Journal of Quality Engineering and Production Optimization

Vol. 5, No. 1, Winter & Spring 2020

http://jqepo.shahed.ac.ir

Research Paper

DOI: 10.22070/JQEPO.2020.5317.1143

A Hybrid Bee Algorithm for Two-Machine Flow-Shop Scheduling Problems with Batch Delivery

Mohammad Rostami [1*], Samira Shad [2]

[1] *Department of Industrial Engineering and management, Shahrood University of Technology, Shahrood, Iran*
[2] *Department of Industrial Engineering, University of Science and Technology, Tehran, Iran*

**** Corresponding Author:** Mohammad Rostami (Email:Rostami_m@shahroodut.ac.ir)*

Abstract *– Because of the high costs for the delivery, manufacturers are generally needed to dispatch their products in a batch delivery system. However, using such a system leads to some adverse effects, such as increasing the number of tardy jobs. The current paper investigates the two-machine flow-shop scheduling problem where jobs are processed in series on two stages and then dispatched to customers in batches. The objective is to minimize the batch delivery cost and tardiness cost related to the number of tardy jobs. First, a mixed-integer linear programming model (MILP) is proposed to explain this problem. Because the problem under consideration is NP-hard, the MILP model cannot solve large-size instances in a reasonable running time. Some metaheuristic algorithms are provided to solve the large-size instances, including BA, PSO, GA, and a novel Hybrid Bees Algorithm (HBA). Using Friedman and Wilcoxon signed-ranks tests, these intelligent algorithms are compared, and the results are analyzed. The results indicate that the HBA provides the best performance for large-size problems.*

Keywords*– Scheduling, Batch Delivery System, Number of Tardy Jobs, Mixed-Integer Linear Programming, Metaheuristic Algorithms.*

I. INTRODUCTION

Tardiness always leads to customers' discontent (in supply chains) or employers (in project implementation). On the one hand, suppliers attempt to deliver orders to the customers with the minimum number of tardy jobs. On the other hand, they tend to reduce the total delivery costs by using the batch delivery system, resulting in increased tardiness. Therefore, achieving a tradeoff between these criteria- i.e., delivery cost and schedule - is of great importance in the SCM (Hall & Potts, 2003).

Many real-world problems involve production lines connected in the series. In the literature, these production lines are known as the "flow-shop system". Flow shops are used in mass production systems of interest to various industries (Mahadevan, 2015). In this system, several machines are placed in series, and some jobs should be processed on these machines sequentially. There are several types of flow shops (Gupta & Stafford, 2006), which the literature on the subject of the two machine flow-shop scheduling problems is presented in section A.

Manuscript Received:1-Jan-2020 & Revised: 25-March-2020 & Accepted: 23-May-2020
ISSN: 2423-3781

From the point of logistic view, there are many restrictions on the distribution of orders. On the one hand, limitations on the number of vehicles and the high transportation cost make managers prefer to dispatch orders to the customer at once (Mazdeh & Rostami, 2014; Herrmann et al., 2003). This decision causes some orders to deliver the customer after the due date. The higher the number of orders delivered to the customer after the due date, the greater the dissatisfaction is. Therefore, there will always be a conflict of decision between production and logistic managers. This paper addresses the decision integration between the two flow-shop scheduling problems and batch delivery costs to respond to this conflict.

There are many examples of the application of this problem in the real world. One of the most famous is the Fast-Moving Consumer Goods (FMCG) supply chain. In this chain, the speed of brand replacement is exceptionally high. Therefore, the delivery of products after due dates causes lost sales. On the other hand, due to these products' small volume, dispatching orders separately for customers is not cost-effective in logistics costs. Therefore, the use of a batch delivery system is considered by managers. It should be noted that the production of the FMCGs is often viewed in two stages, i.e., production and packaging, which can be adapted to the two-machine Flow shop environment.

As this is an NP-hard problem, the MILP model cannot solve large-size instances in a reasonable running time. Also, there is no exact polynomial method for these problems to obtain a globally optimum solution. For this reason, some metaheuristic algorithms, including Bee Algorithm (BA), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and a novel Hybrid Bees Algorithm (HBA), are provided to solve large-size instances. The reason for the importance of speed in reaching a near-optimal solution for the large-size instances is that the problem under consideration in this research is basically at the operational planning level, which sometimes managers have to be decided daily. Therefore, reaching a near-optimal solution is better than not reaching any solution. In the following, a comprehensive review of the literature is presented.

A. Flow-shop scheduling problems

Scheduling and sequencing problems in two machine flow-shop environments have been considered by a large number of researchers. The problem of flow-shop scheduling for two and three machines was first investigated by Johnson (1954), aiming to minimize the makespan. During the next decades, many papers were presented, focusing on flow-shop problems with various objectives, in addition to minimizing the makespan (Panwalkar et al. (2013) and Yenisey et al. (2014)). For two-machine flow-shop scheduling problems to minimize the number of tardy jobs, Lenstra et al. (1977) proved that this problem is NP-complete. However, single-machine scheduling problems can be solved by using Moore's [10] algorithm (1968) with the complexity of *O(nlogn)*. Hariri and Potts(1989) introduced three lower bounds for a branch and bound algorithm to solve two machine flow-shop problems to minimize the number of tardy jobs $F2||\sum U_i$ ($F2||\sum U_i$). Gupta and Hariri (1997) developed an exact algorithm - $O(n^2 logn)$- for multiple machine flow-shop scheduling problems to minimize the number of tardy jobs ($Fm|idm|\sum U_i$). They proved that Moore's algorithm could be used to solve $Fm|ddm|\sum U_i$, where *idm* and *ddm* increase and decrease the series of dominating machines, respectively.

B. Scheduling problems with batch delivery

Many researchers have recently considered scheduling and sequencing problems in a single machine environment with notice to the batch delivery system. It can be cited, among others, Herrmann and Lee (1993) with the objective of minimizing the sum of earliness and tardiness penalties; Cheng et al. (1996) with the objective of minimizing the earliness penalties; Hall and Potts (2005) with a review of multi-objective complex scheduling problems; Pundoor and Chen (2005) with the objective of minimizing the maximum tardiness; Ji et al. (2007) with the objective of minimizing the sum of weighted completion times; Low et al. (2010) through examining the objective of minimizing the makespan with release times; Shabtay (2010) with the aim of minimizing the sum of earliness, tardiness, and inventory costs; Mazdeh et al. (2011) by using a BB algorithm aimed to minimize the total weighted completion times; Hall and Potts (2003) by employing a DP algorithm for different scheduling problems; Mazdeh et al. (2007, 2008) through

investigating supplier and manufacturer viewpoints for a number of customers; and, Hamidinia et al. (2011) by proposing a genetic algorithm with the objective of minimizing the sum of weighted earliness and tardiness. Furthermore, Steiner and Zhang (2011) studied a single-machine scheduling model to minimize the weighted numbers of tardy jobs and the delivery costs, where expected due dates were allocated to all jobs and limitations were imposed batch capacities. However, the same problem was solved by Rasti-Barzoki and Hejazi (2013), using a heuristic algorithm (HA) and a branch and bound method. Assarzadegan and Rasti-Barzoki (2016) proposed and compared three heuristic methods for minimizing the total costs of due date assignment, maximum tardiness, and distribution in an integrated scheduling distribution system. Mazdeh et al. (2013) proposed a BB algorithm for single machine scheduling with batch delivery to minimize the maximum tardiness and delivery costs. Further, Yin et al. (2013) examined a single machine scheduling problem with batch delivery and considering assignable standard due dates. They presented a dynamic programming algorithm to minimize the cost function included earliness, tardiness, job holding, window starting time, window size, and batch delivery. Adding release times to the above problem, Ahmadizar and Farhadi (2015) developed an imperialist competitive algorithm (ICA) to solve large-size problems. Rostami et al. (2015) introduced a new BB algorithm for a single machine scheduling problem considering job release times in a batched delivery system and presented a case study for this procedure. Moreover, by studying on single-machine scheduling with batch deliveries, Cheng et al. (2015) found that when the jobs have same sizes, then it can be found a schedule and delivery plan in O($n$log$n$) time such that the service span is minimum. Rostami et al. (2018) also investigated the integration of production scheduling and distribution in a supply chain network where both deterioration of machine and learning effect have been accordingly addressed, and the jobs must be dispatched to customers via the batch delivery system. Jia et al. (2019) investigated integrated production scheduling and transportation on parallel batch machines via several vehicles. For this problem, they present a mixed-integer programming model and also a deterministic heuristic H, and two heuristics based on ACO. Joo and Kim (2019) developed two VNS algorithms for an integrated production and batch delivery problem with limited capacity trucks. Recently, Ganji et al. (2020) introduced a green multi-objective integrated production and distribution scheduling with VRP and time windows. Their goals include minimizing total costs of distribution, carbon emission, and customer dissatisfaction. Some multi-objective metaheuristic algorithms are presented to solve this problem. In the flow shop environment, Kazemi et al. (2017) developed a two-stage assembly flow-shop scheduling problem considering a batched delivery system where there are m unrelated machines at the first stage processing the components of each job and multiple identical machines at the next assembly stage. Basir et al. (2018) solved the above problem by using a bi-level genetic algorithm.

In the academic literature for two-machine flow-shop scheduling problem via considering the batch delivery system, only six papers can be cited. The first was presented by Lee and Chen (2001), who viewed the processing of unfinished jobs on two separate machines in combination with the distribution problem. Investigating the complexity, the researchers developed polynomial-time or pseudo-polynomial-time algorithms for solving some instances. In this line, Soukhal et al. (2005) evaluated this problem with transportation constraints and proved its complexity. The problems were examined for two cases, with and without buffer capacity limitation. However, Hall and Potts (2003) solved some scheduling problems with the batch delivery system and different goals from three perspectives: the supplier, the manufacturer, and the general supply chain problem (two-machine flow-shop system) used a forward dynamic programming algorithm. Rasti-Barzoki et al. (2013) provided a branch and bound algorithm to minimize the number of weighted tardy jobs and the delivery costs for a single customer. Mazdeh and Rostami (2014) proposed a BB algorithm for the two-machine flow-shop system to minimize the maximum tardiness and delivery costs in a batch delivery system. The same problem is extended to a permutation flow-shop scheduling problem and solved by two simple heuristics and a novel metaheuristic (Wang et al., 2017).

C. Metaheuristic methods in flow-shop scheduling problems

Different solution methods have been developed in the literature for solving scheduling and sequencing problems in the last decades. In general, the solution procedures can be divided into three groups, namely, exact, heuristic, and metaheuristic methods (see Moumene & Ferland, 2009). Since the majority of flow-shop scheduling problems are NP-hard, the metaheuristics algorithms have been received much attention. During the last five years, various

metaheuristics methods have been proposed to provide suitable solutions in an acceptable time for flow-shop scheduling problems. For example, tabu search (Ding et al., 2015), ant colony algorithm (Riahi & Kazemi, 2016), genetic algorithm (Chamnanlor et al., 2014), simulated annealing (Jolai et al., 2013), cuckoo search algorithm (Marichelvam et al., 2014a) and firefly algorithm (Marichelvamet al., 2014b). One metaheuristic technique which has been considered significant in recent years is the bee algorithm family. The bee algorithm family includes bee algorithm (BA), artificial bee colony (ABC), and bee swarm optimization (BSO). In flow-shop scheduling problems, Tasgetiren et al. (2011) developed a discrete artificial bee colony algorithm. This algorithm was hybridized with several greedy algorithms for finding the sequence that obtains the lowest total flow time. Guanlong et al. (2012) presented a hybrid artificial bee colony algorithm for solving the blocking flow-shop scheduling problem. Here, the hybridization was generated by an initialization scheme based on a variant of the Nawaz-Enscore-Ham (NEH) heuristic. Recently, Han et al. (2015) developed a new hybrid artificial bee colony by incorporating the ABC with differential evolution (DE).

