 2$, the system is partially blocked (every blocking situation can be liberated) if the number of robots $k \geq 2$ and eternally blocked if the number of robots $k = 1$

5.2 The proposed Heuristics

In this subsection we propose two dedicated heuristic *(HC1)* and *(HC2)* to the BNWT JSSP. During construction of the algorithm, we complete iteratively a partial schedule S consisting of two vectors: A vector AM representing the scheduling of machine operations and vector AR representing the assignment of robots to transport operations. The scheduling of transport operations is deduced directly from the scheduling of machine operations given the constraints of no wait. U denotes the set of non-scheduled operations. For each iteration of these two heuristic, a machine operation is selected according to eligibility and priority rules. When a machine operation is chosen on the basis of selection rules, the transportation operation which precedes this machine operation is automatically selected and a robot is assigned according to another priority rules.

The Heuristic pseudo code is given on algorithm 1.

Algorithm 1 : *(HC1)* and *(HC2)* Heuristic Pseudocode

Function generate_OM_AR;
Data: Problem inputs (operating range of jobs, processing times, transfer times).
Result: AM : machine selection ; AR : robot assignement
Begin
  $U := O$ /* set of non-scheduled operations */
  $S := \emptyset$ /* set of scheduled operations */
  while $U \neq \emptyset$
    $E := Elire(U,S)$ /* set of eligible operations according to the rules *1, 2* et *3*.*/
    $Oij := Rule\_operation(E,S,J)$ /* Select an operation Oij from the set E according to the rules *4* and *5}*
    $S := Update\ (S,Oij)$ /* Adding Oij to AM*/
    $rs := Rule\_Robot\ (Tij\text{-}1,R,S)$ /* Assign a robot to transport operation *Tij-1* according to the rules *6* and *7.*/
    S := Update (S, r_s) /* Adding r_s to AR */
    U := U - O_{ij} /* Subtract O_{ij} from U */

    /* Case of *(HC2)*/
    if {the selection of O_{ij} lead to $Gs\ (M,J)$ that contains a blocking cycle $B_s = \{O_{ij}, O_{ij+1}, O_{lq}, O_{lq+1}\}$ } **then**
      $S := Update\ (S, \{O_{ij+1}, O_{lq+1}\})$ /* Adding O_{ij+1}, O_{lq+1} to AM to liberate B_s }
      r_s := Rule_Robot (T_{ij},R,S) /* Assign a robot to transport operation T_{ij} according to the rules 6 and 7.*/
      $r_{s'}$:= Rule_Robot (T_{lq},R,S) /* Assign a robot to transport op eration T_{lq} according to the rules 6 and 7.*/
      S := Update (S, $r_s, r_{s'}$) /* Adding $r_s, r_{s'}$ to AR */
      U := U – $\{O_{ij+1}, O_{lq+1}\}$ /* Subtract $\{O_{ij+1}, O_{lq+1}\}$ from U */
    end if

  end while
end

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Eligible Operations:

The Iteration starts with the construction of the set *E* of eligible machine operations. An operation is eligible for partial schedule *S* if it is a non scheduled operation that can start without violating any constraints.

- Rule 1: Eligible operations must respect operations precedence constraints within each job.
- Rule 2: Eligible operations $S(M_p)$ that need to be processed on machine M_p cannot be scheduled as long as machine M_p is occupied by another job. These operations will be eliminated from the set *E*.
- Rule 3 : Eligible operations that could lead a cycle in the graph $G_s = (M,J)$ with $p \geq 2$ will be eliminated from the set *E*. For (HC2), If E is Empty we continue our algorith and select an operation with p=2

Selection rules of machine operations:

After the determination of eligible operations, it remains to appoint the machine operation to be scheduled.

- Rule 4: We associate each operation $O_{ij} \in E$ to a pair (M_p, g) where $M_p = M_{ij}$ and g is the total time of operations $S(M_p) \in U$, g can be seen as the weight of M_p on the set of non scheduled operations *U*. Machine operation to be scheduled is the one associated to a pair (M_p, g) with the largest $(g=g_{max})$.
- Rule 5: If we have on the set *E* two or more operations with the same $g=g_{max}$, we choose the operation that has the longest queue

$$Q(O_{ij})=\sum_{q=j}^{n_1} p_{iq}$$

- Rule 6: To select the robot that will perform the transportation operation T_{ij}, we opted to choose the robot which provides the minimal completion time of the transportation operation f'_{ij}, which involves exploring all robots for each assignment.
- Rule 7: In the case of two or more robots provide the minimal completion time f'_{ij}, we choose the robot that has the minimal empty robot arrival time to arrive at the departure machine for the loaded move T_{ij}.

