Accepted in J Syst Sci Syst Eng

The effect of worker learning on scheduling jobs in a hybrid flow shop; a bi-objective approach

Farzad Pargar[a,*], Mostafa Zandieh[b], Osmo Kauppila[a], Jaakko Kujala[a],

[a]*Industrial Engineering and Management, Faculty of technology, University of Oulu, Oulu, Finland*
[b]*Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, Tehran, Iran*

ABSTRACT

This paper studies learning effect as a resource utilization technique that can model improvement in worker's ability as a result of repeating similar tasks. By considering learning of workers while performing setup times, a schedule can be determined to place jobs that share similar tools and fixtures next to each other. The purpose of this paper is to schedule a set of jobs in a hybrid flow shop (HFS) environment with learning effect while minimizing two objectives that are in conflict: namely maximum completion time (makespan) and total tardiness. Minimizing makespan is desirable from an internal efficiency viewpoint, but may result in individual jobs being scheduled past their due date, causing customer dissatisfaction and penalty costs. A bi-objective mixed integer programming model is developed, and the complexity of the developed bi-objective model is compared against the bi-criteria one through numerical examples. The effect of worker learning on the structure of assigned jobs to machines and their sequences is analyzed. Two solution methods based on the hybrid water flow like algorithm and non-dominated sorting and ranking concepts are proposed to solve the problem. The quality of the approximated sets of Pareto solutions is evaluated using several performance criteria. The results show that the proposed algorithms with learning effect perform well in reducing setup times and eliminate the need for setups itself through proper scheduling.
Keywords: Bi-objective scheduling; hybrid flow shop; learning effect; meta-heuristic.

1. Introduction

Scheduling is a methodology that optimizes the use of resources by ordering a sequence of jobs assigned to each resource. It plays a fundamental role in production planning of manufacturing systems, and it is an important success factor in the modern competitive marketplace (Yue and Wan, 2017). The hybrid flow shop (HFS) is one of the most recognized scheduling problems, and has attracted much attention given its complexity and practical relevance. The topic is intensively studied in all kinds of real world scenarios including the electronics, paper and textile industries (Ruiz and Rodriguez, 2010). A HFS consists of a series of production stages, each of which has several identical machines operating in parallel. Some stages may have only one machine, but at least one stage must have multiple machines to be qualified as a hybrid flow shop. In a HFS environment, a set of *n* jobs is to be processed optimizing a given objective function. A job must pass through all stages and must be processed by exactly one machine at every stage. The layout of a *g*-stage hybrid flow shop environment is illustrated in Figure 1.

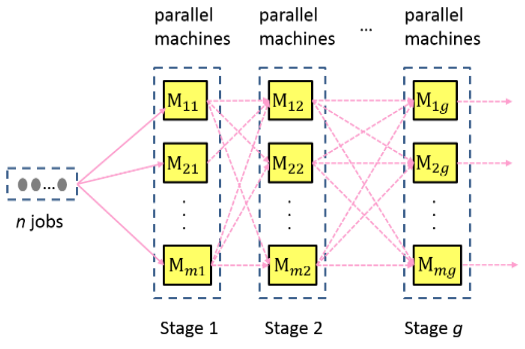


Figure 1. The structure of a typical hybrid flow shop environment.

*Corresponding author. Email: farzad.pargar@oulu.fi

Two decisions are taken at once to solve HFS scheduling problems: determining the assignment of jobs to the parallel machines and the sequence of the jobs allocated to each machine. Between the processing of two successive jobs in each machine a sequence-dependent setup time (SDST) such as cleaning up, changing tools and removal of failures occurs. Setup time is a significant factor for production scheduling, and it may easily consume more than 20% of available machine capacity if not well handled (Pinedo, 2015). The human factor has a significant effect on setup times, and it can be considered that by repeating a similar setup task the worker is able to perform it at an increasing pace. This phenomenon, in which the actual setup time of a job is shorter if it is scheduled later rather than earlier in the sequence is known as the "learning effect" in the literature (e.g. Biskup (1999), Wang et al. (2000)). This effect has been proven to exist by many empirical studies (see the examples in Biskup (2008)), and accounting for it is of importance in today's manufacturing and service organizations where reliable and low-cost products/services should be delivered on time (Soroush, 2015).

An inherent characteristic of scheduling tasks is a high level of human activities, so the number of activities subject to learning is great. Hence, different approaches for modeling learning in scheduling environments have been suggested in the literature (e.g. Cheng and Wang (2000), Kuo and Yang (2006)). The way learning should be modeled depends on the production environment. In this paper, a position-based learning approach is used which means that learning is effected by the pure number of jobs being processed on machines. The position-based approach assumes that the actual processing of the job is machine-driven and has near to none human intervention (Biskup, 1999). Practical examples of the position-based learning approach include the processing of circuit boards and memory chips.

Recently there has been a growing research interest in multi-objective scheduling problems (e.g. Yenisey and Yagmahan (2014), Karimi and Davoudpour (2016)). This paper addresses a bi-objective hybrid flow shop scheduling problem to represent the manufacturers' and the consumers' real world concerns by minimizing makespan and total tardiness. Makespan is a measure to calculate the total length of the schedule and total tardiness is a measure to calculate lateness of jobs which are completed after their due dates. Minimizing the tardiness causes external efficiency and reduces the penalties incurred for late jobs while minimizing the makespan causes internal efficiency and keeps the work-in-process inventory at a low level. Minimization of makespan and total tardiness are in conflict because a single optimal solution may not exist with respect to the two objectives. A sub-schedule which minimizes the total tardiness for the first r jobs may retain a relatively large makespan which worsens the total tardiness for the remaining n−r jobs. Therefore, there is a need for multi-objective scheduling approach to and achieve the objectives simultaneously. The bi-objective HFS problem is NP-hard, since even its simpler version with a single criterion, one machine in the first stage, two machines in the second and no learning effect is already NP-hard (Gupta, 1998). Therefore, it is tough to solve the proposed problem in a reasonable computational time while using traditional approaches (exact methods). This limitation has prompted researchers to develop a variety of heuristics and meta-heuristics to solve these problems quickly with reasonable solution quality. Heuristics are usually problem-specific methods, while meta-heuristics are nature-inspired algorithms for solving tough optimization problems. They are guided random search methods, which mimic the principles of evolution. Meta-heuristics are powerful problem-independent methods that can be applied to a wide range of problems.

According to the best of our knowledge, this study is the first work which examines the impact of worker learning on the structure of assigned jobs to machines and their sequences in bi-objective hybrid flow shop systems. Previous studies on learning effect in the scheduling field have mainly focused on different approaches to modeling learning in various shop environments (e.g. Cheng et al. (2014), Wu and Wang (2016)). This paper presents several illustrative examples to show the potential time reductions that the modeling of learning effect may result in, both through reducing setup times and even through changing the job schedule. The impact of the problem's characteristics, such as the optimization criteria and the learning rates, on the structure of optimal schedule of jobs is visualized. In this paper, the bi-objective and bi-criteria (single objective) versions of the proposed problem are compared to illustrate the complexity of bi-objective optimization. This bi-objective HFS problem with learning effect and SDSTs is formulated as a mixed integer optimization model, and two hybrid meta-heuristic algorithms are developed for solving it.

The rest of the paper is organized as follows: In Section 2, we present a literature review related to hybrid flow shop scheduling and learning effect. The problem formulation and the notation used is given in detail in Section 3. Section 4 deals with the algorithms offered for solving the proposed multi-objective problem. The computational analysis of the developed model and algorithms are given in Section 5. The impact of learning effect on the schedule of jobs and the complexity of the developed bi-objective HFS model are also discussed. Subsequently, non-dominated solutions of the algorithms are evaluated with different performance metrics. Finally, the conclusions and future work are laid out in Section 6.

2. Literature review

In this section, we first survey the literature on the HFS scheduling problem and then continue with the literature on scheduling problem with learning effect. Finally, we review the solution approaches for solving the multi-objective HFS problems.

2.1. Hybrid flow shop scheduling

In the past four decades, extensive work has been done in the field of HFS scheduling approach. The literature on the hybrid flow shop scheduling is abundant, for example, a comprehensive literature review in hybrid flow shop can be found in Quadt and Kuhn, (2007), Ruiz and Vázquez-Rodríguez (2010), and Ribas et al. (2010). Hybrid flow shop as a common manufacturing environment has a "standard" form which can be found in Ruiz and Vázquez-Rodríguez (2010). The modification, removal or addition of assumptions and/or constraints to the standard problem described above leads to different HFS variants. In accordance with Graham et al. (1979), scheduling problems can be described with a triplet α|β|γ. The considered problem in this research which is a *m*-machine *g*-stage flexible (hybrid) flowshop scheduling problem to minimize makespan and total tardiness with sequence-dependent setup times and learning effect will be noted as FHg, $\left(\left(PM^{(t)}\right)_{t=1}^{g}\right)$| Ssd, LE |{ Cmax, ∑Tj } where *P* indicates parallel machines (identical). Here, we only focus on *g*-stage cases of HFS scheduling problems.

One of the most prevailing assumptions by many researchers is the integration of sequence-dependent setup times into different shop scheduling environments. Scheduling problems with SDSTs are among the most difficult classes of scheduling problems. A one-machine scheduling problem with SDST is NP-hard (Zandieh et al., 2006). Kurz and Askin (2003) introduced the mathematical model of the HFS problem with sequence-dependent setup times. A comprehensive review of the literature on scheduling problems involving setup times can be found in Allahverdi (2015).

Ruiz & Vázquez-Rodriguez (2010) reviewed about 200 HFS papers. According to their study, around 55% of the reviewed papers consider *g* stage with the identical parallel machine. This shop configuration has attracted a lot of attention given its complexity and practical relevance. They also showed that the literature is heavily biased towards the makespan criterion with a 60% of the references studying this single objective. Ruiz & Vázquez-Rodriguez (2010) specifically state about the importance of tardiness criterion in the HFS problems: "It is striking to see that from all surveyed papers, only a total of 1% deal with the earliness–tardiness criterion, which is so important for real problems."