Regardless of flow-shop scheduling problems, many researchers have attempted to develop various hybrid algorithms based on the bee algorithm family. Marinakis et al. (2009) introduced a new hybrid method for clustering combined with the artificial bee colony (ABC) and a greedy randomized adaptive search procedure (GRASP). However, Zhao et al. (2010) developed a hybrid metaheuristic approach of an artificial bee colony (ABC) and genetic algorithm (GA). Using the ABC algorithm for hybridizing a quantum evolutionary algorithm (QEA), Duan et al. (2010) designed a new algorithm for solving continuous optimization problems. Also, Li et al. (2011) introduced a hybrid artificial bee colony (HABC) algorithm based on the Pareto front to solve the multi-objective flexible job-shop scheduling problem. Thakare and Chaudhari (2012) performed PSO and BA in parallel and transitional modes and developed two versions of a hybrid bee algorithm to solve the clustering problem. Wu et al. (2012) developed a hybrid harmony search and artificial bee colony algorithm for various optimization issues. Yildiz (2013) examined a novel hybrid optimization method (HRABC) based on an artificial bee colony algorithm and Taguchi's method. Kefayat et al. (2015) and Chun-Feng et al. (2014) proposed a hybrid artificial bee colony (ABC) with PSO for overcoming the disadvantage of the ABC algorithm. Kefayat et al. (2015) developed a hybrid optimization approach that combines ACO and ABC algorithms for probabilistic optimal placement and sizing of distributed energy resources. Recently, Karaboga and Kaya (2016) introduced an adaptive and hybrid artificial bee colony (aABC) algorithm by using the crossover rate and adaptively coefficient. To the best of our knowledge, in the literature there seems to be no research for providing a hybrid metaheuristic algorithm based on Bee Swarm Optimization (BSO) algorithm and particle swarm optimization (PSO) algorithm. In general, hybrid metaheuristic approaches can be classified as either collaborative combinations or integrative combinations. The algorithm developed in this paper is classified as an integrative combination. For further study on the bee algorithm family and its hybridizations, see (Karaboga et al., 2014).

The current study evaluates two-machine flow-shop scheduling problems with a batch delivery system, aiming to minimize the number of tardy jobs and the delivery costs. The significant contributions of this study can be mentioned as follows:

- The class of two-machine flow-shop scheduling problems with batch delivery to multiple customers is introduced to minimize the costs related to the number of tardy jobs and the delivery costs.
- A mixed-integer linear programming model is introduced for obtaining global optimum solutions of small-size problems.
- A novel hybrid bee algorithm (HBA) based on the particle swarm optimization (PSO) algorithm is introduced and extended to solve large-size problems.

The rest of this paper is organized as follows. Section II defines the problem with a mixed-integer linear programming (MILP) model. Section III develops a hybrid bee algorithm and also describes the structure of other metaheuristic methods. Section IV provides the computational results to compare and rank the metaheuristic solution methods. Finally, section V concludes the results and provides some recommendations for future studies.

II. PROBLEM FORMULATION

Let N be the number of jobs that must be processed immediately on two machines in a flow shop environment related to the F customers. All jobs are available at time zero, and pre-emption is not allowed. For these jobs, due dates are defined by customers, denoted by d_{ij}. If a job delivers to the customer after its due date, the lost sale cost, i.e., β, will be reimbursed. To reduce logistics costs, jobs related to each customer can be dispatch in a batch delivery system. Dj indicates the cost of dispatching each batch for customer $_j$. The objective function is minimizing the sum of lost sales costs plus the total delivery cost. When all the jobs belonging to the same batch are completed on machine 2, the batch is rendered. So, the completion time of one batch -C_k - is the completion time of that last job in that batch. Rendition time of one job -R_{ij} - is the completion time of a batch containing that job. The problem assumptions are summarized as follows:

- N is the number of jobs that need to be processed successively on machines.
- F is the number of customers with some orders.
- A machine can process only one job at a time.
- All jobs are available at time zero, and no pre-emption is allowed.
- Only jobs related to one customer may be batched together.

Table I. Description of parameters and variables

		Description	***Values taken***
Indexes	i	the job number	$1,2,\ldots,n_j$
	j	the customer number	$1,2,\ldots,F$
	k	the batch number	$b_{j-1}+1,\ \ldots,b_j$
	z	The machine number	$1,2$
	m	the priority number	$1,2,\ldots,N$
Parameters	N n_j	The number of jobs Total number of jobs belonging to customer j	
	M	An enough big number	
	β	lost sale cost coefficient	
	D_j	Cost of delivery for each batch belong to jth customer	
	p_{ijz}	processing time of the ith job of jth customer on zth machine	
	d_{ij}	due date of the ith job of jth customet	
Decision variables	α_j	number of batches belong to jth customer	
	C_k	Completion time of the kth batch	
	C_{mz}	Completion time of the mth priority on the zth machine	
	C_{jjz}	Completion time of the ith job of the jth customer on the zth machine	
	R_{ij}	Rendition time of the ith job of the jth customer	
	U_{ij}	Equals 1 if the ith job of the jth customer has tardiness	0,1
	x_{ijkm}	equals 1 if the ith job of the jth customer is in kth batch and mth priority	0,1
	y_{jk}	equals 1 if there is a job belonging to the kth batch which relates to the jth customer	0,1

Based on the standard classification of scheduling problems developed by Graham et al. (1979), this problem will be stated $F2||(\beta \sum U_i + \sum_{j=1}^{F} \alpha_j D_j)$. Where $F2$ represents the two-machine flow-shop system and β is the coefficient of lost-sale cost. If a batch is dispatched to the related customer after the due date, then the lost sale cost is imposed on the manufacturer. Typically, this cost is considered equal to the manufacturer's benefit if the order is delivered before its due date. Under such conditions, two parts of the objective function can be both expressed in terms of cost. If the job i has tardiness, then $U_i = 1$; otherwise, it would be zero. α_j indicates the number of batches dispatched to the jth customerD_j while presenting the delivery cost for each batch belonging to the customer j. Other parameters and decision variables are presented in Table I. Since Lenstra et al. (1977) proved that minimizing the number of tardy jobs in a two-machine flow-shop system is NP-complete, the other batch delivery system as the second part of the decision making, the problem will remain NP-complete.

The idea of problem modeling is as follows: First, for each customer, suppose that the number of empty batches is equal to the number of jobs belonging to that customer ($b_j=b_{j-1}+n_j$ and $b_0=0$). Each batch's capacity is supposed that be equal to the number of jobs belonging to the given customer. The model determines which jobs must be placed in each batch. Besides, the N priority number is considered, which indicates the order of processing jobs. Each batch will be delivered to its determined customer while its capacity may not be used ultimately. Note that some batches may remain vacant, but there is no effect on the solution. By considering the above definitions, the model is proposed as follows:

$$Min \ \beta \times \sum_{j=1}^{F} \sum_{i=1}^{n_j} U_{ij} + \sum_{j=1}^{F} \alpha_j D_j \tag{1}$$

Subject to:

$$\begin{cases} -\left(R_{ij} - d_{ij}\right) + MU_{ij} \geq 0 \\ \left(R_{ij} - d_{ij}\right) + M(1 - U_{ij}) > 0 \end{cases} \forall i, \forall j \tag{2}$$

$$R_{ij} = \sum_{k=b_{j-1}+1}^{b_j} \sum_{m=1}^{N} x_{ijkm} \times C_k \qquad \forall i, \forall j \tag{3}$$

$$C_k = \max_{ijm} \{C_{ij2} \times x_{ijkm}\} \qquad \forall k \tag{4}$$

$$C_{ij2} = \sum_{m=1}^{N} \sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \times C_{m2} \quad \forall i, \forall j \tag{5}$$

$$\begin{cases} C_{m1} = C_{m-1,1} + \sum_{j=1}^{F} \sum_{i=1}^{n_j} \sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \times p_{ij1} \\ C_{m2} = \max\left\{C_{m-1,2}, C_{m1}\right\} + \sum_{j=1}^{F} \sum_{i=1}^{n_j} \sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \times p_{ij2} \quad \forall m \\ C_{01} = 0 \\ C_{02} = 0 \end{cases} \tag{6}$$

$$\begin{cases} \sum_{j=1}^{F} \sum_{i=1}^{n_j} \sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} = 1 \qquad \forall m \\ \sum_{k=b_{j-1}+1}^{b_j} \sum_{m=1}^{N} x_{ijkm} = 1 \quad \forall i, \forall j \end{cases} \tag{7}$$

$$\begin{cases} -\sum_{i=1}^{n_j}\sum_{m=1}^{N} x_{ijkm} + My_{jk} \geq 0 \\ \sum_{i=1}^{n_j}\sum_{m=1}^{N} x_{ijkm} + M(1-y_{jk}) > 0 \end{cases} \quad \forall k\ ,\forall j \tag{8}$$

$$\alpha_j = \sum_{k=b_{j-1}+1}^{b_j} y_{jk} \ \forall j \tag{9}$$

$$\begin{aligned} &x_{ijkm}, y_{jk}, u_{ij} \in \{0,1\} \\ &C_k, C_{mz}, C_{ijz}, R_{ij} \geq 0 \\ &\alpha_j \in Integer \end{aligned} \tag{10}$$

The objective function minimizes the costs related to the number of tardy jobs and the sum of delivery costs. Equation (2) states that if the rendition time of the *i*th job of the *j*th customer is more than its due date, it is a tardy job. Equation (3) calculates the rendition time of the *i*th job of the *j*th customer. Equation (4) determines each batch's total completion time by finding the maximum completion times of the jobs placed in each batch. Equation (5) calculates the completion time of the *i*th job of the *j*th customer over machine 2, while equation (6) determines the completion time of each priority on each machine using the recursive relations of the two-flow-shop problem. According to equation (7), one and only one job must be assigned to each priority, and each job related to a customer must be assigned only to one priority and one batch. Equation (8) allows a batch to be formed when at least one job is assigned to the batch; if there is no job assigned to a batch, this batch would be null and equal to zero. Equation (9) determines the number of batches needed to be dispatched for a customer using the value α_j . Equation (10) defines the variables.

