A multistage-graph based procedure for solving a just-in-time flexible job-shop scheduling problem with machine and time-dependent processing costs

Albert Corominas[a], Alberto García-Villoria[a*],

Néstor-Andrés González[b], Rafael Pastor[a]

[a] *Institute of Industrial and Control Engineering. Universitat Politècnica de Catalunya. Av. Diagonal 647, 11th fl., 08028 Barcelona, Spain.*

[b] *Research Group on Industrial Automation. Universidad del Cauca. Popayán, Colombia.*

Abstract. This paper deals with a new flexible job-shop scheduling problem in which the objective function to be minimised is the sum of the earliness and tardiness costs of the jobs and the costs of the operations required to perform the jobs, the latter depending on the machine and the time interval in which they are performed (as happens in many countries with the costs of electric power or those of manpower). We formalise the problem with a mathematical model and we propose a heuristic procedure that is based primarily on constructing a multistage graph and finding in it the shortest path from the source to the sink. We also describe the generation of the data set used in an extensive computational experiment and expose and analyse the obtained results.

Keywords: scheduling; flexible job-shop problem; just-in-time; heuristics

1. Introduction

The job-shop problem (JSP) is an NP-hard problem (Garey *et al.*, 1976) in which *n* jobs must be performed using *m* machines. Each job consists of an ordered set of operations, each operation has to be executed in one previously specified machine and the processing times are known. The problem consists in finding a feasible schedule of the operations that optimises some measure of the quality of the solution (the makespan, $C_{\max}$, being the most usual).

The flexible job-shop problem (fJSP) differs from the JSP in that each operation can be performed in any machine of those belonging to a specific subset associated with the operation. Therefore, the JSP is a particular case of the fJSP in which the subsets of machines contain a single element.

Hence, the fJSP involves two subproblems: that of assigning machines to the operations and that of sequencing the operations (Armentano *et al.*, 2004). As the latter is a JSP, fJSP is at least as hard as JSP.

The problem addressed in this paper is a generalization of the real problem of scheduling that we had occasion to analyse in some manufacturing plants of compound feed. In many real situations, there is a due date for each job (the desired moment to complete the job) and the costs of the operations depend on the time interval in which they are performed. For instance, this may be due to a higher pay-rate for the workforce

* Corresponding author: Alberto García-Villoria, Institute of Industrial and Control Engineering, Av. Diagonal 647 (Edif. ETSEIB), 11th floor, 08028 Barcelona, Spain; tel.: +34 93 4010724; e-mail: alberto.garcia-villoria@upc.edu.

during the night shifts or to dependency on the hours of the day of the electricity charges (for example, a rate for the valley hours and a higher one for the peaks). These cost differences, depending on the time in which tasks are performed, are very relevant in many industries, such as those in which the cost of electricity is a significant part of total costs, as happens, for instance, in the manufacture of compound feed for livestock. Of course, the cost of performing a task may depend also on the machine to which it is assigned.

According to this, we adopt as the objective function of the problem the sum of the costs of the deviations relative to the due dates (earliness and tardiness) and the machine and time dependent costs of the operations. At the best of our knowledge, this variant of the fJSP has not been dealt with before.

The layout of the rest of the paper is as follows. Section 2 outlines the state of the art concerning the fJSP. Section 3 describes the specific variant of the fJSP dealt with in this paper and it is modelised with a binary integer programming (BIP) model. Section 4 is dedicated to a heuristic proposed procedure for solving the problem. Section 5 gives an account of the computational experiment. Lastly, Section 6 contains the conclusions and some ideas for future work.

2. State of the art

A detailed state of the art of the fJSP can be found in Fattahi *et al.* (2007). The first paper tackling the fJSP is Brucker and Schlie (1990), in which a polynomial algorithm is proposed to solve the fJSP with two jobs.

Concerning the criteria to be optimised, $C_{\max}$ (makespan) is adopted, for instance, in Dini and Rossi (2007), Ham *et al.* (2011) and Ziaee (2013). Other papers (Scrich *et al.*, 2004) use tardiness or an aggregate of earliness and tardiness (Akyol and Bayhan, 2005; Wu and Weng, 2005).

Recently, it is usual to consider the fJSP as a multiobjective problem, as in Zhang *et al.* (2009). Loukil *et al.*, (2007), Saad *et al.* (2008), Gholami and Zandieh (2009), Li *et al.* (2010), Chen *et al.* (2012), and Gao *et al.* (2014) take into account earliness and tardiness among the criteria they adopt. Jiang *et al.* (2014) includes energy consumption, under the assumption that it is machine depending. Energy consumption is, too, one of the criteria considered in Liu and Tiwari (2015) and in He *et al.* (2015).

The consideration of the costs of performing the operations is not usual in the scheduling literature, since it is assumed, often implicitly, that they only depend on the operation and not on the machines where it is processed or on the time when it is performed. Several studies take into account only setup times/costs (Allahverdi, 2015). Minimising energy consumption is related to minimising costs, but is not equivalent, because price may depend on time, as happens with electricity when tariffs are time sensitive (time-of-use ─TOU─ electricity prices). As it is pointed out in Zhang and Chiong (2016), the consideration of TOU tariffs is a new research direction in production scheduling. Luo *et al.* (2013) deals with a multi-objective hybrid flow shop scheduling problem in which electric power cost with the presence of TOU tariffs is considered. In Moon *et al.* (2013), which also assumes TOU prices, the objective is to minimise the weighted sum of makespan and electricity cost. Zhang *et al.* (2014)

proposes a mixed integer programming (MIP) model to find, in a flow shop setting, manufacturing schedules that minimise carbon footprint and electricity cost, under TOU tariffs.

Concerning the solution procedures for the general case (with a number of jobs greater than two), two approaches, hierarchical and integrated, have primarily been used. In the hierarchical approach, the two subproblems (assigning machines to operations and sequencing the operations) are dealt with separately. Integrated approaches, on the other hand, tackle assignment and sequencing simultaneously.

Regarding the hierarchical approach, Brandimarte (1993) was the first to adopt it, using tabu search. Arkat *et al.* (2009) use simulated annealing and compare the results with the optimal solutions obtained using branch-and-bound. De Giovanni and Pezzella (2010) use a hierarchical procedure that combines a genetic algorithm with local search procedures.

The integrated approach is adopted by Hurink *et al.* (1994) and by Dauzère-Pérès and Paulli (1997), who use tabu search. Mastrolilli and Gambardella (2002) present two local optimisation procedures that improve the technique proposed by Dauzère-Pérès and Paulli (1997). Hmida *et al.* (2010) use discrepancy search, and Thammano and Phuang (2013) use a hybrid artificial bee colony algorithm. Gao *et al.* (2014) solve the fJSP using the so-called discrete harmony search algorithm and Gao *et al.* (2015) use the same approach considering fuzzy processing times. González *et al.* (2015) apply path relinking and tabu search in the frame of scatter search. Fattahi *et al.* (2009), Roshanaei *et al.* (2013) and Birgin *et al.* (2014), propose mixed integer linear programming (MILP) models to optimise fJSP.

Summing up, regarding the types of problems treated in the literature, there are papers that consider earliness and tardiness as criteria to optimise in the fJSP and others, dealing with different scheduling problems, which take into account machine or time dependent costs. However, to the best of our knowledge there is no published work on the fJSP with the objective function considered in the present paper, i.e., the sum of the costs of earliness and tardiness and the costs of performing the operations, dependent on the machine and on the time interval in which the operations are processed.

3. Description and formulation of the problem

First, we describe the problem dealt with in this paper (Subsection 3.1). Then, we formalise it with a binary integer programming (BIP) model (Subsection 3.2). The model is useful for a formalisation purpose and for obtaining benchmark solutions of small instances as well.

