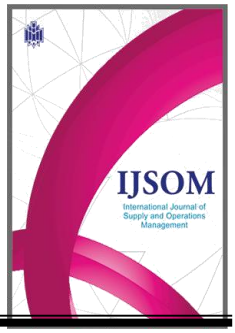
International Journal of Supply and Operations Management

IJSOM

February 2017, Volume 4, Issue 1, pp. 78- 89

ISSN-Print: 2383-1359
ISSN-Online: 2383-2525
www.ijsom.com



Modeling and solving the distributed and flexible job shop scheduling problem with WIPs supply planning and bounded processing times

M. Ziaee[a]

[a] *University of Bojnord, Bojnord, Iran*

Abstract
In this paper, for the first time in the literature, we integrated production scheduling decisions and WIPs planning decisions in a distributed environment. We study the distributed and flexible job shop scheduling problem (DFJSP) which involves the scheduling of jobs (products) in a distributed manufacturing environment, under the assumption that the shop floor of each factory/cell is configured as a flexible job shop. It is also assumed that the work-in-process (WIP) parts can be bought from the market instead of manufacturing them in-house, and they also can be sold in the market instead of processing their remaining operations and selling the end products. Moreover, the processing times of the operations can be decreased by paying a cost. However, there are a lower limit and an upper limit for the processing time of each operation. We formulate this general problem as a mixed integer linear programming (MILP) model. A fast heuristic algorithm is also developed to obtain good solutions in very short time. The algorithm is tested on some problem instances in order to evaluate its performance. Computational results show that the proposed heuristic is a computationally efficient and practical approach.

Keywords:
Distributed Scheduling; WIPs Supply Planning; Flexible Job Shop; Bounded Processing Times.

1. Introduction

The significance of distributed manufacturing has been recognized by many researchers and industrialists in recent years due to the changes in the mode of today's production environment. Single factory or centralized production environment in traditional manufacturing systems has been gradually replaced by more flexible distributed settings, including multi-factory networks or multi-cell job shops due to the trend of globalization. The distributed manufacturing enables the enterprises to be closer to their customers and suppliers, to produce and market their products more effectively, to be responsive to market changes more quickly, to achieve better product quality, lower production cost, reduced management risk, and better utilization of production resources (Chan et al., 2006b; Rosenau, 1996; Wang, 1997).

The distributed manufacturing takes place in a multi-factory environment including several factories, which may be geographically distributed in different locations, or in a multi-cell environment including several independent manufacturing cells located in the same plant. Each factory/cell is capable of manufacturing a variety of product types. In addition, the factories or cells have different production efficiencies and lead times, operating costs, constraints, etc. (Chan et al., 2006b; Giovanni and Pezzella, 2010). An important issue in dealing with the production in this decentralized manner is the scheduling of operations of products (jobs) in the distributed manufacturing system (Jia et al., 2003; Wang and Shen, 2007).

*Corresponding author email address: ziaee@iust.ac.ir

In this paper, we study the distributed and flexible job shop scheduling problem (DFJSP) which involves the scheduling of jobs in a distributed manufacturing environment described above, under the assumption that the shop floor of each factory/cell is configured as a flexible job shop (Therefore, each factory/cell will be hereafter called flexible manufacturing unit (FMU)). The flexible job shop scheduling problem (FJSP) is a generalization of the classical job shop scheduling problem(JSP). The $n \times m$ classical JSP is defined as follows: set of n jobs must be processed on a group of m machines, where the processing of each job i consists of J_i operations which should be performed on these machines. Each job has a specified processing order on the machines which is fixed and known in advance, i.e., each operation has to be performed on a given machine. In the FJSP however each operation is allowed to be processed on any machines among the set of available ones; and thus, the scheduling problem is choosing a machine and a starting time at which the operation must be processed, for each operation (Baker, 1974; Pinedo, 2002). The DFJSP consists of the following two subproblems which can be solved sequentially or simultaneously: (1) the assignment of each job to exactly one FMU (which corresponds to one cell in a multi-cell environment or one factory in a multi-factory setting), and (2) the scheduling of the jobs in each FMU, i.e. solving a FJSP for each FMU. Once a job is assigned to a FMU and starts its processing (but not finished), it is usually impossible or uneconomical to transfer this unfinished job to another FMU for the remaining operations (Chan et al., 2006b). Therefore, we assume that once a job is allocated to a FMU, all its operations have to be processed in the same FMU. We also suppose that all the FMUs belong to the same company. Accordingly, they work cooperatively to define an optimal production plan maximizing the revenue of the company as a whole. Another assumption considered in this paper is that the WIPs can be bought from the market instead of manufacturing them in-house, and they also can be sold in the market instead of processing their remaining operations and selling the end products. In other words, for each job assigned to each FMU, the FMU can either process all the operations of the job or process only several consecutive operations. These options, i.e. buying/selling the WIPs, lead to the best use of existing capacities and a maximum profit by an integrated planning model for the WIPs and therefore, can have many practical advantages and be used for many production and even non-production environments. It is also assumed that the processing times of the operations can be decreased by paying a cost. However, there are a lower limit and an upper limit for the processing time of each operation. We formulate this general problem as a mixed integer linear programming (MILP) model to integrate the decisions on production scheduling and WIPs supply planning.

Development of MILP models for more complex scheduling problems such as FJSP has become increasingly important in recent years, because they can be easily solved using a robust software system and used for benchmarking the heuristic algorithms and understanding the structure of the problems (Unlu and Mason, 2010). A review of MILP formulations for the job shop and the flow shop scheduling problems presented in the literature is provided by Pan (1997). For the FJSP, a review of the mathematical models developed for this problem is presented by Özgüven et al. (2010). A more recent literature review and comparative evaluation of mathematical models for the FJSP is provided by Demir and Isleyen (2013). They investigate all the mathematical models for the FJSP existing in the literature in terms of binary variables that they rely on for sequencing operations on machines, and computationally compare the models assuming that the objective function is makespan. They conclude that the model presented by Özgüven et al. (2010) is the best model in the literature in terms of the computational time. For the FJSP with bounded processing times, there is only one contribution from reference (Zhang et al., 2012) in which the authors present a mathematical model and a genetic algorithm with tabu search procedure to solve the problem. Even for simpler scheduling problems such as parallel machines or flowshop scheduling problems, a common assumption in the literature is that the processing times of the jobs are fixed and known in advance. However, in some real scheduling problems, this assumption may not be realistic. Moreover, the lower and upper bounds on processing times are often very easy to obtain in practical cases.