For (HC2), if the selection of O_{ij} lead to $Gs\ (M,J)$ that contains a blocking cycle $B_s = \{O_{ij}, O_{ij+1}, O_{lq}, O_{lq+1}\}$ then the machine operation O_{ij+1}, O_{lq+1} are also added to *AM* to liberate B_s and a robots are assigned to transport operations T_{ij} and T_{lq}

6. NUMERICAL RESULTS

The evaluation of our methods is carried out using a first set of instances from a well Known Benchmark for the problem of a job shop with transport and second set of instances developed by ourselves. The Well-known benchmark is suggested by [23] and has been used by [36]. This instance set encompass two subsets *P1* and *P2* whose size *(n * m)* is respectively *(6 * 6)* and *(10 * 10)*. This set of instances is used to compare MILP results between NWBT JSSP and T JSSP for instances of size *(6*6*1)*.

The second set of instances comprises *30* instances grouped into three subsets *S1*, *S2* and *S3* with respective sizes of *(4 * 4), (6 * 6), (10 * 10).* For these instances, the jobs routing are randomly generated and the processing times and transfer times are randomly distributed respectively over *[10; 100]* and *[1; 20]*. The evaluations are carried out using respectively *1* robot and *2* robots. For all instances, that encompass more than one robots, the robots are considered as similar.

In the table 2 & 3, we report the results of our heuristics and MILP. These results are obtained used the *C++* programming language for the construction Heuristic Program and *CPLEX 12.2* as LP solver. The *C++* and the linear program using *CPLEX 12.2* program were run on a simple desktop PC running Windows 7 with an Intel Core i5 processor running at 2.40GHz with 4,00GB of memory.

The notations below are used in the numerical results table:

- Problem size *n *m *k*: *n:* number of jobs, *m*: number of machines *k:* number of robots ;
- *C_max[1]:* The MILP solution found by [36] for the job shop problem with transportation (T JSSP).
- *C_max[2]:* The optimal solution found by the execution of MILP for the NWBT JSSP. We allowed a maximum computation time of 4 hour;
- *FS-(HC1):* The number of feasible solution found by *(HC1).*
- *FS-(HC2):* The number of feasible solution found by *(HC2).*
- *C_max[3]:* the approached solution found by the Construction Heuristic algorithm *(HC1).*
- *C_max[4]:* the approached solution found by the Construction Heuristic algorithm *(HC2).*
- *AM_{1-2}:* the relative gain obtained by using *2* robots instead of *1* robot;
- *AM_{2-3}:* the relative gain obtained by using *3* robots instead of *2* robot;
- *AM_{1-3}:* the relative gain obtained by using *3* robots instead of *1* robot;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- *Dev_[1]:* The relative deviation between *C_max[1]* and the *C_max[2]*.
- *Dev_[2]:* The average relative deviation between *C_max[2]* and *C_max[3]*
- *Dev_[3]:* The average relative deviation between *C_max[2]* and *C_max[4]*.
- *R: Robot*

Table 2: Numerical results for P1- one robot

	C_max[1]	*C_max[2]*	*Dev_[1]*	*Dev_[2][3]*
D1-d1	60	94	57%	122%
D1-t1	62	82	32%	132%
D2_d1	71	152	114%	80%
D3-d1	84	171	104%	98%
T2-t1	68	112	65%	100%
T3-t0	58	77	33%	136%
Tkl.1	62	104	68%	102%

Table 3: (HC1) and (HC2) Heurictics numercial results for (P1-P2) and (S1-S2-S3) instances.

		S1	S2	S3
FS-(HC1)		70%	50%	0
FS-(HC2)		100%	90%	60%
Dev_[2]	1-Robot	47%	82%	
	2-Robot	43%	75%	
Dev_[3]	1-Robot	49%	84%	
	2-Robot	46%	77%	

Since there is no benchmark results for the NWBT JSSP, the evaluation of the performance of our constructive heuristics is done by comparing the results of the two heuristics with each other and with the MILP for the second set of instances. The first set of instances served us to evaluate the effects of blocking and no wait constraints by comparing the optimal solutions found for the BNWT JSSP and the T JSSP.