3.1. Description of the problem

In the following, we present the specific variant of the fJSP dealt with in this paper, in which n jobs $(j = 1,...,n)$ must be performed using m machines $(i = 1,...,m)$. Each job j consists of an ordered set of h_j operations, where O_{jh} $(j = 1,...,n;\ h = 1,...,h_j)$ denote the h-th operation of job j. Each operation has to be performed in any machine

of those belonging to a specific subset associated with the operation, M_{jh} $\left(j=1,...,n;\ h=1,...,h_j\right)$. Processing operation h of job j $\left(O_{jh}\right)$ in machine i $\left(i\in M_{jh}\right)$ requires a predetermined processing time, p_{ijh}, and a predefined amount of a resource, such as energy, P_{ijh} $\left(j=1,...,n;\ h=1,...,h_j;\ i\in M_{jh}\right)$. Moreover, each job j has a release date (earliest start date), r_j.

The problem consists in finding a feasible schedule of the operations that minimises the sum of the costs of earliness and tardiness of the n jobs, with respect to their due date d_j $\left(j=1,...,n\right)$, and the costs of performing the operations, dependent on the machine and on the time interval in which the operations are processed (Eq. 1). Concerning the costs corresponding to deviations from the due date, we have assumed that they are linear and quadratic relative to earliness and tardiness, respectively (however, these assumptions can be modified without altering the structure of the proposed algorithm). Regarding the costs of performing the operations, to fix ideas hereinafter we will identify them with those of the electrical energy required to process the operations on the machines (to adapt the procedure to other kinds of costs depending on the machines and on the time interval it suffices to change the terminology).

$$MIN\ [z]=\sum_{j=1}^{n}\left[C_E\left(E_j\right)+C_T\left(T_j\right)+U_j\right] \tag{1}$$

Therefore, the objective function includes three kinds of variables:

- The earliness cost of job j with respect to its due date d_j:

$$C_E\left(E_j\right)=\delta_j\cdot E_j \tag{2}$$

  where $E_j=\max\left(0,d_j-c_j\right)$ is the earliness, c_j is the completion time of job j (i.e., the instant when the last operation O_{jh_j} is finished), and δ_j $\left(\delta_j\geq 0\right)$ is the parameter that specifies the linear relation between the earliness and its cost.

- The tardiness cost of job j with respect to its due date d_j:

$$C_T\left(T_j\right)=\beta_j\cdot T_j^2+\gamma_j\cdot T_j \tag{3}$$

  where $T_j=\max\left(0,c_j-d_j\right)$ is the tardiness, β_j $\left(\beta_j>0\right)$ and γ_j $\left(\gamma_j\geq\delta_j\right)$ are the coefficients that specify the quadratic cost function of the tardiness of job j; we assume that $\gamma_j\geq\delta_j$ to enforce that for any given value of the discrepancy from the due date, the earliness cost is lower than the tardiness cost.

- U_j is the cost of the energy required to produce job j:

$$U_j=\sum_{h=1}^{h_j}U_{i_{(jh)}jh} \tag{4}$$

$$U_{i_{(jh)}jh} = \left(\Omega_{jh} \cdot C^V + \left(1-\Omega_{jh}\right) \cdot C^P\right) \cdot P_{i_{(jh)}jh} \quad (5)$$

where $i_{(jh)}$ $\left(i_{(jh)} \in M_{jh}\right)$ is the machine to which operation h of job j $\left(O_{jh}\right)$ has been assigned; $U_{i_{(jh)}jh}$ corresponds to the cost of the energy required to process operation O_{jh} on machine $i_{(jh)}$; Ω_{jh} and $\left(1-\Omega_{jh}\right)$ indicate the proportion of operation O_{jh} processed during the valley and the peak hours, respectively; C^V and C^P are the unit costs of energy at valley and peak hours, respectively; and $P_{i_{(jh)}jh}$ is the energy required to process operation O_{jh} on machine $i_{(jh)}$.

Next, there is a summary of the assumptions of the solved problem:

1. The sequence of operations for each job is fixed.
2. There are no precedence constraints among operations of different jobs.
3. Each machine can process at most one operation at a time.
4. The time is discretised in regular intervals. Without loss of generality, all times are considered multiple of the time intervals.
5. The operations cannot be interrupted.
6. There is unlimited buffer space between the machines.
7. For each job j the release date, r_j, is also known.
8. Ideally, each job j should be completed on its due date d_j.
9. The set-up times are independent of the sequence and are included in the processing times of the jobs.
10. The initial availability of the machines is known and it is defined by the set of time intervals, Q_i, at which machine i $\left(i = 1,...,m\right)$ is initially occupied by previously scheduled operations.
11. Each machine i requires a predefined amount of energy, P_{ijh}, to process each operation O_{jh}.
12. The costs of processing the operations depend on the machine and on the time interval in which they are performed.

Assumptions 1 to 6 are common in scheduling problems and compatible with the real problem underlying that we are dealing with in this article. Accepting the possibility of releasing dates different from 0 is less usual, because it adds difficulty, but it overcomes the rigidity of imposing that all of them are null. As the unavailability of the machines during some periods is seldom considered, assumption 10, which is essential in many real settings, contributes also to make the formalization of our problem more general. Assumptions 11 and 12, as well as the objective function, for which the consideration of due dates is essential, distinguishes the fJSP considered in this work from previous fJSPs. Assumption 9 is somehow restrictive, since it does not fit in many real settings; although the heuristic algorithm that we propose could be adapted straightforwardly to take into account sequence dependent set-up times when computing the completing times of the operations, their consideration in the mathematical programming model and

in the Step 1 (Section 4.1) of the heuristic requires further research, as we point out in Section 6.

3.2. A BIP model

Ku and Beck (2016) compare four mathematical models for the JSP with the objective of minimising the makespan. The best one is called *disjunctive* MIP model because of the way in which the time-overlapping of the operations in the machines is avoided. Thus, the model that we propose also uses *disjunctive* non-overlapping constraints. The model is formulated as follows:

Additional data

T_{ijh} set of integer times in which operation O_{jh} can start on machine i tacking into account the release dates of the jobs (r_j) and the initial occupancies of the machines ($j=1,...,n;\ h=1,...,h_j;\ i \in M_{jh}$).

c_{ijht} cost associated with the start of operation O_{jh} on machine i at time t ($j=1,...,n;\ h=1,...,h_j;\ i \in M_{jh};\ t \in T_{ijh}$). If operation O_{jh} is not the last operation of job j ($h \le h_j - 1$), c_{ijht} is only the energy cost of performing O_{jh} at machine i starting at time t; otherwise ($h = h_j$), c_{ijht} is the energy cost of performing O_{jh} at machine i starting at time t plus the earliness or tardiness cost of the discrepancy between d_j (due date) and $t + p_{ijh}$ (completion time of job j).

V_{ijh} big enough value used in Eqs. 9 and 10: $V_{ijh} = p_{ijh} + \max_{t \in T_{ijh}} t$

Variables

$x_{ijht} \in \{0,1\}$ 1 if operation O_{jh} is started at instant t on machine i ($j=1,...,n;\ h=1,...,h_j;\ i \in M_{jh};\ t \in T_{ijh}$); 0 otherwise

$z_{ijh\kappa\iota} \in \{0,1\}$ auxiliary variables to ensure that operations O_{jh} and $O_{\kappa\iota}$ are not scheduled on machine i at the same time ($j=1,...,n;\ h=1,...,h_j;\ \kappa = j+1,...,n;\ \iota = 1,...,h_\kappa;\ i \in M_{jh} \cap M_{\kappa\iota}$). If operations O_{jh} and $O_{\kappa\iota}$ are performed in the same machine then $z_{ijh\kappa\iota}$ is 1 if O_{jh} is performed before operation $O_{\kappa\iota}$ and 0 otherwise; if O_{jh} and $O_{\kappa\iota}$ are performed in different machines then the value of $z_{ijh\kappa\iota}$ is irrelevant.