As the developed model contains the binary variables and some nonlinear equations (Equation (3), (4), (5) and (6)) so, it is a mixed-integer nonlinear programming model. However, the linearization of each equation can be simply obtained by using equalities (11), (12), (13), and (14), respectively. Here, the linearization methods presented in the study of Chen et al. (2011) are inspired.

$$R_{ij} \geq C_k + M(\sum_{m=1}^{N} x_{ijkm} \text{-} 1) \qquad \forall i, j, k \tag{11}$$

$$\mathrm{C}_k \geq \mathrm{C}_{ij2} + \mathrm{M}(\sum_{m=1}^{N} x_{ijkm} \text{-} 1) \quad \forall i, j, k \tag{12}$$

$$\mathrm{C}_{ij2} \geq \mathrm{C}_{m2} + \mathrm{M}(\sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \text{-} 1) \quad \forall i, j, m \tag{13}$$

$$\begin{cases} C_{m1} = C_{m-1,1} + \sum_{j=1}^{F}\sum_{i=1}^{n_j}\sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \times p_{ij1} \\ \mathrm{C}_{m2} \geq \mathrm{C}_{m-1,2} + \sum_{j=1}^{F}\sum_{i=1}^{n_j}\sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \times p_{ij2} \quad \forall m \\ \mathrm{C}_{m2} \geq \mathrm{C}_{m1} + \sum_{j=1}^{F}\sum_{i=1}^{n_j}\sum_{k=b_{j-1}+1}^{b_j} x_{ijkm} \times p_{ij2} \qquad \forall m \\ C_{01} = 0 \\ C_{02} = 0 \end{cases} \tag{14}$$

In equation (11), if the ith job of the jth customer is not assigned to the kth batch, $\left(\sum_m x_{ijkm} - 1\right)$ equals -1. Furthermore, setting a large enough positive value for *M* represents a negative value for the equation's right-hand side. Otherwise, $\left(\sum_m x_{ijkm} - 1\right)$ equals 0, so the real value of R_{ij} is obtained. According to equation (12), if the ith job of the jth customer does not belong to the kth batch, $\left(\sum_m x_{ijkm} - 1\right)$ equals -1 and, again, gives a negative value for the right-hand side of the equation. Otherwise, $\left(\sum_m x_{ijkm} - 1\right)$ equals 0 and the value of the right side of the equation shows each job's completion time. So, the real value is obtained as C_k has to be greater than all these values.

Equation (13) states that if the ith job of the jth customer is not assigned to the *m*th priority, $\left(\sum_k x_{ijkm} - 1\right)$ equals -1, and this leads to a negative value for the right-hand side of the equation. Otherwise, $\left(\sum_k x_{ijkm} - 1\right)$ equals 0, so the real value of C_{ij2} is obtained.

By using the new equation instead of the nonlinear equations, the mathematical model proposed here is linearized. This model can be solved by using a commercial solver. However, no software can promise to find optimum solutions for large-size instances in a reasonable running time. This issue will be discussed in detail.

III. METAHEURISTIC METHODS

In this section, a hybrid bee algorithm is first provided, which combines the particle swarm optimization (PSO) algorithm with a bee algorithm (BA). After describing BA, PSO, and HBA, a novel genetic algorithm is extended, and the related functions are expressed.

A. Hybrid Bee Algorithm

The bee algorithm (BA) is derived from the honeybee behavior idea, based on swarm intelligence and food searching system. This method was first provided by Pham et al. (2005), which is a population-based search algorithm. A numerical optimization algorithm based on honeybees' foraging behavior, called Artificial Bee Colony (ABC), was proposed by Karaboga (2005). Later, a new method of bees algorithm, called bee swarm optimization (BSO), was introduced by Akbari et al. (2010). This method has a high efficiency in numerical optimization. Also, Preux and Talbi (1999) reviewed the hybrid evolutionary algorithms and proposed a general classification of how hybridization can be conducted. In this section, a hybrid bee algorithm (HBA) is provided, an extended model from the BSO through the PSO algorithm. Kennedy and Eberhart (1995, 1997) introduced the Particle swarm optimization (PSO) algorithm inspired by birds and fish's social behavior. In PSO, particles with a similar neighborhood type are related by transmitting data about each particle's specific position. So, all particles are inclined into one particle that seems to have the best-known position. At below, the structure of a hybrid bee algorithm (HBA) algorithm is described.

A.1. Framework & Bee Representation in HBA Algorithm

According to the idea of this algorithm, bees of the initial population are divides into four group bases on the values of their fitness function, and particular works are done on each group so that their position will be changed in the next iteration and this process is continued until the termination criterion is satisfied. This algorithm employs the idea of changes in position and velocity in the particle swarm optimization (PSO) algorithm to obtain better solutions in less time.

In this algorithm, each bee *z* is shown as $x_z = \left(x_{1111}, x_{1112}, ..., x_{111n}, ..., x_{11n_1n}, ..., x_{n_F Fn_F n}\right)$ where $x_{ijkm} \in \{0,1\}$. x_{ijkm}

equals 1 if the *i*th job belonging to the *j*th customer is placed in the *k*th batch and the *m*th place of the job sequence. The present idea assumes that for each customer *j*, (j=1, 2, …, F), there are n_j empty batches (n_j presents the number of jobs belonging to the customer *j*). In this case, feasible solutions will always be obtained. It means that only jobs belonging to a single customer can be placed on a batch. Fig. (1) shows an example of how hypothetical batches are formed for two customers with 2 and 3 jobs, respectively.

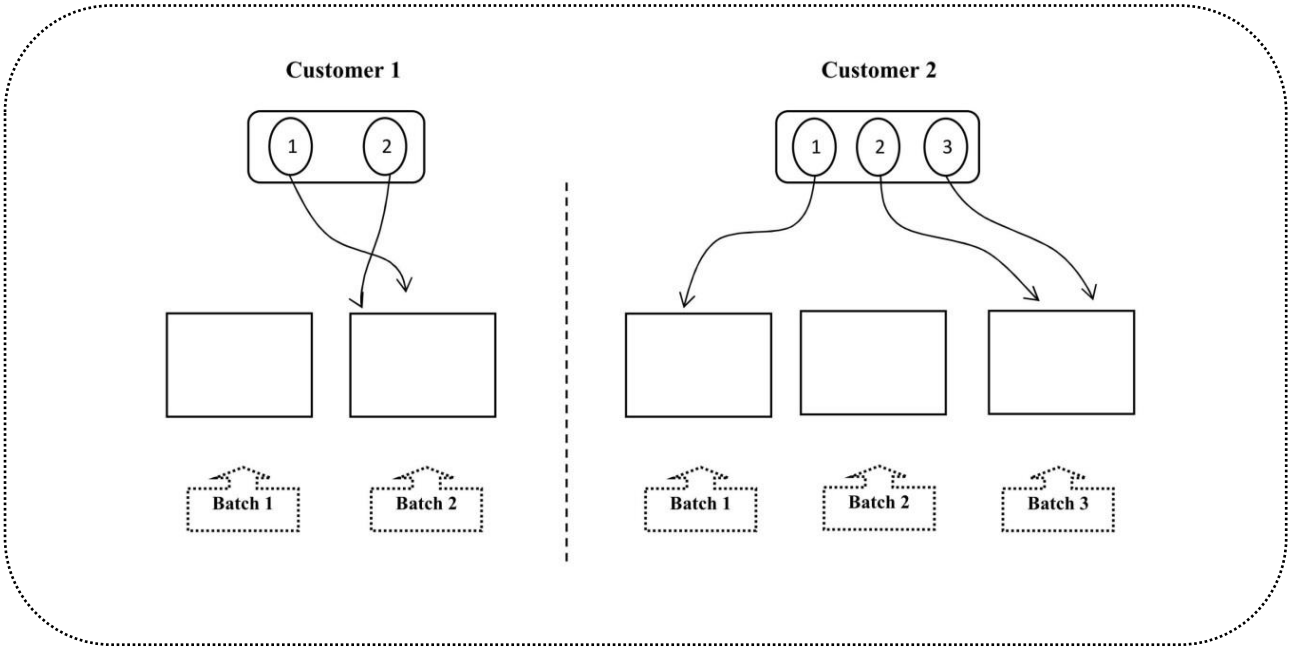

Fig.1. Formation of hypothetical batches for each customer

The N_p bees are selected, which are the initial population. These bees go to some random places, and for each place, the values of fitness function are measured. In this stage, the bees are divided into four groups: *ne* bees with the best fitness function are selected as the *elite bees* from which *nep* bees (60% of N_s) are sent for the neighborhood searching, *nm* bees with useful fitness function, but not as much as the elite bees, are selected as the *distinguished bees* from which *nsp* bees (40% of N_s) are sent for searching the neighborhood. It should be noted that N_s is the number of neighborhood search bees. *nt* bees with the worst fitness function are selected as the *explorer bees*. The rest of the population is considered as the *onlooker bees*, represented by *no*. In each iteration, three best schedules obtained in the whole algorithm up to that iteration are stored in the memory as $gbest_1$, $gbest_2$, $gbest_3$.

A.2. Initial Population Generation

As mentioned above, each bee *z* is shown as a vector whose components are the binary numbers of 0 and 1. First, Np bees are allocated to Np nutrition sources using random numbers for each component for generating the initial population. Then, to calculate fitness functions, bee position vectors are converted to schedules consisting of sequencing and job batching mood. Based on the fitness function definition, a better solution has a better fitness score; therefore, it is suitable to use the fitness function having the same feature. As the model's aim is in the minimization mode, we use the inverse of the objective function. The fitness function can be given from equation (15):

$$fit(x_i) = \frac{1}{\beta \sum U_i + \sum_{j=1}^{F} \alpha_j D_j} \tag{15}$$

A.3. Inertia Vector & Its Updates

To update bee positions, a vector, which is called the inertia vector and represented as $\tau_z = \left(\tau_{1111}, \tau_{1112}, ..., \tau_{111n}, ..., \tau_{11n_1n}, ..., \tau_{n_F F n_F n}\right)$, is defined where $\tau_{ijkm} \in \left[-\tau_{\max}, +\tau_{\max}\right]$. Here, $\tau_{\max} = 4$ and indicates allowable bound for each component of the inertia vector. When each component exceeds the limits, the component is given one of the limits. The idea of updating bee positions in this algorithm is based on a structure combined with vectors of velocity and position changes in the PSO algorithm. First, the inertia vector is updated for the bees; and then through this vector, new positions of the bees are updated by using the inertia-to-position function, which will be discussed in Section A.4. As stated, the bee population is divided into four groups. For each group, there is a particular method for updating the inertia vector. The methods are discussed below.

A.3.1. Explorer bees

This group could not find rich nutrition sources. Therefore, our strategy for updating their inertia vectors is to randomly develop a new position, provided that their fitness functions would be better than that of the expected fitness function, and then, the inertia vector is updated by Equation (16). The reason is that (a) due to the disproportional position of this group, it is unlikely to rely on their previous position, and (b) keeping such strategy in each stage can search new points from the search space.

$$\tau_{new} = \omega \tau_{old} + c_1 r_1 (RP_z - x_{old}) \tag{16}$$

The inertia weight of bees is determined as decreasing, based on a linear relation and calculated by Equation (17).

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{iter_{\max}} \times iter \tag{17}$$

Here $iter_{\max}$ is the number of total algorithm iterations, while the number is allocated for algorithm iteration at that stage. The value decreases since the initial iteration search other spaces and final iterations to obtain better solutions. This parameter is similar to the inertia weight in the PSO algorithm. In equation (16) r_1 is a random number between 0 and 1; the cognitive learning factor indicates the self-reliance rate each bee has. RP_z indicates the randomly-generated position for the *z* bee.