From our computational results in Table 3 and over the *30* test instances, we noted that *(HC1)* found a solution in *40%* cases whereas *(HC2)* found a solution in *83,3%*.

These results are consistent with the way we have constructed our two heuristics. For Heuristic *(HC2)*, the fact that we have authorized the selection of operation that lead to cycle $p = 2$ allowed us to greatly improve the ability of this heuristic to find feasible solutions. In contrast, Heuristic *(HC1)* was not able to find feasible solutions for the majority of test instances.

The absolute quality of the results obtained by *(HC1)* and *(HC2)* are only compared for small and medium size instances. For large instances, we could not obtain an optimal solution within a reasonable computational time by MILP to compare it with the heuristics solutions. When solutions are feasible, the results show that the average relative error for the heuristic *(HC1)* is *62%* and the relative error for Heuristic (HC2) is *64%*.

It is also interesting to observe the effects of the blocking and no-wait constraints on the value of the optimal solutions. As it is noted in table 2, the optimal solution of BNWT JSSP is *67%* larger, on average, than the optimal solution of T JSSP.

7. CONCLUSIONS

This paper addresses the NWBT JSSP problem with the objective of minimizing the make span. We developed a MILP to solve the problem exactly. We have modelized the problem by a disjunctive graph and proposed two heuristic based on priority rules and avoiding blocking situations. Since no benchmark tests for the NWBT JSSP problem were available in the literature, we have evaluated our methods by comparing our results firstly with the T JSSP benchmark results and secondly the results obtained by our heuristics with the MILP results.

For the problem of job shop with blocking and no-wait constraints, greedy heuristics which involves enlarging step by step a partial schedule must deal with the quality and feasibility issue as a feasible partial schedule cannot always be extended to a complete feasible schedule. This difficulty is well illustrated by *(HC1) & (HC2)* heuristic numerical results.

Numerical application show that heuristic *(HC2)* is quite efficient in terms of obtaining feasible solutions. The comparison with MILP shows that there is still room of improvements in terms of the quality. However, the application of the *(HC2)* heuristic remains more interesting than the MILP for very large instances in that it allows us to find approximate solutions in reasonable times.

For further research, it would be interesting to ameliorate the solution obtained by our greedy heuristic by investigating other heuristics. These heuristics can be Meta –heuristics, such as Tabu Search, simulated annealing, or other special greedy heuristics.

REFRENCES:

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

[2] N.G. Hall, C. Sriskandarajah, A survey of machine scheduling problems with blocking and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

no-wait in process, Operations Research 44 (3) (1996) 510-525.

[3] O. Candar, Machine scheduling problems with blocking and no-wait in process, Working Paper [April-99], Department of Industrial Engineering, Bilkent University, Ankara, Turkey, 1999.

[4] Mascis, D. Pacciarelli, Machine scheduling via alternative graphs, Research Report, RT-DIA-46-2000, Italy, 2000.

[5] A. Mascis, D. Pacciarelli, Job-shop scheduling with blocking and no-wait constraints, European Journal of Operational Research 143 (2002) 498-517.

[6] Y. Mati, N. Rezg, X. Xie, Geometric approach and taboo search for scheduling flexible manufacturing systems, IEEE Transactions on Robotics and Automation 17 (6) (2001) 805-818.

[7] C.A. Brizuela, Y. Zhao, N. Sannomiya, No-wait and blocking job-shops: Challenging problems for GA's, IEEE 0-7803-77-2/ 01 (2001) 2349-2354.

[8] A. Klinkert, Optimization in design and control of automated high-density warehouses, Doctoral Thesis No. 1353, University of Fribourg, Switzerland, 2001.

[9] H. Groflin, A. Klinkert, Local search in job shop scheduling with synchronization and blocking constraints, Internal working paper [04-06], Department of Informatics, University of Fribourg, Switzerland, 2004.

[10] P. Brucker, T. Kampmeyer, Cyclic job shop scheduling problems with blocking,Ann. Oper. Res. 159 (2008) 161–181.

[11] A. AitZai, B. Benmedjdoub, M. Boudhar, A branch and bound and parallel genetic algorithm for the job shop scheduling problem with blocking, Int. J. Oper. Res.14 (3) (2012) 343–365.

[12] J. Grabowski, J. Pempera, Sequencing of jobs in some production system, European Journal of Operation Research 125 (2000) 535–550.