Model

$$[\text{MIN}] \sum_{j=1}^{n} \sum_{h=1}^{h_j} \sum_{i \in M_{jh}} \sum_{t \in T_{ijh}} c_{ijht} \cdot x_{ijht} \tag{6}$$

$$\sum_{i \in M_{jh}} \sum_{t \in T_{ijh}} x_{ijht} = 1 \qquad j=1,...,n;\ h=1,...,h_j \tag{7}$$

$$\sum_{i\in M_{jh}}\sum_{t\in T_{ijh}} t\cdot x_{ijht} \geq \sum_{i\in M_{j,h-1}}\sum_{t\in T_{i,j,h-1}} \left(t+p_{i,j,h-1}\right)\cdot x_{i,j,h-1,t} \qquad j=1,...,n;\ h=2,...,h_j \tag{8}$$

$$\sum_{t\in T_{ijh}} t\cdot x_{ijht} \geq \left(\sum_{t\in T_{i\kappa\iota}} \left(t+p_{i\kappa\iota}\right)\cdot x_{i\kappa\iota t}\right) - V_{i\kappa\iota}\cdot z_{ijh\kappa\iota}$$
$$j=1,...,n;\ h=1,...,h_j;\ \kappa=j+1,...,n;\ \iota=1,...,h_\kappa;\ i\in M_{jh}\cap M_{\kappa\iota} \tag{9}$$

$$\sum_{t\in T_{i\kappa\iota}} t\cdot x_{i\kappa\iota t} \geq \left(\sum_{t\in T_{ijh}} \left(t+p_{ijh}\right)\cdot x_{ijht}\right) - V_{ijh}\cdot\left(1-z_{ijh\kappa\iota}\right)$$
$$j=1,...,n;\ h=1,...,h_j;\ \kappa=j+1,...,n;\ \iota=1,...,h_\kappa;\ i\in M_{jh}\cap M_{\kappa\iota} \tag{10}$$

The objective function (6) states the minimisation of the total cost. Constraints (7) impose that each operation starts once and is scheduled on one machine. Constraints (8) impose that all operations of a job are performed in the right order. Finally, constraints (9) and (10) ensure that two operations cannot be performed on the same machine at the same time.

4. A heuristic procedure for solving the problem

Next, we present the structure of the developed solution procedure, which can be considered as an integrated approach (Section 2). The heuristic procedure, named HeufJSP, consists of two sets of iterations. In the first one (lines 1 to 5 of Figure 1), which may be seen as the diversification phase of the algorithm, two steps are repeated for different values of parameter ζ $(0\leq\zeta\leq 1)$. At the first step, a job sequence is obtained; at the second one, when the jobs have been ordered in a sequence, an optimal schedule of the operations of each job is obtained constructing a multistage graph and finding in it the shortest path from the beginning to the end. Once the better value, ζ^*, of the parameter ζ is obtained, a second round of iterations is performed (lines 6 to 10 in Figure 1), in order to intensify the search in a neighbourhood of ζ^*.

Algorithm HeufJSP

ζ : parameter

ζ^*: ζ value from which the best solution is obtained in the first loop

Δ_ζ^1, Δ_ζ^2 ($\Delta_\zeta^2 < \Delta_\zeta^1$): increase values of parameter ζ

Υ, Z : current solution and its value, respectively

Υ^*, Z^*: best obtained solution and its value, respectively

0 $Z^* = \infty$

1 For ($\zeta = 0; \zeta \le 1; \zeta = \zeta + \Delta_\zeta^1$)

2 Π = Initial_Job_Sequence(ζ)

3 Υ = Optimal_Schedule(Π)

4 If $Z < Z^*$ then $\zeta^* = \zeta$, $\Upsilon^* = \Upsilon$, $Z^* = Z$ End if

5 End for

6 For ($\zeta = \max\left(0, \zeta^* - 0.5 \cdot \Delta_\zeta^1\right); \zeta \le \min\left(1, \zeta^* + 0.5 \cdot \Delta_\zeta^1\right); \zeta = \zeta + \Delta_\zeta^2$)

7 Π = Initial_Job_Sequence(ζ)

8 Υ = Optimal_Schedule(Π)

9 If $Z < Z^*$ then $\Upsilon^* = \Upsilon$, $Z^* = Z$ End if

10 End for

Figure 1. Structure of the developed algorithm HeufJSP

When coding HeufJSP, the following two improvements are introduced to reduce cpu time:

- The same sequence of jobs may be obtained with different values of ζ (Subsection 4.1). However, solution Υ is calculated only once for the set of different values of ζ that yield the same sequence of jobs.

- Function Optimal_Schedule(Π) (Subsection 4.2) is applied from the position in which the current job sequence Π differs from the previous job sequence (i.e., the job sequence obtained in the previous iteration).

4.1 *Step 1: Obtaining an initial job sequence*

Before generating a solution, jobs j $(j = 1,...,n)$ are sorted in the non decreasing order of their priority indexes IP_j:

$$IP_j = \zeta \cdot D_j - (1 - \zeta) \cdot A_j \tag{11}$$

where parameter ζ $(0 \le \zeta \le 1)$ is used to prioritise the jobs, D_j is the normalised due date of job j, and A_j is a normalised estimation of the energy required to process job j.

D_j and A_j are obtained as follows: $D_j = \frac{d_j}{\max\limits_{s=1...n} d_s}$ and $A_j = \frac{\lambda_j}{\max\limits_{s=1...n} \lambda_s}$, where $\lambda_j = \sum_{h=1}^{h_j} \sigma_{jh}$ is an estimation of the energy required to process job j, being $\sigma_{jh} = \sum_{i \in M_{jh}} \frac{P_{ijh}}{|M_{jh}|}$ an estimation of the energy required to process operation O_{jh}.

Note that parameter ζ and its complement weight the importance given to the due date and to the estimation of the energy requirements, respectively, when ordering the jobs. For example, if ζ is equal to 1, then the jobs would be sorted according to the EDD (earliest due date) rule.

This way of generating different orderings of the jobs captures an small but logical subset of all possible $n!$ orderings. The adopted approach intends to be a first reasonable compromise between diversifying more and avoiding prohibitive computing times.

4.2 *Step 2: Obtaining an optimal schedule of each job*

After a sequence of jobs is obtained, the subproblem of scheduling optimally the operations of each job j is solved successively according to the order of the jobs in the sequence. When scheduling the operations of a given job, the availability of the machines is that resulting from the decisions corresponding to all the preceding jobs. The solution indicates the start and finish times for processing each operation, as well as the assignment of the operations to the machines.

In order to represent and solve the subproblem of assigning the operations of job j to the machines and the timing of these operations, we propose to construct and find the shortest path in a multistage graph. Although the calculation of the minimum path is done as it advances the construction of graph, for the sake of clarity, we first expose separately the process of constructing the multistage graph (Section 4.2.1) and the process of finding the shortest path (Section 4.2.2). Then, the overall pseudocode for obtaining the optimal schedule of job j is given (Section 4.2.3). Appendix A1 describes the process of obtaining an optimal schedule of a given job for a numerical example.

4.2.1 *Constructing the multistage graph*

Given a job j, a multistage graph with $h_j + 2$ stages (from stage 0 to stage $h_j + 1$) is constructed. The step from stage $h-1$ to stage h $\left(h = 1,...,h_j\right)$ corresponds to the scheduling of operation O_{jh}; the step from stage h_j to stage $h_j + 1$ represents the completion of job j.

The stages of the graph, from stage 1 to stage h_j, contain one or several nodes, v_{ht}, corresponding to the instants t at which the processing of operation O_{jh} can be

completed. In addition, there are two nodes that represent the beginning and end of the graph: node α (at stage 0) and node ω (at stage $h_j + 1$).