The DFJSP is more complicated than the traditional FJSP in single FMU, because it involves not only the FJSP in each FMU, but also the problem in an upper level that is the assignment of the jobs to the FMUs. This problem is strongly NP-hard, since the problem with one FMU, i.e. the FJSP, is already strongly NP-hard (Garey et al., 1976). Therefore, the more general problem considered in this paper is also strongly NP-hard. This justifies the need for developing efficient heuristic algorithms to obtain approximate solutions of good quality at little computational cost. These heuristic algorithms are very fast in comparison with the exact methods and the metaheuristic algorithms; and if they have been proved to produce solutions with adequate accuracy then they can be the best approach. Therefore, this paper proposes a powerful heuristic algorithm to solve the problem.

Almost all existing studies in the field of production scheduling deal with the centralized or non-distributed production environments, and only few papers investigate the distributed scheduling problems, especially those involving the DFJSP (Wang and Shen, 2007). Due to the complexity of the distributed scheduling problems mentioned above, approximate algorithms, mainly metaheuristics, have been often used to solve the problem. A review of the heuristics developed to solve these problems can be found in references (Giovanni and Pezzella, 2010; Wang and Shen, 2007). Wang et al. (2013) consider the distributed permutation flow-shop scheduling problem and present an effective estimation of distribution algorithm (EDA) to solve the problem. This problem is also examined in (Gao et al., 2013) and solved by a tabu search algorithm. Jia et al. (Jia et al. 2002, 2003) present a modified genetic algorithm to solve the distributed scheduling problem in a multi-factory network in which, each factory is configured as a job shop. Chan et al. (2005, 2006a) consider the DFJSP and present a genetic algorithm with dominated genes (GADG) to solve the problem and demonstrate the

performance of the method by using several new DFJSP instances. Also, Chan et al. (2006b) study a generalized version of the DFJSP in which, machine maintenance constraints are considered and it is assumed that the maintenance time is related to the machine age. A mathematical model and a genetic algorithm with dominant genes (GADG) are developed for the problem. For both with and without maintenance constraints, the performance of the presented genetic algorithm is evaluated using some new test instances. Giovanni and Pezzella (2010) present an improved genetic algorithm to solve the DFJSP. The proposed approach is compared with other algorithms for distributed scheduling and tested on a large set of DFJSP instances derived from well-known FJSP benchmarks, providing good results. As it can be seen in the above review of literature on the distributed scheduling problems, studies on methods for solving these problems are in the early stage. Almost all the proposed methods are metaheuristic algorithms and very time-consuming in comparison with the heuristic algorithms, because they perform the search procedure in larger part of the solution space. For the general problem considered in this paper, i.e. the DFJSP with the WIPs supply planning and bounded processing times, to the best of author's knowledge, there is no contribution in the literature.

In this study, we present a MILP model to solve the DFJSP with the WIPs supply planning and bounded processing times (section 3). The problem is also solved using a heuristic algorithm (section 4). In order to evaluate the performance of the proposed heuristic, we implement it using several randomly generated test problems and present the results of the computational experiments (section 5). The results show that the proposed heuristic method can obtain good solutions in very short time. Concluding remarks are given in the last section.

2. Assumptions and notations

The assumptions considered in this paper are as follows:

(1) Each FMU can produce all jobs with different efficiencies.
(2) For each job, all FMUs have the same number of operations.
(3) Jobs are independent of each other.
(4) Setup and transportation times are negligible.
(5) There are no precedence constraints among the operations of different jobs.
(6) Preemption is not allowed, i.e. a started operation cannot be interrupted during its processing.
(7) Each machine can process at most one operation at the same time.
(8) An operation cannot be performed by more than one machine at the same time.
(9) All the jobs have equal priorities.
(10) Machines never break down and are available throughout the scheduling period.
(11) For each job assigned to each FMU, the FMU can buy one of its WIPs and neglect the processing of all the operations belonging to the bought WIP. Also, the FMU can stop the processing of the job in an arbitrary stage, sell the corresponding WIP and neglect the processing of its remaining operations. In other words, for each job assigned to each FMU, the FMU can process either all the operations of the job or only several consecutive operations.
(12) For each job, at most one WIP can be bought by the whole company.
(13) For each job, the processing time of each operation can be decreased by paying a cost. However, there are a lower limit and an upper limit for the processing time of each operation.
(14) All FMUs, machines, jobs and bought WIPs are available at time zero.

The notations used throughout the paper are as follows:
Indices and parameters:
l: number of FMU's.
n: number of jobs.
f: index of FMU's; f=1,…,l.
i,i',z: index of jobs; i,i',z=1,…,n.
m_f: number of machines in FMU f.
J_i: number of operations of job i.
$maxJ$: maximum number of operations per job (i.e., $maxJ= \max_i J_i$).
j,j': index of operations; j=1,…, J_i; j'=1,…, $J_{i'}$.
k,k',y: index of machines; k,k',y=1,…, m_f.
lt_{ijyf}: Lower limit of processing time of operation j of job i on machine y of FMU f.
ut_{ijyf}: Upper limit of processing time of operation j of job i on machine y of FMU f.
a_{ijf}: set of machines in FMU f which are capable of executing operation j of job i.
na_{ijf}: number of members of the set a_{ijf}.
s'_{ijf}: mean processing time of operation j of job i over the machines belonging to the set a_{ijf} (i.e., $s'_{ijf} = (\sum_{y \in a_{ijf}} ut_{ijyf}) / na_{ijf}$).

sj_{if}: total mean processing time of job i in FMU f (i.e., $sj_{if} = \sum_{j=1}^{J_i} s'_{ijf}$).

sk_{yf}: total weighted processing time on machine y of FMU f which is calculated as follows: $sk_{yf} = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ if\ y \in a_{ijf}}}^{J_i} \frac{ut_{ijyf}}{na_{ijf}}$.

sp_{ijf}: selling price of WIP (i,j,f). WIP (i,j,f) consists of all operations j': $j' \leq j$, which may be either executed in FMU f or bought by FMU f from an outsource. Therefore, for WIP (i,j,f), all operations j': $j'>j$, are unprocessed and need to be executed either by FMU f or by outsourcing. WIP (i,0,f) is a base part/raw material and WIP (i,J_i,f) is a finished job.
pc_{ijf}: procurement cost of WIP (i,j−1,f).
fc_f: fixed (overhead) cost of FMU f per time unit. This cost is therefore not dependent to the number of jobs processed by the FMU and its scheduling plan.

tfc: the sum of fixed costs, i.e. $tfc = \sum_{f=1}^{l} fc_f$. According to this definition, if the makespan, i.e. the maximum completion time over all the FMUs, is equal to $Cmax$ then total fixed cost for the problem is equal to $tfc.Cmax$.
c_{ijyf}: processing cost of operation j of job i processed on machine y of FMU f per time unit.
c'_{ijyf}: cost of decreasing the processing time of operation j of job i on machine y of FMU f per time unit.
According to the above two definitions for c_{ijyf} and c'_{ijyf}, if the processing time of operation j of job i on machine y of FMU f is equal to T_{ijyf}, $lt_{ijyf} \leq T_{ijyf} \leq ut_{ijyf}$, then total processing cost of it is equal to $T_{ijyf}.c_{ijyf} + (ut_{ijyf} - T_{ijyf}).c'_{ijyf}$.
M: a large number.