A.3.2. Onlooker Bees

This group does not provide useful fitness functions but not as worth as the explorer bees. These bees use data from the global best (gbest) bees given by their dancing to change their inertia vector. This group does not rely on its neighbors. Each onlooker bee always selects one of the global best bees, and the selection is based on the probability values obtained from equation (18). For example, the selection probability will be more for gbest1 than gbest2.

$$p_i = \frac{fit(gbest_i)}{\sum_{j=1}^{3} fit(gbest_j)} \tag{18}$$

Updating the inertia vector for the onlooker bees is based on equation (19).

$$\tau_{new} = \omega \tau_{old} + c_2 r_2 (gbest_i - x_{old}) \tag{19}$$

Where r_2 is a random number between 0 and 1, and c_2 is the social learning factor indicating the reliance degree on the best solutions?

A.3.3. Distinguished Bees

These bees select relatively right places. Compared to onlooker bees, this group relies on the so-far best places explored (gbest1) and the best places in their neighborhood. This updating manner seems similar to the displacement in particle positions in the PSO algorithm, where particles consider cognitive learning, in addition to social learning. However, in the proposed HBA algorithm, we try to give such a feature only to strong and distinguished bees. Therefore, updating the inertia vector of these bees can be calculated by Equation (20):

$$\tau_{new} = \omega\tau_{old} + c_1 r_3 (b_z - x_{old}) + c_2 r_4 (gbest_1 - x_{old}) \tag{20}$$

Where r_3 and r_4 are some random numbers between 0 and 1. b_z indicates the best position obtained in the neighborhood of the bee *z* in that iteration. This neighboring is obtained from the neighborhood search process explained in Section A.6. The number of the neighborhood is equal to *nsp*.

A.3.4. Elite Bees

This group indeed consists of the best and strongest bees of the population. The preference is to rely on these bees. For this reason, updating the inertia vector is only based on the reliance on the best position explored in the neighborhood. The process is given by Equation (21):

$$\tau_{new} = \omega\tau_{old} + c_1 r_5 (b_z - x_{old}) \tag{21}$$

Where r_5 is a random number between 0 and 1. b_z indicates the best position obtained from the neighborhood of the bee z in that iteration. Again, this neighboring is obtained through the neighborhood search process explained in Section A.6. The number of neighbors generated is equal to *nep*.

A.4. Inertia-to-Position Function

New positions for bees can be achieved by using the updated inertia vector. Note that the position vector is given as discrete numbers between 0 and 1, while the inertia vector could select as the real numbers. This results in impossible solutions. For this reason, the inertia vector must be converted to the probability value. The probability values are calculated from the sigmoid function (22). The value of $e(\tau_{ijkm})$ indicates the probability as x_{ijkm} takes 1.

$$e(\tau_{ijkm}) = \frac{1}{1+\exp(-\tau_{ijkm})} \tag{22}$$

Step1. Set

$$E = \left\{ e\left(\tau_{1111}\right), e\left(\tau_{1112}\right), \ldots, e\left(\tau_{111n}\right), \ldots, e\left(\tau_{11n_{in}}\right), \ldots, e(\tau_{n_F F n_{F^n}}) \right\}$$

and $S = \varnothing$

Step 2. Choose the biggest value from E, e.g., $e\left(\tau_{lqrs}\right)$;

and assign this unscheduled job l belong to customer q to S (*i.e.* $x_{lqrs} = 1$)

and delete $e(\tau_{ijks})$ *from* E ($i.e.\ x_{ijks} = 0$),
where $i = 1, 2, .., n_F$, $j = 1, 2, \ldots, F$ *and* $k = 1, 2, \ldots, n_F$.
and delete $e(\tau_{lqkm})$ *from* E ($i.e.\ x_{lqkm} = 0$),
where $k = 1, 2, \ldots, n_q$, $m = 1, 2, \ldots, n$.

*Step*3. *Repeat step* 2 *until* E *equals* $\varnothing$

New particles can be conducted according to the values of $e(\tau_{ijkm})$ and using the most significant position value (LPV) rule by considering such a feature. So, the conducted particles will always be feasible.

A.5. Termination Criterion

In iterative algorithms, different criteria can be imposed as the termination criterion. In this algorithm, reaching a pre-determined number for iterations is selected as the termination criterion. After the algorithm is terminated, the global best solution (gbest1) is selected as the best solution.

A.6. Neighborhood Search Scheme

In the problem under investigation, the mode of batching should be determined in addition to the sequencing. Hence, the solution space is vast. For the neighborhood search scheme, two approaches are jointly used to develop neighbors. In the first approach, the sequence of jobs and the batch number of two components are changed. So, two different components of a single bee with the value of 1 are selected, and their sequences are exchanged (index m). The exchanged batch number should always be within the range of permitted values for a given customer to change the batch number (index k). If the exchanged batch number exceeds the range of permitted values for the related customer, the maximum allowable batch number is considered. By this approach, the neighborhood would be far. Only the job sequences' values or the batch number of two selected components will be changed in the second approach. With this approach, the neighbors would be closer. For the proposed HBA, half of the neighbors are developed using the first approach, and the rest are selected through the second approach. As can be seen, the use of this neighborhood search scheme always guarantees to generate feasible solutions.

B. Genetic Algorithm

Now, a genetic algorithm will be extended in order to compare and evaluate the HBA model. In general, the chromosome structure is used in the genetic algorithm to represent the solutions. The initial population is generated either randomly or through a heuristic algorithm by using this structure. Members of the population are given as the numbers of the fitness function. The higher values the fitness function has, the higher chance the individual has to be selected. Genetic operators are used on selected members to generate a new population. This process is continued until the termination criterion will be satisfied. Since the GA is a well-known technique in the optimization context, a brief description of the algorithm structure is presented here.

In this paper, a string consisting of two substrings is used for representing the chromosomes. The first substring shows the order of jobs, and the second substring indicates the number of batches. In each substring, the index represents the number of orders. For example, the number placed on the *i*th cell of the *sequence* substring shows the sequence of the job *i* among all jobs. Similarly, the number placed on the *i*th cell of the *batch* substring represents the batch to which the job *i* belongs. *Fig.2* illustrates the solution of an example with five jobs and two customers. The first two jobs belong to customer 1, and the last three jobs belong to customer 2. In this illustration, the second job for customer 1 placed on the third position of the sequence is batched with the same customer's first job.

Additionally, the first job of customer 2 placed on the fifth position of the sequence is batched by itself. The initial population consisting of N_p chromosomes, is randomly generated. Based on the fitness function in equation (15), each population is assigned some values.

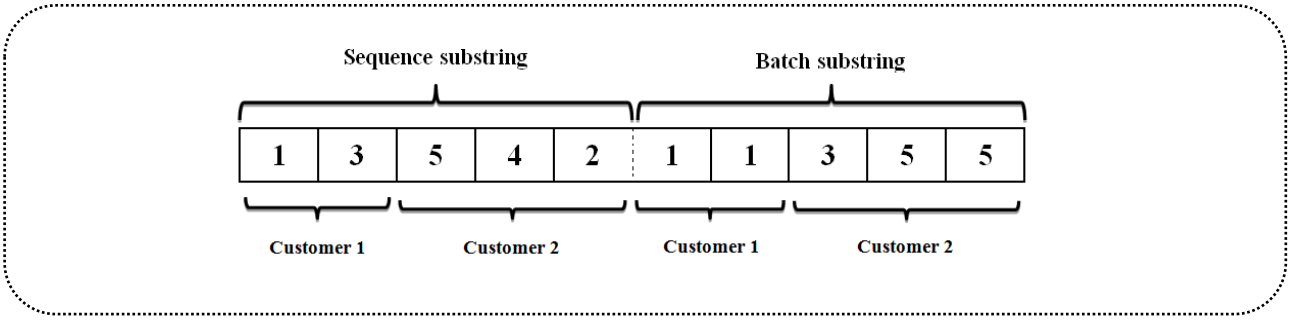

Fig. 2. Illustration of the solution for the example

A technique called the roulette wheels is used and selected as $P_c\%$ of the population as a parent to select members as the parents of a new generation. Then, crossing and mutation operators are employed for the selected members. For each substring, the operators perform individually. A "*one-point*" crossing operator is used for every two substrings. It should be noted that the diversity of genes must be followed for the sequence substring of children. Each child can be mutated by the chance of $P_m\%$. A "*random derangement*" mutation operator is provided for one sequence substring, while for another, a "*replacement*" mutation operator is used ($\sqrt{n}$ is the number of mutated genes in a selected chromosome). N_p better chromosomes are transferred into the next generation to develop a new generation and combine with the old one. Reaching a pre-determined maximum number for iterations meets the termination criterion. So, the algorithm terminates, and the best solution is provided.

$\sqrt{n}$C. Bee algorithm and particle swarm optimization

As mentioned above, the BA was first developed by Pham et al. (2005), inspired by the honeybee behavior idea. Also, Kennedy and Eberhart (1995) introduced the Particle swarm optimization (PSO) algorithm inspired by birds and fish's social behavior. In PSO, particles with a similar neighborhood type are related by transmitting data about each particle's specific position. So, all particles tend into one particle that seems to have the best-known position. Since the HBA model studied in this paper consists of both BA and PSO, these two algorithms' detailed structure is excluded here. For more information on BA and PSO, see (2005) and (1997), respectively.

IV. EXPERIMENTAL RESULTS

A. Parameters setting

This section evaluates the MILP model's performance, HBA, BA, PSO, and GA algorithms. To this end, the MILP model is encoded by using GAMS software and solved in CPLEX, while the metaheuristic algorithms are encoded by using C# and run on a system with a CPU of Core2duo 2.00 GHz and 4 GB of RAM. The performance of the metaheuristic algorithms presented here is directly related to the parameters' values, so it is essential to determine accurate values. Here, we have used Taguchi's method, which can reduce the number of required experiments. For each algorithm, the parameters are examined in three levels. Table II shows that the algorithm parameter ranges along with their levels.

Fig. (3) to Fig. (6) present the results obtained from Taguchi's optimization procedure for each metaheuristic algorithm. The exact values of S/N ratios are illustrated in Table XII in Appendix A.