The arcs that link nodes at stage $h-1$ to nodes at the next stage h $\left(h = 1,...,h_j\right)$ correspond to the scheduling of operation O_{jh} on a machine i $\left(i \in M_{jh}\right)$. The cost associated with these arcs is that of the energy required to perform the operation, U_{ijh}.

All nodes of stage h_j are linked with node ω of stage $h_j + 1$ (which represents the completion of job j). The costs associated with these arcs correspond to earliness E_j or tardiness T_j with respect to the due date d_j of job j.

If there are two arcs going from the same node at a stage $h < h_j$ to the same node at stage $h+1$ (of course, these arcs correspond to different machines), one of them can be omitted, according to the following rule (rule R1), which comprises two cases:

- Case R1a: If an arc has a cost greater that of another arc, the former is omitted.
- Case R1b: If two arcs have the same cost, only the arc corresponding to machine i with the lower estimation of future workload q_{ijh} is retained; if there is a tie, the retained arc is that corresponding to the machine with the lower value of i.

Case R1b (that is not a dominance rule) is oriented to favour the assignment of operations to the machine with a lower estimation of future workload. q_{ijh} is obtained as the processing time of O_{jh} in machine i plus the sum of the quotients of the processing times of the operations of the other jobs that could still be processed in machine i by the number of machines on which each operation can be processed. Appendix A2 describes the process of calculating q_{ijh} for a numerical example.

A rule for dominance between arcs emanating from the same node at a stage $h \leq h_j - 1$ and leading to different nodes (rule R2) can also be applied:

- Case R2a, for arcs emanating from the same node at a stage $h < h_j - 1$: If the cost of an arc is not better than that of another arc and leads to a node that represents a later point in time, the former arc is omitted, since if it were used instead of the latter the cost of the energy would not be better and nor either would be the cost associated with the difference between the completion time of the job and its due date For instance, in the graph of the example given in Appendix A1 (Figure A1.1), arc $\alpha \xrightarrow{8,m_2} 240$ is omitted because it is dominated by arc $\alpha \xrightarrow{2,m_1} 60$ (see Appendix A1 for the arc notation).
- Case R2b, for arcs emanating from the same node at a stage $h = h_j - 1$ and leading to different nodes $v_{h_j t}$ with $t \geq d_j$: If the cost of an arc is not better than that of another arc and leads to a node that represents a later point in time, the former arc is

omitted because if it were used instead of the latter, the cost of the energy corresponding to the last operation of the job would not be better and the tardiness cost would be worse. In the graph in Figure A1.1, arc $240 \xrightarrow{2,m_2} 2400$ is omitted since it is dominated by arc $240 \xrightarrow{2,m_2} 2340$.

- Case R2c, for arcs emanating from the same node at a stage $h = h_j - 1$ and leading to different nodes $v_{h_j t}$ with $t < d_j$: If the cost of an arc is not better than that of another arc and leads to a node that represents an earlier point in time, the former arc is omitted, since if it were used, the energy cost of performing the last operation of the job would not better and the earliness cost would be worse. In the graph in Appendix A1 (Figure A1.1), arc $240 \xrightarrow{2,m_2} 780$ is omitted since it is dominated by arc $240 \xrightarrow{2,m_2} 1140$.

The arcs omitted because of applying R2a (when the cost of the dominated arc is greater than that of the dominant one), R2b, and R2c could not belong to an optimum path. When dominant and dominated arcs (case R2a) have the same cost, the dominated one can be omitted without detriment of the cost of the optimal path; the way chosen to break the tie favours the assignments that allow completing the operations sooner.

4.2.2 *Finding the shortest path*

The subproblem of assigning operations to machines and the subproblem of sequencing the operations on the machines are optimally solved, for each job j of the ordered sequence Π, by calculating the minimum cost path, between the initial node α and the final node ω. This process is performed simultaneously with the construction of the multistage graph, as mentioned above.

The process to find the shortest path in the multistage graph of job j is carried out using the expressions (12) and (13):

$$\kappa_{0\alpha} = 0 \tag{12}$$

$$\kappa_{ht'} = \min_{\forall t | v_{h-1,t} \in \Gamma^-_{ht'}} \left(\kappa_{h-1,t} + c_{htt'} \right) \quad \left(h = 1, ..., h_j + 1 \right) \tag{13}$$

where: $\kappa_{0\alpha}$ is the value of the initial node α

κ_{ht} is the value of node v_{ht}

$c_{htt'}$ is the cost of the arc that connects node $v_{h-1,t}$ to node $v_{ht'}$

$\Gamma^-_{ht'}$ is the set of nodes at stage $h-1$ that are connected to node $v_{ht'}$

As the least cost path is being calculated, ties may occur between arcs leading to the same node and yielding the minimum value of the expression $\kappa_{h-1,t} + c_{htt'}$. To break these ties we use a rule R3 (that is not a dominance rule), which considers the three following cases:

- Case R3a, for arcs yielding to nodes at stage $h \le h_j$: If the tie is among arcs with the same value associated with the source nodes, $\upsilon_{h-1,t}$, and that correspond to the same machine, the arc that it is retained is that emanating from the node corresponding to the earliest time at which the previous operation $O_{j,h-1}$ is completed.
- Case R3b, for arcs yielding to node ω (at stage h_{j+1}): The arc that will be retained is that emanating from the node corresponding to the earliest time at which the last operation, O_{jh_j}, is completed.
- Case R3c, for arcs yielding to nodes at stage $h \le h_j$: If the tie is among arcs corresponding to different machines, the arc that will be retained is that which corresponds to machine i with a lower estimation of the workload, q_{ijh}. If the tie remains, then the arc that is retained is that emanating from the node corresponding to the earliest time at which the previous operation $O_{j,h-1}$ is completed.

Note that cases R3a and R3b favour that the operations are performed as early as possible, whereas case R3c favours the assignment of the operations to the machines with a possible lower future workload.

Furthermore, rule R4 concerning the relations between the nodes at the same stage $\left(1 \le h \le h_j - 1\right)$ can also be applied. According to this rule, if there are two nodes at the same stage, v_{ht} and $v_{ht'}$, such that $\kappa_{ht} \le \kappa_{ht'}$ and $t < t'$, then node v_{ht} dominates $v_{ht'}$ which is eliminated.

4.2.3 *Procedure for calculating the optimal schedule of a job*

The overall procedure for simultaneously constructing the multistage graph and finding the minimum path is shown in Figure 2.

Given job j:

1 Add node α (stage 0)
2 For each stage $h = 1...h_j$
2 For each node $v_{h-1,t}$ of stage $h-1$
3 Add arcs and nodes $v_{h,t'}$ for $t' = t + \Delta, t + 2 \cdot \Delta, ...$ (where Δ is the time unit) such as there exists at least one arc yielding to that node. If $h \leq h_j - 1$, when an arch yielding to node $v_{h,t'}$ has the lowest possible energy cost, then no more nodes (with $t' > t$) are added because the arcs to these nodes will be dominated according to rule R2. If $h = h_j$, additionally it is ensured that there is at least one node $v_{h,t'}$ such as $t' \geq d_j$
4 Apply rules R1 and R2, and eliminate the nodes $v_{h,t'}$ without arcs yielding to them
5 Find the minimum cost path to nodes $v_{h,t'}$ using rule R3
6 If $h \leq h_j - 1$, eliminate nodes $v_{h,t'}$ according to rule R4
7 End for
8 End for
9 Add node ω and the arcs from nodes to ω
10 Return the minimum path to ω

Figure 2. Procedure for building the graph and calculating the minimum path

5. Computational experiment

The algorithm HeufJSP was coded in Java and the computational experiment was executed on a PC 3.16 GHz Intel Core 2 Duo E8500 with 3.5 GB of RAM. The values of parameters Δ_ζ^1 and Δ_ζ^2 (see Figure 1) are set to 0.1 and 0.01, respectively.