Variables:
U_{ijf}: binary variable that takes the value 1 if WIP (i,j,f) is sold by FMU f, and 0 otherwise.
V_{ijf}: binary variable that takes the value 1 if WIP (i,j−1,f) is bought by FMU f, and 0 otherwise.
T_{ijyf}: processing time (duration) of operation j of job i on machine y of FMU f.
CT_{ijf} / CT_{ijyf}: completion time of operation j of job i on (machine y) FMU f.
IF_f: number of jobs assigned to FMU f.
$Y_{iji'j'}$: binary variable that defines the processing order of the operations (i, j) (i.e. operation j of job i) and (i', j') belonging to two different jobs ($i<i'$) on the same machine; it takes the value 1 if operation (i', j') precedes operation (i, j), and 0 otherwise.
X_{ijkf}: binary variable that takes the value 1 if operation j of job i is processed on machine k of FMU f, and 0 otherwise.
$Cmax$: makespan, i.e. the overall completion time of all jobs on all FMUs.
Obj: objective function value that is the total profit and equal to the sum of revenues of selling the WIPs minus the total costs including the fixed costs, the processing costs and the procurement costs; it is computed by the following equation:

$$Obj = \sum_i \sum_j \sum_f (sp_{ijf}.U_{ijf}) - (fc_1 + fc_2 + fc_3).Cmax$$
$$- \sum_i \sum_j \sum_f \sum_{k \in a_{ijf}} (c_{ijkf}.T_{ijkf})$$
$$- \sum_i \sum_j \sum_f \sum_{k \in a_{ijf}} (c'_{ijkf}.(ut_{ijkf} - T_{ijkf}))$$
$$- \sum_i \sum_j \sum_f (pc_{ijf}.V_{ijf}).$$

3. MILP formulation

Here, a MILP formulation is presented for the problem. This MILP model simultaneously makes a plan for buying and selling the WIPs, determines the values of the processing times of the operations selected for processing, and makes a scheduling plan for the operations.

Minimize

$$Obj = \sum_i \sum_j \sum_f (sp_{ijf}.U_{ijf}) - (fc_1 + fc_2 + fc_3).Cmax - \sum_i \sum_j \sum_f \sum_{k \in a_{ijf}} (c_{ijkf}.T_{ijkf}) -$$
$$\sum_i \sum_j \sum_f \sum_{k \in a_{ijf}} (c'_{ijkf}.a'_{ijkf}) - \sum_i \sum_j \sum_f (pc_{ijf}.V_{ijf}),$$

Subject to:

$$(\sum_{k' \in a_{ijf}} CT_{ijk'f}) - (CT_{i(j+1)kf} - T_{i(j+1)kf}) \le M.(1 - X_{i(j+1)kf}) + M.U_{ijf}$$

$$\forall i,\ \forall j < J_i,\ \forall f,\ k \in a_{i(j+1)f},$$

$$CT_{ijkf} - T_{ijkf} - CT_{i'j'kf} \ge -M.(1 - Y_{iji'j'}) - M.(2 - X_{ijkf} - X_{i'j'kf})$$

$$\forall i,\ \forall j \le J_i,\ \forall i' > i,\ \forall j' \le J_{i'},\ \forall f,\ k \in (a_{ijf} \cap a_{i'j'f}),$$

$$CT_{i'j'kf} - T_{i'j'kf} - CT_{ijkf} \ge -M.Y_{iji'j'} - M.(2 - X_{ijkf} - X_{i'j'kf})$$

$$\forall i,\ \forall j \le J_i,\ \forall i' > i,\ \forall j' \le J_{i'},\ \forall f,\ k \in (a_{ijf} \cap a_{i'j'f}),$$

$$ut_{ijkf} - T_{ijkf} \le a'_{ijkf} + M.(1 - X_{ijkf}) \qquad \forall i,\ \forall j \le J_i,\ \forall f,\ k \in a_{ijf},$$

$$(\sum_{k \in a_{ijf}} X_{ijkf}) - (\sum_{j' \le j} V_{ij'f}) - (\sum_{j^* \ge j} U_{ij^*f}) >= -1 \qquad \forall i,\ \forall j \le J_i, \forall f,$$

$$M.(1 - X_{ijkf}) + CT_{ijkf} - T_{ijkf} \ge 0 \qquad \forall i,\ \forall j \le J_i,\ \forall f,\ k \in a_{ijf},$$

$$CT_{ijkf} + T_{ijkf} + a'_{ijkf} - M.X_{ijkf} \le 0 \qquad \forall i,\ \forall j \le J_i,\ \forall f,\ k \in a_{ijf},$$

$$C\,max \ge \sum_f \sum_{k \in a_{ijf}} CT_{ijkf} \qquad \forall i, \forall j \le J_i,$$

$$(\sum_j \sum_f V_{ijf}) \le 1 \qquad \forall i,$$

$$(\sum_j \sum_f U_{ijf}) \le 1 \qquad \forall i,$$

$$V_{ijf} - (\sum_{j' \ge j} U_{ij'f}) \le 0 \qquad \forall i,\ \forall j,\ \forall f,$$

$$(\sum_j U_{ijf}) - (\sum_{j'} V_{ij'f}) = 0 \qquad \forall i,\ \forall f,$$

$$lt_{ijkf} \le T_{ijkf} + M.(1 - X_{ijkf}) \qquad \forall i,\ \forall j \le J_i,\ \forall f,\ k \in a_{ijf},$$

$$T_{ijkf} \le ut_{ijkf} + M.(1 - X_{ijkf}) \qquad \forall i,\ \forall j \le J_i,\ \forall f,\ k \in a_{ijf},$$

$$X_{ijkf} \in \{0,1\} \qquad \forall i,\ \forall j \le J_i,\ \forall f,\ k \in a_{ijf},$$

$$Y_{iji'j'} \in \{0,1\} \qquad \forall i,\ \forall j \le J_i,\ \forall i' > i,\ \forall j' \le J_{i'},$$

$$U_{ijf}, V_{ijf} \in \{0,1\} \qquad \forall i,\ \forall j \le J_i,\ \forall f.$$