2.2. Learning effect and scheduling problems

Learning and its effect on productivity are well recognized in different industrial settings. Although the learning effect has been applied to industry for more than sixty years, it has been adopted in the scheduling field just in the recent years (Cheng et al., 2014). There exist a growing interest in the literature to study scheduling problems with learning effect. Biskup (1999) and Cheng and Wang (2000) were pioneers that introduced the concept of learning to the scheduling problems. Biskup (1999) considered a single-machine scheduling problem with a position-based learning effect and assumed the two objectives of minimizing the weighted sum of completion time deviations from a common due date and the sum of job completion times. Cheng and Wang (2000) considered a single machine scheduling problem to minimize the maximum lateness by using a piecewise linear processing time function to model the learning effect. Eren and Güner (2008) considered learning effect in a two-machine flow shop scheduling. They analyzed the bi-criteria flow shop problem to minimize a linear combination of the makespan and the total completion time. Eren and Güner (2009) considered learning effect in a bi-criteria parallel machine scheduling problem to minimize the weighted sum of total completion time and total tardiness.

The HFS problem with learning effects has been also investigated in several studies. Pargar and Zandieh (2012) introduced the learning effect into hybrid flow shop scheduling problems with the objective of minimization of the weighted sum of makespan and total tardiness. Behnamian and Zandieh (2013) considered a position-based learning effects to solve HFS with the consideration of tardiness and earliness penalties as the objective function. Mousavi et al. (2016) considered a position-based learning effect to solve re-entrant HFS problem with the objective of minimizing makespan and total tardiness. A comprehensive review of scheduling problems with learning effects can be found in Biskup (2008) and Agnetis et al. (2014).

2.3. Solution approaches

Researchers have proposed many different approaches to solving the HFS problems. Exact methods, heuristics, and meta-heuristics are the three important solution approaches addressed in the literature (Govindan et al., 2017). Meta-heuristics have attracted significant research effort during the past twenty years to solve multi-objective problems, and they are still one of the popular research topics in the field of evolutionary computation (Zhou et al., 2011). The main reason for metaheuristics popularity for solving multi-objective problems can be can be described by their population-based nature and ability to find multiple optimal simultaneously. Multi-objective meta-heuristics can approximate the Pareto front in a single optimization run. Various meta-heuristic algorithms have ever been derived for multi-objective optimization problems. Multi-objective meta-heuristics differ by their fitness assignment procedure, elitism, or diversification approaches. A summary of basic definitions of multi-objective optimization problem can be found in (Tavakkoli-Moghaddam et al., 2008). Choong et al. (2011) review different meta-heuristic methods used in hybrid flow shop scheduling problem.

The first method for solving multi-objective problems by modifying the GA was vector evaluated genetic algorithm proposed by Schaffer, (1985). Murata, Ishibuchi, & Tanaka, (1996) proposed a multi-objective genetic algorithm to transform the multiple objectives into single objective by using dynamic weighting which randomly assigns different weight values to different objectives. Non-dominated sorting genetic algorithm version 2 (NSGA-II) was developed by Deb et al. (2002) to alleviate difficulties of NSGA algorithm which was developed by Srinivas and Deb (1995). Jadaan et al. (2008) developed a non-dominated sorting genetic algorithm (NRGA) to combine the new ranked based roulette wheel selection algorithm with Pareto-based population ranking algorithm. Rashidi et al. (2010) proposed a hybrid multi-objective parallel genetic algorithm (GA) for the HFS problem with sequence-dependent setup times, unrelated parallel machines and processor blocking to minimize the maximum tardiness and makespan. Mousavi et al. (2012) reviewed several kinds of evolutionary algorithms based on the GA which are applied for scheduling multi-objective HFS models.

Multi-objective meta-heuristics differ by their fitness assignment procedure, elitism, or diversification approaches. In the recent past, several nature-inspired multi-objective meta-heuristics such as the differential evolution, particle swarm optimization, and bacterial foraging optimization have been applied to solve the multi-objective problems. Comprehensive surveys of multi-objective meta-heuristics can be found in Jones et al. (2002), Coello et al. (2007) and Zhou et al. (2011).

3. Mixed integer linear programming formulation

The bi-objective HFS problem presented in this paper is formulated as a mixed integer programming model. Our proposed model considers position-based learning effect in a static and deterministic environment. All jobs are always available and processed with no breakdowns or scheduled/unscheduled maintenance. All other assumptions in the standard form of the HFS problem are considered in this study. The notation used for problem formulation is summarized in Table 1.

Table 1. The notation used for mathematical modeling.

Sets	
I	Set of jobs, with its elements numbered for convenience from *0* to |*n*|
J	Set of jobs, with its elements numbered for convenience from *1* to |*n*|
M	Set of machines, with its elements numbered for convenience from *1* to |m_t|
R	Set of positions, with its elements numbered for convenience from *1* to |*n*|
T	Set of stages, with its elements numbered for convenience from *1* to |*g*|
Parameters	
n	Number of jobs to be scheduled
g	Number of stages in sequence
m_t	Number of parallel machines in stage *t*
p_{jt}	Processing time for job *j* at stage *t*, $p_{jt} = p_{jmt}$ (parallel machines are identical)
s_{ijt}	Machine setup time for job *i* to job *j* at stage *t*, $s_{ijt} = s_{ijmt}$
d_j	The due date of job *j*
LR	Learning rate
a	Learning index (a $= \log_2 LR$, negative parameter)
r^{a}	Learning effect on *r*th position, $r = 1, \ldots, n_m$, (n_m is the number of jobs assigned to machines in stages)
Decision variables	
x_{ijmrt}	1, if job *i* scheduled immediately before job *j* on machine *m* in position *r* at stage *t*; otherwise= *0*

T_j	Tardiness of job *j*
C_{jt}	Completion time of job *j* at stage *t*
C_{max}	Makespan ($C_{max} = \max_{j\in\{1,\dots,n\}}\{C_{jg}\}$)
S_{ijrt}	Setup time of job *i* to job *j*, scheduled in position *r* at stage *t*

Due dates of all *n* jobs are generated based on the Equations (1)-(3). In Equation (1) total processing times of each job on all *g* stages are computed. Equation (2) computes average setup times for all possible subsequent jobs and sum it for all *g* stages. Finally, Equation (3) determines a due date for each job. Where *random* is a random number over the range (0, 1).

$$p_j = \sum_{t=1}^{g} p_{jt} \ , \quad \forall j \in n \tag{1}$$

$$s_j = \sum_{t=1}^{g} \frac{\sum_{i=1,i\neq j}^{n} s_{ijt}}{n-1} \ , \quad \forall j \in n \tag{2}$$

$$d_j = (p_j + s_j)\times(1 + random \times 3) \ , \quad \forall j \in n \tag{3}$$

The objective functions of our study are formulated as follows. The objective (4) minimizes makespan, and the objective (5) minimizes the total sum of tardiness.

$$\text{Minimize} \quad C_{max} \tag{4}$$
$$\text{Minimize} \quad \sum T_j \tag{5}$$

Constraint sets (6) and (7) ensure that each job is scheduled exactly once in one of the positions of machines available at every stage. They also ensure each job is processed once at each stage in the successor of another job.

$$\sum_{m=1}^{m_t}\sum_{r=1}^{n}\sum_{i=0,i\neq j}^{n} x_{ijrmt} = 1 \ , \quad \forall\, t,j \tag{6}$$

$$\sum_{i=0}^{n}\sum_{j=1,i\neq j}^{n} x_{ijrmt} \leq 1 \ , \quad \forall\, r,m,t \tag{7}$$

Constraint set (8) ensures that position on each machine should be filled in sequence. Constraint set (9), which is complementary to constrain (8), is a flow balance constraint that guarantees jobs are performed in well-defined sequence and ensures each job has a predecessor and a successor on the machine where the job is processed. In fact, if job *j* is processed directly after job *i* on machine *m* in position *r* at stage *t*, job *k* that is the successor of the job *j*, should be processed in position *r+1* on the machine *m* at stage *t*.

$$\sum_{i=0}^{n}\sum_{j=1,i\neq j}^{n} x_{ijrmt} \leq \sum_{i=0}^{n}\sum_{j=1,i\neq j}^{n} x_{ij(r-1)mt} \ , \quad \forall\, r \geq 2, m, t \tag{8}$$

$$\sum_{i=0}^{n} x_{ij(r-1)mt} \geq \sum_{k=1,k\neq j}^{n} x_{jkrmt} \ , \quad \forall\, j, r \geq 2, m, t \tag{9}$$

Learning effect of workers on the processing of similar setups is modeled by using the Equation (10), where s_{ijt} is the input parameter for setup time from job *i* to job *j* at stage *t*, and s_{ijrt} is the updated setup time (by considering learning effect) for job *j* if it is performed after job *i* on *r*th position of the machine in stage *t*. Learning effect on *r*th position on a machine is shown by r^{a}, where a is the learning index and can be calculated by the logarithm to the base 2 of the learning rate (LR). The lower the learning rate the higher the effects from learning.

$$S_{ijrt} = s_{ijt} \times r^{\mathrm{a}} \ , \quad \mathrm{a} = \log_2 LR \quad \forall\, i,j,t,r \tag{10}$$

Constraint set (11) forces job *j* to follow job *i* by at least *i*'s processing time plus the setup time from *i* to *j*. The value *M* is an upper bound on the completion of processing time at stage *t*.

$$C_{jt} - C_{it} + M(1 - x_{ijrmt}) \geq p_{jt} + S_{ijrt} \ , \quad \forall\; i, j, i \neq j, r, m, t \tag{11}$$

Constraint set (12) implies that the completion time of job *j* at stage *t* is larger than or equal to the sum of completion time of job *j* at stage *t-1*, the processing time of job *j* at stage *t*, and the setup time from its predecessor to job *j*.

$$C_{jt} - C_{j(t-1)} + M(1 - x_{ijrmt}) \geq p_{jt} + S_{ijrt} \quad , \quad \forall \ i, j, i \neq j, r, m, t>1 \tag{12}$$

Constraints (13) and (14) bound the objective functions and link the decision variables.

$$T_j \geq C_{ig} - d_j \quad , \quad \forall j \tag{13}$$

$$C_{\max} \geq C_{jg} \quad , \quad \forall j \tag{14}$$

Constraint sets (15) and (16) ensure that in the first position of each machine, nominal job *0* should be placed, and the rest of positions should not be filled by job *0*.

$$x_{ij1mt=0} \quad , \quad \forall \ i \geq 1, j, i \neq j, m, t \tag{15}$$

$$x_{0jrmt} = 0 \quad , \quad \forall \ r \geq 2, j, i \neq j, m, t \tag{16}$$

Constraints (17)-(19) represent the conditions on the decision variables.

$$x_{ijrmt} \in \{0,1\} \quad , \quad \forall \ i, j, i \neq j, r, m, t \tag{17}$$

$$C_{jt} \geq 0 \quad , \quad \forall \ j, t \tag{18}$$

$$T_j \geq 0 \quad , \quad \forall \ j \tag{19}$$

In this paper, the learning effect of workers on the processing of similar setups is modeled by using the Equation (10). In accordance with Biskup (2008), learning indices were generated uniformly (and rounded) between a 70% learning rate (a=-0.514) and a 90% learning rate (a=-0.152). Table 2 shows the effect of different learning rates on performing setups in different positions on a machine. As can be seen in Table 2, by considering a lower learning rate, more reduction in the setup times would be expected. Lower learning rate means higher learning effect which yields to better performance and more reduction on job's completion time. In Table 2, the learning rate of 100% is also considered which means that there would be no learning effect. In some production systems, the responsible worker for each machine is changed frequently and the learning effect of workers is negligible. From Equation (10), it can be concluded that there would be no learning effect for the first position (r=1) on machines.

