Annals of Operations Research (2022) 311:899–920
https://doi.org/10.1007/s10479-020-03600-0

S.I.: MOPGP19

A simulated multi-objective model for flexible job shop transportation scheduling

Yiyi Xu[1,2] · M'hammed Sahnoun[1] · Fouad Ben Abdelaziz[2] · David Baudry[1]

Published online: 17 April 2020

Abstract
This paper proposes a new dynamic algorithm based on simulation approach and multi-objective optimization to solve the FJSP with transportation assignment. The objectives considered in scheduling jobs and transportation tasks in a flexible job shop manufacturing system include makespan, robot travel distance, time difference with due date and critical waiting time. The results obtained from the computational experiments have shown that the proposed approach is efficient and competitive.

Keywords Flexible job shop scheduling · Multi-objective programming · Simulation · Multi-agent systems · Transportation

1 Introduction

Scheduling is a process of assigning sets of tasks, or jobs to different resources. The classic job shop scheduling problem (JSP) has been considerably investigated by researchers (Adams et al. 1988; Van Laarhoven et al. 1992). In the JSP, there are n jobs and m machines in a manufacturing system. A job is comprised of j operations with a pre-defined order of execution. Generally, the JSP strives to find a feasible schedule to execute all operations on the machines available. However, jobs are independent of each others and can be processed on machines in any sequence. Sotskov and Shakhlevich (1995) proved that a JSP with three jobs and three machines $J = 3 \mid M = 3 \mid Cmax$ is an NP-hard problem.

✉ Fouad Ben Abdelaziz
fouad.ben.abdelaziz@neoma-bs.fr

Yiyi Xu
yxu@cesi.fr; yiyi.xu.16@neoma-bs.com

M'hammed Sahnoun
msahnoun@cesi.fr

David Baudry
dbaudry@cesi.fr

[1] CESI-LINEACT laboratory, 80 rue Edmund Halley Rouen Madrillet Innovation, 76800 Saint-Étienne-du-Rouvray, France

[2] Neoma Business School, 1 Rue du Marchal Juin, 76130 Mont-Saint-Aignan, France

The increasing variety of manufactured products helps customization service to satisfy customers, while results in more and more complex manufacturing systems (Doganis and Sarimveis 2008). The demand of intelligent data analytic tools and flexible manufacturing systems (FMS) are increasing under the move to Industry 4.0 (Lee et al. 2014). The flexible job shop scheduling extends the classic job shop, assuming that more than one machine is able to run a specific operation (Brandimarte 1993). It allows a workshop to carry out more operations without investing extra resources or machines. Since the JSP is considered as a particular case of the flexible job shop problem (FJSP), the FJSP is also regarded as NP-hard (De Giovanni and Pezzella 2010; Tay and Ho 2008).

In this paper, we employ simulation along with optimization techniques to solve the FJSP with various uncertain parameters inherently existent in the manufacturing environment. Our main contributions are as follows:

- Development of a novel algorithm to solve the FJSP with transportation assignment.
- Optimization of the FJSP with multiple objectives. Previous research hasn't considered robot transportation distance as an objective.
- Application of dynamic lexicographic order objectives to avoid issues with one absolute importance order given by decision makers.
- Embedding the proposed method in a simulation model which reduces computational time significantly while providing equivalent results.
- Obtaining a unique solution, as opposed to a set of Pareto solutions, to make decision-making more effective.

The rest of the paper is organized as follows. Section 2 gives a review of the recent research on the FJSP. In Sect. 3, the FJSP is redefined and our programming background is presented. Section 4 describes the proposed simulation model and its input requirements, whereas the dynamic scheduling algorithm is explained in Sect. 5. In Sect. 6, the test data and computational results are shown. Finally, we provide our conclusions and suggestions for future research in Sect. 7.

2 Related work

Brucker and Schlie (1990), among the first to deal with the FJSP, proposed a polynomial algorithm to solve a two-job flexible job shop problem with one objective to minimize the makespan. Since then, $Makespan$, the completion time of finishing all jobs, has been taken as mono-objective to solve the FJSP by many researchers (Liouane et al. 2007; Pezzella et al. 2008; Xing et al. 2010; Li et al. 2011; Yuan et al. 2013). Gambardella and Mastrolilli (1996) introduced a tabu search (TS) based algorithm with two novel neighborhood functions and demonstrated the effectiveness of his algorithm through a benchmark on five sets of experiments. Zandieh et al. (2008) proposed a genetic algorithm (GA) with several initial populations and several strategies for generating new populations. Nouiri et al. (2018) proposed hybrid metaheuristics based on a clustered holonic multi-agent model to solve the FJSP with many robots. Buddala and Mahapatra (2019) incorporated a new local search technique to teaching–learning-based optimization to solve the FJSP with one objective to minimize makespan.

However, minimizing one objective is unsuitable in real manufacturing systems as different criteria are often considered simultaneously over the production flow (Thörnblad et al. 2013). The multi-objective FJSP has gained the attention of some researchers (Kacem et al. 2002; Gao et al. 2008; Moslehi and Mahnam 2011; Karthikeyan et al. 2015; Kumar and Pandey

2015) over the last two decades. Brandimarte (1993) described a hierarchical algorithm based on a tabu search metaheuristic and a two-way information flow, with makespan and tardiness as separate objectives. Xia and Wu (2005) developed a hybridized particle swarm optimization (PSO) and simulated annealing (SA) to minimize makespan, total machine workload and maximum machine workload at the same time. Gao et al. (2016) proposed a discrete harmony search algorithm (DHS) with a weighted combination of makespan, the mean of earliness and tardiness. Li et al. (2019) introduced an elitist non-dominated sorting hybrid algorithm to solve the FJSP by minimizing makespan and total setup costs.

In general, the FJSP can be decomposed into two sub-tasks: assignment of an operation to an appropriate machine, and sequencing the operations on each machine. For solving the FJSP with more than two jobs, two basic approaches are available (Brandimarte 1993). Concurrent/integrated approaches are proposed to solve assignment and sequencing problems simultaneously (Hurink et al. 1994; Dauzère-Pérès and Paulli 1997; Gao et al. 2008; Yuan and Xu 2013; Buddala and Mahapatra 2019), whereas hierarchical approaches addresses the two sub-tasks separately with the aim to reduce complexity (Brandimarte 1993; Xia and Wu 2005; Gao et al. 2015; Karthikeyan et al. 2015; Wang et al. 2018). However, most the above mentioned references assume that move times between different machines are negligible, and transportation resources are unlimited during the production.

Langston (1987) and some other researcher (Boudhar and Haned 2009; Naderi et al. 2009, 2010) considered uniformly distributed or sequence-dependent transportation times between different stages in the flow shop scheduling problem(FSP). In the classic job shop problem (JSP), researchers (Hurink and Knust 2005; Zhang et al. 2014) introduced disjunctive graph models with machine-dependent transportation times as constraints. For the FJSP, researchers (Rossi and Dini 2007; Zhang et al. 2012; Rossi 2014; Nouri et al. 2016; Karimi et al. 2017) used transportation time matrices, including the times to move a job between two different machines, as input data to develop mathematical models or metaheuristics to find optimal schedules.