The objective (1) is to maximize the profit that is the total revenue of selling the WIPs minus total costs including total processing costs, total fixed costs and the total costs of procuring the WIPs. Constraint set (2) ensures that the precedence restrictions between consecutive operations of each job are met, it means that the completion time of jth operation of job *i* should not be greater than the start time of its (*j*+1)th operation. Constraint sets (3) and (4) define the order of any two operations processed on the same machine and ensure that they will not clash. For any two operations (*i, j*) and (*i', j'*), (*i*≠*i'*), constraint (3) is active only if the two operations can be processed on the same machine (k) (i.e. $X_{ijkf} = X_{i'j'k'f'} = 1$) and operation (i, j) is processed after operation (*i', j'*) (i.e. $Y_{iji'j'} = 1$); otherwise, i.e. if $X_{ijkf} = 0$ or $X_{i'j'k'f'} = 0$ or $Y_{iji'j'} = 0$, this constraint is inactive. Note that, operation (*i, j*) is not necessarily positioned immediately after operation (*i', j'*) when $Y_{iji'j'} = 1$. Clearly, for any two operations processed on the same machine, either constraint (3) or constraint (4) is active. a'_{ijkf} in constraint set (5) is the difference between the processing time of operation (*i,j,k,f*) and its upper limit, and is used for computing the processing cost of the operation in the objective function. Constraint set (6) means that each operation (*i, j*) to be processed in FMU f is assigned to one machine. According to constraint set (7), if an operation is selected for processing and scheduling (i.e. $X_{ijkf} = 1$), then its completion time must be greater than or equal to its processing time, i.e. its starting time must be greater than or equal to time zero. If operation (*i,j,k,f*) is not selected for processing (i.e. $X_{ijkf} = 0$), then constraint (8) sets CT_{ijkf} , T_{ijkf} and a'_{ijkf} equal to zero. Constraint set (9) determines the overall completion time (makespan), i.e. the completion time of each operation on each machine and each FMU should not be greater than the makespan. Constraint set (10) ensures that for each job, at most one WIP is bought by the whole company. Constraint set (11) ensures that for each job, at most one WIP can be sold by the whole company. Constraint set (12) means that if a WIP of a job is to be bought, then one of its subsequent WIPs must be sold. According to constraint set (13), for each job, the number of bought WIPs is equal to the number of sold WIPs, i.e. either no WIP is bought and no WIP is sold or one WIP is bought and one WIP is sold. Constraints (14) and (15) ensure that the processing time of each operation (*i,j,k,f*) does not

violate its lower limit (lt_{ijkf}) and upper limit (ut_{ijkf}). Constraint sets (16), (17) and (18) define the binary variables X_{ijkf}, and $Y_{jii'j'}$, U_{ijf} and V_{ijf}, respectively.

4. Proposed heuristic approach

In order to reduce the computational time, instead of solving the MILP model presented in the previous section, we propose an efficient heuristic method for the problem. An outline of the proposed heuristic algorithm is given in Figure 1. This method consists of four algorithms run consecutively and described as follows.

Apply Algorithm 1 to find a solution in which, for each job, the first WIP is bought and the last WIP is sold, i.e.

$\sum_{f=1}^{l} V_{i1f} = \sum_{f=1}^{l} U_{iJ_i f} = 1, \forall i.$

- Beginning of Algorithm 1 (Ziaee, 2014)

for *j*:= 1 to *maxJ* do
{
until the *j*th operation of all jobs are scheduled, repeat
{

- Find *i, k, f* (such that: 1. (*j*=1), or (*j*>1 and job *i* is assigned to FMU *f*); 2. $j \le J_i$; and 3. *j*th operation of job *i* is an unscheduled operation and machine *k* of FMU *f* is capable of processing this operation) that minimizes *Tcrit*.

- If job *i* is an unassigned job (i.e. a job that is not already assigned to any FMU), then assign it to FMU *f*; and schedule *j*th operation of this job at the last position of current partial sequence on machine *k* of FMU *f*.

}
}

- End of Algorithm 1
- Repeat:
  - Apply Algorithm 2 to improve the quality of solution obtained by Algorithm 1,
  - Apply Algorithm 3 to improve the quality of solution obtained by Algorithm 2,

  Until (no improvement occurs over the best solution)
- Apply Algorithm 4 to improve the quality of solution obtained by Algorithms 2 and 3.

Figure 1. General outline of the proposed heuristic algorithm

4.1 Algorithm 1

This algorithm is based on the heuristic algorithm presented in (Ziaee, 2014). We have revised this algorithm in order to cover the WIPs supply planning and bounded processing times. In this algorithm, each unscheduled operation (*i, j*) (i.e. operation *j* of job *i*) to be scheduled on machine *y* of FMU *f* is evaluated by the following criterion, and the unscheduled operation with minimum *Tcrit* is selected for scheduling.

$Tcrit = \sum_{r=1}^{6} W_r . crit_r ,$

such that,

$crit_1 = ((\max(Cmax_{yf}, CT_{i,j-1,f}) + T_{ijyf}) - Cmax_{yf}).tfc$
$crit_2 = \max(0, (CT_{i,j-1,f} - Cmax_{yf}))$
$crit_3 = T_{ijyf} . c_{ijyf} + (ut_{ijyf} - T_{ijyf}).c'_{ijyf}$
$crit_4 = \begin{cases} \sum_{x=1}^{J_i} (pc_{ixf} - sp_{ixf}) / J_i & \text{if } j = 1 \\ 0 & \text{if } j > 1 \end{cases}$
$crit_5 = sk_{yf}/n$
$crit_6 = sj_{if}/maxJ$

Tcrit is weighted sum of some criteria which are established based on the factors affecting the objective function value. W_r (r=1,2,…,6) are integer variables used to increase the flexibility and effectiveness of criterion *Tcrit*. The coefficients W_r are variables bounded in a given range and used to refine the *Tcrit*. $Cmax_{yf}$ is the maximum completion time across all the operations scheduled on machine *y* of FMU *f;* that is, $Cmax_{yf}$ is equal to the completion time of the operation situated just before operation *j* of job *i* on machine *y* of FMU *f*. $crit_1$ is equal to the fixed cost relating to the scheduling of operation *j* of job *i* on machine *y* of FMU *f*. $crit_1$ and $crit_2$ are applied to decrease $Cmax_{yf}$ and idle times, respectively; clearly, both these objectives affect the makespan. For assigning operations to a machine, their processing costs are also taken into account by $crit_3$. $crit_4$ is the average procurement cost minus average selling price and computed only for the first operation of each job, leading to a good assignment of the job to one FMU. $crit_5$ and $crit_6$ are used for taking into account the total weighted processing time of machines, and the total mean processing time of jobs, respectively. It must

be noted that these criteria ($crit_1$ to $crit_6$) do not have the same dimension, however, the algorithm itself adjusts their effects by using variables Wr.

In Algorithm 1, all the processing times (T_{ijyf}, $\forall$ i,j,y,f) are set to either their upper limit (ut_{ijyf}) or their lower limit (lt_{ijyf}). Binary variable W_7 is applied for setting the values of processing times (i.e. either the upper limits or the lower limits), it takes a value of 1 for upper limits and 0 for lower limits. For other details concerning Algorithm 1 refer to (Ziaee, 2014). Pseudocode of this algorithm is available by request to the author.