Table 2. The effect of learning rate on completion of setup activities.

Learning rate: LR	Learning index: a	Learning effect for different positions on machine		
		r =2	r =3	r =4
70 %	$a = \log_2 0.70 = -0.514$	$r^a = 2^{-0.514} = 0.7$	0.56	0.49
90 %	$a = \log_2 0.90 = -0.152$	$r^a = 2^{-0.152} = 0.9$	0.84	0.81
100% *	$a = \log_2 1 = 0$	$r^a = 2^0 = 1$	1	1

* No learning

4. Non-dominated sorting and ranking water flow-like algorithms (NSWFA and NRWFA)

Two new hybrid meta-heuristic algorithms named NSWFA and NRWFA are offered here for solving the developed bi-objective optimization problem. The novelty of the developed algorithms lies in hybridizing water flow like algorithm (WFA) with non-dominated sorting and ranking concepts. The main reason for using this approach is that the problem under study is NP-hard and both non-dominated sorting concept and WFA have been demonstrated to be cost-effective for solving this type of problem. In addition, multiple and dynamic number of solution agents in WFA can be used to conduct an efficient and effective solution search in multi-objective optimization.

The proposed algorithms are somewhat similar to the NSGA-II. NSGA-II is an elitist multi-objective evolutionary algorithm which approximates the Pareto front based on the non-dominance concept (Deb et al., 2002). In NSGA-II, a ranking procedure is performed at each generation to achieve different Pareto fronts. The main stages of the NSGA-

II are the creation of an initial population, the selection of parents, the creation of children and the finding of non-dominated solutions. In the following, we extensively describe the key steps of the proposed algorithms.

Step 1: Encoding and decoding of solutions

In NSWFA and NRWFA, each solution is represented as a water flow, and the objective function surface on the solution space is modeled as the terrain traversed by the flows. The sub-flows in water flow-like algorithm are like the offspring in the genetic algorithm. We can split our randomly generated solutions iteratively into sub-flows to move toward the optimal solution. A $1 \times n$ array is applied to display a solution which n denotes the number of jobs that should be scheduled in each stage. To represent both assignment and sequence of jobs simultaneously, we use the random-key representation method to generate initial solutions. In this method, we assign a real number to each job whose integer part is the machine number to which the job is allocated and the fractional part is used to sort the jobs allocated to each machine. The fractional part of the numbers determines the sequence of jobs. For example, Figure 2 below shows a solution for an HFS model with six jobs, single stage, and three machines with the use of random-key representation method.

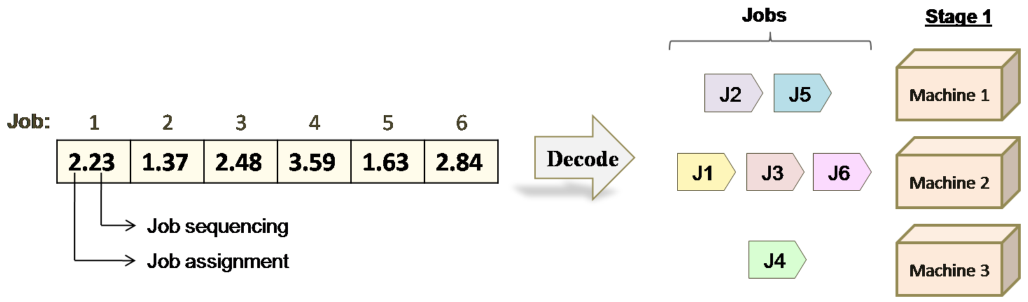


Figure 2. Representation method of solutions.

This solution presents a schedule where machine 1 has jobs 2 and 5 in that order; machine 2 has jobs 1, 3 and 6 in that order, and machine 3 has job 4. This information, when combined with the other problem data, is used to determine the objective function values for each solution. This representation method is used in the first stage, and the schedule of jobs in the subsequent stages follows the method used in Shortest Processing Time Cyclic Heuristic (SPTCH) (Kurz and Askin, 2004). In SPTCH, jobs are ordered at stage *1* in increasing order of the modified processing times $\tilde{p}_{jt}$ ($\tilde{p}_{jt}$ for job *j* in stage *t* is defined as $\tilde{p}_{jt} = p_{jt} + min_i \, s_{ijt}$). This time represents the minimum time *t* that job *j* should elapse at a stage to be completed. At the subsequent stages, jobs are assigned to a machine that allows it to complete at the earliest time as measured in a greedy fashion (Kurz and Askin, 2004).

Step 2: Initialization

In this step, we set the algorithm parameters such as the number of initial population (N), minimum number of flows (k), maximum number of sub-flows (nf), number of neighborhood searches (ns), threshold for job assignment operator (r), and threshold for precipitation and evaporation (u). Then, we generate the initial population of flows randomly

Step 3: Non-dominated sorting and ranking procedure

In this step, we use a fast non-dominated sorting approach to classifying the solutions into successive non-dominated fronts (Deb et al., 2002). We assign each solution of the same Pareto layer, a fitness value equal to its non-domination level. Obviously, the first front solutions dominate solutions of all other fronts, and they have more chance to reproduce the next generation.

To rank the solutions in each layer, the crowding distance of each solution with respect to every other solution on the same front will be computed. The crowding distance operator is used to maintain sustainable diversity in a population. The crowding distance, which is calculated front wise, is a measure of how close an individual is to its neighbors. For computing crowding-distance, we should sort the solutions according to each objective function value in ascending order. After that, for each objective function, the solutions with smallest and largest function values are assigned a big number as a distance value, and the distance value of all other intermediate solutions are determined based on the absolute normalized difference in the function values of two adjacent solutions. This quantity serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices. Larger values of the crowding distance indicate more diversity of the solutions. Figure 3 shows a schematic view of the computation procedure of the crowding distance for *i*th individual in the first frontier of solutions.

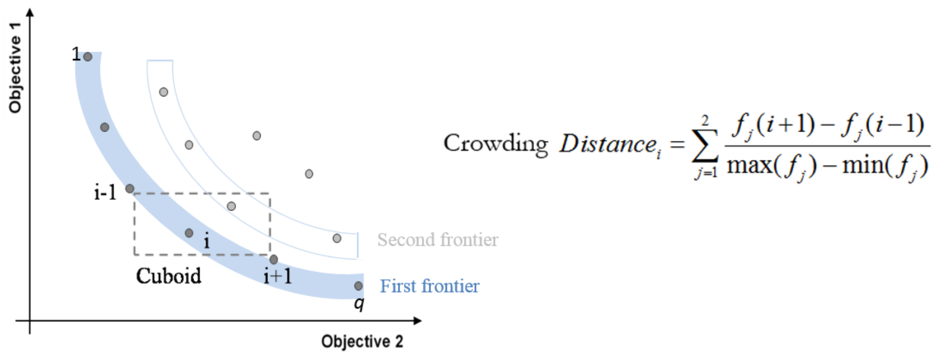

Objective 1
Objective 2
1
i-1
i
i+1
q
Cuboid
Second frontier
First frontier
$\text{Crowding } Distance_i = \sum_{j=1}^{2} \frac{f_j(i+1) - f_j(i-1)}{\max(f_j) - \min(f_j)}$

Figure 3. The calculation of crowding-distance for solutions on the same Pareto fronts.

Step 4: Selection scheme

In this step, first we count the number of solutions in the first frontier (*q*) and then select *h* solutions (*h*=max [*k*, *q*]) out of *N* solutions in the population to act as main flows. The flows in water flow-like algorithm are like the parents in genetic algorithm. The tournament selection is used for non-dominated sorting algorithm to reproduce the next generation. The roulette wheel selection is used for non-dominated ranking algorithm to reproduce the next generation. In non-dominated sorting algorithms, we assign each solution of the same Pareto layer equal fitness value (equal to its non-domination level). However, in non-dominated ranking algorithms, each solution is assigned a fitness value equal to its rank in the population. In fact, fronts are ranked based on the non-dominated rank and individuals in a front are ranked based on their crowding distance. The selection probability for non-dominated ranking algorithm is calculated as given in Equation (20).

$$p_i = \frac{2 \times Rank}{N \times (N+1)} \tag{20}$$

It is good to note that we combine the population of flows and sub flows in each iteration of the algorithm. Then, we perform non-dominated sorting and ranking to create the next population of size *N*.

Step 5. Generating sub-flows

An advantage of the NSWFA and NRWFA is that the number of sub-flows forked from a flow at each iteration is not constant. Three operators are used for generating the dynamic population of sub-flows, which are: flow splitting and moving operation, water evaporation and precipitation operation, and flow-merging operation. The mission of flow splitting and moving operations is to search for better neighborhood solutions and ultimately select the best solution for the current flow. Water evaporation and precipitation operation is used to avoid being trapped and explore more solution spaces. The flow-merging operation is used to reduce the number of solution agents. These operators are designed based on the properties of water flow such as movement from higher to lower areas and spreading on the ground.

The locations of the split sub-flows (offsprings) are derived from the neighboring locations of the original flow. The number of sub-flows for each flow is randomly generated in this interval: [*N/2k* , *nf*], where *nf* is a parameter called a maximum number of sub-flows. Two steps have been used for splitting and moving flows, namely, job assignment and job sequencing.