One disadvantage of considering transportation times between machines is the underestimate of empty move times, the time a robot moves without a job between machines M_i. In practice, a certain distance separates machine input and output buffers. Therefore, the empty move time from M_l to M_l cannot be neglected. The extended shifting bottleneck heuristic (SBH) developed by Drießel and Mönch (2012) considers transportation times between machine buffers/stockers, rather than between machines to solve a JSP where a path to move a job is static since each operation can only be executed on one machine (or one of a group of identical machines). Nevertheless, to the best of our knowledge, none has considered transportation times between buffers in the FJSP.

In this paper, we develop a multi-agent simulation model to solve the multi-objective FJSP with transportation assignment of robots, a problem hasn't been studied yet. Compared to other simulation techniques such as discrete-event simulation and micro simulation, agent-based simulation is more straight-forward and adequate to deal with heterogeneous agents (Macal and North 2005; Siebers et al. 2010) and agents with geo-spatial movement (Siebers et al. 2010). A novel dynamic scheduling algorithm is proposed and embedded in the simulation model to lexicographic-optimize transportation operation schedules. All critical real-time information, such as remaining jobs, robot positions and machine conditions, can be obtained from the simulation model and inputted into the dynamic algorithm.

3 Problem formulation

The FJSP has a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ and a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job J_i has j operations $O_i = \{O_{i1}, O_{i2}, \ldots, O_{ij}\}$ with a precedence constraint and each operation O_{ij} can be processed on any among a subset $M_{ij} \subseteq M$ of compatible machines.

In this paper, we address the FJSP with transportation assignment of robots. Each machine M_m is with an input buffer B_m^I and an output buffer B_m^O. Transfers between machines and their own buffers are automatic. Additionally, there are a system input buffer B_0^I and a system output buffer B_0^O to put raw jobs and completely finished jobs. A set of p robots $R = \{R_1, R_2, \ldots, R_p\}$ transfers jobs between buffers, where the transportation time is distance-dependent. The path and distance between two different buffers are predetermined and fixed. In Fig. 1, we show the whole process of J_1 production and transportation T_{1i}. The five transportation tasks $T = \{T_{1,1}, \ldots, T_{1,5}\}$ are possible to be operated by different robots. The processing time of an operation O_{ij} on a machine M_{ijm} is operation-dependent and machine-dependent , as shown in Table 1. Production and transportation can be processed independently, which allows robots to transfer some jobs while machines are treating others. The problem thus involves three sub-tasks: assignment of operations to machines, assignment of robots to operations and sequencing the operations on each machine. As we are not solving these sub-problems separately, we integrate them and optimize the schedule dynamically at each transportation assignment, step by step. The algorithm is detailed in Sect. 5.

The proposed model includes the following assumptions:

- All jobs, machines, buffers and robots are available at time 0.
- Machine breakdown and robot failure are ignored (but possible).
- Jobs:
  - Jobs are independent of each other.
  - Each job can be processed on one and only one machine at a time.
  - Preemption of operations is not allowed.

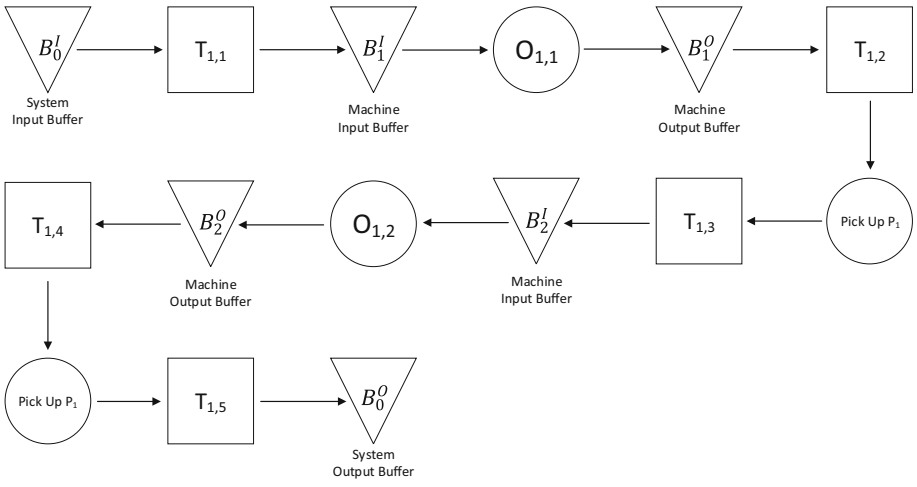

Fig. 1 Production process and transportation

Table 1 Processing times of operations

Jobs	Operations	Machines			
		M_1	M_2	M_3	M_4
J_1	$O_{1,1}$	15	20	18	–
	$O_{1,2}$	25	40	–	30
J_2	$O_{2,1}$	–	25	30	20
	$O_{2,2}$	30	–	–	25
J_3	$O_{3,1}$	–	15	25	40
	$O_{3,2}$	60	–	45	–

- Machines:
  - Machines are independent of each other.
  - Each machine can process one and only one operation at a time.
  - Machine setup times are ignored (but possible).
- Buffers:
  - Machine buffers have an identical capacity limit.
  - System buffers are unlimited.
- Robots:
  - Robots are independent of each other.
  - Each robot can transfer one and only one job at a time.
  - Time required to load and unload jobs from buffers is included in processing time.
  - Robot collisions are ignored.

Our objectives are to minimize the following:

- F1: makespan, the completion time of all jobs.
- F2: transportation distance, total distance passed by all robots.
- F3: time difference between job finishing time and due time (including both earliness and tardiness).
- F4: critical waiting time, the longest time a job waits in a buffer.

Generally, the approaches focused on solving multi-objective optimization problems (MOO) can be divided into two classes (Fonseca and Fleming 1998):

- Pareto-based approaches, optimizing all objectives concurrently.
- Non-Pareto approaches, treating objectives separately.

In this paper, our algorithm lies within the first group, Pareto-based approaches. We optimize all objectives simultaneously with dynamic lexicographic orders. Lexicographic ordering (Marchi and Oviedo 1992) requires decision makers to prioritize the objectives according to their absolute importance before the solution process. A more important objective is infinitely more important than a less important objective (Branke et al. 2008). After the ordering step is completed, the solution process starts working minimizing the most important objective function. If a unique solution is found, the solution process will stop. Otherwise, the second most important objective function will be considered and so on until a unique solution is found. The solution of lexicographic ordering can be proven to be Pareto optimal as it is a unique optimal solution of one of the objectives.

However, it is not that easy to derive the importance order of all objectives in practice and lexicographic orderings do not allow a small increment of a more important objective function to be traded off with a great decrement of a less important objective, which can be attractive to decision makers (Roy and Mousseau 1996).

To avoid these drawbacks, we use dynamic lexicographic orders based on two concepts, entropy (Shannon 1948) and goal programming (Charnes et al. 1955), at all transportation assignment steps. The importance orders of the objectives are dependent on their entropy values. When considering a group of jobs based on one objective, the larger entropy value the group has, the more similarity the jobs have, which leads to more difficulty in prioritizing members. At each assignment step, a robot obtains a new lexicographic order based on updated objective entropy values from the real-time information, such as job next operations, robot positions, jobs already in machine input buffers and so on. Before using goal programming to generate a solution, decision makers often need to specify aspiration levels for the objective functions and then, deviations from these aspiration levels are minimized in the solution process. An objective function combined with an aspiration level is regarded as a goal. In our algorithm, instead of aspiration levels given by decision makers, we use the best value of an objective and an acceptable deviation range to get the aspiration level of this objective. The solution process starts with the most important objective function based upon entropy calculation, all jobs better than the aspiration level of this objective are kept, then turns to the next most important objective. The solution process stops once we find a unique solution or the minimization of all objectives finishes. We provide a more detailed description and explanation of the proposed algorithm in Sect. 5.