In order to evaluate the heuristic, we generated and solved a set of test instances. The generation of these instances is explained in Section 5.1. The obtained results are discussed in Section 5.2.

5.1. *Test instances*

A set of 2592 instances were generated randomly. Next, we detail the characteristics of these instances:

- m: 10, 20, 30, 40, 50, 60.
- n : 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120.
- h_j: $U[1,10]$ ($j = 1,...,n$), where $U[a,b]$ is a discrete uniform distribution between a and b.
- M_{jh} (set of machines that can process operation O_{jh}; $j = 1,...,n$, $h = 1,...,h_j$). This set is calculated as follows. For each machine $i \in M$, there is a probability pr_j that

machine i can process operation O_{jh} (probability pr_j is the same for all operations of job j) and it is ensured that $M_{jh} \neq \varnothing$ (i.e., O_{jh} can be processed in at least one machine). Probability pr_j (for each job j) is generated at random and 2 scenarios are considered: $pr_j = U[0.1, 0.4]$ (low versatility scenario) or $pr_j = U[0.5, 0.9]$ (high versatility scenario).

- p_{ijh}: $U[1,30]$, $U[1,90]$ or $U[1,240]$ (processing time of operation O_{jh} in machine i; $j = 1,...,n$, $h = 1,...,h_j$, $i \in M_{jh}$).
- P_{ijh}: $U[1.1 \cdot p_{ijh}, 1.9 \cdot p_{ijh}]$ (energy required, in kWh, to process operation O_{jh} in machine i; $j = 1,...,n$, $h = 1,...,h_j$, $i \in M_{jh}$).
- r_j: $U[0,359]$ (release date of job j; $j = 1,...,n$).
- C^P, C^V (costs, in €/kWh, of energy at peak and valley hours, respectively): 0.15 and 0.06, respectively. Each day has 10 peak hours and 14 valley hours and the first instant of the scheduling horizon coincides with the beginning of a peak period.
- $(\beta_j, \gamma_j, \delta_j)$: $(0.02,1,0.90)$, $(1/9,1,0.5)$ or $(0.4,1,0.2)$, coefficients in the objective function related to costs associated with the due dates of job j ($j = 1,...,n$). Note that these scenarios correspond to a low, medium and high ratio between tardiness and earliness costs, respectively.

In order to generate realistic due dates and initial occupancies of the machines, the following two mechanisms are used.

- d_j (due date of job j; $j = 1,...,n$). Let *ins* be an instance in which the values of m, n, h_j, M_{jh}, p_{ijh}, P_{ijh}, r_j, C^P, C^V, β_j, γ_j, δ_j and Q_i have been set. Let *insAux*1 an auxiliary instance in which the values of the aforementioned data, except those of r_j, C^P, C^V and β_j, have been set to the values of instance *ins*; in *insAux*1, the release dates and energy costs are set to 0, and β_j values are set to a very big value. The idea is that the jobs are scheduled as soon as possible when *insAux* is solved with the proposed heuristic with $\zeta = 0$. Let e_j be the instant in which job j is finished in the obtained solution. The due date of job j of instance *ins* is calculated as $d_j = \max(TIM, e_j + \{U[-0.05 \cdot TIM, 2 \cdot TIM]\})$, where $[x]$ returns the integer value closest to x and *TIM* is the expected value of the total processing time of a job; that is, TIM is the product of the expected number of operations and the expected processing time of an operation.
- Q_i (initial occupancy of machine i; $i = 1,...,m$). Let *ins* be the instance to generate in which $Q_i = \varnothing \ \forall i$ (i.e., all machines are fully available from the initial instant) and the values of m, n, h_j, M_{jh}, p_{ijh}, P_{ijh}, r_j, C^P, C^V, β_j, γ_j and δ_j have been set. Let *insAux*2 an auxiliary instance in which the values of the aforementioned parameters, except those of r_j, have been set to the values of instance *ins*; the release dates of the jobs of *insAux*2 are set to 0 (i.e., the jobs are available from the start). First, the due dates of the jobs of *insAux*2 are set using the mechanism described above. Then the proposed heuristic with $\zeta = 0.5$ is applied to

solve $insAux2$. Finally, the initial occupancies of the machines of ins are set to the occupancies of the obtained solution of $insAux2$ (according to the assignment of the operations to the machines).

For each combination of the parameters, 2 instances are generated, giving a total of 2592 test instances. All instances are available at https://www.ioc.upc.edu/EOLI /research/.

5.2. *Results*

All test instances were solved with the algorithm HeufJSP. For each one, the cpu time, the costs (differentiating energy and deviation from due date costs), average (*Aver.*) and maximum (*Max.*) earliness and tardiness times were recorded (the minimum earliness and tardiness times are not reported in Tables 1-6 since they are always 0). Table 1 shows the averages of the aforementioned values for all instances.

cpu time (s)	***Cost (€)***			***Earliness***		***Tardiness***	
	Total	***Energy***	***Deviation from due dates***	***Aver.***	***Max.***	***Aver.***	***Max.***
57.05	15812.83	778.64	15034.19	2.11	53.24	3.91	48.57

Table 1. Average values of the heuristic solutions

The implemented heuristic obtains a solution in around 1 minute on average. On average, average tardiness times are around 2 times greater than average earliness times, although their maximum values are similar (around 50 units of time).

Next, we analyse the solutions according to the characteristics of the instances. Table 2 groups the results according to the ratio between number of jobs and number of machines, n/m (between parentheses, it is shown the number of instances in each group).

n/m	*cpu time (s)*	*Cost (€)*			*Earliness*		*Tardiness*	
		Total	*Energy*	*Deviation from due dates*	*Aver.*	*Max.*	*Aver.*	*Max.*
(0,1] (756)	10.25	288.62	246.13	42.49	0.28	3.79	0.26	3.36
(1,2] (756)	62.34	983.13	603.29	379.84	0.73	15.30	0.78	14.26
(2,3] (432)	82.54	3471.10	855.09	2616.01	1.88	43.55	2.23	36.03
(3,4] (216)	77.48	8628.20	1116.41	7511.79	3.06	69.41	4.87	78.26
(4,5] (108)	67.23	35763.56	1264.70	34498.87	4.14	93.52	9.52	115.75
(5,6] (108)	83.67	37987.94	1538.12	36449.82	5.30	134.51	10.54	146.73
7 (36)	71.51	86524.00	1544.68	84979.32	8.51	174.31	19.60	199.83
8 (36)	79.47	92510.07	1744.51	90765.56	10.32	303.00	20.55	225.56
9 (36)	93.17	105650.01	1968.49	103681.52	9.83	258.97	20.94	203.89
10 (36)	120.73	123219.70	2276.23	120943.47	11.24	393.22	24.97	233.83
11 (36)	132.48	194383.90	2559.01	191824.88	11.20	355.89	27.36	266.33
12 (36)	177.99	194852.13	2763.24	192088.89	10.06	323.94	30.22	308.08

Table 2. Average values grouped by the ratio *n*/*m*

As we expect, earliness and especially tardiness times and costs tend to increase when the ratio n/m increases. When there are not more jobs than machines $\left(n/m \in (0,1]\right)$, the main costs are energetic (85.28% on average) but this percentage decreases quickly when there are much more jobs than machines (1.42% on average when $n/m = 12$). We can see that earliness and tardiness times, on average, have a tendency to grow with similar proportions as the ratio n/m grows. Regarding the cpu time, we cannot observe any tendency with regard to n/m, because it depends mainly of the number of jobs: when $n = 10$ and $n = 120$, the cpu time averages are 1.33 s and 141.72 s, respectively.