In the first step (job assignment), we find a rough direction for moving flow to the neighborhoods of the current solution by reassigning a job to any machine other than the current one based on a predefined probability *r*, which is set to a comparatively low value (e.g. 0.3). For example, for each job in the current solution, a random number over the range (0, 1) is first selected. If the value is greater than *r*, then the job is assigned to another machine randomly; otherwise, it processed on the current machine.

In the second step (job sequencing), we find the best sequence of jobs allocated to each machine, by introducing a parameter called the number of neighborhood search (ns). The neighborhood of a solution is a set of feasible solutions which are reachable by a single move of jobs on each machine. For example, for the neighborhood search with the value of five, we should change the sequence of two jobs assigned to that machine five times, and choose the best sequence (out of five sequences) that maximizes the sum of improvement in both of objective functions.

Since job sequencing uses the result of job assignment, the job assignment plays an important role in the goodness of the final solution. In Figure 4, we demonstrate the splitting and moving operations proposed for searching neighborhood solutions on the HFS problem. It is assumed that the minimum number of flows is equal to six, and we are going to generate two sub-flows per each flow.

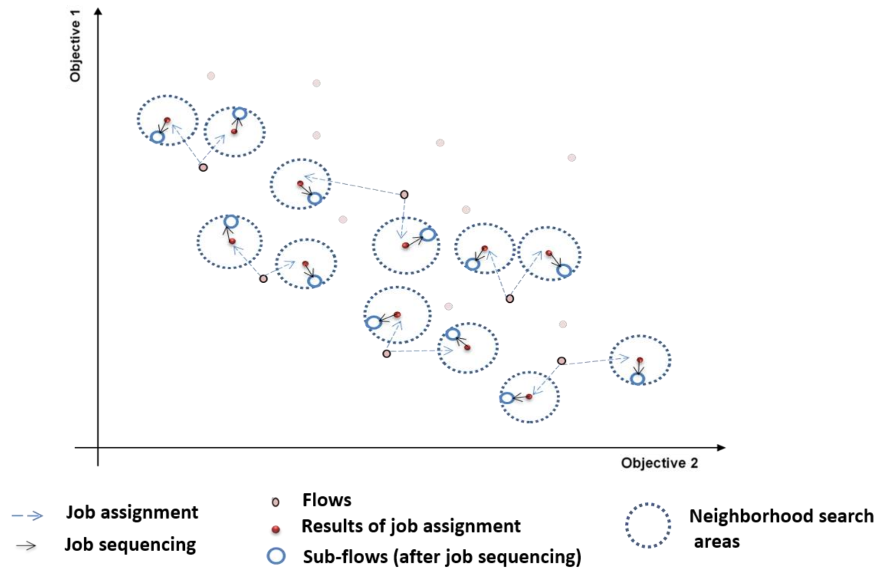

Objective 1
Objective 2
Job assignment
Job sequencing
Flows
Results of job assignment
Sub-flows (after job sequencing)
Neighborhood search areas

Figure 4. Searching neighborhood solutions with flow splitting and moving operation.

In this study, water evaporation and precipitation operators are used to avoid being trapped when there is no improvement in the flow splitting and moving. We propose a probabilistic mechanism to simulate the behavior of water in evaporation and precipitation by defining a predefined probability u, which is set to a comparatively high value (e.g. 0.7). Each time we generate a random number from (0, 1) and we exchange the stopped flow with another one if the generated number is greater than the defined threshold. The flow-merging operator is used to avoiding redundant searches. It eliminates the redundant flows when multiple agents result in the same objective value.

Step 6. Stopping rule

The stopping criterion is set to a CPU time limit fixed to $ST_{max} = \left(n^2 \times \sum_{t=1}^{g} m_t \,/ g\right) \times 3$ milliseconds. This stopping rule is responsive to the number of jobs, stages, and the number of machines at each stage. If the stopping criterion is met then stop; otherwise, go to Step 3.

Figure 5 presents the flowchart of the proposed algorithms for solving the bi-objective HFS problem. The algorithm step of NRWFA is the same as NSWFA, and the only difference is in fitness assignment procedure (see Step 4).

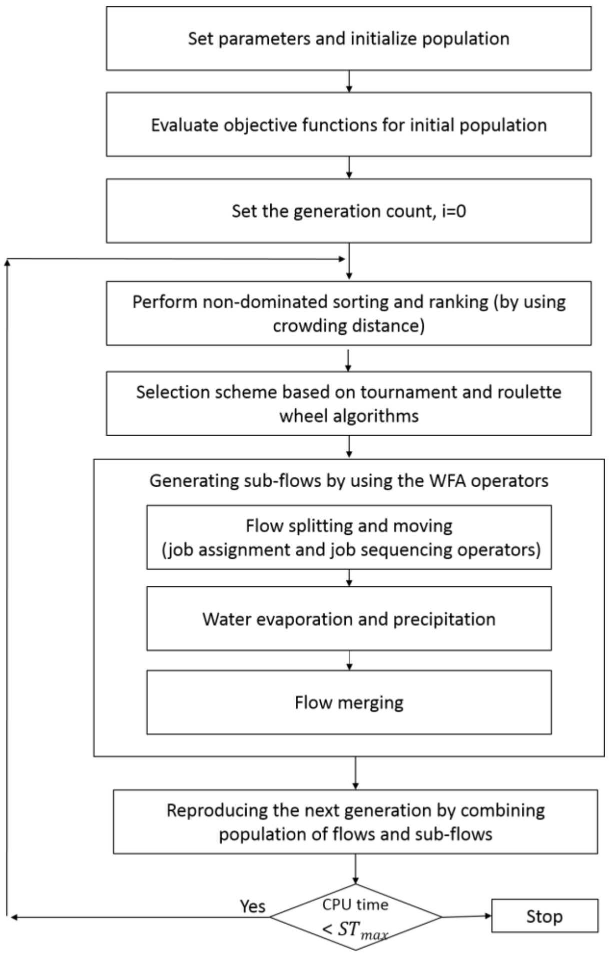

Set parameters and initialize population
Evaluate objective functions for initial population
Set the generation count, i=0
Perform non-dominated sorting and ranking (by using crowding distance)
Selection scheme based on tournament and roulette wheel algorithms
Generating sub-flows by using the WFA operators
Flow splitting and moving (job assignment and job sequencing operators)
Water evaporation and precipitation
Flow merging
Reproducing the next generation by combining population of flows and sub-flows
Yes
CPU time $< ST_{max}$
Stop

Figure 5. Flowchart for the NSWFA and NRWFA algorithms.

Although the proposed hybrid algorithms follow some of the elementary structure of NSGA-II (e.g., non-dominated sorting, and crowding distance), they benefit from using the operators of the WFA algorithm which are adapted for the bi-objective HFS problem. The main advantages of the proposed hybrid algorithms are: (1) multiple and non-fixed number of flows dependent on the first Pareto layer, (2) dynamic size of sub-flows by considering the possibility of generating new flows, and (3) efficient multi-direction search along with the objective space.

5. Computational results

In this section, the impact of learning on the schedule of jobs and the complexity of the developed bi-objective model are analyzed. The efficiency of the proposed algorithms is also evaluated by comparing their results against those obtained from NSGA-II and NRGA, which are more commonly used in solving similar problems. This section also contains the method of generating test problems, performance metrics, the parameter settings of the algorithms, and explains the experimental results for comparing the efficiency of the algorithms. Extensive experimentations have been conducted to assess the effectiveness of different algorithms proposed in Section 4. All the algorithms are implemented in Java and ran on a PC with a 2.50 GHz Intel Core 2 Duo processor and 4.00 GB of RAM memory.

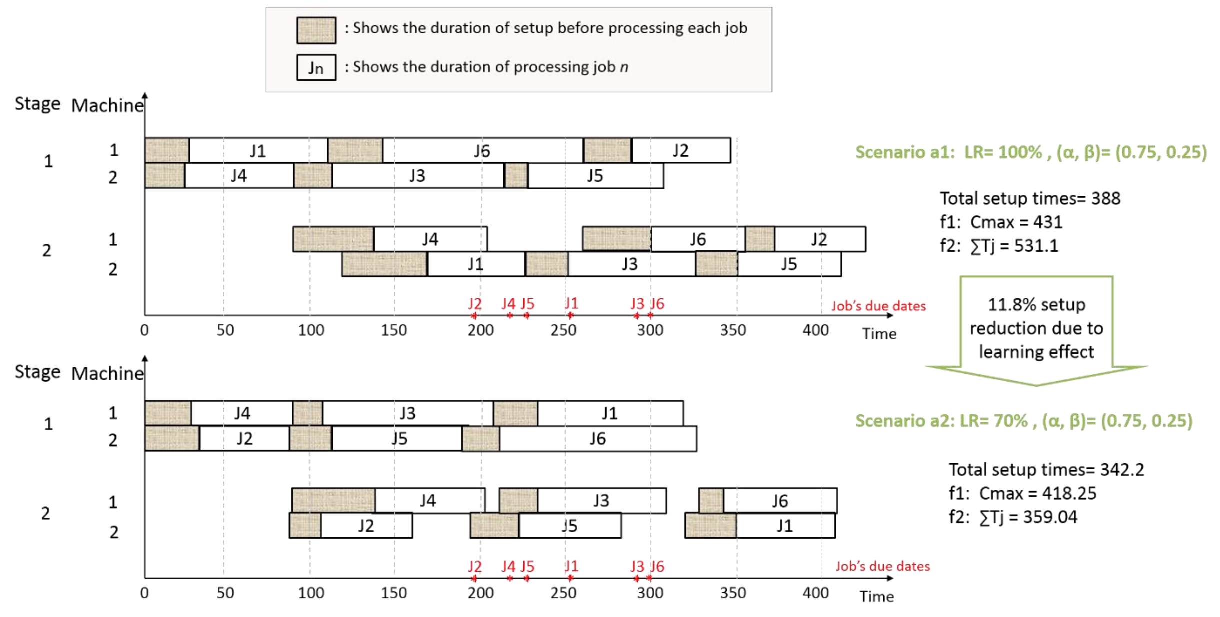
5.1. Impact of learning on scheduling jobs

Several experiments were conducted to show the significance of the learning effect on the schedule of jobs in hybrid flow shop systems. An example including six jobs and two stages with two machines in each is used as the basis for illustration. The data set relevant to this example is shown below in Table 3. Processing times of jobs in stage 1 and 2 are shown by p_{j1} and p_{j2}, respectively. Due date of each jobs is shown by d_j.