4 Simulation model

In this section, we describe the developed simulation model. First, we discuss the required inputs and then move to present the simulation framework comprised of three main steps.

4.1 Inputs

To simulate different scenarios in the FJSP, we develop a multi-agent Netlogo model with four interdependent agents: **Job, Machine, Buffer and Robot**. Operations of jobs are defined as properties of Job Agents.

The following inputs are required to set up the model:

- Global Values
  - objective list
  - deviation space
  - due time list
  - simulation step (tick value)
- Job Agent:
  - ID and initial location (X- and Y-coordinates)
  - operations with a precedence constraint
- Machine Agent:
  - ID and location (X- and Y-coordinates)

- associated input buffer and output buffer
- processing times of compatible operations (a sample in Table. 1)

○ Buffer Agent:

- ID and location (X- and Y-coordinates)
- buffer type (system input/output, machine input/output)
- associated machine if it is machine buffer
- capacity limit (no capacity limit for system buffers)

○ Robot Agent:

- ID and initial location (X- and Y-coordinates)
- move speed.

4.2 Framework

The simulation model and the proposed method are coded in Netlogo 6.0.1. This software allows modellers to define hundreds or even thousands of independent agents for modeling complex systems, which could help us solve the FJSP at large scale. The provided interface works as a desktop application, making the proposed model able to be used directly and easily for industry users.

The proposed model is shown conceptually in Fig. 2. We list four agents and three main model steps. **Input** is the first step and includes model setup, capture and storage of all the required data. In the **GO** step, all agents work independently and robots/agents perform transportation assignments and optimize schedules as follows: `TransportationAssignment` function (defined and described in Sect. 5). The `TransportationAssignment` function gives each robot a task list, including job ID, machine ID and machine input buffer ID. If no job in the feasible set satisfies constraints (for example, all machine input buffers have reached their capacity limit), then no task is given to robots. After assignment, a robot leaves its current position to the system input buffer or the machine output buffer where the target job is in, then picks up the job and transfer it to the assigned machine input buffer. Machines execute jobs according to the arrival sequence. The **GO** process repeats until all jobs finish. Once all job operations complete, **Output** produces the final result and provides the job–machine–robot schedule and all relevant objective values.

5 Dynamic multi-objective scheduling algorithm

Most previous algorithms were applied to solve the FJSP with predetermined conditions, such as fixed numbers of jobs, job release times (Kacem et al. 2002) and machine breakdowns (Al-Hinai and ElMekkawy 2011). Ahmadi et al. (2016) addresses the stable scheduling of the FJSP with random machine breakdown, where the interval between every two breakdown occurrences follows an exponential distribution. They evaluated the change before and after machine breakdown to find optimal solutions with the maximal stability. Unlike the existing approaches, our dynamic algorithm makes it possible to adjust the remaining solution process whenever extra information are inputted into the system. For example, when new orders arrive or resources change unpredictably (machine breakdown or robot failure), robots (which are still working) will make decisions based on the updated information, whereas the finished operations are kept as a part of the final feasible solution.

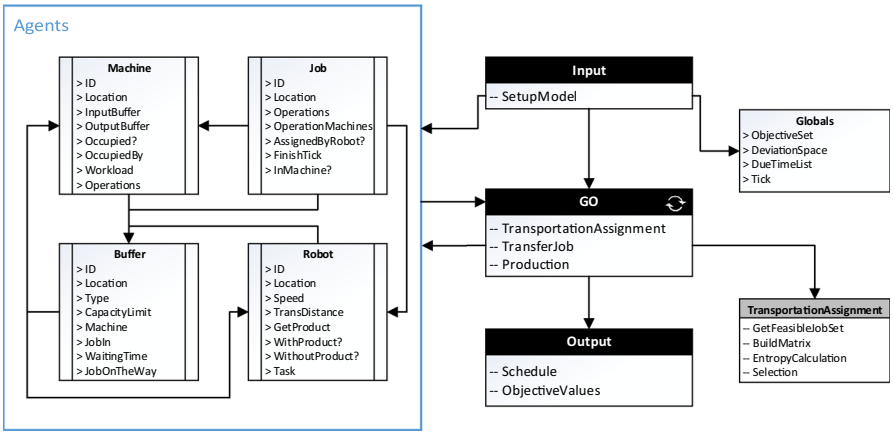

Fig. 2 Proposed model with key agents and steps

In this section, we describe the proposed `TransportationAssignment` function in Fig. 2. As mentioned in Sect. 3, we apply dynamic lexicographic ordering based on entropy and use goal programming at transportation assignment steps. We extend the combination of entropy and lexicographic ordering introduced by Eloundou (2016). For the FJSP at large scale, the values regarding one objective are rarely identical. That means the optimization process very likely stops after minimizing the first objective, since a unique solution is found. In order to avoid this problem, we modify the procedure of obtaining lexicographic orders and introduce objective aspiration levels to allow a trade-off between a small decrement for a more important objective and a substantial increment for a less important objective.

Before transferring the last job to the system output buffer, `Transportation Assignment` function is called by different robots repeatedly. It contains three phases, namely the decision matrix formation phase, the lexicographic ordering phase and the job selection phase. We describe in detail as follows:

Phase I: The decision matrix formation phase

During this phase, the decision matrix, including all feasible jobs waiting for transfer and their corresponding objective values, is formatted according to the real-time information derived from the model.

When a robot is available, it checks the system input buffer and all machine output buffers to find feasible jobs. If no job is found (for example, the production of jobs is almost done and only one or two jobs are being processed in machines), the robot skips all remaining steps and waits on the spot. If only one job is feasible, the robot chooses this job directly without going through the algorithm. When two or more jobs ($n \geq 2$) are feasible, the robot builds a decision matrix (Table 2) with n rows and 7 columns (3 columns job information and 4 columns relevant objective values).

The operations of a job J_i follow a precedence constraint, therefore the next operation O_{ij} of this job is easy to acquire. To select feasible machines M_m from a subset M_{ij} of compatible machines which can process O_{ij}, we need to respect the machine buffer capacity constraint. If none of the buffers allow the transportation of J_i due to previous transferred jobs, then J_i is deleted from the feasible job set. Otherwise, we list all input buffers B_{im} of feasible machines in the matrix. For the right part of the decision matrix, recall that we optimize

Table 2 Decision matrix

Job information			Objective values			
Job	Next operation	Next buffer	F1	F2	F3	F4
J_1	O_{1j}	B_{1m}	f_{11}	f_{21}	f_{31}	f_{41}
J_2	O_{2j}	B_{2m}	f_{12}	f_{22}	f_{32}	f_{42}
…	…	…	…	…	…	…
J_i	O_{ij}	B_{im}	f_{1i}	f_{2i}	f_{3i}	f_{4i}
…	…	…	…	…	…	…
J_n	O_{nj}	B_{nm}	f_{1n}	f_{2n}	f_{3n}	f_{4n}

makespan (F1), transportation distance (F2), time difference between job finishing time and due time (F3), and critical waiting time (F4). Since we schedule transportation assignments dynamically based on real-time information, the objectives listed in the matrix do not mean the total results, but basic values. For example, the makespan value f_{1i} of J_i is the finishing time of O_{ij} on selected M_m. The critical waiting time value f_{4i} is the maximal waiting time in the system after the robot puts J_i into B_{im}. Once all jobs and objective values are inserted in the decision matrix, the formation phase ends.