Table II. Parameter ranges of metaheuristic algorithms

Algorithm	*Parameters*	*Parameter level*		
		Level 1	*Level 2*	*Level 3*
BA	*scout bees* (A)	5%	10%	15%
	best selected bee and their recruits (B)	20%	30%	40%
	N_p (C)	40	50	60
PSO	c_1 (A)	1.5	2	2.5
	c_2 (B)	1.5	2	2.5
	ω_{min} (C)	0.1	0.2	0.3
	ω_{max} (D)	0.7	0.8	0.9
	N_p (E)	40	50	60
GA	P_c (A)	30%	40%	50%
	P_m (B)	5%	10%	15%
	N_p (C)	40	50	60
HBA	c_1 (A)	1.5	2	2.5
	c_2 (B)	1.5	2	2.5
	ω_{min} (C)	0.1	0.2	0.3
	ω_{max} (D)	0.7	0.8	0.9
	ne (E)	3%	5%	7%
	nm (F)	35%	40%	45%
	nt (G)	10%	12%	14%
	N_s (H)	10%	15%	20%
	N_p (I)	40	50	60

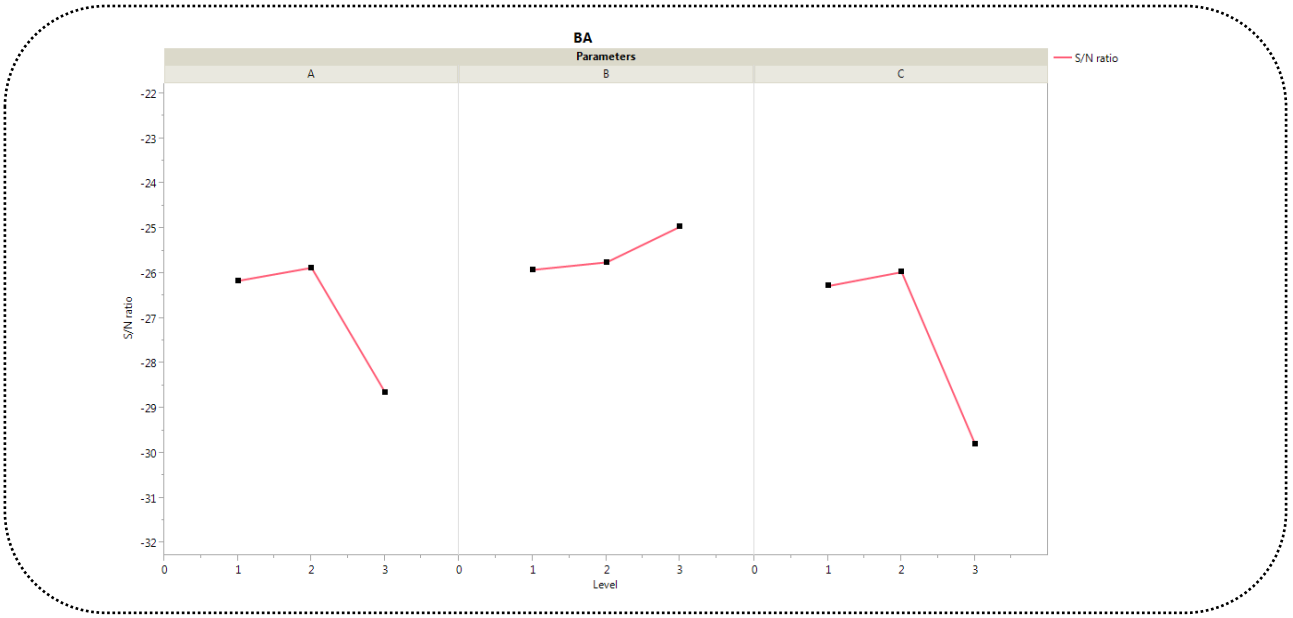

Fig.3. S/N ratio for parameters of BA

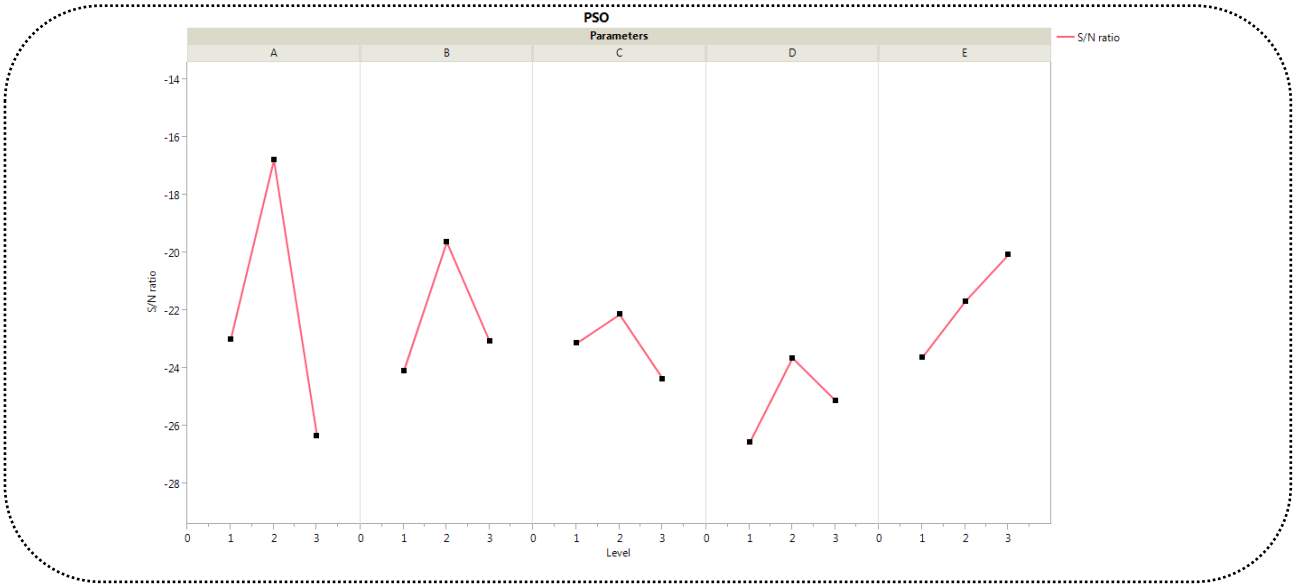

Fig.4. S/N ratio for parameters of PSO

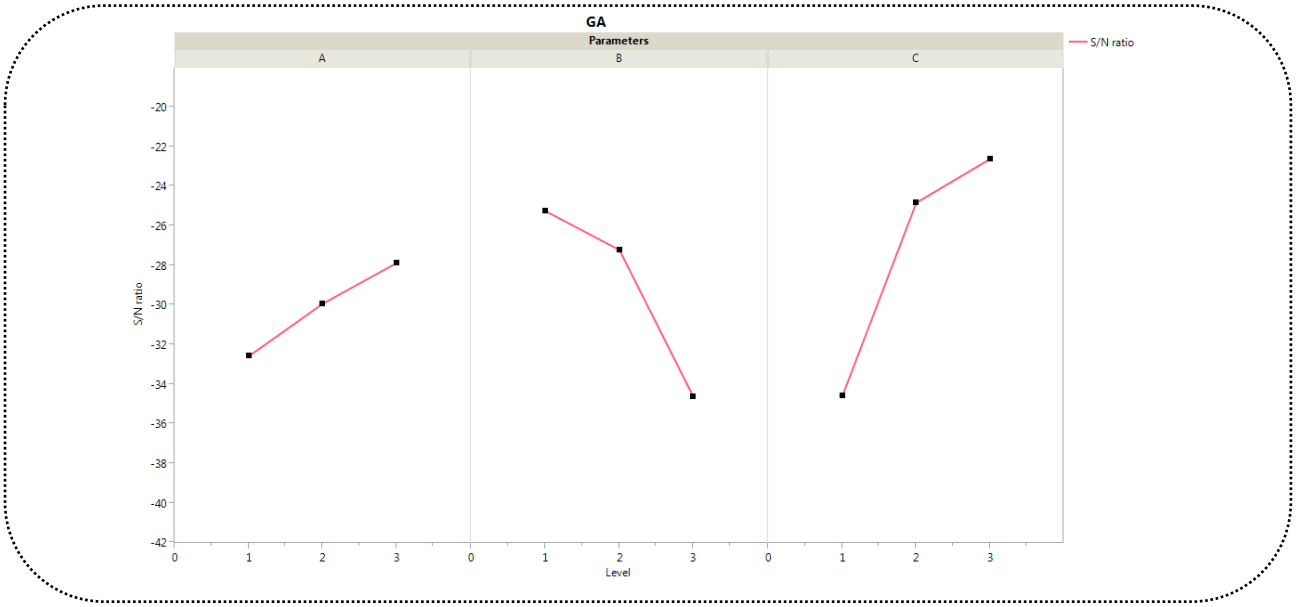

Fig.5.S/N ratio for parameters of GA

As seen from Fig. (3), the following values have the highest S/N ratio for BA:

- *scout bees* = 10% of population size
- best selected bee and their recruits = 40% of population size
- $N_p = 50$

According to Fig. (4), the following values have the highest S/N ratio for PSO:

- $c_1 = 2$ and $c_2 = 2$
- $\omega_{\max} = 0.8$ and $\omega_{\min} = 0.2$
- $N_p = 60$

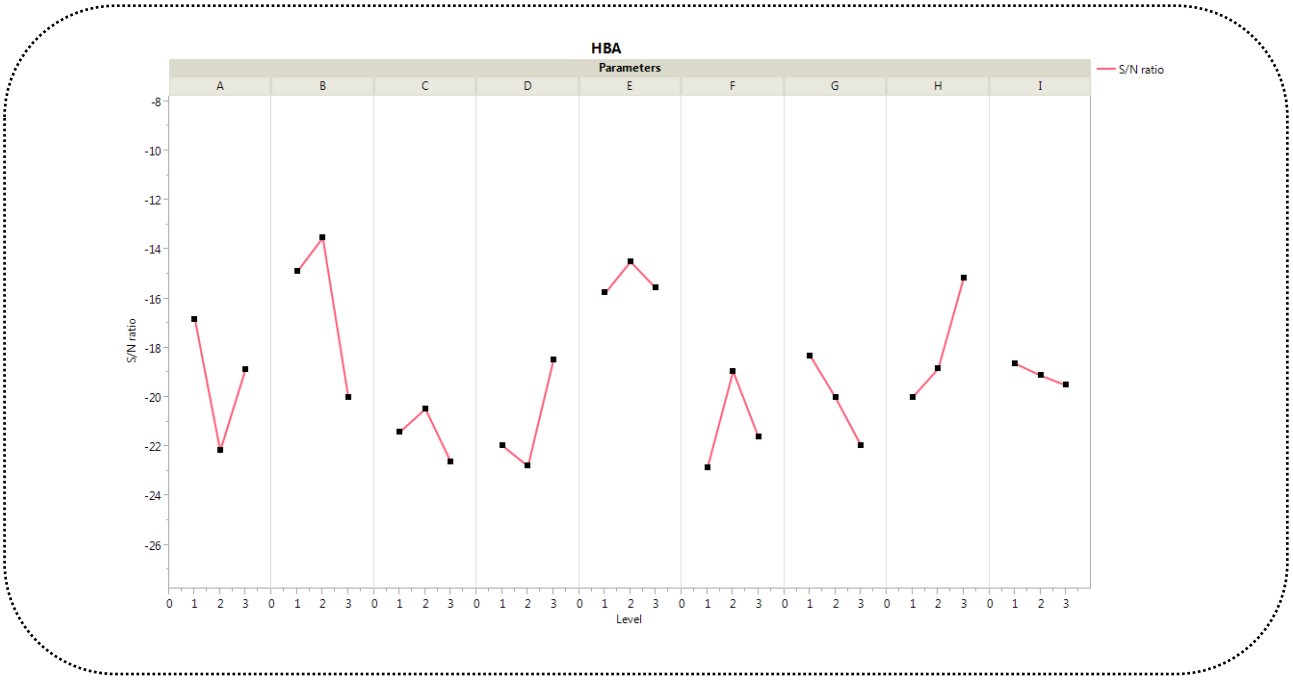

Fig.6. S/N ratio for parameters of HBA

Also, as shown in Fig. (5), the following values have the highest S/N ratio for GA:

- P_c = 50% of population size
- P_m = 5% of population size
- N_p = 60

As seen from Fig. (6), the following values have the highest S/N ratio for HBA:

- c_1 = 1.5 and c_2 = 2
- $\omega_{\max} = 0.9$ and $\omega_{\min} = 0.2$
- *ne* = 5% of population size
- *nm* = 40% of population size
- *nt* = 10% of population size; *no* selects the rest
- N_s = 20% of population size
- N_p = 40

B. Computational results

There is no well-known benchmark test for evaluating the performance of solution methods. The method provided by Bulfin and M'Hallah (2003) is employed to generate random problems. Processing times on two machines are selected from a uniform distribution function on [1, 100]. Here, d^l and d^u are selected as $d^l \in \{0.2, 0.4, 0.6, 0.8\}$ and $d^u \in \{0.4, 0.6, 0.8, 1\}$, respectively; where $d^l < d^u$. These randomly-generated parameters specify the lower and upper bounds for due dates. A uniform distribution function on [Pd^l, Pd^u] is used to determine due dates where P shows the highest completion time predicted. The value of P is given by Equation (23):

$$P = \frac{\sum_{i=1}^{n}\sum_{j=1}^{2} p_{ij} + \sum_{i=1}^{n}(n-1)p_{i2}}{n} \tag{23}$$

A uniform distribution function calculates the batches' values of delivery costs $[(£-1)\times 20,\ £\times 20]$ based on the delivery cost level. Also, the values β are $£ \in \{1,2,3\}$ generated randomly on [20, 50]. In this study, the termination criterion for all algorithms is to reach 200 iterations. Table III represents the results for the CPLEX solver based on the delivery cost level $£ = 1$. The CPLEX cannot solve instances with more than 26 jobs in reasonable running time (in this study, the reasonable time is equal to 3600 seconds). It also has inappropriate CPU running times for small instances. Table IV presents the results for solving 20 (5*2*2) different random small-size instances and compares BA, PSO, GA, and HBA results. In this table, for each instance, the metaheuristic algorithms are repeated ten times, and the Avg.Gap from the global optimum solution and Avg.CPU running time is presented. The below equation gives the average Gap:

$$Avg.\ Gap = \frac{1}{10}\sum_{i=1}^{10}\frac{solution_i - best\ \ solution}{best\ \ solution} \tag{24}$$

Here, the $solution_i$ shows the result of instance *i* from the metaheuristic algorithm, and the *best solution* indicates the result of the CPLEX solver. As shown in Table IV, all heuristic methods solve small-size problems with the Avg. Gap is less than 1%. Among the solution methods proposed here, the HBA model can reach solutions with the minimum Gap, although this algorithm spends more CPU running time than others. According to the findings, as the number of customers increases, the problems and then the values of Avg. become more difficult. CPU running time and Avg. Gap also increases. In contrast, while $£$ increased, the problem tends to deliver jobs with fewer batches, so the solution space becomes smaller. Consequently, the values of Avg. CPU running time and Avg. Gap will decrease. In fact, β and $£$ determine the significance of each part in the objective function. As $£$ increases (or β decreases), delivering those jobs with fewer batches and Avg's values would be more beneficial. CPU running time and Avg. Gap will decrease, and vice versa.

However, to evaluate the performance of proposed metaheuristic methods for large-size problems, 45 (5*3*3) different random instances are solved. In this experiment, the factors are considered according to Table V. Since for large-size problems, there is no global optimum solution, each metaheuristic algorithm will solve the problem, and the best solution provided by four algorithms will be selected as the best-known solution (BKS). In Eq. (24), the *best solution* is replaced by *BKS*. In addition, a useful metric - i.e., Marginal Improvement per CPU (MIC) index - is proposed to analyze quality of solutions. This index has been inspired by the MIC index introduced by Osman (2003). The MIC is given by Equation (25):

$$MIC = \frac{100}{Avg.\ Gap \times Avg.\ CPU\ running\ time} \tag{25}$$

The evaluation results of large-size problems are shown in Fig. (7). As seen, when the number of jobs increases, the HBA is more efficient than other algorithms. This result is correct also for the number of customers. According to the results, when the delivery cost level increases, the feasible solution space will become smaller, so the MIC index will increase.

Table III. Results for MILP model

Instance	*No. Jobs*	*No. Customer*	*CPU running time (Sec.)*	*Instance*	*No. Jobs*	*No. Customer*	*CPU running time (Sec.)*
1	6	2	16.8	7	18	3	283.8
		3	14.3			4	278.9
2	8	2	35.8	8	20	3	395.5
		3	32.1			4	391.3
3	10	2	63.8	9	22	3	680.1
		3	61			4	668.7
4	12	2	102.5	10	24	3	1091.8
		3	99.8			4	1085.2
5	14	2	155.2	11	26	3	2428.4
		3	150.6			4	2385.5
6	16	3	208.4	12	28	3	>3600
		4	201.5			4	>3600

Table IV. Comparison of BA, PSO, GA and HBA for small-size problems

Setting	*No. Jobs*	*No. Customer*	*Delivery cost level (£)*	*BA results*		*PSO results*		*GA results*		*HBA results*	
				Avg. Gap (%)	*Avg. CPU running time (Sec.)*	*Avg. Gap (%)*	*Avg. CPU running time (Sec.)*	*Avg. Gap (%)*	*Avg. CPU running time (Sec.)*	*Avg. Gap (%)*	*Avg. CPU running time (Sec.)*
1	6	2	1	0.10%	0.4	0.14%	0.5	0.09%	0.3	0.08%	0.5
2		2	2	0.10%	0.4	0.13%	0.4	0.08%	0.3	0.06%	0.4
3		3	1	0.12%	0.4	0.18%	0.5	0.12%	0.4	0.10%	0.6
4		3	2	0.12%	0.3	0.17%	0.4	0.10%	0.3	0.07%	0.5
5	8	2	1	0.41%	1.4	0.78%	1.7	0.36%	1.3	0.22%	1.6
6		2	2	0.39%	1.2	0.71%	1.5	0.30%	1.2	0.19%	1.5
7		3	1	0.69%	1.7	1.01%	1.9	0.55%	1.5	0.49%	1.9
8		3	2	0.67%	1.4	0.97%	1.8	0.52%	1.3	0.44%	1.8
9	10	2	1	1.03%	1.9	1.06%	2.1	0.90%	1.8	0.75%	2.2
10		2	2	0.97%	1.8	1.01%	2	0.85%	1.6	0.74%	2.2
11		3	1	1.09%	1.9	1.10%	2.3	0.98%	1.9	0.88%	2.3
12		3	2	1.02%	1.7	1.09%	2.2	0.91%	1.8	0.85%	2.1
13	12	2	1	1.15%	4.3	1.16%	4.7	1.11%	4.1	1.07%	4.8
14		2	2	1.14%	4.2	1.13%	4.6	1.11%	4	1.06%	4.6
15		3	1	1.17%	4.4	1.19%	4.9	1.13%	4.2	1.12%	5.1
16		3	2	1.14%	4.2	1.20%	4.6	1.12%	4.2	1.10%	5
17	14	2	1	1.51%	6.8	1.67%	7.4	1.53%	6.1	1.42%	7.3
18		2	2	1.41%	6.5	1.65%	7	1.50%	6.2	1.44%	7.1
19		3	1	1.58%	7.1	1.69%	7.6	1.62%	6.5	1.55%	7.6
20		3	2	1.58%	6.9	1.66%	7.5	1.61%	6.2	1.49%	7.5
Average				0.87%	2.95	0.99%	3.28	0.82%	2.76	0.76%	3.33

Table V. The level of factors

Factors	Level
Number of jobs	20, 40, 60, 80, 100
Number of customers	2, 3, 4
Delivery cost level (£)	1, 2, 3

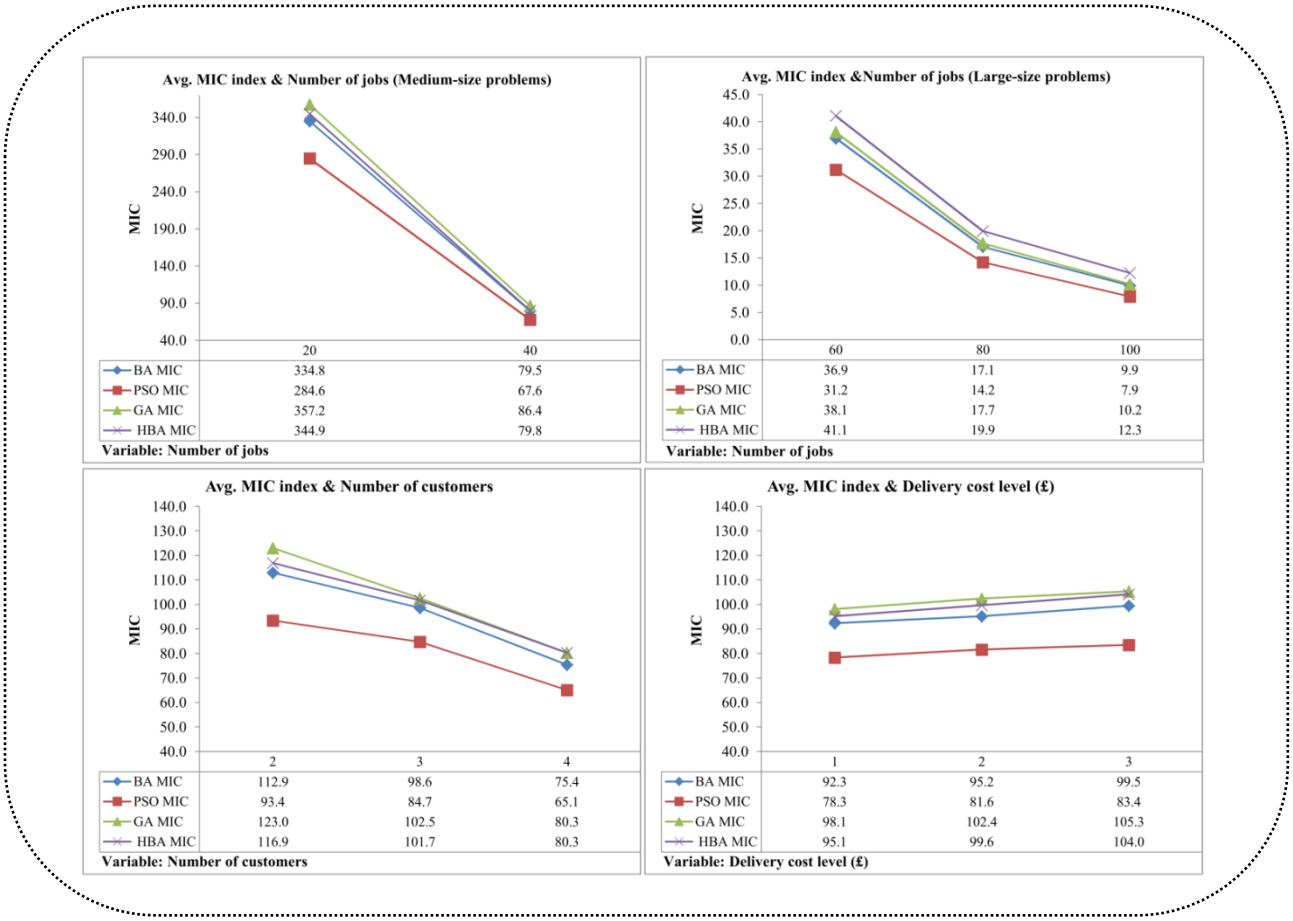

Fig. 7. Avg. MIC index for medium and large-size instances

C. Ranking the methods

In the previous section, various instances with different variable settings are solved by the proposed metaheuristic algorithms. Here, using the Friedman and Wilcoxon signed-ranks test, a guide table is presented for ranking the methods. Initially, normality tests with the help of two popular methods are performed, including Kolmogorov-Smirnova and Shapiro-Wilk. The results show the significant deviation of the data from the normal distribution (see Table VI).