Table 3. Sample problem data for HFS scheduling problem.

Job	1	2	3	4	5	6
p_{j1}	90	54	99	59	69	118
p_{j2}	61	55	75	64	60	65
d_j	254	192	286	218	224	296
Sequence-dependent setup times between jobs on stage 1						
From/to	1	2	3	4	5	6
0	26	35	55	31	55	24
1	33	32	37	40	48	30
2	61	44	61	57	47	56
3	32	51	61	55	21	24
4	43	49	32	61	39	47
5	25	50	57	58	45	47
6	41	28	43	36	61	32
Sequence-dependent setup times between jobs on stage 2						
From/to	1	2	3	4	5	6
0	50	20	63	49	56	20
1	30	42	30	46	57	39
2	30	46	22	48	41	47
3	56	43	43	43	24	31
4	39	45	34	47	54	27
5	56	42	60	36	29	58
6	47	20	48	40	45	55

Here, four scenarios are evaluated to show the significance of the learning effect and the coefficient of optimization criteria on the assignment of jobs to machines and their sequence. The weights for makespan and total tardiness are respectively shown by α and β. The above example is solved with IBM ILOG CPLEX Optimization Studio 12.6. We used the bi-criteria version of the proposed mathematical model in Section 3 while considering two different learning rates (LR=70% and LR=100%) and two different weight vectors for the optimization criteria (α=0.75, β=0.25, and α=0.25, β=0.75). Figure 6 below shows the optimum schedule of the proposed four scenarios.



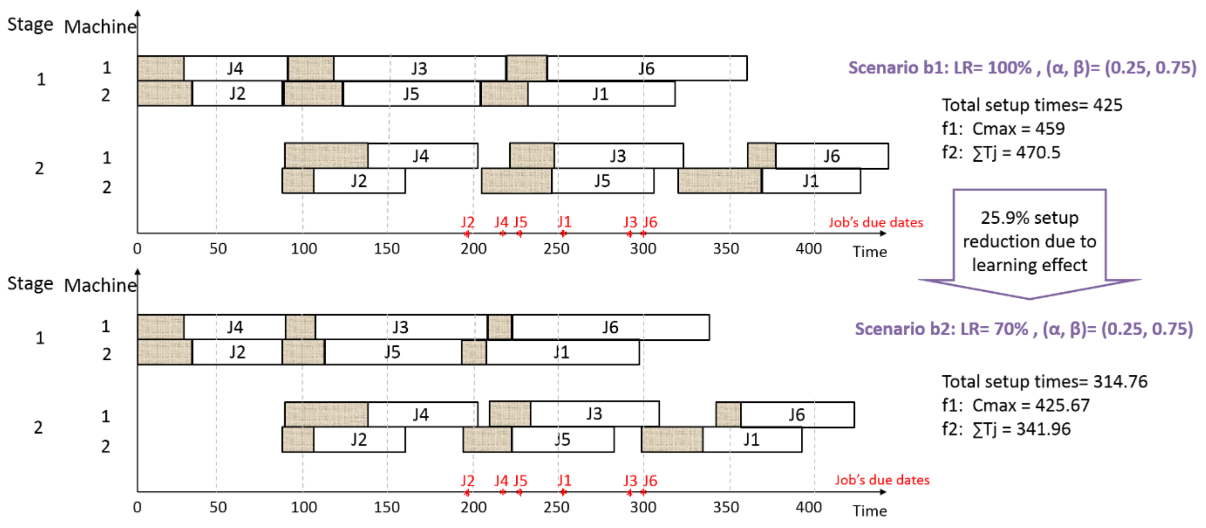


Figure 6. The impact of learning effect and objective coefficients on the (optimum) schedule of jobs for an individual problem.

Figure 6 aims to visually evaluate the impact of learning effect and the objective coefficients on the (optimum) schedule of jobs. As can be seen in Figure 6 (a1 and a2), the weight vectors are identical and considering the lower learning rate leads to reducing the makespan, total tardiness and total setup time. Figure 6 shows that learning effect not only reduces the optimization criteria but also changes the optimal schedule of jobs. In Figure 6 (b1 and b2), the weight vector is changed to (0.25, 0.75) and by lowering the learning rates, the optimum schedule remains the same and the optimization criteria are reduced as a result of a reduction in setup times. It is good to note that all the data parameters such as processing time of jobs and their due dates remained the same in all the scenarios. The above example illustrates that the learning effect performs well in reducing setup times through proper scheduling. Further analyses on the structure of optimum job scheduling show that the makespan criterion has more dependency on the learning effect (see Table 5).

5.2. Complexity of the developed bi-objective model

A comprehensive numerical example is presented in this section to verify the behavior of the bi-objective proposed model and evaluate its complexity. First, we combine the two objectives into a single objective (bi-criteria) by using the weighted sum method. The weighted sum approach takes its basic premise from conventional multi-objective optimizations (Tabucanon, 1988). Then, we analyze the complexity of the bi-objective HFS problem (minimize: C_{max} and $\sum T_j$) against the bi-criteria one (minimize:$\alpha C_{max} + \beta \sum T_j$) through numerical examples.

Here, four small problems are considered to verify the behavior of the bi-criteria model on changing the learning rate and objective coefficient. The processing time of jobs is from a uniform distribution between 40 and 120 and integer setup times selected from a uniform distribution between 20 and 64. The parameters of small-sized problems and additional problem characteristics are given in Table 4. The second problem of Table 4 is the same example which was discussed in Section 5.1.

Table 4. Characteristics of four small problems.

Problem	Number (#) of jobs	# of stages	# of machines	# of variables	# of constraints
1	6	2	1	331	575
2	6	2	2	643	1125
3	10	2	1	1671	2719
4	10	2	2	3311	5397

The developed mixed integer programming model is used to find the optimal solutions of the above problems with IBM ILOG CPLEX Optimization Studio 12.6. Since the process of finding an optimal solution can take a long time, each problem was allowed a maximum of 7200 seconds (two hours) runtime. We used three different weight factors (for α and β) and three different learning rates for the problems in Table 4. The best-obtained makespan and total tardiness within the two-hour time limit are given in Table 5.

Table 5. Problem solutions based on different learning rates and weights. The solutions marked with an asterisk indicate obtaining a solution with a distinct schedule of jobs in comparison with solving the same problem with other learning rates.

(α,β)	Problem	LR	Makespan	Total tardiness	Cplex results
(0.25,0.75)	1	100 %	753	1463.7	Optimal
		90 %	751.79	1338.87	Optimal
		70 %	691.63	1130.74	Optimal
	2	100 %	459	470.5	Optimal
		90 %	447.34	426.28	Optimal
		70 %	425.67	341.96	Optimal
	3	100 %	1426	5744.48	Gap [a]: 99.07%
		90 %	1294.86	4988.74	Gap: 99.02%*
		70 %	1154.91	4486.48	Gap: 99.06%*
	4	100 %	720	2316.42	Gap: 96.70%*
		90 %	754.14	2217.61	Gap: 97.97%
		70 %	692.55	1930.92	Gap: 97.53%
(0.75,0.25)	1	100 %	753	1463.7	Optimal
		90 %	722.07	1367.91	Optimal*
		70 %	691.63	1130.74	Optimal*
	2	100 %	431	531.1	Optimal
		90 %	421.26	497.48	Optimal*
		70 %	418.25	359.04	Optimal*
	3	100 %	1304	5534.5	Gap: 94.70%*
		90 %	1317.74	5257.86	Gap: 94.21%
		70 %	1135.91	4389.48	Gap: 94.36%
	4	100 %	747	2340.88	Gap: 83.43%
		90 %	716.89	2170.96	Gap: 89.99%
		70 %	654.46	1695.87	Gap: 82.27%*

[a] Gap = |bestbound-bestinteger|/|bestinteger|.

In general, optimization methods for scheduling problems are only applicable to relatively small problems (Naderi et al., 2009). The problem studied in this paper is NP-hard, and it is not time efficient to solve this type of problems using a MIP-solver. Due to complexity of the developed MIP model, increasing the problem size would result in higher gaps, highlighting the need to use heuristics or meta-heuristics as solution methods. As can be seen in Table 5, only problem 1 and 2 were solved optimally in the two-hour time limit. They have been solved on average after 10 and 13 seconds, respectively. The other problems were stopped due to the time limit before finding the optimal solution. As discussed in Section 5.1, lower learning rate yields to more reduction in the optimization criteria. This impact of learning rates on the objective criteria is summarized in Figure 7.

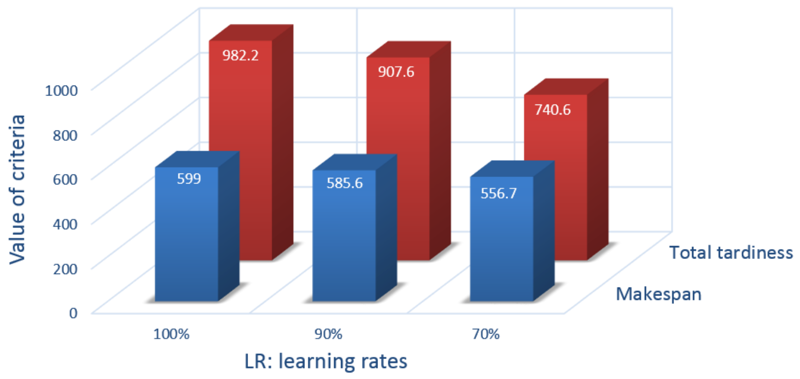
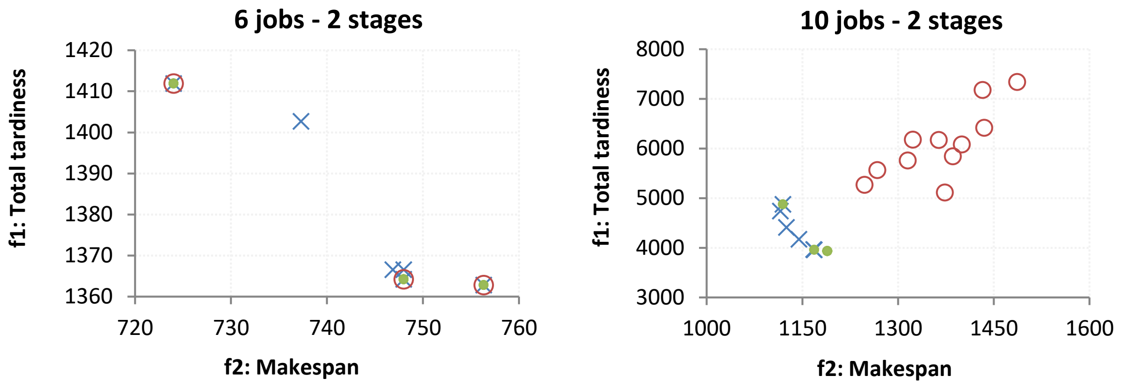


Figure 7. Mean plot of optimization criteria for different learning rates.