4.2 Algorithm 2

Algorithm 2 improves the quality of solution obtained by Algorithm 1. The pseudocode of this algorithm is shown in Figure 2. At each step of this algorithm, the value of *crit'* is first computed for all scheduled operations. All the scheduled operations are then sorted in increasing order of their $crit'_{ijf}$, and for each operation taken in this order, the elimination of the corresponding WIP is tested, i.e. if this elimination can lead to better objective function value, then it is eliminated from the scheduling plan. $crit'_{ijf}$ is the relative profit resulting from eliminating the corresponding WIP.

Repeat:
For f= 1 to l do
For i= 1 to n do
For j= 1 to J_i do
For y= 1 to m do
{
If (V_{ijf}=0 and U_{ijf}>0 and machine y is capable of processing operation (i,j,f)) then
{
$crit'_{ijf} = sp_{ijf} - sp_{i(j-1)f} - c_{ijyf} \cdot T_{ijyf} - tfc \cdot T_{ijyf}$.
}
If (V_{ijf}>0 and U_{ijf}>0 and machine y is capable of processing operation (i,j,f)) then
{
$crit'_{ijf}=sp_{ijf} - pc_{ijf} - c_{ijyf} \cdot T_{ijyf} - tfc \cdot T_{ijyf}$.
}
If (V_{ijf}>0 and U_{ijf}=0 and machine y is capable of processing operation (i,j,f)) then
{
$crit'_{ijf} = pc_{i(j+1)f} - pc_{ijf} - c_{ijyf} \cdot T_{ijyf} - tfc \cdot T_{ijyf}$.
}
}

Sort all operations (i,j,f) in increasing order of their $crit'_{ijf}$, and for each operation taken in this order do:
{
If (V_{ijf}=0 and U_{ijf}>0 and machine y is capable of processing operation (i,j,f)) then
{
U_{ijf} = 0,
$U_{i(j-1)f}$ = 1,
}
If (V_{ijf}>0 and U_{ijf}>0 and machine y is capable of processing operation (i,j,f)) then
{
V_{ijf} = 0,
U_{ijf} = 0,
}
If (V_{ijf}>0 and U_{ijf}=0 and machine y is capable of processing operation (i,j,f)) then
{
V_{ijf} = 0,
$V_{i(j+1)f}$ = 1,
}

Considering the eliminated operation, compute the completion time of all remaining operations without changing their position in the schedule. If the objective value of the obtained solution is less than the best objective value obtained so far, then consider it as the best solution obtained so far.
}
Until (no improvement occurs over the best solution)

Figure 2. Pseudocode of Algorithm 2

4.3 Algorithm 3

Algorithm 3 improves the quality of solution obtained by Algorithm 2. The pseudocode of this algorithm is shown in Fig. 3. In this algorithm, the value of *crit''* is first computed for all scheduled operations. All the scheduled operations are then sorted in increasing order of their $crit''_{ijf}$, and for each operation taken in this order, the decreasing of the processing time of the operation up to ($ut_{ijyf} - lt_{ijyf}$) is tested, i.e. if this decrement can yield better objective function value, then its processing time is decreased up to ($ut_{ijyf} - lt_{ijyf}$). $crit''_{ijf}$ is the relative profit resulting from decreasing the processing time of the operation.

For the solution generated by Algorithm 2 (denoted by δ^*) perform the following algorithm.

For f= 1 to l do
For i= 1 to n do
For j= 1 to J_i do
For y= 1 to m do
{
If (operation (i,j,f) is a scheduled operation in δ^*, and machine y is capable of processing this operation) then
{
$crit''_{ijf} = (c'_{ijyf} - c_{ijyf} - tfc).(ut_{ijyf} - lt_{ijyf})$,
}

Sort all operations (i,j,f) in increasing order of their $crit''_{ijf}$, and for each operation taken in this order, if processing time of this operation is set to its upper limit, then set the processing time to its lower limit; and considering this change, recompute the completion time of all operations without changing their position in the schedule. If the objective value of the obtained solution is less than the best objective value obtained so far, then consider it as the best solution obtained so far.
}

Figure 3. Pseudocode of Algorithm 3

4.4 Algorithm 4

In this algorithm, it is first assumed that no WIP is bought, i.e. a solution in which the value of *Cmax* and objective function value are equal to zero. The algorithm considers the best solution obtained by Algorithm 1 (denoted by δ), and for each operation scheduled in δ, the buying of the WIP corresponding to this operation is tested. If the buying of this WIP can improve the objective function value then it is selected to be bought and scheduled in new generated solution (δ') such that its machine and its position are the same as those in δ. The pseudocode of this algorithm is shown in Fig. 4. Let $Cmax(\delta')$ be the makespan across all the operations scheduled in δ', and $Cmax_{yf}$ be the maximum completion time across all the operations scheduled on machine y of FMU f.

Assume that no WIP is bought, i.e. a solution in which the value of *Cmax* and objective function value are equal to zero, and call this solution δ' with the objective value called Obj_1. Consider the schedule generated in Algorithm 1 (denoted by δ) and perform the following algorithm.

For j= 1 to $maxJ$ do
For f= 1 to l do
For i= 1 to n do
For y= 1 to m do
{
If ($j \le J_i$, and operation (i,j,y,f) is a scheduled operation in δ) then
{
If ($pc_{ijf} - sp_{ijf} + c_{ijyf}.lt_{ijyf} + tfc.(\max(0, Cmax_{yf} + \max(0, CT_{i(j-1)f} - Cmax_{yf}) + lt_{ijyf} - Cmax(\delta'))) + c'_{ijyf}.(ut_{ijyf} - lt_{ijyf}) < 0$)
{
If (j=1)
{
V_{ijf} = 1,
U_{ijf} = 1,
}
If (j>1 and $U_{i(j-1)f}$=0 and $\sum_{x=1}^{j-1} V_{ixf} = 0$)
{
V_{ijf} = 1,
U_{ijf} = 1,
}
}
If($sp_{i(j-1)f} - sp_{ijf} + c_{ijyf}.lt_{ijyf} + tfc.(\max(0, Cmax_{yf} + \max(0, CT_{i(j-1)f} - Cmax_{yf}) + lt_{ijyf} - Cmax(\delta'))) + c'_{ijyf}.(ut_{ijyf} - lt_{ijyf}) < 0$)
{
If (j>1 and $U_{i(j-1)f}$>0)
{
$U_{i(j-1)f}$ = 0,
U_{ijf} = 1,
}
}
}

If (U_{ijf} > 1)
{
Schedule operation (i,j,f) in δ' assuming that its machine and its position are the same as those in δ. Considering this change, recompute the completion time of all operations in δ'. If the objective value of the obtained solution is less than Obj_1, then set Obj_1 to it.
}
}

If the objective value of the best obtained solution (i.e. Obj_1) is less than the best objective value obtained so far, then consider it (i.e. the solution corresponding to Obj_1) as the best solution obtained so far.