For ranking, the problem is categorized into three groups; namely, small-size (with 10 and 20 jobs), medium-size (with 40 and 60 jobs), and large-size (with 80 and 100 jobs). In each group, 18 (2*3*3) different random instances are solved by the methods, and the results of the MIC index are reported. Table XIII in Appendix B shows these results. Table VII provides the results of the Friedman test for each group.

Table VI. Normality tests result

Methods	*Kolmogorov-Smirnov*		*Shapiro-Wilk*	
	Statistic	*Sig.*	*Statistic*	*Sig.*
BA	.435	.000	.517	.000
PSO	.439	.000	.521	.000
GA	.440	.000	.516	.000
HBA	.431	.000	.521	.000

Table VII. Friedman test result

Class	*Method*	*Mean Rank*	*Chi-Square*	*df*	*Asymp. Sig.*
Small-size	BA	2.72	48.200	3	.000
	PSO	4.00			
	GA	1.06			
	HBA	2.22			
Medium-size	BA	2.78	42.467	3	.000
	PSO	4.00			
	GA	1.50			
	HBA	1.72			
Large-size	BA	3.00	54.000	3	.000
	PSO	4.00			
	GA	2.00			
	HBA	1.00			

The null hypothesis of the Friedman test is rejected in all groups. For this reason, the pairwise comparisons seem necessary to be done. In this paper, this comparison is performed by using the Wilcoxon signed-ranks test. Tables VIII to X show the Wilcoxon signed-ranks test results for small, medium, and large-size problems, respectively.

Table VIII. Result of Wilcoxon signed ranks test for small-size problems

Pairwise comparisons	*PSO - BA*	*GA - BA*	*HBA - BA*	*GA - PSO*	*HBA - PSO*	*HBA - GA*
Z	-3.906[a]	-3.874[b]	-1.964[b]	-4.146[b]	-3.866[b]	-3.586[a]
Asymp. Sig. (2-tailed)	.000	.000	.050	.000	.000	.000
a. Based on negative ranks. b. Based on positive ranks.						

Table IX. Result of Wilcoxon signed ranks test for medium-size problems

Pairwise comparisons	*PSO - BA*	*GA - BA*	*HBA - BA*	*GA - PSO*	*HBA - PSO*	*HBA - GA*
Z	-3.947[a]	-3.906[b]	-2.937[b]	-3.834[b]	-3.792[b]	-.831[a]
Asymp. Sig. (2-tailed)	.000	.000	.003	.000	.000	.406
a. Based on negative ranks. b. Based on positive ranks.						

Table X. Result of Wilcoxon signed ranks test for large-size problems

Pairwise comparisons	*PSO - BA*	*GA - BA*	*HBA - BA*	*GA - PSO*	*HBA - PSO*	*HBA - GA*
Z	-4.243[a]	-4.243[b]	-4.243[b]	-4.243[b]	-4.243[b]	-4.243[b]
Asymp. Sig. (2-tailed)	.000	.000	.000	.000	.000	.000
a. Based on negative ranks. b. Based on positive ranks.						

The results show the GA has an upper MIC than other metaheuristic methods for small-size problems. Also, in these problems, the PSO finds the worst MICs among all proposed metaheuristic methods. However, no significant difference can be observed between HBA and BA. For medium-size problems, there is no significant difference between HBA and GA; and the BA has an upper MIC than PSO. Finally, for large-size problems, the HBA shows the best MICs among all proposed metaheuristic methods. To summarize these results, Table XI guides the decision-maker to select the appropriate method concerning the size of problems.

Table XI. Guidance for selecting the best algorithm

Class	*Ranking methods*
Small-size problems	1st. **GA**
	2nd. **HBA** and **BA** (No preference)
	3rd. **PSO**
Medium-size problems	1st. **GA** and **HBA** (No preference)
	2nd. **BA**
	3rd. **PSO**
Large-size problems	1st. **HBA**
	2nd. **GA**
	3rd. **BA**
	4th. **PSO**

V. CONCLUSION

Since the sequence of jobs has a significant effect on the costs related to customer satisfaction, scheduling, and distribution problems are of great importance. Without considering transportation costs, it will be a weak strategy to decide how to schedule jobs. Hence, a successful approach needs dealing with both aspects at the same time. The current paper proposed a two-machine flow-shop scheduling problem in a batch delivery system to minimize the number of tardy jobs and the delivery costs. The problem was proved as NP-hard. First, a MILP model was introduced that can solve small instances by specific software. Additionally, as the number of jobs and problem constraints increases, the solvers' ability may decrease. A novel hybrid bee algorithm (HBA) was then developed to solve the large-size problems in reasonable CPU running time. Three metaheuristics were provided for evaluating the HBA performance, including a genetic algorithm, bee algorithm, and particle swarm optimization. So, 65 different instances were randomly generated and run by the algorithms into two groups of small-size and large-size. The results revealed that in small-size problems, the GA shows the highest efficiency, while the HBA can provide the best performance for large-size problems. This approach can be enhanced by using a wide range of techniques to search the solution space under different situations when space is more extensive. According to the results, the HBA has more capacity to search

solution spaces to offer acceptable solutions for even smaller population sizes. The reason is that the HBA employs a neighborhood search process to find solutions and to prevent trapping at the local optimum. Due to the HBA algorithm's acceptable MIC index compared to other presented methods, this algorithm is recommended for the systems that perform their production and distribution process daily and need accurate solutions.

Future researchers are recommended to study the same problem using exact methods, such as dynamic programming or branch-and-bound algorithms. These methods are attractive for researchers, and the industries achieving the optimal solution are critical, such as the aerospace industry. Furthermore, vehicles with a limited capacity can be appealing. The application of different delivery costs appropriate with batching volumes is also suggested for further study.

REFERENCES

Ahmadizar, F., & Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers & Operations Research, 53*, 194-205.

Akbari, R., Mohammadi, M., & Ziarati, K. (2010). A novel bee swarm optimization algorithm for numerical function optimization. *Journal of Communications in Nonlinear Science and Numerical Simulation, 15*, 3142–3155.

Assarzadegan, P., & Rasti-Barzoki, M. (2016). Minimizing sum of the due date assignment costs, maximum tardiness and distribution costs in a supply chain scheduling problem. *Applied Soft Computing, 47*, 343-356.

Basir, S. A., Mazdeh, M. M., & Namakshenas, M. (2018). Bi-level genetic algorithms for a two-stage assembly flow-shop scheduling problem with batch delivery system. *Computers & Industrial Engineering, 126*, 217-231.

Bulfin, R. L., & M'Hallah, R. (2003). Minimizing the weighted number of tardy jobs on a two-machine ow shop. *Computers & Operations Research, 30*, 1887–1900.

Chamnanlor, C., Sethanan, K., Chien, C.-F., & Gen, M. (2014). Re-entrant flow shop scheduling problem with time windows using hybrid genetic algorithm based on auto-tuning strategy. *International Journal of Production Research, 52*(9), 2612-2629.

Chen, D., Batson, R., & Dang, Y. (2011). Applied integer programming: modeling and solution. 2011. In: John Wiley & Sons.

Cheng, B. Y., Leung, J. T., Li, K., & Yang, S. L. (2015). Single batch machine scheduling with deliveries. *Naval Research Logistics (NRL), 62*(6), 470-482.

Cheng, T. C. E., Gordon, V. S., & Kovalyov, M. Y. (1996). Single machine scheduling with batch deliveries. *European Journal of Operational Research, 94*(2), 277-283.

Chun-Feng, W., Kui, L., & Pei-Ping, S. (2014). Hybrid artificial bee colony algorithm and particle swarm search for global optimization. *Mathematical Problems in Engineering, 2014*.

Ding, J.-Y., Song, S., Gupta, J. N., Zhang, R., Chiong, R., & Wu, C. (2015). An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing, 30*, 604-613.

Duan, H.-b., Xu, C.-f., & Xing, Z.-h. (2010). A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems. *International Journal of Neural Systems, 20*(01), 39-50.

Ganji, M., Kazemipoor, H., Molana, S. M. H., & Sajadi, S. M. (2020). A green multi-objective integrated scheduling of production and distribution with heterogeneous fleet vehicle routing and time windows. *Journal of Cleaner Production*, 120824.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics*, 287-326.

Guanlong, D., Zhenhao, X., & Xingsheng, G. (2012). A discrete artificial bee colony algorithm for minimizing the total flow time in the blocking flow shop scheduling. *Chinese Journal of Chemical Engineering*, *20*(6), 1067-1073.

Gupta, J., & Hariri, A. (1997). Two machine ow shop to minimize number of tardy jobs. *Journal of the Operational Research Society*, *48*, 212–220.

Gupta, J. N., & Stafford Jr, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, *169*(3), 699-711.

Hall, N., & Potts, C. (2005). The coordination of scheduling and batch deliveries. *Annals of Operations Research*, *135*(1), 41-64.

Hall, N. G., & Potts, C. N. (2003). Supply chain scheduling: Batching and delivery. *Operations Research*, *51*(4), 566-584.

Hamidinia, A., Khakabimamaghani, S., Mahdavi Mazdeh, M., & Jafari, M. (2011). A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers & Industrial Engineering*, doi:10.1016/j.cie.2011.1008.1014.

Han, Y.-Y., Gong, D., & Sun, X. (2015). A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. *Engineering Optimization*, *47*(7), 927-946.

Hariri, A., & Potts, C. (1989). A branch and bound algorithm to minimize number of late jobs in a permutation ow shop. *European Journal of Operational Research*, *38*, 228–237.

Herrmann, J. W., & Lee, C.-Y. (1993). On scheduling to minimize earliness-tardiness and batch delivery costs with a common due date *European Journal of Operational Research*, *70*(3), 272-288.

Ji, M., He, Y., & Cheng, T. C. E. (2007). Batch delivery scheduling with batch delivery cost on a single machine. *European Journal of Operational Research*, *176*, 745–755.

Jia, Z.-h., Zhuo, X.-x., Leung, J. Y., & Li, K. (2019). Integrated production and transportation on parallel batch machines to minimize total weighted delivery time. *Computers & Operations Research*, *102*, 39-51.

Johnson, S. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*(1), 61-68.

Jolai, F., Asefi, H., Rabiee, M., & Ramezani, P. (2013). Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Scientia Iranica*, *20*(3), 861-872.