Phase II: The lexicographic ordering phase

In this phase, we calculate entropy values and sort objectives in decreasing order of entropy. First, we normalize the objective values, because the value range and unit of objectives we consider in this model have large differences. Then we perform entropy calculation of each normalized value and sum entropy values in the same column to get objective entropy (E_k). The applied formulas are as followed:

$$N_{ki} = \frac{f_{ki}}{\sum_{i=1}^{n} f_{ki}} \tag{1}$$

$$E_{ki} = -N_{ki} \times \ln N_{ki} \tag{2}$$

$$E_k = \sum_{i=1}^{n} E_{ki} \tag{3}$$

The interval of N_{ki} is (0,1) and $\ln N_{k,i}$ is a negative value with an interval $(-\infty, 0)$, therefore $E_{k,i}$ is always positive. From 0 to $\frac{1}{e}$, E_{ki} increases while from $\frac{1}{e}$ to 1, E_{ki} decreases. The decision matrix with entropy values is shown in Table 3.

The value of an objective entropy shows the level of difficulty in choosing jobs regarding this objective. If $E_k > E_l$, it means, compared to objective k, robots have more difficulties in distinguishing jobs at minimizing objective l, because the jobs have more similarities with each other in their objective l values. To avoid rejecting potential good solutions, the job selection starts from the objective with the largest entropy. We sort all objectives in decreasing order of entropy and relist these objectives in the matrix according to the order.

Phase III: The job selection phase

During this phase, we optimize objectives in the matrix from left to right sequentially based on their corresponding aspiration levels (AL). Recall that we use the best value of an objective and a corresponding acceptable deviation range to get the aspiration level of

Table 3 Decision matrix with entropy value

Job information			Objective values			
Job	Next operation	Next buffer	F1	F2	F3	F4
J_1	O_{1j}	B_{1m}	E_{11}	E_{21}	E_{31}	E_{41}
J_2	O_{2j}	B_{2m}	E_{12}	E_{22}	E_{32}	E_{42}
…	…	…	…	…	…	…
J_i	O_{ij}	B_{im}	E_{1i}	E_{2i}	E_{3i}	E_{4i}
…	…	…	…	…	…	…
J_n	O_{nj}	B_{nm}	E_{1n}	E_{2n}	E_{3n}	E_{4n}
			E_1	E_2	E_3	E_4

the objective. Once a unique solution is found or all objectives are optimized, the selection process stops. If more than one job remains in the matrix after all objectives are optimized, robots choose the job with the minimal value of the last optimized objective. For a dynamic feasible job set, the solution obtained by the decision matrix is efficient. We call this set 'partwise' efficient, which can be proven as follows:

For the optimization problem:

$$\min_{j \in J} f(f_1, \ldots, f_k)$$

Proposition *let $\hat{j}$ be the final solution from the matrix such that for at least one $k \in K$, we have $f_k(\hat{j}) \leq f_k(j)$ for all $j \in J$. Then $\hat{j} \in J_E$, where J_E is the set of partwise efficient solutions.*

Proof Suppose that $\hat{j}$ is not efficient. Then there is a $j \in J$ such that $f(j) < f(\hat{j})$. Let $K^* := \{k \in K : f_k(j) < f_k(\hat{j})\}$. So we have $f_k(j) < f_k(\hat{j})$ for $k \in K^*$, and $f_k(j) = f_k(\hat{j})$ for $k \in K$ and $k \notin K^*$, which is contradicting the assumption that $f_k(\hat{j}) \leq f_k(j)$ for at least one $k \in K$. □

The selection starts with the first left objective column. First, we sort all jobs in increasing order of objective values (the decision matrix used here is the original one (Table 2), not the matrix with entropy (Table 3)). Then we screen out jobs based on the minimal objective value in the matrix and the objective aspiration level. The process stops when we find a unique solution. Otherwise, the process goes to the next objective. We depict the three phases of our dynamic scheduling algorithm in Fig. 3 and present a simple example in Table 4.

Assume we have five feasible jobs. The objective with the maximal entropy is F3, namely time difference between due time and finishing time, and its aspiration level is 120% of the best value. Therefore, the aspiration level here is 1368 s, then J_4 and J_5 are screened out. For the second objective F2, if the aspiration level is the same as that of F3, then only J_3 is left in the matrix. In this case, the selection process stops and J_3 is assigned to the robot as its transportation task. When a jobs is assigned, even if it still resides in a buffer, other robots are no longer able to consider it as feasible.

We display a detailed description of how a robot uses the proposed algorithm to find a proper job to transfer in Algorithm. 1.

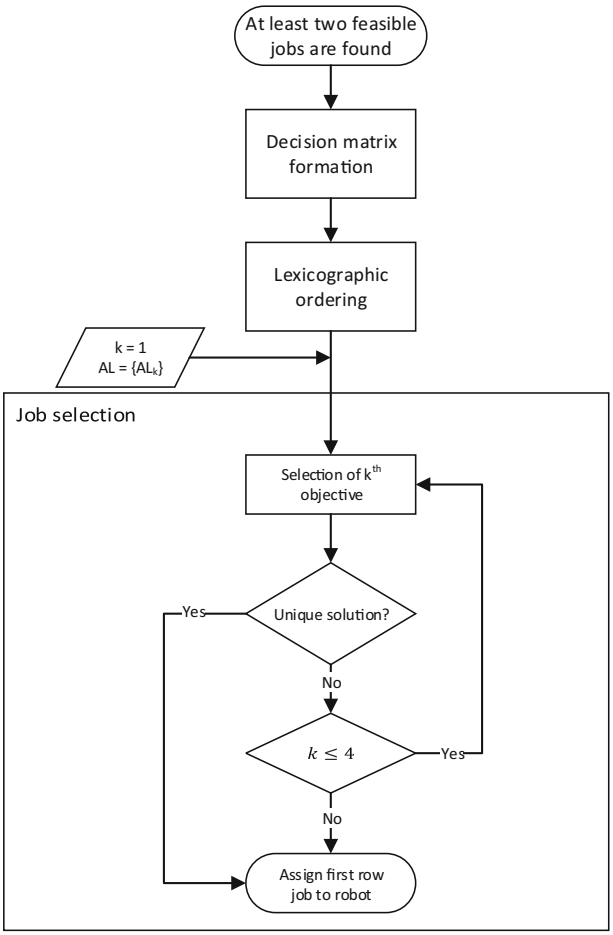

Fig. 3 Dynamic scheduling algorithm

Table 4 A job selection example

Job information			Objective values			
Job	Next operation	Next buffer	F3	F2	F1	F4
J_1	O_{11}	B_{13}	1140	620	f_{11}	f_{41}
J_2	O_{22}	B_{22}	1230	590	f_{12}	f_{42}
J_3	O_{31}	B_{22}	1298	450	f_{13}	f_{43}
J_4	O_{43}	B_{41}	1567	490	f_{14}	f_{44}
J_5	O_{52}	B_{53}	1800	700	f_{15}	f_{45}

Algorithm 1 Transportation Assignment Procedure

Require:

- Robot is available for transportation assignment.
- At least one feasible job exists in the system.