Here, the complexity of the bi-objective model against the bi-criteria is shown by solving test problems 1 and 3 from Table 4 by using exact and meta-heuristic algorithms. NSWFA and WFA are the two meta-heuristic algorithms for solving the bi-objective and bi-criteria versions of the problem, respectively. Branch and bound as an exact method is used for solving the bi-criteria problem. Eleven weights vectors (α, β) are used to combine the two objectives into a single objective, which are: (0, 1), (0.1, 0.9), (0.2, 0.8), (0.3, 0.7), (0.4, 0.6), (0.5, 0.5), (0.6, 0.4), (0.7, 0.3), (0.2, 0.8), (0.9, 0.1), (1, 0). These weights present the preference information of a decision-maker to each criterion. The solutions of solving both the bi-objective model and the bi-criteria model are shown in Figure 8. Due to the complexity of the proposed model, the problems were run up to two hours for the exact method and up to five minutes for the metaheuristics.



(○) shows the solutions of the bi-criteria problem with the exact method, (●) shows the solutions of the bi-criteria problem with metaheuristic algorithm (WFA algorithm), and (×) shows the solutions of the bi-objective problem with metaheuristic algorithm (NSWFA hybrid algorithm). For the problem with ten jobs and two stages, solutions of the bi-criteria problem with the exact method are not optimal (solution procedure was stopped due to the two-hour time limit).

Figure 8. The complexity of the bi-objective model and the inability of weighted sum method on finding the Pareto solutions.

As can be seen in Figure 8, NSWFA performs best in approximating the Pareto-optimal front. The number of solutions and the diversity of solutions found by solving the bi-objective problem with NSWFA is higher than those of solving the bi-criteria problems. WFA works better than the exact method for solving the bi-criteria problem. However, by considering the proposed (eleven) weight vectors for solving the bi-criteria problem, just a few solutions are found. The results of analysis also show that the proposed bi-objective model is a computationally intensive problem and multi-objective meta-heuristic methods should be used even for small-sized problems to find good Pareto solutions.

5.3. Test problems

Data required for generating test problems consist of the number of jobs (n), the number of stages (g), the number of machines per stage (m_t), the range of processing times of jobs in stages (p_{jt}) and the range of sequence dependent setup times of jobs in stages (s_{ijt}). Due dates need not to be generated and should be calculated based on Equations (17)-(19).

We categorize our test problems into large, medium, and small problems. These categories are defined based on the number of jobs (n) and stages (g). The number of machines in each stage can be generated uniformly between interval [1, 5] or [2, 8] based on the size of the problem. The processing times and setup times are generated by the uniform distribution over the range [40,120] and [20, 64], respectively. We defined 16 problem sets comprising combinations of n and g and generated two instances for each problem set. Therefore, 32 problem instances were generated and each instance is tackled five times to yield more reliable information. We have used the same stopping rule which is explained in Section 4 for all the algorithms. Stopping criterion for all the algorithms is a CPU time limit which is dependent on the size of the test problem. When the stopping criterion is met then the algorithm will stop and we can register the values of both objective functions for the obtained non-dominated solutions.

5.4. Performance metrics

The concept of performance metrics is used for comparison amongst algorithms. Since no single metric can entirely capture the performance of multi-objective meta-heuristics, we use the following five performance criteria to measure the quality of the approximated sets of Pareto solutions in a quantitative way:

- The number of Pareto solution (*NPS*): The more the number of non-dominated solutions, the more alternatives for decision makers to decide through.

- The spread of non-dominance solution (*SNS*): First, we should calculate mean ideal distance of each non-dominated solution (where k is the number of non-dominated sets) by ideal point (0, 0) and then SNS expressed by the following equation:

$$c_i = \sqrt{f_{1i}^2 + f_{2i}^2} \quad , \quad MID = \frac{\sum_{i=1}^{k} c_i}{k} \tag{21}$$

$$SNS = \sqrt{\frac{\sum_{i=1}^{k} (MID - c_i)^2}{k-1}} \tag{22}$$

  The higher the value of *SNS*, the better solution quality we have (more diversity in the obtained solution).
- Diversification metric (*D*): The diversification metric measures the spread of the solution set. Its definition is the following:

$$D = \sqrt{\sum_{i=1}^{k} \max\left(\left\| x_i' - y_i' \right\|\right)} \tag{23}$$

  Where $\left\| x_i' - y_i' \right\|$ is the Euclidean distance between the non-dominated solution x_i and the non-dominated solution y_i.
- The rate of achievement to two objectives simultaneously (*RAS*): The balance in reaching to objective functions.

$$RAS = \frac{\sum_{i=1}^{k} \left(\frac{f_{1i} - F_i}{F_i}\right) + \left(\frac{f_{2i} - F_i}{F_i}\right)}{k} \tag{24}$$

  Where F_i = min { f_{1i} , f_{2i} }. The lower value of *RAS*, the better of solution quality we have.
- Set coverage (*SC*): set coverage compares the percentage of the final solutions found by one algorithm covered by the solutions found by another algorithm (domination of two populations in a pair-wise manner). *SC* is defined as the mapping of the ordered pair (X', X'') to the interval [0, 1], as follows:

$$SC(X', X'') = \frac{\left| a'' \in X''; \forall a' \in X' : a' \geq a'' \right|}{\left| X'' \right|} \tag{25}$$

  Where x', x'' are two sets of decision vectors. If all points in x' dominate or are equal to all points in x'', then by definition $SC = 1$. In general, both $SC(X', X'')$ and $SC(X'', X')$ have to be considered due to set intersections not being empty.

5.5. Experimental results

In this section, we compare the proposed NSWFA and NRWFA with NSGA-II and NRGA in different test problems. In general, the performance of evolutionary algorithms is sensitive to its parameters. Therefore, we conduct a series of experiments to determine the optimal parameter combinations before conducting actual runs to collect the results. The crossover probability (*Pc*) is selected between 0.75 and 0.90, in steps of 0.01 and for each *Pc* performance is analyzed. It is found that *Pc* = 0.79, produces the best results. Other parameters such as mutation probability (*Pm*) is selected as *1/n* (where n is the number of variables), as recommended by Deb et al. 2002. The parameters for the algorithms are given in Table 6.

Table 6. Parameters for multi-objective evolutionary algorithms.

Parameters	Parameter values
Population size, *N*	100
Minimum number of flows, *k*	5
Maximum number of sub-flows, *nf*	15
Number of neighborhood searches, *ns*	10
Threshold for job assignment operator, *r*	0.25
Threshold for precipitation and evaporation, *u*	0.75
Crossover probability, *Pc*	0.79
Mutation probability, *Pm*	1/n (where n is the number of variables)

The performance of the algorithms was evaluated by comparing the quality of non-dominated solutions obtained by each algorithm. Table 7 represents a comparison of performances of the algorithms with respect to NPS, SNS, D, and RAS metrics. As it can be seen in the average row, the NRWFA is superior to the other algorithms with regard to *NPS*, *SNS*, and *D*. However, NSWFA provides a better result than others with regard to *RAS* index.

Table 7. Evaluation of non-dominated solutions of the algorithms with different performance metrics.

		NPS				SNS				D				RAS			
Problem sets	*n*×*g*	NSGA-II	NRGA	NSWFA	NRWFA	NSGA-II	NRGA	NSWFA	NRWFA	NSGA-II	NRGA	NSWFA	NRWFA	NSGA-II	NRGA	NSWFA	NRWFA
Small	6×4	3	3.3	2	2	18.44	14.86	32.59	33.20	98.60	85.27	125.01	125.31	0.484	0.461	0.408	0.409
	6×8	3.3	3.3	1	2	36.19	37.83	0	69.43	121.36	132.45	0	129.50	0.176	0.167	0.121	0.1632
	10×4	2.3	5.6	2.3	3	80.23	59.97	162.99	97.37	143.47	251.11	317.09	242.35	1.449	1.371	1.239	1.299
	10×8	3.3	4	4	2.3	186.34	98.21	138.54	143.23	428.38	281.08	450.753	339.66	0.752	0.688	0.718	0.685
Medium	20×2	8.3	6.6	6	5.6	220.43	131.73	110.00	295.68	674.69	386.36	313.01	631.70	5.66	5.705	4.715	5.151
	20×4	8.3	5	6	3.6	181.54	242.38	151.46	205.51	604.36	552.49	453.14	500.42	5.773	5.760	5.373	4.9264
	20×8	5.3	4.6	3.3	3	92.24	80.41	171.32	615.33	255.21	210.59	295.80	907.95	4.314	4.323	4.029	4.2
	40×2	4.6	4.6	6	6.3	100.99	137.77	234.08	217.60	284.93	375.68	650.66	526.84	10.588	10.725	9.954	9.9806
	40×4	5.6	4	5.3	10	354.94	271.27	832.03	848.52	902.44	572.20	1744.25	2266.68	14.64	14.491	13.982	14.253
	40×8	4	4.6	6.3	6.6	447.50	375.22	618.42	771.77	1208.85	890.54	1543.06	1909.47	13.09	13.109	12.615	12.773
Large	80×2	4.6	3.6	4.3	5.6	1475.95	765.18	3279.8	2999.5	3378.30	1290.19	7851.77	7756.42	35.38	35.197	34.319	34.321
	80×4	2.6	3	5.6	6.6	1209.26	1478.18	2102.38	1911.77	2340.58	2157.85	5562.11	5225.99	33.300	33.310	31.450	31.969
	80×8	3	2	7.3	4.3	1000.04	256.05	1839.76	2767.33	1947.55	471.74	5259.40	4554.88	29.785	29.793	28.893	29.344
	100×2	6.6	5	7.3	9	996.10	858.73	3012.27	2865.13	2729.48	1999.42	8154.62	7133.51	43.100	43.241	40.980	41.315
	100×4	4	3.3	6	5.6	952.65	831.39	1823.22	3180.66	2316.50	1928.35	4539.66	7713.02	41.544	41.471	38.979	40.022
	100×8	6	4.3	8	11	2465.25	1395.65	2826.70	3815.49	6092.78	3278.61	7869.88	11844.0	41.784	41.762	40.234	40.847
Average		**4.67**	**4.17**	**5.04**	**5.40**	**613.63**	**439.67**	**1083.48**	**1302.3**	**1470.46**	**928.99**	**2820.64**	**3237.98**	**17.615**	**17.598**	**16.751**	**16.978**