Figure 4. Pseudocode of Algorithm 4

5. Computational results

In this section, the results of computational experiments are presented. As mentioned earlier, to the best of our knowlede, there is no reference in the literature regarding the general problem defined in this paper and so there is no mathematical model and no solution method for the problem in the literature. Therefore, we compared the computational results of the proposed heuristic method with those of the proposed MILP model. First, some preliminary experiments have been conducted for the parameter settings of Algorithm 1. Regarding the test on various values for the parameters of this algorithm and considering the computational results, we have used the settings of Table 1 for benchmarking the presented heuristic. As it can be seen in the table, we let variables W_r to take the value 0 because of their different dimensions.

Table 1. Parameter settings for Algorithm 1

Parameter	L_W_1	L_W_2	L_W_3	L_W_4	L_W_5*	L_W_6*	L_W_7	L_W_8	L_W_9
Value	0	0	0	0	–2	0	0	0	0
Parameter	U_W_1	U_W_2	U_W_3	U_W_4	U_W_5	U_W_6	U_W_7	U_W_8	U_W_9
Value	1	1	1	1	0	2	1	1	1

* $W_6 \in \{-2,0\}$, $W_7 \in \{0,2\}$; i.e. the step length for W_6 and W_7 is 2, and the step length for all other W_r is 1.

The benchmark problems used were derived from 20 FJSP benchmark problems of (Fattahi et al., 2007) (denoted by SFJS1-SFJS10 and MFJS1-MFJS10). The instances have been achieved by supposing that for each instance, all the FMUs share the same characteristics and by replicating the data describing the related FJSP for each FMU. In other words, for each instance, the following data are the same for all the FMUs: n, m_f, J_i, a_{ijf}, lt_{ijyf}, ut_{ijyf}, c_{ijyf}, c'_{ijyf}. The data related to n, m_f, J_i and a_{ijf} are the same as those from (Fattahi et al., 2007). The values of lt_{ijyf} are also the same as T_{ijy} (processing times) in (Fattahi et al., 2007). Other parameters have been obtained by the following equations:

fc_f = INT(Random(0,1)*50)+50, for instances: SFJS1-SFJS10,
fc_f = INT(Random(0,1)*100)+100, for instances: MFJS1-MFJS10,
pc_{ijf} = INT(Random(0,1)*10000)+10000(j+1), for instances: SFJS1-SFJS10,
sp_{ijf} = INT(Random(0,1)*10000)+ 10000(j+3), for instances: SFJS1-SFJS10,
pc_{ijf} = INT(Random(0,1)*10000)+10000(j), for instances: MFJS1-MFJS10,
sp_{ijf} = INT(Random(0,1)*10000)+ 10000(j+3), for instances: MFJS1-MFJS10,
ut_{ijyf} = 1.3 lt_{ijyf}, c_{ijyf} = INT(Random(0,1)*15)+5, c'_{ijyf} = 2 c_{ijyf}.

Random(0,1) is a random real number between 0 and 1 generated from uniform distribution. The function INT(α) rounds the real number α down to the nearest integer. Cases with two, three or four FMUs have been taken into account, leading to a total of 60 instances. The distance from FMUs to jobs has been considered negligible.

The platform of our experiments was a Pentium IV, 2.2 GHz and 2.0 GB RAM PC. All the benchmark instances were solved by the proposed MILP model and by using the software LINGO Release 8.0 (LINDO Systems Inc., 1999) that uses the algorithm of Branch and Bound to optimally solve the problem. We used the default settings of the solver. The running time limit for each instance was 1800 CPU seconds and no MFJS instance could be optimally solved in this time. The algorithms of the heuristic were coded in C language. The computational results for the three categories of the problem instances, i.e. the instances with two, three and four FMUs are shown in Table 2(a,b). In this table, the first column indicates the name of each problem instance. The second column refers to the size of each instance consisting of four numbers which mean the number of jobs, operations, machines and FMUs, respectively. Note that, in any of these test problems, all the jobs have the same number of operations. The first set of columns denoted by *MILP* contains the results of the proposed MILP model obtained by the Lingo software, and the second set denoted by *Heuristic* contains those of the proposed heuristic approach. *Obj* and *CPU time (s)* represent the objective function value and the computational time in seconds, for each instance, respectively. It must be noted that, the computational time of the

heuristic for SFJS instances was very low and only one sec on average, so its elapsed times are not reported in the table; and the computational time of the MILP for all MFJS instances is 1800 sec as mentioned earlier. RPD is the relative percentage deviation and calculated as follows:

$$RPD = \frac{Obj^{**} - Obj^{*}}{Obj^{**}} \times 100 ,$$

where Obj^* is the objective function value obtained by the corresponding method and Obj^{**} is the best objective function value achieved by the two models (i.e. either the MILP or the Heuristic). The best values of variables W_r (i.e. W_r^*), r=1,2,…,9; are also reported in the table. The average value of each variable W_r, r=1,2,…,6 can be considered as the relative effect of the corresponding criterion on the quality of solutions. For example, the average value of W_2 is near zero in the table that means $crit_2$ has little effect on *obj*. Of course, as it can be seen, the values of each variable W_r, r=1,2,…,6 have relatively high variance in the table, meaning that they are dependent on the specifications of problem instance under consideration and on the values of other variables W_r. The proposed algorithm selects for each instance, the best combination of values of W_r, leading to the best result. Average value of W_5 is negative in the table, that means it has adverse effect on *obj*. Average value of W_9 in the table is nearer 0 than 1. It is because the machines with larger sk_{yf} which are firstly selected for scheduling have more sensibility and effect on the objective value. As shown in the table, average RPD value for the MILP and SFJS instances is 0 compared to 3.75 for the heuristic; and the average computational time for the heuristic is one sec compared to 23.63 sec for the MILP. Similarly, for the MFJS instances, average RPD value for the MILP is 5.76 compared to 5.07 for the heuristic; however, the average computational time for the heuristic is 206.25 sec compared to 1800 sec for the MILP. For the MFJS instances, despite the significant difference between the computational time of the two methods, average RPD value of the heuristic is approximately the same as that of the MILP; the heuristic can also find better objective value than the MILP for 13 out of 30 problems, that means the proposed heuristic is an effective and efficient method, especially for larger size problems.