Feasible jobs checking
if only one feasible job **then**
  assign this job to robot
  end the procedure
else
  let J be the set of feasible jobs
  go to the next step
end if

Decision matrix formation
for all $J_i \in J$ **do**
  get the next operation O_{ij}
  let M_{ij} be the set of machines which are able to process O_{ij}
  for all $M_m \in M_{ij}$ **do**
    if the input buffer B_m^I of M_m has sufficient capacity to accept J_i **then**
      calculate corresponding objective values, $f_{1i}, f_{2i}, f_{3i}, f_{4i}$
      insert (list $J_i, O_{ij}, B_m^I, f_{1i}, f_{2i}, f_{3i}, f_{4i}$) in the matrix
    end if
  end for
  if no B_m^I of $M_m \in M_{ij}$ has sufficient capacity **then**
    delete J_i from J
  end if
end for
if the decision matrix is not empty **then**
  go to the next step
else
  end the procedure
end if

Lexicographic ordering
let K be the set of objectives
for all $k \in K$ **do**
  calculate the objective entropy E_k
end for
sort objective $k \in K$ in decreasing order of entropy E_k
relist objectives in the matrix based on the order
go to the next step

Job selection
for all $k \in K$ **do**
  sort jobs $J_i \in J$ in increasing order of objective value f_{ki}
  calculate objective aspiration level AL_k
  screen out all jobs with $f_{ki} > AL_k$
  if a unique solution is found **then**
    assign this job to robot
    end the procedure
  end if
end for
if no unique solution **then**
  assign first row job in the matrix to robot
end if

6 Computational results and comparisons

In this section, we study the performance of our dynamic scheduling algorithm by experiments. First, we show some simulation results to illustrate that our model significantly reduces the computational time required to solve the FJSP at large scale. Since no existing method has been introduced to solve the same problem with the same objectives, we carry out some experiments to compare our method with another algorithm proposed for the FJSP without transportation. Finally, we introduce a novel genetic algorithm in order to compare our proposed method with the results of the metaheuristic.

6.1 Simulation results

In this section, we present test instances that we have simulated to validate the efficiency of the suggested approach. To ease the simulation, we assume that:

- The workshop has a fixed number of machines. It is possible to shut off some machines for small size productions, but not possible to add more machines since in practice it is difficult to get additional capital investments.
- In instances with different numbers of machines, the setting of machines, such as operation processing times or locations, remains the same. The distance between different buffers is fixed.

In Fig. 4, we show the workshop environment for all instances. The system has 4 machines, 8 machine buffers and 2 system buffers. The number of jobs and the number of robots can vary.

In Table 5, we present the results of different test. We show CPU times in seconds. Instances at really large scale (i.e., 200–500 jobs) not previously considered by other researchers, appear to be solvable in acceptable time frames by utilizing our algorithm.

6.2 Comparisons in the FJSP without transportation

To show the flexibility of our simulation model and evaluate the efficiency of the proposed scheduling method, we carry out some experiments to compare our algorithm with another Pareto-based method proposed for solving the FJSP without transportation.

Kacem et al. (2002) applied a hybrid evolutionary algorithm with fuzzy logic (FL+EA) based on the Pareto concept to solve the multi-objective FJSP where move times between

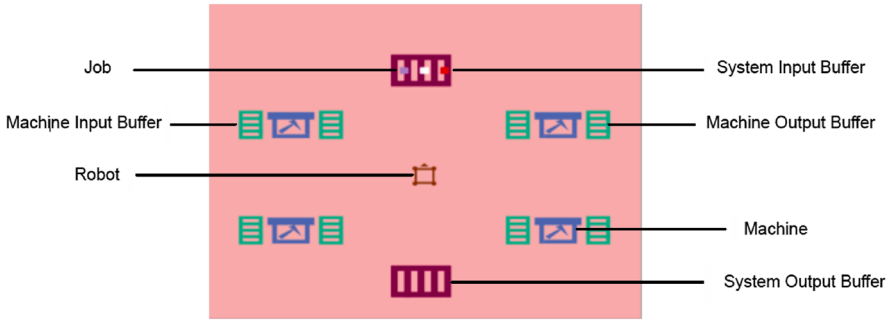

Fig. 4 Workshop layout

Table 5 Simulation results

Instances		Objectives*				CPU (s)
Problem size	$J \times M \times R$	F1	F2	F3	F4	
Small	$2 \times 2 \times 1$	719	116	570	300	0.721
	$2 \times 3 \times 2$	369	117	253	60	0.649
	$3 \times 3 \times 2$	389	186	147	143	0.867
Middle	$9 \times 3 \times 2$	1034	617	742	602	0.857
	$15 \times 3 \times 3$	1579	1023	348	921	1.051
	$15 \times 4 \times 3$	1398	1058	458	863	0.997
Large	$30 \times 4 \times 3$	2429	2100	691	1384	2.686
	$48 \times 4 \times 3$	4402	3437	1180	2775	1.824
	$100 \times 4 \times 3$	9016	7215	989	2519	16.778
	$200 \times 4 \times 4$	17,917	14,343	8061	2397	251.546
	$500 \times 4 \times 4$	43,567	36,298	7490	2399	5029.476

*F1: makespan; F2: transportation distance; F3: time difference; F4: critical waiting time.
All objective values are in seconds

Table 6 Comparison results of FL+EA and our method

Problem size	Objectives	FL + EA					Our	CPU (s)
$J \times M$		1	2	3	4	5	Method	
4×5	Makespan	18	18	16	16	–	16	0.563
	Critical workload	8	7	9	10	–	9	
	Total workload	32	33	35	34	–	32	
10×7	Makespan	15	17	18	16	16	16	0.707
	Critical workload	11	10	10	10	12	12	
	Total workload	61	64	63	66	60	65	
10×10	Makespan	8	8	7	–	–	8	1.070
	Critical workload	7	5	5	–	–	7	
	Total workload	41	42	45	–	–	41	

machines (buffers) are ignored. The objectives optimized are makespan, critical workload and total workload. They also considered the release times of jobs (some jobs are not available at time 0).

The comparison results are shown in Table 6 and we present the detailed solutions obtained by our model in Tables 7 and 8. The values of computational time are not provided in their paper. However, based on their algorithm, we can conclude that our proposed method is more effective as it generates efficient solutions with much less computational times.

6.3 Comparisons with genetic algorithm

GA methods are inspired by Darwin's principle of natural evolution. Starting with a set of candidate solutions, the best candidates in each generation are selected to form offspring as new candidates through crossover and mutation operations. These offspring are inserted into the population by replacing some of the weaker individuals in the previous generation.