[13] D. Pacciarelli, M. Pranzo, Production scheduling in a steelmaking-continuous casting plant, Computer and Chemical Engineering 28 (2004) 2823–2835.

[14] J. Romero, L. Puigjaner, T. Holczinger, F. Friedler, Scheduling intermediate storage multipurpose batch plants using the S-graph, AIChE J. 50 (2004) 403–417.

[15] L. Chen, N. Bostel, P. Dejax, J. Cai, L. Xi, A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, European Journal of Operation Research 18 (2007) 40–58.

[16] A. D'Ariano, D. Pacciarelli, M. Pranzo, A branch and bound algorithm for scheduling trains in a railway network, European Journal of Operation Research 183 (2007) 643–657.

[17] J. Hurink ,S. Knust, Tabu search algorithms for job-shop problems with a single transport robot, European Journal of Operational Research 2005;162(1):99–111.

[18] J. Blazewicz,H. Eiselt ,G. Finke,G. Laporte,J. Weglarz, Scheduling tasks and vehicles in a flexible manufacturing system, International Journal of Flexible Manufacturing Systems 1991;4(1):5–16.

[19] U. Bilge,G. Ulusoy, A time window approach to simultaneous scheduling of machines and material handling system in an FMS. Operations Research 1995;43(6):1058–70.

[20] G. Ulusoy, F. Sivrikaya-erifolu, U. Bilge, A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles, Computers \& Operations Research 1997;24(4):335–51.

[21] G. Alvarenga,G. Mateus, G. De Tomi, A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows, Computers \& Operations Research 2007;34(6):1561–84.

[22] M. Soylu,N. O¨ zdemirel, S. Kayaligil, A self-organizing neural network approach for the single AGV routing problem, European Journal of Operational Research 2000;121(1):124–37.

[23] J. Hurink,S. Knust, A tabu search algorithm for scheduling a single robot in a job shop environment, Discrete Applied Mathematics 2002;119(1–2):181–203.

[24] T. Abdelmaguid,A. Nassef, B. Kamal, M. Hassan, A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Research 2004;42(2):267–81.

[25] P. Lacomme,M. Larabi, N. Tchernev, A disjunctive graph for the job shop with several robots, In: MISTA conference; 2007. p. 285–92.

[26] L. Deroussi, M. Gourgand, N. Tchernev, A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Research 2008;46(8):2143–64.

[27] H. Gong, L. Tang, Two-machine flow shop scheduling with intermediate transportation under job physical space consideration,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Computers \& Operations Research 2011;8(9):1267–74.

[28] M. Mateo, R. Companys, J. Bautista, Resolution of graphs with bounded cycle time for the cyclic hoist scheduling problem, In: 8th International Workshop on Project Management and Scheduling.

[29] A. Caumond, P. Lacomme, A. Moukrim, N. Tchernev, An MILP for scheduling problems in an FMS with one vehicle, European Journal of Operational Research 2009;199(3):706–22.

[30] Y. Crama, J. Van de Klundert, Cyclic scheduling of identical parts in a robotic cell. Operations Research 1997;45(6):952–65.

[31] N. Brauner, G. Finke, V. Lehoux-Lebacque, C. Potts, J. Whitehead, Scheduling of coupled tasks and one machine no wait robotic cells, Computers \& Operations Research 2009;36(2):301–7.

[32] AI. Corréa, A. Langevin, L-M. Rousseau, Scheduling and routing of automated guided vehicles: a hybrid approach. Computers \& Operations Research 2007; 34(6):1688–707, [part special Issue: Odysseus 2003 second international workshop on freight transportation logistics.

[33] E. Stafford, F. Tseng,J. Gupta, Comparative evaluation of MILP flow shop models, Journal of Operations Research Society 2005; 56;88–101.

[34] A. Manne, On the job shop scheduling problem. Operations Research 1960;8,219–22

[35] Roy, B., Sussmann, B. Les problemes d'ordonnancement avec constraintes disjonctives, Note DS no. 9 bis, SEMA, 1964, Paris.

[36] LARABI, Mohand. Le problème de job-shop avec transport: modélisation et optimisation. 2010. Thèse de doctorat. Université Blaise Pascal-Clermont-Ferrand II.

[37] A Mascis, D Pacciarelli, Job-shop scheduling with blocking and no-wait constraints, European Journal of Operational Research, 2002, Volume 143, Issue 3, Pages 498-517.