Table 2(a). Computational results for the proposed MILP model and heuristic method

Name	Size	MILP			Heuristic									
		Obj	CPU time (s)	RPD	Obj	RPD	W1	W2	W3	W4	W5	W6	W7	W8	W9
SFJS1-2FMUs	2.2.2.2	56396	0.00	0.00	56396	0.00	0	0	1	0	-2	2	1	0	0
SFJS2-2FMUs	2.2.2.2	37952	0.00	0.00	37952	0.00	0	0	0	0	0	0	1	0	1
SFJS3-2FMUs	3.2.2.2	65478	1.00	0.00	65312	0.25	1	0	1	1	-2	2	0	0	0
SFJS4-2FMUs	3.2.2.2	58928	1.00	0.00	58600	0.56	1	0	0	1	-2	2	0	0	0
SFJS5-2FMUs	3.2.2.2	75747	1.00	0.00	73366	3.14	0	0	1	1	-2	2	1	0	0
SFJS6-2FMUs	3.3.3.2	70864	8.00	0.00	70754	0.16	0	0	1	1	-2	2	1	0	1
SFJS7-2FMUs	3.3.5.2	71149	5.00	0.00	66704	6.25	0	0	0	1	-2	2	0	0	0
SFJS8-2FMUs	3.3.4.2	66363	17.00	0.00	66098	0.40	0	0	1	1	-2	2	1	0	0
SFJS9-2FMUs	3.3.3.2	93473	15.00	0.00	89884	3.84	0	0	1	1	-2	2	1	0	0
SFJS10-2FMUs	4.3.5.2	72883	89.00	0.00	69831	4.19	0	0	1	0	-2	2	1	0	0
SFJS1-3FMUs	2.2.2.3	46908	1.00	0.00	43749	6.73	1	0	0	1	-2	2	1	0	0
SFJS2-3FMUs	2.2.2.3	31960	0.00	0.00	30741	3.81	1	0	0	0	-2	2	0	0	0
SFJS3-3FMUs	3.2.2.3	67264	2.00	0.00	63229	6.00	0	1	0	0	-2	0	1	1	0
SFJS4-3FMUs	3.2.2.3	32181	3.00	0.00	30863	4.10	0	0	0	1	0	2	0	0	1
SFJS5-3FMUs	3.2.2.3	67631	4.00	0.00	67292	0.50	0	0	1	0	-2	2	1	0	0
SFJS6-3FMUs	3.3.3.3	55891	16.00	0.00	54780	1.99	1	0	0	0	-2	2	0	0	0
SFJS7-3FMUs	3.3.5.3	41731	18.00	0.00	37816	9.38	0	0	1	0	0	2	0	1	0
SFJS8-3FMUs	3.3.4.3	85093	21.00	0.00	82560	2.98	1	0	0	1	-2	2	1	0	0
SFJS9-3FMUs	3.3.3.3	76873	26.00	0.00	75289	2.06	0	1	0	1	0	2	0	0	0
SFJS10-3FMUs	4.3.5.3	59312	146.00	0.00	56944	3.99	1	0	0	1	-2	2	0	0	0
SFJS1-4FMUs	2.2.2.4	50223	1.00	0.00	48902	2.63	0	0	1	0	-2	0	1	0	0
SFJS2-4FMUs	2.2.2.4	42248	1.00	0.00	41513	1.74	1	0	0	1	-2	2	1	0	0
SFJS3-4FMUs	3.2.2.4	53603	6.00	0.00	51492	3.94	1	0	0	0	-2	2	0	0	0
SFJS4-4FMUs	3.2.2.4	33033	3.00	0.00	29312	11.26	0	0	0	0	0	2	0	0	1
SFJS5-4FMUs	3.2.2.4	73937	2.00	0.00	70212	5.04	0	0	0	1	0	0	1	0	0
SFJS6-4FMUs	3.3.3.4	57878	13.00	0.00	54502	5.83	0	0	0	0	0	0	0	1	1
SFJS7-4FMUs	3.3.5.4	55201	10.00	0.00	51674	6.39	0	1	0	0	0	0	1	1	1
SFJS8-4FMUs	3.3.4.4	69883	19.00	0.00	67934	2.79	0	1	0	1	0	2	1	0	0
SFJS9-4FMUs	3.3.3.4	73976	35.00	0.00	71374	3.52	0	0	1	1	-2	2	1	0	0
SFJS10-4FMUs	4.3.5.4	48851	245.00	0.00	44420	9.07	0	0	1	1	-2	2	0	0	0
Average			23.63	0.00		3.75	0.30	0.13	0.40	0.57	-1.40	1.60	0.57	0.13	0.20

Table 2(b). Computational results for the proposed MILP model and heuristic method

Name	Size	MILP		Heuristic										
		Obj	RPD	Obj	CPU time (s)	RPD	W1	W2	W3	W4	W5	W6	W7	W8	W9
MFJS1-2FMUs	5.3.6.2	124282	0.00	121413	10	2.31	0	0	1	0	0	2	0	1	0
MFJS2-2FMUs	5.3.7.2	119693	0.00	107124	12.344	10.50	1	0	1	0	-2	2	0	0	0
MFJS3-2FMUs	6.3.7.2	164710	0.00	160937	22.797001	2.29	0	1	1	0	0	2	0	1	0
MFJS4-2FMUs	7.3.7.2	182612	0.00	173234	33.75	5.14	0	0	1	0	-2	2	0	0	0
MFJS5-2FMUs	7.3.7.2	221849	0.00	183652	43	17.22	0	0	1	1	-2	2	0	0	0
MFJS6-2FMUs	8.3.7.2	251564	0.00	231823	53.639999	7.85	0	0	1	0	0	2	0	0	0
MFJS7-2FMUs	8.4.7.2	185832	6.98	199771	139.078003	0.00	0	1	1	0	-2	2	1	0	0
MFJS8-2FMUs	9.4.8.2	216609	3.54	224553	218.360001	0.00	0	0	1	0	0	0	0	1	0
MFJS9-2FMUs	11.4.8.2	308290	0.00	290876	494.578003	5.65	0	0	0	1	0	2	0	0	1
MFJS10-2FMUs	12.4.8.2	274744	6.40	293533	600.531006	0.00	0	1	1	1	0	2	1	0	0
MFJS1-3FMUs	5.3.6.3	100378	0.00	90924	14.156	9.42	0	0	0	1	0	2	1	0	0
MFJS2-3FMUs	5.3.7.3	112290	0.00	108298	13.157	3.56	1	0	0	1	-2	2	1	0	0
MFJS3-3FMUs	6.3.7.3	133365	1.57	135496	30.514999	0.00	0	1	1	1	-2	2	1	0	0
MFJS4-3FMUs	7.3.7.3	143262	7.47	154828	44.172001	0.00	0	1	1	1	0	2	0	0	0
MFJS5-3FMUs	7.3.7.3	186368	0.00	166471	49.264999	10.68	0	0	1	1	-2	2	1	0	0
MFJS6-3FMUs	8.3.7.3	215946	0.00	181144	84	16.12	0	0	0	1	0	2	0	0	1
MFJS7-3FMUs	8.4.7.3	159075	17.00	191658	185.828995	0.00	0	0	1	1	-2	2	0	0	0
MFJS8-3FMUs	9.4.8.3	210714	0.00	208668	297.078003	0.97	0	0	1	0	0	0	0	1	0
MFJS9-3FMUs	11.4.8.3	207229	19.48	257348	631.937012	0.00	0	0	1	0	-2	2	0	1	0
MFJS10-3FMUs	12.4.8.3	255893	11.44	288952	790.625	0.00	1	0	0	1	-2	2	0	0	0
MFJS1-4FMUs	5.3.6.4	86936	0.00	68222	16.063	21.53	1	0	0	1	-2	2	1	0	0
MFJS2-4FMUs	5.3.7.4	99869	0.00	91247	16.875	8.63	0	1	0	1	0	0	0	0	1
MFJS3-4FMUs	6.3.7.4	121187	0.00	86875	35.905998	28.31	1	0	0	0	-2	0	0	1	0
MFJS4-4FMUs	7.3.7.4	149112	0.00	147122	53.5	1.33	0	0	1	1	-2	2	1	0	0
MFJS5-4FMUs	7.3.7.4	118810	10.60	132892	50.922001	0.00	1	0	0	1	-2	2	0	0	0
MFJS6-4FMUs	8.3.7.4	142371	0.00	141632	88.578003	0.52	0	1	0	0	0	0	0	0	0
MFJS7-4FMUs	8.4.7.4	110570	24.23	145937	224.686996	0.00	0	0	0	0	0	2	0	1	0
MFJS8-4FMUs	9.4.8.4	105725	27.92	146680	306.157013	0.00	0	0	0	0	0	0	1	0	0
MFJS9-4FMUs	11.4.8.4	178345	23.00	231615	670.640015	0.00	0	1	0	0	0	0	0	0	0
MFJS10-4FMUs	12.4.8.4	209749	13.09	241352	955.453003	0.00	0	0	1	1	-2	2	0	0	0
Average			5.76		206.25	5.07	0.20	0.27	0.57	0.53	-1.00	1.53	0.30	0.23	0.10