Table 7 Our solution of instance 4×5

J_i	O_{ij}	M_{ij}	t_{ij}, tf_{ij}
J_1	O_{11}	4	3, 4
	O_{12}	2	4, 8
	O_{13}	4	8, 12
J_2	O_{21}	1	5, 7
	O_{22}	5	7, 12
	O_{23}	1	12, 16
J_3	O_{31}	3	1, 7
	O_{32}	2	8, 9
	O_{33}	1	9, 11
	O_{34}	4	12, 13
J_4	O_{41}	1	7, 8
	O_{42}	2	9, 10

Table 8 Our solutions of instance 10×7 and of instance 10×10

Instance 10×7				Instance 10×10			
J_i	O_{ij}	M_{ij}	t_{ij}, tf^*_{ij}	J_i	O_{ij}	M_{ij}	t_{ij}, tf_{ij}
J_1	O_{11}	5	2, 5	J_1	O_{11}	1	0, 1
	O_{12}	6	5, 6		O_{12}	3	1, 2
	O_{13}	1	6, 10		O_{13}	4	3, 4
J_2	O_{21}	7	4, 7	J_2	O_{21}	1	2, 4
	O_{22}	1	10, 12		O_{22}	10	4, 5
					O_{23}	3	5, 7
J_3	O_{31}	7	11, 12				
	O_{32}	3	12, 13	J_3	O_{31}	10	0, 1
	O_{33}	5	13, 15		O_{32}	4	2, 3
					O_{33}	7	5, 6
J_4	O_{41}	6	6, 10				
	O_{42}	4	13, 14	J_4	O_{41}	7	0, 1
	O_{43}	1	14, 15		O_{42}	2	1, 3
					O_{43}	4	6, 7
J_5	O_{51}	2	7, 8				
	O_{52}	1	12, 14	J_5	O_{51}	9	0, 2
	O_{53}	2	15, 16		O_{52}	9	2, 3
					O_{53}	4	5, 6
J_6	O_{61}	3	5, 9				
	O_{62}	3	9, 10	J_6	O_{61}	6	2, 4
	O_{63}	3	10, 12		O_{62}	9	4, 6
					O_{63}	9	6, 7

Table 8 continued

Instance 10 × 7				Instance 10 × 10			
J_i	O_{ij}	M_{ij}	t_{ij}, tf^*_{ij}	J_i	O_{ij}	M_{ij}	t_{ij}, tf_{ij}
J_7	O_{71}	7	7, 9				
	O_{72}	7	12, 15	J_7	O_{71}	1	1, 2
	O_{73}	6	15, 16		O_{72}	3	2, 3
					O_{73}	4	4, 5
J_8	O_{81}	2	4, 5				
	O_{82}	4	5, 13	J_8	O_{81}	5	0, 2
	O_{83}	2	13, 15		O_{82}	2	3, 6
					O_{83}	2	6, 8
J_9	O_{91}	3	1, 5				
	O_{92}	2	5, 7	J_9	O_{91}	6	1, 2
	O_{93}	7	9, 11		O_{92}	7	2, 3
					O_{93}	6	4, 5
J_{10}	O_{101}	2	0, 4				
	O_{102}	6	4, 5	J_{10}	O_{101}	6	0, 1
	O_{103}	1	5, 6		O_{102}	4	1, 2
					O_{103}	7	3, 5

*t_{ij}, tf_{ij}: starting time and finishing time of operation O_{ij}

The proposed GA method follows the following general steps of evolutionary algorithms:

1. *Initialization:* The aim here is to obtain some feasible solutions with a given population size.
2. *Fitness evaluation:* We employ NSGA-II to find non-dominated solutions with the same four objectives as we optimize in our dynamic algorithm. We explain the details of this in Sect. 6.3.2.
3. *Generation of offspring:* Based on the results of the fitness evaluation, some of individuals are randomly selected as parents to generate offspring through crossover and mutation operations.
4. The second and third steps are repeated until a set of solutions satisfies decision maker or until reaching the stopping criterion. In our case, we use a given number of generations as the stopping criterion.

6.3.1 Chromosome presentation

In existing GA research work to solve the FJSP (Pezzella et al. 2008; Zandieh et al. 2008; Chaudhry et al. 2013), researchers built their chromosomes with triples, namely operation, job and machine (Fig. 5). In order to implement our GA to solve the FJSP with transportation assignment, we form adapted triples (J_i, B_m, R_p), one for each transportation task, where:
J_i is the job to transfer;
B_m is the buffer where the robot transfers J_i to;
R_p is the robot which executes this transportation task.

The length of the chromosome corresponds to the total number of transportation tasks. Recall that for each job, transportation tasks include transferring the job to machines which execute its operations and transferring it to system output system. Therefore, the number of transportation tasks for a job equals the number of operations belonging to this job added by 1.

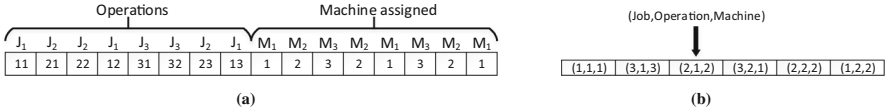

Fig. 5 Chromosomes proposed by Chaudhry et al. (2013) (a) and Zandieh et al. (2008) (b)

However, for the FJSP with partial flexibility, not all machines can process all operations resulting in the generation of offspring naturally deemed infeasible when the crossover and mutation operations execute. In that case, we have to regenerate offspring to manage the population size. Therefore, we think it is better to use a triple string of indexes between 0 and 1 to code chromosomes and interpret the string depending on actual transportation assignment circumstances. To articulate the coding and interpretation process, we give a simple example here.

Assume we have 2 jobs (J_1, J_2) each with two operations (O_{i1}, O_{i2}), two machines (M_1, M_2) both able to process all operations, and two robots (R_1, R_2). The length of chromosomes is the total number of transportation tasks, which equals 6. Then we generate a random initial chromosome as

Job	0.23	0.53	0.32	0.35	0.75	0.67
Buffer	0.81	0.83	0.25	0.62	0.96	0.27
Robot	0.07	0.43	0.82	0.13	0.32	0.29

To interpret, we follow these steps:

Step 1: List all feasible jobs, $J = \{J_1, J_2\}$. Use the function

$$I(x) = \lfloor x \times N + 1 \rfloor$$

where

- $I(x)$:the item position in the list
- N: the number of items in the list
- $\lfloor\rfloor$: floor function, taking as input a real number and gives as output the greatest integer less than or equal to

to interpret the index 0.23 in the first gen, then the corresponding job is J_1, the first item ($I(0.23) = \lfloor 0.23 \times 2 + 1 \rfloor = 1$) in J.

Step 2: Find a subset of compatible machines for the next operation of J_1, $M = \{M_1, M_2\}$. By the same function, we can get M_2, the second item ($I(0.81) = \lfloor 0.81 \times 2 + 1 \rfloor = 2$) in M.

Step 3: With the robot list $R = \{R_1, R_2\}$, we get R_1, the first item in R.

After obtaining the first transportation task, the process is repeated with new feasible lists until we process all indexes. Therefore, the chromosome has the schedule as below:

Job	J_1	J_2	J_1	J_1	J_2	J_2
Buffer	B_2^I	B_2^I	B_1^I	B_0^O	B_2^I	B_0^O
Robot	R_1	R_1	R_2	R_1	R_1	R_1

6.3.2 Fitness evaluation

In NSGA-II methods, no objective is regarded as having priority over others. All individuals are sorted based on non-domination into each front. The individuals in the first front do not have any individual who dominates them in the current population, and the individuals in the second front are only dominated by the individuals in the first front. Individuals in different fronts are assigned different fitness values according to the rank of their front, for example, 1 for the first front and 2 for the second front.

The concept of crowding distance is introduced to rank individuals in the same front. It calculates how close an individual is to its neighbors. Fitness value and crowding distance are double standards to select elitist individuals from the population. Individuals are selected when their fitness value are minimal. If two individuals have the same fitness value the one with the greater crowding distance is preferential. The adopted method to take non-dominated sort and to calculate crowding distance follows the one proposed by Deb et al. (2002).