All the criteria used in Table 7 have some disadvantages. For example, NSWFA only obtained one non-dominated solution in solving the second problem of Table 7. This single non-dominated solution of NSWFA dominate all the non-dominated solutions of the other algorithms. On the other hand, *SNS* and *D* would be equal to zero for NSWFA and this decreases the average diversity of NSWFA algorithm. It can be concluded that the performance criteria shown in Table 7, cannot be solely used to compare the efficiency of the algorithms. For further analysis of the results, the means plot and least significant difference (LSD) intervals of the algorithms for different performance criteria are shown in Figure 9. We set a 95% confidence interval for the Tukey tests. Obviously, the less the limits obtained for an algorithm is overlapping with that of other algorithms, the more statistically significant difference between their performances are deemed to be. Although solution obtained by NSWFA are superior to those by other algorithms, the benchmark results in Figure 9 do not reveal a significant difference (at least) between the NSWFA and NRWFA.

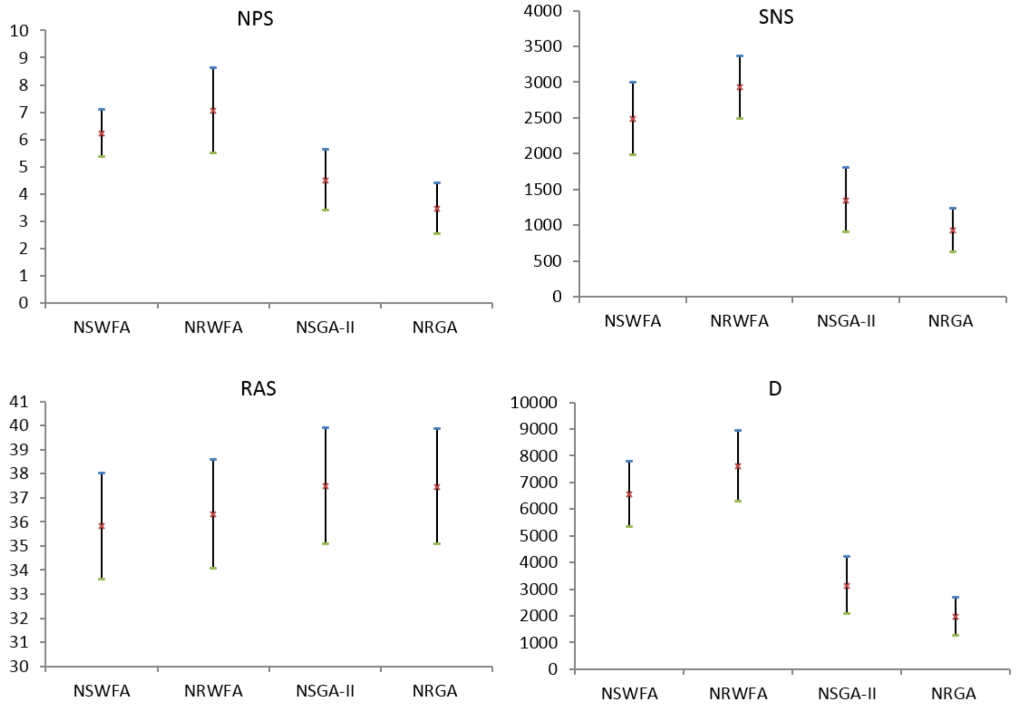


Figure 9. Diagram of means and LSD intervals (at the 95% confident level) for NPS, SNS, RAS, and D of different algorithms (relevant to large-sized problems).

Among the discussed performance criteria, set coverage has a better illustration of how two algorithms dominate each other, and it produces a comprehensive comparison index. In performance evaluation of bi-objective meta-heuristic algorithms, comparing measures such as *SNS*, *D*, and *RAS* is useful when the non-dominated solutions of algorithms have almost equal set coverage values.

To obtain reliable results in the calculation of the set coverage index, first, non-dominated solutions of each algorithm should be updated after five replications and then compared together. As can be seen in Table 8, the results of set coverage criteria demonstrate the high performance of NSWFA, NRWFA against NSGA-II and NRGA. The results show that the Pareto solutions obtained by NSGA-II and NRGA cannot dominate those by NSWFA and NRWFA.

Table 8. Average value of the set coverage criteria for paired comparison of the algorithms {SC(X',X")}.

X'	X''			
	NSGA-II	NRGA	NSWFA	NRWFA
NSGA-II	-	0.37	0.04	0.04
NRGA	0.83	-	0.1	0.05
NSWFA	1	1	-	0.76
NRWFA	1	1	0.36	-

To visually evaluate the quality of solutions on the Pareto-optimal front, Figure 10 represents the non-dominated solutions of the proposed algorithms after five replications for solving small, medium and large problems.

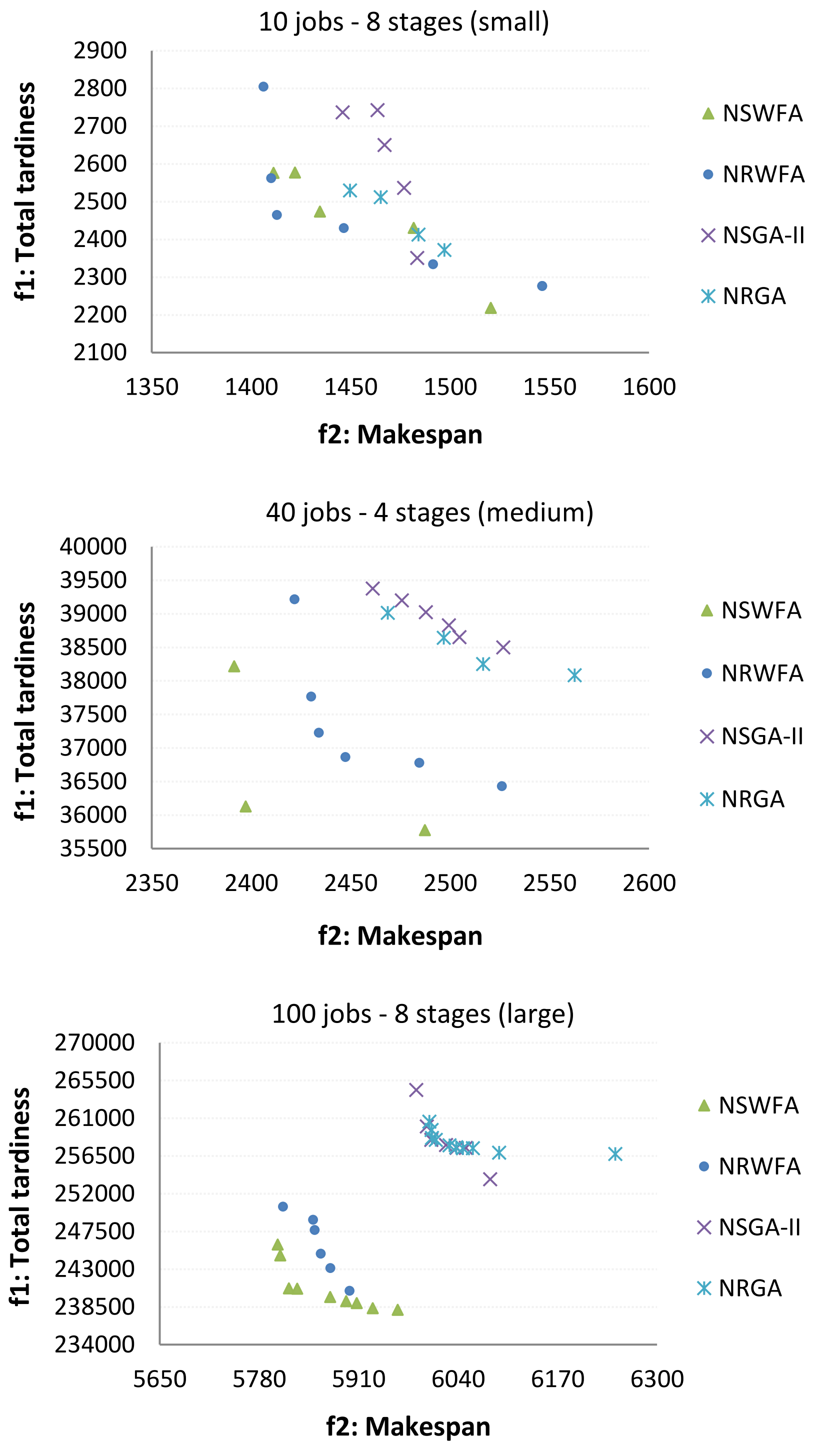


Figure 10. Non-dominated solutions of different algorithms for small, medium and large problems.

As can be seen in Figure 10, NSWFA performs best by increasing the problem size. The results shown in Figure 10 confirm the results of *SC* index. The computational results show that the proposed NSWFA works effectively for scheduling jobs in bi-objective HFS scheduling problem with learning effect.

6. Conclusion

In this paper, we have studied a hybrid flow shop scheduling problem with a learning effect of workers performing sequence-dependent setup times. Our objective was to determine a schedule that minimizes both total tardiness and makespan simultaneously. A bi-objective mixed integer linear programming formulation was presented to formulate the problem. Several experiments were conducted to show the significance of the learning effect on minimizing the optimization criteria and its impact on the structure of assigned jobs to machines and their sequences. It has been shown that the learning effect not only reduces the setup times but also results in schedules in which jobs that share similar tools and fixtures are placed adjacent to each other.