Table 3 groups the results according to the versatility of the machines (*Vers.*). When the versatility of the machines is high (i.e., the number of machines capable of processing an operation is, on average, relatively high) the average earliness and tardiness times are reduced around 3 and 13 times, respectively, with respect to the times corresponding to a low versatility of the machines (in whose case, on average, the number of machines capable of processing an operation is low). Thus, the versatility of the machines has a high influence on the costs of the solutions. Moreover, we can see that with high versatility the energy cost is an important term of the total cost (40.33% on average) whereas the energy cost is relatively insignificant with low versatility (3.56% on average). Regarding cpu times, on average the heuristic applied to instances with high versatility lasts 1.2 times more with respect to the solution of low versatility instances.

Vers.	*cpu time (s)*	*Cost (€)*			*Earliness*		*Tardiness*	
		Total	*Energy*	*Deviation from due dates*	*Aver.*	*Max.*	*Aver.*	*Max.*
Low	49.02	30452.57	1084.14	29368.43	3.24	83.00	7.29	83.61
High	65.09	1173.08	473.14	699.94	0.98	23.49	0.54	13.53

Table 3. Average values grouped by the versatility of the machines

Table 4 groups the results according to the expected value of the processing times of the operations (*PT Var.*): low ($U[1,30]$), medium ($U[1,90]$) and high ($U[1,240]$). This characteristic has some influence in the cpu time: with high processing times the heuristic takes twice as long than it does with low processing times. On the other hand, as expected, earliness and tardiness times and costs (both energy and due date costs) increase along with processing times. Here, when processing times are low, the energy cost is an important term of the total cost (25.25% on average) whereas the energy cost is relatively insignificant with high processing times (3.69% on average).

PT Var.	*cpu time (s)*	*Cost (€)*			*Earliness*		*Tardiness*	
		Total	*Energy*	*Deviation from due dates*	*Aver.*	*Max.*	*Aver.*	*Max.*
Low	36.80	1033.07	261.00	772.07	0.29	8.21	0.94	10.06
Medium	56.70	7422.65	636.11	6786.54	1.33	35.85	3.10	37.75
High	77.67	38982.75	1438.80	37543.95	4.70	115.67	7.70	97.90

Table 4. Average values grouped by the expected value of the processing times of the operations

Finally, Table 5 groups the results according to the ratio between tardiness and earliness cost parameters, β_j/δ_j: low $\left(\beta_j/\delta_j = 0.0\hat{2}\right)$, medium $\left(\beta_j/\delta_j = 0.\hat{2}\right)$ and high $\left(\beta_j/\delta_j = 2\right)$. We can see that average cpu times are very similar. Regarding the other results, the tendency is that the tardiness averages decrease and the earliness averages increase when the tardiness cost weight increases, as it can be expected.

β_j/δ_j	*cpu time (s)*	*Cost (€)*			*Earliness*		*Tardiness*	
		Total	*Energy*	*Deviation from due dates*	*Aver.*	*Max.*	*Aver.*	*Max.*
Low	55.45	2667.13	776.14	1890.99	1.34	35.83	4.03	49.00
Medium	57.92	9655.36	778.81	8876.55	2.15	54.66	3.83	48.90
High	57.79	35115.99	780.97	34335.01	2.83	69.25	3.88	47.82

Table 5. Average values grouped by the ratio β_j/δ_j

As it has been stated in Section 2, there is not any published procedure for solving the specific variant of the fJSP problem dealt with in the present paper. Barr *et al*. (1995) suggest that new methods can be compared with a simple random restart procedure. Therefore, as a rough first assessment of the quality of the solutions provided by the proposed procedure we compared them with those obtained with MSfJSP, which in fact

is a multi-start greedy randomised algorithm that takes into account the deviations from the due dates.

MSfJSP generates a random solution at each iteration as follows. First, an operation is selected at random among a set of candidate operations. Operation O_{jh} is candidate if it has not been scheduled and it is the first operation of its job ($h=1$) or the previous operation ($O_{j,h-1}$) has been scheduled. Then the selected operation is assigned to a compatible machine (i.e., a machine in set M_{jh}) selected at random. If O_{jh} is not the last operation ($h < h_j$) then it is assigned to the first possible time interval; otherwise it is assigned to the best feasible time interval (i.e., to the feasible time interval involving the minimum increment of costs).

All test instances were solved with MSfJSP. In order to make a fair comparison, the cpu time per instance was the same time used by HeufJSP. The average number of iterations (generated solutions) per instance of MSfJSP is 1888.41. HeufJSP yields better results for all the instances, improving the total costs given by MSfJSP between a minimum of 33.86% and a maximum of 99.99%. Table 6 compares the average results obtained by HeufJSP and MSfJSP. We can see clearly the benefits of solving the problem with HeufJSP.

Procedures	***Cost (€)***			***Earliness***		***Tardiness***	
	Total	***Energy***	***Deviation from due dates***	***Aver.***	***Max.***	***Aver.***	***Max.***
HeufJSP	15812.83	778.64	15034.19	2.11	53.24	3.91	48.57
MSfJSP	17119560.12	3296.72	17116263.40	2.36	98.24	546.30	1391.74

Table 6. Average values of the HeufJSP and MSfJSP solutions

Finally, the HeufJSP and MSfJSP solutions are compared with the solutions obtained with the BIP model. BIP was solved using IBM ILOG CPLEX 12.6 and the cpu time was limited to 1 hour per instance. The maximum possible start time of the operations when calculating the sets T_{ijh} (Section 3.2) is set to $2 \cdot \max_{j=1}^{n} d_j$, which is a big enough value.

For the instances with 10 and 20 jobs (432 instances), BIP was solved optimally 320 times; for larger instances, optimal (or even feasible) solutions were rarely found. For those 320 instances that were solved optimally with BIP (grouped by versatility of the machines, *vers.*, and number of machines, *m*), Table 7 shows the number of instances, in each group, that were solved optimally (column *#ins*), the averages of the cpu times (in s) and the total costs (in €) of BIP, HeufJSP and MSfJSP (columns t_M, Z_M, t_H, Z_H, t_S, and Z_S respectively), the gap (%) between the average costs of HeufJSP and the BIP model (column $GAP_H = 100 \cdot (Z_H - Z_M)/Z_H$) and the gap (%) between the average costs of MSfJSP and BIP (column $GAP_S = 100 \cdot (Z_S - Z_M)/Z_S$).

vers. / m	*BIP*			*HeufJSP*			*MSfJSP*		
	#ins	t_M	Z_M	t_H	Z_H	GAP_H	t_S	Z_S	GAP_S
Low / 10	29	1006.81	292.91	2.11	1387.12	78.88	2.12	81638.46	99.64
Low / 20	28	539.53	247.35	2.20	351.35	29.60	2.20	69835.82	99.65
Low / 30	28	611.69	208.18	2.30	217.42	4.25	2.30	103644.63	99.80
Low / 40	29	585.63	156.37	1.53	156.79	0.27	1.54	144712.00	99.89
Low / 50	32	761.28	172.25	1.87	173.05	0.46	1.88	214230.99	99.92
Low / 60	32	1019.08	139.62	1.54	140.13	0.37	1.54	155915.90	99.91
High / 10	26	682.57	158.09	2.89	167.74	5.75	2.88	197031.37	99.92
High / 20	29	910.78	117.10	3.28	117.46	0.30	3.27	140543.41	99.92
High / 30	26	804.44	72.02	2.57	72.10	0.11	2.55	69950.60	99.90
High / 40	22	864.27	50.94	1.95	50.94	0.00	1.95	25393.13	99.80
High / 50	21	912.14	48.86	1.64	48.76	0.00	1.63	80761.86	99.94
High / 60	18	1096.22	38.28	1.85	38.28	0.00	1.85	77853.40	99.95

Table 7. Comparison of the heuristic solutions versus the optimal solutions of BIP model

We can see that the heuristics need a tiny fraction of the cpu time spent by BIP. With respect to the quality of the HeufJSP solutions, when the machines are highly versatile, on average the heuristic performs very well on instances with 20 or more machines (gaps are not greater than 0.3%) and performs quite well on instances with 10 machines (gap is equal to 5.75%). And when the machines have low versatility, HeufJSP performs also very well on instances with 30 or more machines (gaps are smaller than 0.5%) and quite well with 20 machines (gap is equal to 4.25%). In contrast, the results of MSfJSP show that these instances are not trivial to solve and very poor quality solutions are obtained with randomness.