6. Conclusion

In this paper, for the first time in the literature, we integrated production scheduling decisions and WIPs planning decisions in a distributed environment. The paper investigated the distributed and flexible job shop scheduling problem (DFJSP) under the assumption that the WIPs can be bought from the market instead of manufacturing them in-house, and they also can be sold in the market instead of processing their remaining operations and selling the end products. It was also assumed that the processing time of each operation is a variable that has a known lower limit and a known upper limit. Moreover, there is a processing cost for each operation. We proposed a MILP model to solve this general problem whose optimal solution simultaneously makes a plan for buying and selling the WIPs, makes a scheduling plan for all the FMUs, and determines the values of the processing times of the operations. The optimal solution obtained by this MILP model allows heuristic algorithms to be compared to an optimal solution rather than a solution obtained by another heuristic. A fast heuristic algorithm was also developed to solve the problem very quickly. The two proposed methods, i.e. the MILP model and the heuristic, were compared with each other using some problem instances. Computational results showed that the proposed heuristic is computationally efficient and useful in practical problems. Some points and directions for future research are as follows:

- Different sequences of algorithms 2-4 may lead to better performance of the heuristic.
- At each step of Algorithm 2 ,the elimination of only one WIP was tested. One can simultaneously test the elimination of different sets of WIPs at each step and select one with best objective function value.
- At each step of Algorithm 3, the decreasing of the processing time of each operation up to ($ut_{ijyf} - lt_{ijyf}$) was tested. One can test the decreasing of the processing times in lower amounts, for example 1/2($ut_{ijyf} - lt_{ijyf}$) or 1/3($ut_{ijyf} - lt_{ijyf}$).

References

Baker K. (1974). *Introduction to sequencing and scheduling*. NewYork: Wiley.

Chan F.T.S., Chung S.H. and Chan P.L.Y. (2005). An adaptive genetic algorithm with dominated genes for distributed scheduling problems. *Expert Systems with Applications*, Vol.29, pp. 364–371.

Chan F.T.S., Chung S.H. and Chan P.L.Y. (2006a). Application of genetic algorithms with dominated genes in a distributed scheduling problem in flexible manufacturing. *International Journal of Production Research*, Vol. 44 (3), pp. 523–543.

Chan F.T.S., Chung S.H., Chan L.Y., Finke G. and Tiwari M.K. (2006b). Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach. *Robotics and Computer-Integrated Manufacturing*, Vol. 22, pp. 493–504.

Demir Y. and Isleyen K. (2013). Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modeling*, Vol. 37, pp. 977–988.

Fattahi P., Mehrabad M.S. and Jolai F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, Vol. 18, pp. 331–342.

Gao J., Chen R. and Deng W. (2013). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research*, Vol. 51 (3), pp. 641-651.

Garey M.R., Johnson D.S. and Sethi R. (1976). The complexity of flow shop and job-shop scheduling. Mathematics of Operations Research, Vol. 1 (2), pp. 117–129.

Giovanni L. D. and Pezzella F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European Journal of Operational Research*, Vol. 200, pp. 395–408.

Jia H.Z., Fuh J.Y.H., Nee A.Y.C. and Zheng Y.F. (2002). Web-based multi-functional scheduling system for a distributed manufacturing environment. *Concurrent Engineering-Research and Applications*, Vol.10(1), pp. 27–39.

Jia H.Z., Fuh J.Y.H., Nee A.Y.C. and Zhang Y.F. (2003). A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing*, Vol. 15, pp. 351–362.

LINDO Systems Inc. (1999). *LINGO User's Guide. LINDO Systems Inc*.: Chicago.

Özgüven C., Ozbakır L. and Yavuz Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, Vol. 34, pp. 1539–1548.

Pan C.H. (1997). A study of integer programming formulations for scheduling problems. *International Journal of Systems Science*, Vol. 28 (1), pp. 33–41.

Pinedo M. (2002). *Scheduling: theory, algorithms and systems*. Englewood cliffs, NJ: Prentice-Hall.

Rosenau M.D. (1996). *The PDMA Handbook of New Product Development*. Wiley, New York.

Unlu Y. and Mason S.J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers and Industrical Engineering*, Vol. 58, pp. 785–800.

Wang B. (1997). *Integrated product, process and enterprise design*. Chapman & Hall, London.

Wang L. and Shen W. (2007). *Process Planning and Scheduling for Distributed Manufacturing*. Springer, London.

Wang S.-Y., Wang L., Liu M. and Xu Y. (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics*, Vol. 145 (1). pp. 387–396..

Zhang Q., Manier H. and Manier M.-A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, Vol. 39, pp. 1713–1723.

Ziaee M. (2014). A heuristic algorithm for the distributed and flexible job-shop scheduling problem. *Journal of Supercomputing*, Vol. 67, pp. 69–83.