In order to solve this bi-objective problem, two hybrid meta-heuristic algorithms, NSWFA and NRWFA, were developed. These hybrid algorithms combine elements such as non-dominated sorting, and crowding distance from NSGA-II with operators of the WFA. The main advantages of these algorithms are in multiple and dynamic number of solution agents with efficient multi-direction search along with the objective space. The performance of these algorithms is compared with two commonly used multi-objective genetic algorithms using sixteen sets of test problems. We used five different performance criteria, namely the number of Pareto solutions, the spread of non-dominance solutions, the rate of achievement to the objectives, diversification metric and set coverage. The results illustrate that the set coverage is the most comprehensive performance metric for mutual comparison of algorithms in bi-objective optimization. The computational analysis demonstrates the superiority of NSWFA to approximate the efficient set of solutions.

The complexity of the developed bi-objective model with learning effect was also analyzed. We combined the two objectives into a single objective by using the weighted sum method and compared its performance with the developed bi-objective model. The results show that several runs of solving the bi-criteria problem with different weight vectors could not cover the solutions obtained from a single run of the proposed hybrid algorithms.

As a future research direction, it is interesting to work on the problem considering other representation methods of learning effects such as time-based and experience-based learning. An analysis of the impact of the objective coefficients on the complexity of the problem could also be interesting. Another direction for future research is to extend the assumptions of the proposed model and consider other objectives or even presenting other performance metrics. Moreover, an extension of the proposed hybrid meta-heuristic algorithms for other fields of scheduling such as open shop and job shop could be effective.

References

Agnetis, A., Billaut, J.-C., Gawiejnowicz, S., Pacciarelli, D. & Soukhal, A. (2014). Multiagent scheduling. Berlin Heidelberg: Springer Berlin Heidelberg. Doi, 10(1007), 978–3.

Behjat, S. and Salmasi, N. (2017). Total completion time minimisation of no-wait flowshop group scheduling problem with sequence dependent setup times. European Journal of Industrial Engineering, 11(1), 22-48.

Behnamian, J. and Zandieh, M. (2013). Earliness and tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic. Arabian Journal for Science and Engineering, 1-14.

Biskup, D. (1999). Single machine scheduling with learning considerations, European Journal of Operational Research, 115(1), 173–178.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effect, European Journal of Operational Research, 188, 315–329.

Cheng, T.C.E., Kuo, W.H. and Yang, D.L. (2014). Scheduling with a position-weighted learning effect. Optimization Letters, 8(1), 293-306.

Cheng, T.C.E. and Wang, G. (2000). Single machine scheduling with learning effect considerations, Annals of Operations Research, 98(1-4), 273–290.

Choong, F., Phon-Amnuaisuk, S., and Alias, M.Y. (2011). Meta-heuristic methods in hybrid flow shop scheduling problem, Expert Systems with Applications, 38 (9), 10787-10793.

Coello, C.A.C., Lamont, G.B. and Van Veldhuizen, D.A. (2007). Evolutionary Algorithms for Solving Multi-Objective Problems, Springer.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 6(2), 182-197.

Eren, T. and Güner, E. (2008). A bi-criteria flow shop scheduling with a learning effect, Applied Mathematical Modelling, 32, 1719–1733.

Eren. T. and Güner, E. (2009). A bicriteria parallel machine scheduling with a learning effect, International Journal of Advanced Manufacturing Technology, 40, 1202-1205.

Govindan, K., Balasundaram, R., Baskar, N. and Asokan, P. (2016). A hybrid approach for minimizing makespan in permutation flowshop scheduling. Journal of Systems Science and Systems Engineering, 1-27.

Gupta, J. (1988). Two-stage hybrid flow shop scheduling problem, Journal of Operational Research Society, 39 (4), 359–364.

Jadaan, O., Rajamani, L. and Rao, C. (2008). Non-dominated ranked genetic algorithm for solving multi-objective optimization problems: NRGA, Journal of Theoretical and Applied Information Technology, 4(1), 61-68.

Jones, D.F., Mirrazavi, S.K. and Tamiz, M. (2002). Multi-objective meta-heuristics: an overview of the current state of the art, European journal of operation research, 137(1), 1-9.

Karimi, N. and Davoudpour, H. (2016). Multi-objective colonial competitive algorithm for hybrid flowshop problem. Applied Soft Computing, 49, 725-733.

Kuo, W.-H., Yang, D.-L. (2006). Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect, European Journal of Operational Research, 174, 1184–1190.

Kurz, M.E., and Askin, R.G. (2003). Comparing scheduling rules for flexible flow lines, International Journal of Production Economics, 85 (3), 371–388.

Kurz, M.E., and Askin, R.G. (2004). Scheduling flexible flow lines with sequence dependent setup times, European Journal of Operational Research, 159 (1), 66–82.

Mousavi, S.M., Mahdavi, I., Rezaeian, J. and Zandieh, M. (2016). An efficient bi-objective algorithm to solve re-entrant hybrid flow shop scheduling with learning effect and setup times. Operational Research, 1-36.

Mousavi, S.M. and Zandieh, M. (2016). An Efficient Hybrid Algorithm for a Bi-objectives Hybrid Flow Shop Scheduling. Intelligent Automation & Soft Computing, 1-8.

Murata, T., Ishibuchi, H. and Tanaka, H. (1996). Multi-objective genetic algorithm and its application to flowshop scheduling, Computer and Industrial Engineering, 30, 957–968.

Naderi, B., Zandieh, M. and Roshanaei, V. (2009). Scheduling hybrid flow shops with sequence dependent setup times to minimize makespan and maximum tardiness, International Journal of Advanced Manufacturing Technology, 41 (11–12), 1186–1198.

Pargar, F. and Zandieh, M. (2012). A bi-criteria SDST hybrid flow shop scheduling with learning effect of setup times: Water flow-like algorithm approach, International Journal of Production Research, 50(10), 2609-2623.

Pinedo, M., 2015. Scheduling; Theory, Algorithms, and Systems. Springer.

Quadt, D. and Kuhn, H. (2007). A taxonomy of flexible flow line scheduling procedures. European Journal of Operational Research, 178(3), 686-698.

Ribas, I., R. Leistein, and J.M. Framiñan. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solution procedure perspective, Computers and Operational Research, 37 (8), 1439–1454.

Ruiz, R. and Rodriguez, J.A. (2010). The hybrid flow shop scheduling problem, European Journal of Operational Research, 205(1), 1–18.

Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of First International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, Pittsburgh, PA, USA, 93–100.

Soroush, H.M. (2015). Scheduling with job-dependent past-sequence-dependent setup times and job-dependent position-based learning effects on a single processor. European Journal of Industrial Engineering, 9(3), 277-307.

Srinivas, N. and Deb, K., (1995). Multi-objective function optimization using non-dominated sorting genetic algorithms, Evolution Computing, 2(3), 221–248.

Tabucanon, M.T. (1988). Multiple criteria decision making in industry (Vol. 8). Elsevier Science Ltd.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A.R. and Mirzaei, A.H. (2008). Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm. The International Journal of Advanced Manufacturing Technology, 36(9-10), 969-981.

Wang, L.Y., Wang, J.J., Wang, J.B. and Feng, E.M. (2011). Scheduling jobs with general learning functions. Journal of Systems Science and Systems Engineering, 20(1), 119-125.

Wu, Y.B. and Wang, J.J. (2016). Single-machine scheduling with truncated sum-of-processing-times-based learning effect including proportional delivery times. Neural Computing and Applications, 27(4), 937-943.

Yenisey, M.M. and Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. Omega, 45, 119-135.

Yue, Q. and Wan, G. (2017). Order scheduling with controllable processing times, common due date and the processing deadline. Journal of Systems Science and Systems Engineering, 1-20.

Zandieh, M., Ghomi, S.F. and Husseini, S.M. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Applied Mathematics and Computation, 180(1), 111-127.

Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N. and Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation, 1, 32–49.

Farzad Pargar obtained his doctoral degree in industrial engineering and management from the University of Oulu, Finland. He is currently working as a postdoctoral researcher in the same unit. His work focuses on optimization modeling and management of complex real-world systems using quantitative techniques. The topic of his doctoral thesis is related to resource optimization techniques in maintenance and production planning. His research interests include scheduling, optimization, project management, maintenance & production planning, supply chain management, system dynamics and agent-based simulation. He has published several journal and conference papers and acted as reviewer of many prestigious journals. He is an editorial board member of International Journal of Data Analysis Techniques and Strategies and acted as guest editor for applied Computational Intelligence and Soft Computing Journal.

Mostafa Zandieh accomplished his BSc in industrial engineering at Amirkabir University of Technology, Tehran, Iran, and MSc in industrial engineering at Sharif University of Technology, Iran. He obtained his PhD in industrial engineering from Amirkabir University of Technology, Iran. Currently, he is an Associate Professor at Industrial Management Department, Shahid Beheshti University, Iran. His research interests are production planning and scheduling, financial engineering, quality engineering, applied operations research, simulation and artificial intelligence techniques in the areas of manufacturing systems design.

Osmo Kauppila received his doctoral degree in industrial engineering and management from the University of Oulu, Finland. He currently holds a position of University Lecturer of Process and Quality Management at this institution. His research interests and teaching responsibilities span across a large variety of quantitative methods that support decision making. Some particular topics of interest include design of experiments, linear programming, regression analysis and machine learning, measurement system analysis, discrete event simulation and Bayesian methods. Lately he has worked on the application of operations research and statistical process control methods in railway environments. Despite having the main focus of his work on teaching and industry collaboration, Kauppila has authored or co-authored over twenty journal articles, conference papers and other scientific publications. He is a Distinguished Reviewer of the Project Management Journal and a certified Six Sigma Black Belt.

Jaakko Kujala is a professor in project and quality management. He currently works at the research unit of industrial engineering and management, University of Oulu, Finland. He has over ten years' experience in international system industry before his career in the academia. He has initiated and served as a director for several large research projects with the participation of leading project-based firms, such as Nokia, Kone, Wärtsilä, Metso, and Outotec. His publications include more than 150 academic papers, book chapters and books. His current research interests include simulation and modelling in the context of complex systems, governance of complex inter-organizational project networks and value creation in the context of project business.