On the other hand, in the scenario of 10 or 20 machines with low versatility, HeufJSP does not perform well (gaps are equal to 29.6% and 78.88%, respectively) and MSfJSP performs much worse (gaps greater than 99.6%).

To sum up, for the most of the instances with 10 and 20 jobs, HeufJSP performs well when compared to the optimal solutions obtained with BIP. For larger and more realistic instances with more than 20 jobs, optimal (or even feasible) solutions were rarely found with the BIP model whereas the proposed heuristic can calculate always a feasible solution in a short cpu time (around 1 minute on average).

6. Conclusions and prospects

This paper introduces a new just-in-time scheduling problem in which the costs of performing the operations depend on time. This problem is found in many real industrial situations where the costs of the energy or of the work force are a significant component of the total costs and the operations can be performed in any time of the day.

We have formulated the problem with a mathematical programming model and proposed a quick enough heuristic algorithm to find feasible solutions fulfilling the

constraints on the availability of the machines and taking into account the costs of performing the operations and those corresponding to the deviations from the due dates.

This is a first step in a research agenda that includes the relaxation of some of the assumptions enumerated in 3.1 (specifically, those concerning the precedence relationships between the operations of each job and between the operations of different jobs, the size of buffers and the set-up times) and the application of metaheuristics and matheuristics, with the aim to introduce a greater diversification, i. e., to explore ordering jobs different from those corresponding to the diverse values of the parameter ζ. The benchmark solutions obtained with BIP model points out that the proposed heuristic performs very well on scenarios with a not too small number of machines or with machines with high versatility, but for the other scenarios the resolution of the problem has to be improved.

References

Akyol, D.E., Bayhan, G.M., (2005). A Coupled Gradient Network Approach for the Multi Machine Earliness and Tardiness Scheduling Problem. *Lecture Notes in Computer Science*, 3483, 596–605.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246, 345–378.

Arkat, J., Fattahi, P., Jolai, F. (2009). "Flexible job shop scheduling with overlapping in operations". *Applied Mathematical Modelling*, 33, 3076–3087.

Armentano, V.A., Laguna, M., Scrich, C.R. (2004). Tardiness minimization in a flexible job shop: a tabu search approach. *Journal of Intelligent Manufacturing*, 15, 103–115.

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G.C., Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1, 9-32.

Birgin, E.G., Feofiloff, P., Fernandes, C.G., Melo, E.L., Oshiro, M.T.I., Ronconi, D.P. (2014). A MILP model for an extended version of the Flexible Job Shop Problem. *Optimization Letters*, 8, 1417–1431.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41, 157–183.

Brucker, P., Schlie, R. (1990). Job shop scheduling with multi-purpose machines. *Computing*, 45, 369–375.

Chen, J.C., Cheng-Chun, W., Chia-Wen, C., Kou-Huang, C. (2012). Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm. *Expert Systems with Applications*, 39, 10016–10021.

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

De Giovanni, L., Pezzella, F. (2010). An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. *European Journal of Operational Research*, 200, 395–408.

Dini, G., Rossi, A. (2007). Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimization method. *Robotics and Computer-Integrated Manufacturing*, 23, 503–516.

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

Fattahi, P., Jolai, F., Arkat, J. (2009). Flexible job shop scheduling with overlapping in operations. *Applied Mathematical Modelling*, 33, 3076–3087.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., Chong, C. S. (2014). Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Information Sciences*, 289, 76-90.

Gao, K. Z., Suganthan, P. N., Pan, Q. K.,Tasgetiren, M. F. (2015). An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time. *International Journal of Production Research*, 53, 5893-5911.

Garey, M.R., Johnson, D.S., Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1, 117–129.

Gholami, M., Zandieh, M. (2009). Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. *Journal of Intelligent Manufacturing*, 20, 481-498.

González, M. A., Vela, C. R. Vela, Varela, R. (2015). Scatter Search with Path Relinking for the Flexible Job Shop Scheduling problem. *European Journal of Operational Research*, 245, 35-45.

Ham, M., Lee, Y.H., Kim, S.H. (2011). Real-time scheduling of multi-stage flexible job shop floor. *International Journal of Production Research*, 49, 3715–3730.

He, Y., Li, Y., Wu, T., Sutherland, J. W. (2015). An energy-responsive optimization method for machine tool selection and operation sequence in flexible machining job shops. *Journal of Cleaner Production*, 87, 245-254.

Hmida, A. B., Haouari, M., Huguet, M.J., Lopez, P. (2010). Discrepancy search for the flexible job shop scheduling problem. *Computers & Operations Research*, 37, 2192-2201.

Hurink, E., Jurisch, B., Thole, M. (1994). Tabu search for the job shop scheduling problem with multi-purpose machines. *Operations Research Spektrum*, 15, 205–215.

Jiang, Z., Zuo, L., E, M. (2014). Study on Multi-objective Flexible Job-shop Scheduling Problem considering energy Consumption. *Journal of Industrial Engineering and Management*, 7, 589-604.

Ku, W.-Y., Beck, J. C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, 73, 165-173.

Li, J., Pan, Q., Xie, S., Liang, J. (2010). A Hybrid Pareto-Based Tabu Search for Multi-objective Flexible Job Shop Scheduling Problem with E/T Penalty. *Lecture Notes in Computer Science*, 6145, 620–627.

Liu, Y., Tiwari, A. (2015). An Investigation into Minimising Total energy Consumption and Total Completion Time in a Flexible Job Shop for Recycling Carbon Fiber Reinforced Polymer. *Procedia CIRP*, 29, 722-727.

Loukil, T., Teghem, J., Fortemps, P. (2007). A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research*, 179, 709–722.

Luo, H., Du, B., Huang, G. Q., Chen, H., Li, X. (2013). Hybrid flow shop scheduling considering machine electricity consumption cost. *International Journal of Production Economics*, 146, 423-439.

Mastrolilli, M., Gambardella, L.M. (2002). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, 3, 3–20.

Moon, J.-Y., Shin, K., Park, J. (2013). "Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *International Journal of Advanced Manufacturing Technology*, 68, 523-535.

Roshanaei, V., Azab, A., ElMaraghy, H. (2013). Mathematical modelling and a meta-heuristic for flexible job shop scheduling. *International Journal of Production Research*, 51, 6247–6274.

Saad, I., Hammadi, S., Benrejeb, M., Borne, P. (2008). Choquet integral for criteria aggregation in the flexible job-shop scheduling problems. *Mathematics and Computers in Simulation*, 76, 447-462.

Scrich, C.R., Armentano, V.A., Laguna, M. (2004). Tardiness minimization in a flexible job shop: A tabu search approach. *Journal of Intelligent Manufacturing*, 15, 103–115.

Thammano, A., Phu-ang, A. (2013). A Hybrid Artificial Bee Colony Algorithm with Local Search for Flexible Job-Shop Scheduling Problem. *Procedia Computer Science*, 20, 96-101.

Wu, Z., Weng, M. X. (2005). Multiagent Scheduling Method With Earliness and Tardiness Objectives in Flexible Job Shops. *IEEE Transactions on Systems, Man, and Cyberrnetics-Part B: Cybernetics*, 35, 293-301.

Zhang, R., Chiong, R. (2016). Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production*, 112, 3361-3375.

Zhang, G., Shao, X., Li, P., Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering*, 56, 1309-1318.