Note that we use the Netlogo simulator to calculate values of objectives for each individual. Rather than estimating the distance between different buffers, using the simulator to simulate how robots work and obtain the distance directly from the model could produce more accurate results and get closer to practical applications.

In our NSGA-II method, we sort the population with the same four objectives: makespan, transportation distance, time difference and critical waiting time. In the end, it offers users a set of Pareto solutions, rather than a single solution. This method requires decision makers to pick their preferred solution after seeing all Pareto solutions. In the case study, we show feasible solutions without bias and compare the results from the dynamic scheduling algorithm we discussed in Sect. 5.

6.3.3 Genetic operators

The NSGA-II method uses crossover and mutation operators to generate offspring. We assume that it is possible to generate good children even if the parents are relatively bad in the population. In each iteration, we generate children with the population size by randomly selecting parents from the population. The crossover operator divides the two parent chromosomes in the middle, and the second parts of the parent chromosomes are swapped to get new off-springs. The chromosome structure avoids generating infeasible off-springs and legalizing after. The mutation operator simulates the mutation observed in nature. With a given mutation rate, each point on a gen can be changed randomly.

The offspring and the parent populations are combined and sorted together based on non-domination. The new generation is filled by each front subsequently until it reaches the population size. If it needs to pick some individuals from a front, individuals are selected by the crowding distance in descending order. These processes repeat until reaching the stopping criterion.

6.4 Comparison results of GA and our method

To obtain potential efficient solutions, the population size and the number of generation are 50 and 1000, respectively. Table 9 indicates the comparison of the solutions by the two approaches. We have observed that the solutions from our method dominate most of GA solutions.

Table 9 Comparison results of GA and our method

Problem size	Obj.*	GA							Our	CPU
$J \times M \times R$		1	2	3	4	5	6	CPU	method	
$2 \times 2 \times 1$	F1	730	743	800	–	–	–	996.657	719	0.721
	F2	179	177	209	–	–	–		116	
	F3	543	616	588	–	–	–		570	
	F4	347	354	287	–	–	–		300	
$3 \times 3 \times 2$	F1	391	461	407	390	–	–	1040.235	389	0.867
	F2	192	217	204	210	–	–		186	
	F3	352	110	341	278	–	–		147	
	F4	194	171	131	117	–	–		143	
$6 \times 4 \times 2$	F1	618	638	710	684	–	–	1468.847	581	0.867
	F2	459	423	415	435	–	–		440	
	F3	166	174	425	520	–	–		128	
	F4	230	242	300	301	–	–		283	
$9 \times 3 \times 2$	F1	1045	1010	1045	1173	1125	–	3080.132	1034	0.857
	F2	618	645	654	651	662	–		617	
	F3	746	759	673	872	733	–		742	
	F4	675	590	649	605	585	–		602	
$15 \times 4 \times 3$	F1	1319	1413	1425	1419	1440	–	2651.562	1398	0.997
	F2	1085	1078	1212	1103	1066	–		1058	
	F3	504	866	493	870	862	–		458	
	F4	1017	1013	1239	940	1185	–		863	
$52 \times 4 \times 3$	F1	4960	5162	5191	5290	4659	5012	37,890.129	4873	5.728
	F2	3993	4131	3941	4032	4231	4057		4048	
	F3	1491	1322	1531	1127	1052	1601		1041	
	F4	2773	2427	2646	2557	2457	2005		2109	

*F1: makespan; F2: transportation distance; F3: time difference; F4: critical waiting time

7 Conclusion and future work

Research on the FJSP with transportation assignment is relatively rare. In this work, we develop a novel scheduling algorithm to solve the multi-objective flexible job shop problem with transportation assignment. A simulation model with 4 agents is proposed to simulate the whole production process. Whenever a robot agent is available, it employs the algorithm with all real-time data derived from the simulation model to generate partwise efficient dynamic feasible solutions.

We test the proposed algorithm by using several benchmarks. The comparative results show that our algorithm outperforms all the benchmarked algorithms. One of the major advantages of our method is the reduction of computational time required while providing equivalent or better results. Additionally, the single solution provided by the proposed method makes it easier for decision makers to execute production plans. For automatic manufacturing workshops, this model can be implemented directly to help robots execute transportation tasks. We recognize that the assumption constraints of our model impose limitations that would benefit from additional work. Some future research directions can be recommended.

Additional machine conditions could be considered, such as setup time, maintenance, or breakdown situations, which makes the assumptions more realistic and more complete. The insertion or withdrawal of robots during production peak or slack period also can be interesting to investigate.

Acknowledgements Acknowledgement is made to the Normandy region and the European Union for the support of this research through the European FEDER program by funding project entitled Xterm.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, *34*(3), 391–401. https://doi.org/10.1287/mnsc.34.3.391.

Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & Operations Research*, *73*, 56–66. https://doi.org/10.1016/j.cor.2016.03.009.

Al-Hinai, N., & ElMekkawy, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, *132*(2), 279–291. https://doi.org/10.1016/j.ijpe.2011.04.020.

Boudhar, M., & Haned, A. (2009). Preemptive scheduling in the presence of transportation times. *Computers & Operations Research*, *36*(8), 2387–2393. https://doi.org/10.1016/j.cor.2008.09.006.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, *41*(3), 157–183. https://doi.org/10.1007/BF02023073.

Branke, J., Branke, J., Deb, K., Miettinen, K., & Slowiński, R. (2008). *Multiobjective optimization: Interactive and evolutionary approaches* (Vol. 5252). Berlin: Springer.

Brucker, P., & Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, *45*(4), 369–375. https://doi.org/10.1007/BF02238804.

Buddala, R., & Mahapatra, S. S. (2019). An integrated approach for scheduling flexible job-shop using teaching–learning-based optimization method. *Journal of Industrial Engineering International*, *15*(1), 181–192. https://doi.org/10.1007/s40092-018-0280-8.

Charnes, A., Cooper, W. W., & Ferguson, R. O. (1955). Optimal estimation of executive compensation by linear programming. *Management Science*, *1*(2), 138–151. https://doi.org/10.1287/mnsc.1.2.138.

Chaudhry, I. A., Khan, A. M., & Khan, A. A. (2013). A genetic algorithm for flexible job shop scheduling. *Proceedings of the World Congress on Engineering*, *1*, 1–6. https://doi.org/10.1109/ROBOT.1999.772512.

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. https://doi.org/10.1023/A:1018930406487.

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*(2), 395–408. https://doi.org/10.1016/j.ejor.2009.01.008.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017.

Doganis, P., & Sarimveis, H. (2008). Optimal production scheduling for the dairy industry. *Annals of Operations Research*, *159*(1), 315–331. https://doi.org/10.1007/s10479-007-0285-y.

Drießel, R., & Mönch, L. (2012). An integrated scheduling and material-handling approach for complex job shops: A computational study. *International Journal of Production Research*, *50*(20), 5966–5985. https://doi.org/10.1080/00207543.2011.639099.

Eloundou, J. (2016). *Modélisation multi-contraintes d'un système de production flexible*. Ph.D. thesis, INSA de Rouen

Fonseca, C. M., & Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *28*(1), 26–37. https://doi.org/10.1109/3468.650319.

Gambardella, L., & Mastrolilli, M. (1996). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, *3*(3), 3–20. https://doi.org/10.1002/(SICI)1099-1425(200001/02)3:1<3::AID-JOS32>3.0.CO;2-Y.