Zhang, H., Zhao, F., Fang, K., Sutherland, J. W. (2014). Energy-conscious flow shop scheduling under time-of-use electricity tariffs. *CIRP Annals – Manufacturing Technology*, 63, 37-40.

Ziaee, M. (2013). A heuristic algorithm for the distributed and flexible job-shop scheduling problem. *The Journal of Supercomputing*, 67, 69–83.

Appendix A1. Process of obtaining an optimal schedule of a given job for a numerical example

Next, an example of obtaining an optimal schedule of a job is shown. Let the data of the example set as follows:

$n = 2$

$m = 4$

$h_1 = 3,\ h_2 = 2$

Table A1.1 shows the values of O_{jh}, M_{jh} and p_{ijh}.

		p_{ijh}			
		m_1	m_2	m_3	m_4
Job $j=1$	O_{11}	60	240	×	×
	O_{12}	×	180	120	×
	O_{13}	180	120	×	180
Job $j=2$	O_{21}	180	240	×	120
	O_{22}	120	×	60	×

Table A1.1. Values of O_{jh}, M_{jh} and p_{ijh}

$\Pi = (1,2)$ (job $j=1$ is processed before job $j=2$) and let $j=1$ be the job to be scheduled.

Table A1.2 shows the values of P_{ijh} (only for job $j=1$ to be scheduled).

		P_{i1h}			
		m_1	m_2	m_3	m_4
Job $j=1$	O_{11}	10	40	×	×
	O_{12}	×	30	20	×
	O_{13}	30	20	×	30

Table A1.2. Values of P_{ijh} for job $j=1$

$d_1 = 1800$

$r_1 = 1$

$\beta_1 = 1/10;\ \gamma_1 = 1;\ \delta_1 = 1/30$

$C^P = 0.2;\ C^V = 0.1.$

Table A1.3 shows the initial occupancy of each machine i, Q_i, the peak and valley periods and the associated energy cost.

	t	m_1	m_2			m_3	m_4
Peak period	1-60	2	8				
	61-120						
	121-180						
	181-240	2					
	241-300						
	301-360						
	361-420						
	421-480						
	481-540						
	541-600						
Valley period	601-660						
	661-720						
	721-780	1					
	781-840						
	841-900						
	901-960						
	961-1020						
	1021-1080						
	1081-1140						
	1141-1200						
	1201-1260						
	1261-1320						
	1321-1380						
	1381-1440						
Peak period	1441-1500						
	1501-1560	2					
	1561-1620	2					
	1621-1680						
	1681-1740						
	1741-1800	2					
	1801-1860	2					
	1861-1920	2					
	1921-1980						
	1981-2040						
Valley period	2041-2100						
	2101-2160						
	2161-2220	1					
	2221-2280	1	4				
	2281-2340	1		4			
	2341-2400	1			4		
	2401-2460	1				4	
	2461-2520	1	4				
	2521-2580	1		4			
	2581-2640	1			4		
	2641-2700	1				4	
	2701-2760	1					
	2761-2820	1					
	2821-2880	1					

Table A1.3. Initial occupancy of machines and the energy costs of performing operation O_{11} for each machine and set of periods in which the operation can be processed.

Figure A1.1 shows the multistage graph corresponding to job $j=1$. The value inside the nodes is the completion time of the corresponding operations. The first value over the arc leading at the stage h $\left(1 \le h \le h_j\right)$ is the cost $U_{i_{(jh)}jh}$ (the cost of the energy required to process operation O_{jh} on machine $i_{(jh)}$); and the second value is the machine $i_{(jh)}$ $\left(i_{(jh)} \in M_{jh}\right)$ to which operation h of job j $\left(O_{jh}\right)$ has been assigned. The value over the arc leading at the stage h_j+1 is the cost $C_E\left(E_j\right)$ (the earliness cost of job j with

respect to its due date d_j) plus the cost $C_T\left(T_j\right)$ (the tardiness cost of job j with respect to its due date d_j). The first value below the nodes corresponds to κ_{ht} (the value of node v_{ht}); and the second value identifies the node that precedes v_{ht}) in the shortest path.

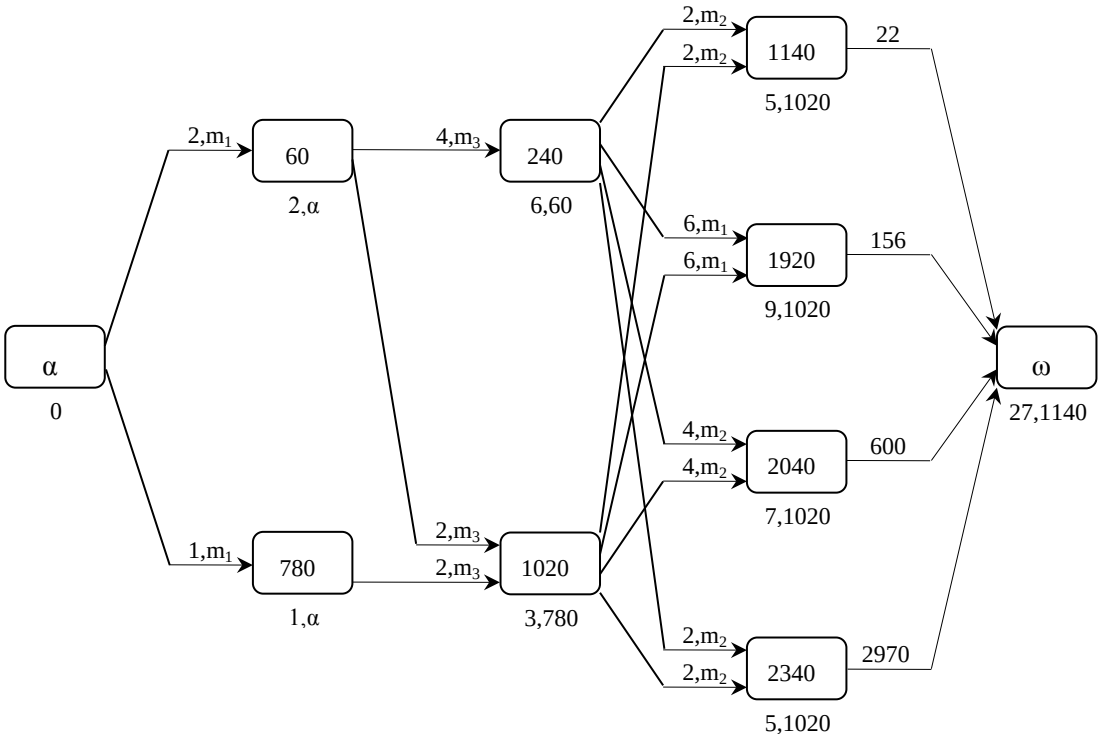

Figure A1.1. Multistage graph corresponding to job $j=1$

In summary, the least cost path (27) represents the ordered sequence of operations (O_{11}, O_{12}, O_{13}) and their respective associated costs ($U_{111}=1, U_{312}=2, U_{213}=2$, $C_E\left(E_1\right)+C_T\left(T_1\right)=22$), as well as the times (780, 1020, 1140) at which the operations finish being processed on the machines (m_1, m_3, m_2, respectively).

Appendix A2. Process of calculating q_{ijh} for a numerical example

Next, an example of calculating q_{ijh} is shown: $n=2$; $m=4$; $h_1=3$, $h_2=2$; Table A1.1 shown the values of O_{jh}, M_{jh} and p_{ijh}; $\Pi=(1,2)$ (job $j=1$ is processed before job $j=2$); and let O_{11} be the operation to be scheduled.

The obtained values of q_{ijh} are:

$$q_{111}=60+\frac{180}{3}+\frac{120}{2}=180$$

$$q_{211}=240+\frac{240}{3}=320$$