Gao, J., Sun, L., & Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, *35*(9), 2892–2907. https://doi.org/10.1016/j.cor.2007.01.001.

Gao, K. Z., Suganthan, P. N., Chua, T. J., Chong, C. S., Cai, T. X., & Pan, Q. K. (2015). A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert Systems with Applications*, *42*(21), 7652–7663. https://doi.org/10.1016/j.eswa.2015.06.004.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. S. (2016). Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. *Journal of Intelligent Manufacturing*, *27*(2), 363–374. https://doi.org/10.1007/s10845-014-0869-8.

Hurink, J., & Knust, S. (2005). Tabu search algorithms for job-shop problems with a single transport robot. *European Journal of Operational Research*, *162*(1), 99–111. https://doi.org/10.1016/j.ejor.2003.10.034.

Hurink, J., Jurisch, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, *15*(4), 205–215. https://doi.org/10.1007/BF01719451.

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, *60*(3), 245–276. https://doi.org/10.1016/S0378-4754(02)00019-8.

Karimi, S., Ardalan, Z., Naderi, B., & Mohammadi, M. (2017). Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*, *41*, 667–682. https://doi.org/10.1016/j.apm.2016.09.022.

Karthikeyan, S., Asokan, P., Nickolas, S., & Page, T. (2015). A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems. *International Journal of Bio-inspired Computation*, *7*(6), 386–401. https://doi.org/10.1007/s00170-014-5753-3.

Kumar, R., & Pandey, V. (2015). Job shop scheduling with alternate process plan by using genetic algorithm. *International Journal of Research in Advent Technology*, *3*(9), 91–98. https://doi.org/10.1016/j.eswa.2012.01.211.

Langston, M. A. (1987). Interstage transportation planning in the deterministic flow-shop environment. *Operations Research*, *35*(4), 556–564. https://doi.org/10.1287/opre.35.4.556.

Lee, J., Kao, H. A., & Yang, S. (2014). Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp*, *16*, 3–8. https://doi.org/10.1016/j.procir.2014.02.001.

Li, J. Q., Pan, Q. K., Suganthan, P., & Chua, T. (2011). A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *52*(5–8), 683–697. https://doi.org/10.1007/s00170-010-2743-y.

Li, Z., Qian, B., Hu, R., Chang, L., & Yang, J. (2019). An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups. *Knowledge-Based Systems*, *173*, 83–112. https://doi.org/10.1016/j.knosys.2019.02.027.

Liouane, N., Saad, I., Hammadi, S., & Borne, P. (2007). Ant systems & local search optimization for flexible job shop scheduling production. *International Journal of Computers Communications & Control*, *2*(2), 174–184. https://doi.org/10.15837/ijccc.2007.2.2350.

Macal, C. M., & North, M. J. (2005). Tutorial on agent-based modeling and simulation. In *Proceedings of the winter simulation conference, 2005* (p. 14). IEEE. https://doi.org/10.1109/WSC.2005.1574234

Marchi, E., & Oviedo, J. A. (1992). Lexicographic optimality in the multiple objective linear programming: The nucleolar solution. *European Journal of Operational Research*, *57*(3), 355–359. https://doi.org/10.1016/0377-2217(92)90347-C.

Moslehi, G., & Mahnam, M. (2011). A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, *129*(1), 14–22. https://doi.org/10.1016/j.ijpe.2010.08.004.

Naderi, B., Javid, A. A., & Jolai, F. (2010). Permutation flowshops with transportation times: Mathematical models and solution methods. *The International Journal of Advanced Manufacturing Technology*, *46*(5–8), 631–647. https://doi.org/10.1007/s00170-009-2122-8.

Naderi, B., Zandieh, M., Balagh, A. K. G., & Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications*, *36*(6), 9625–9633. https://doi.org/10.1016/j.eswa.2008.09.063.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 603–615. https://doi.org/10.1007/s10845-015-1039-3.

Nouri, H. E., Driss, O. B., & Ghédira, K. (2016). Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model. *Computers & Industrial Engineering*, *102*, 488–501. https://doi.org/10.1016/j.cie.2016.02.024.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, *35*(10), 3202–3212. https://doi.org/10.1016/j.cor.2007.02.014.

Rossi, A. (2014). Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *International Journal of Production Economics*, *153*, 253–267. https://doi.org/10.1016/j.ijpe.2014.03.006.

Rossi, A., & Dini, G. (2007). Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. *Robotics and Computer-Integrated Manufacturing*, *23*(5), 503–516. https://doi.org/10.1016/j.rcim.2006.06.004.

Roy, B., & Mousseau, V. (1996). A theoretical framework for analysing the notion of relative importance of criteria. *Journal of Multi-Criteria Decision Analysis*, *5*(2), 145–159. https://doi.org/10.1002/(SICI)1099-1360(199606)5:2<145::AID-MCDA99>3.0.CO;2-5.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

Siebers, P. O., Macal, C. M., Garnett, J., Buxton, D., & Pidd, M. (2010). Discrete-event simulation is dead, long live agent-based simulation!. *Journal of Simulation*, *4*(3), 204–210. https://doi.org/10.1057/jos.2010.14.

Sotskov, Y. N., & Shakhlevich, N. V. (1995). Np-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, *59*(3), 237–266. https://doi.org/10.1016/0166-218X(95)80004-N.

Tay, J. C., & Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, *54*(3), 453–473. https://doi.org/10.1016/j.cie.2007.08.008.

Thörnblad, K., Strömberg, A. B., Patriksson, M., & Almgren, T. (2013). *A competitive iterative procedure using a time-indexed model for solving flexible job shop scheduling problems*. Gothenburg: Department of Mathematical Sciences: Division of Mathematics, Chalmers University of Technology and University of Gothenburg.

Van Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, *40*(1), 113–125. https://doi.org/10.1287/opre.40.1.113.

Wang, H., Jiang, Z., Wang, Y., Zhang, H., & Wang, Y. (2018). A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization. *Journal of Cleaner Production*, *188*, 575–588. https://doi.org/10.1016/j.jclepro.2018.03.254.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *48*(2), 409–425. https://doi.org/10.1016/j.cie.2005.01.018.

Xing, L. N., Chen, Y. W., Wang, P., Zhao, Q. S., & Xiong, J. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing*, *10*(3), 888–896. https://doi.org/10.1016/j.asoc.2009.10.006.

Yuan, Y., & Xu, H. (2013). An integrated search heuristic for large-scale flexible job shop scheduling problems. *Computers & Operations Research*, *40*(12), 2864–2877. https://doi.org/10.1016/j.cor.2013.06.010.

Yuan, Y., Xu, H., & Yang, J. (2013). A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Applied Soft Computing*, *13*(7), 3259–3272. https://doi.org/10.1016/j.asoc.2013.02.013.

Zandieh, M., Mahdavi, I., & Bagheri, A. (2008). Solving the flexible job-shop scheduling problem by a genetic algorithm. *Journal of Applied Sciences*, *8*(24), 4650–4655. https://doi.org/10.3923/jas.2008.4650.4655.

Zhang, Q., Manier, H., & 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*, *39*(7), 1713–1723. https://doi.org/10.1016/j.cor.2011.10.007.

Zhang, Q., Manier, H., & Manier, M. A. (2014). A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *International Journal of Production Research*, *52*(4), 985–1002. https://doi.org/10.1080/00207543.2013.828164.