Joo, C. M., & Kim, B. S. (2019). Variable Neighborhood Search Algorithms for an Integrated Manufacturing and Batch Delivery Scheduling Minimizing Total Tardiness. *Applied Sciences*, *9*(21), 4702.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization.* Erciyes University.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21-57.

Karaboga, D., & Kaya, E. (2016). An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training. *Applied Soft Computing*, *49*, 423-436.

Kazemi, H., Mazdeh, M. M., & Rostami, M. (2017). The two stage assembly flow-shop scheduling problem with

batching and delivery. *Engineering Applications of Artificial Intelligence, 63*, 98-107.

Kefayat, M., Ara, A. L., & Niaki, S. N. (2015). A hybrid of ant colony optimization and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources. *Energy Conversion and Management, 92*, 149-161.

Kennedy, J., & Eberhart, R. C. (1995). *Particle swarm optimization.* Paper presented at the IEEE International Conference, Neural Networks.

Kennedy, J., & Eberhart, R. C. (1997). *A discrete binary version of the particle swarm algorithm.* Paper presented at the the World Multiconference on Systemics, NJ.

Lee, C.-Y., & Chen, Z.-L. (2001). Machine scheduling with transportation considerations. *JOURNAL OF SCHEDULING, 4*, 3-24.

Lenstra, J. K., Karl, A. H. G. R., & Brucker, P. (1977). Complexity of machine scheduling problems. *Ann. of Discrete Math*, 343-351.

Li, J.-q., Xie, S.-x., Pan, Q.-k., & Wang, S. (2011). A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. *International Journal of Computers Communications & Control, 6*(2), 286-296.

Low, C., Hsu, C.-J., & Su, C.-T. (2010). A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance. *Expert Systems with Applications, 37*, 6429–6434.

Mahadevan, B. (2015). *Operations management: Theory and practice*: Pearson Education India.

Marichelvam, M., Prabaharan, T., & Yang, X.-S. (2014a). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Applied Soft Computing, 19*, 93-101.

Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014b). A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. *IEEE transactions on evolutionary computation, 18*(2), 301-305.

Marinakis, Y., Marinaki, M., & Matsatsinis, N. (2009). *A hybrid discrete artificial bee colony-GRASP algorithm for clustering.* Paper presented at the Computers & Industrial Engineering, 2009. CIE 2009. International Conference on.

Mazdeh, M. M., & Rostami, M. (2014). A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs. *International Journal of Systems Science: Operations & Logistics, 1*(2), 94-104.

Mazdeh, M. M., Rostami, M., & Namaki, M. H. (2013). Minimizing maximum tardiness and delivery costs in a batched delivery system. *Computers & Industrial Engineering, 66*, 675–682.

Mazdeh, M. M., Sarhadi, M., & Hindi, K. S. (2007). A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. *European Journal of Operational Research, 183*, 74–86.

Mazdeh, M. M., Sarhadi, M., & Hindi, K. S. (2008). A branch-and-bound algorithm for single-machine scheduling with batch delivery and job release times. *Computers & Operations Research, 35*, 1099–1111.

Mazdeh, M. M., Shashaani, S., Ashouri, A., & Hindi, K. S. (2011). Single-machine batch scheduling minimizing weighted flow times and delivery costs. *Applied Mathematical Modelling, 35*, 563–570.

Moore, J. M. (1968). Sequencing n jobs on one machine to minimize the number of late jobs. *Management Science, 15*(1), 102–109.

Moumene, K., & Ferland, J. A. (2009). Activity list representation for a generalization of the resource-constrained

project scheduling problem. *European Journal of Operational Research, 199*(1), 46-54.

Osman, I. (2003). *Meta-heuristics: Models, analysis, and directions.* Paper presented at the A tutorial paper presented at the joint EURO/INFORMS Meeting, Istanbul, July.

Panwalkar, S., Smith, M. L., & Koulamas, C. (2013). Review of the ordered and proportionate flow shop scheduling research. *Naval Research Logistics (NRL), 60*(1), 46-55.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2005). *The Bees Algorithm. Technical Note.* Cardiff University.

Preux, P., & Talbi, E. G. (1999). Towards hybrid evolutionary algorithms. *International Transactions in Operational Research, 6*(6), 557-570. doi:10.1111/j.1475-3995.1999.tb00173.x.

Pundoor, G., & Chen, Z.-L. (2005). Scheduling a Production–Distribution System To Optimize the Tradeoff between Delivery Tardiness and Distribution Cost. *Wiley InterScience*.

Rasti-Barzoki, M., & Hejazi, S. R. (2013). Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries for multiple customers in supply chains. *European Journal of Operational Research, 228*(2), 345-357. doi:http://dx.doi.org/10.1016/j.ejor.2013.01.002.

Rasti-Barzoki, M., Hejazi, S. R., & Mazdeh, M. M. (2013). A branch and bound algorithm to minimize the total weighed number of tardy jobs and delivery costs. *Applied Mathematical Modelling, 37*(7), 4924-4937.

Riahi, V., & Kazemi, M. (2016). A new hybrid ant colony algorithm for scheduling of no-wait flowshop. *Operational Research*, 1-20.

Rostami, M., Kheirandish, O., & Ansari, N. (2015). Minimizing maximum tardiness and delivery costs with batch delivery and job release times. *Applied Mathematical Modelling, 39*(16), 4909-4927.

Rostami, M., Nikravesh, S., & Shahin, M. (2018). Minimizing total weighted completion and batch delivery times with machine deterioration and learning effect: a case study from wax production. *Operational Research*. doi:10.1007/s12351-018-0373-6.

Shabtay, D. (2010). Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs. *Int. J. Production Economics, 123*, 235–242.

Soukhal, A., Oulamara, A., & Martineau, P. (2005). Complexity of flow shop scheduling problems with transportation constraints. *European Journal of Operational Research, 161*, 32-41.

Steiner, G., & Zhang, R. (2011). Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries. *Annals of Operations Research, 191*(1), 171-181.

Tasgetiren, M. F., Pan, Q.-K., Suganthan, P. N., & Chen, A. H. (2011). A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Information Sciences, 181*(16), 3459-3475.

Thakare, A. D., & Chaudhari, M. S. M. (2012). Introducing a Hybrid Swarm Intelligence Based Technique for Document Clustering. *International Journal of Engineering Research and Applications*, 1455-1459.

Wang, K., Luo, H., Liu, F., & Yue, X. (2017). Permutation flow shop scheduling with batch delivery to multiple customers in supply chains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48*(10), 1826-1837.

Wu, B., Qian, C., Ni, W., & Fan, S. (2012). Hybrid harmony search and artificial bee colony algorithm for global optimization problems. *Computers & Mathematics with Applications, 64*(8), 2621-2634.

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

Yildiz, A. R. (2013). A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing. *Applied Soft Computing, 13*(5), 2906-2912.

Yin, Y., Cheng, T., Hsu, C.-J., & Wu, C.-C. (2013). Single-machine batch delivery scheduling with an assignable common due window. *Omega, 41*(2), 216-225.

Zhao, H., Pei, Z., Jiang, J., Guan, R., Wang, C., & Shi, X. (2010). A hybrid swarm intelligent method based on genetic algorithm and artificial bee colony. *Advances in Swarm Intelligence*, 558-565.

Appendix A

Table XII. Detailed S/N ratios of Taguchi method

Descriptions	***S/N ratio***		
BA parameters	***Level 1***	***Level 2***	***Level 3***
scout bees (A)	-26.16	-25.87	-28.64
best selected bee and their recruits (B)	-25.92	-25.75	-24.96
N_p (C)	-26.28	-25.97	-29.79
PSO parameters			
c_1 (A)	-22.98	-16.78	-26.34
c_2 (B)	-24.08	-19.62	-23.06
$\omega_{\min}$ (C)	-23.13	-22.13	-24.36
$\omega_{\max}$ (D)	-26.57	-23.64	-25.11
N_p (E)	-23.62	-21.67	-20.06
GA parameters			
P_c (A)	-32.57	-29.92	-27.86
P_m (B)	-25.24	-27.21	-34.64
N_p (C)	-34.57	-24.82	-22.61
HBA parameters			
c_1 (A)	-16.84	-22.16	-18.85
c_2 (B)	-14.88	-13.51	-20.00
$\omega_{\min}$ (C)	-21.39	-20.46	-22.60
$\omega_{\max}$ (D)	-21.95	-22.79	-18.47
ne (E)	-15.76	-14.49	-15.56
nm (F)	-22.83	-18.93	-21.62
nt (G)	-18.31	-20.01	-21.94
N_s (H)	-19.98	-18.83	-15.14
N_p (I)	-18.63	-19.11	-19.50

Appendix B

Table XIII. Avg. MIC index results for 54 small, medium and large-size instances

Class	*No. Jobs*	*No. Customer*	*Delivery cost level (£)*	*Avg. MIC index*			
				BA	*PSO*	*GA*	*HBA*
Small-size	10	2	1	5109.9	4492.4	6172.8	6060.6
	10	2	2	5727.4	4950.5	7352.9	6142.5
	10	2	3	6038.6	5482.5	7002.8	6613.8
	10	3	1	4828.6	3952.6	5370.6	4940.7
	10	3	2	5767.0	4170.1	6105.0	4902.0
	10	3	3	5711.0	4672.9	7102.3	5681.8
	10	4	1	3852.1	3255.2	4290.0	3467.4
	10	4	2	3935.5	3871.5	4965.2	4048.6
	10	4	3	4464.3	3819.7	5291.0	4091.7
	20	2	1	393.7	321.9	426.2	387.0
	20	2	2	408.9	337.4	451.8	423.2
	20	2	3	432.2	352.7	472.8	456.1
	20	3	1	335.0	291.1	342.6	340.2
	20	3	2	348.1	303.3	364.5	354.7
	20	3	3	368.6	311.8	373.6	374.9
	20	4	1	230.2	205.7	253.2	247.6
	20	4	2	240.9	221.6	261.4	257.2
	20	4	3	255.8	215.8	268.6	263.1
Medium-size	40	2	1	82.9	71.3	90.1	82.9
	40	2	2	83.2	73.2	95.1	84.9
	40	2	3	86.2	75.3	96.0	87.2
	40	3	1	78.8	66.8	85.0	77.5
	40	3	2	79.6	67.6	86.4	78.8
	40	3	3	79.9	70.1	88.8	79.6
	40	4	1	73.6	60.4	77.5	74.4
	40	4	2	75.0	60.9	78.6	75.8
	40	4	3	76.1	62.4	80.2	77.3
	60	2	1	40.8	33.1	41.9	43.6
	60	2	2	41.2	33.9	42.3	44.0
	60	2	3	41.5	34.5	42.2	44.8
	60	3	1	35.5	30.0	37.0	40.5
	60	3	2	35.9	30.6	37.4	40.9

Continue Table XIII. Avg. MIC index results for 54 small, medium and large-size instances

Class	*No. Jobs*	*No. Customer*	*Delivery cost level (£)*	*Avg. MIC index*			
				BA	*PSO*	*GA*	*HBA*
Medium-size	60	3	3	36.4	33.5	38.2	42.0
	60	4	1	33.7	27.9	34.3	37.4
	60	4	2	33.8	28.5	34.5	38.0
	60	4	3	33.8	28.7	35.2	38.3
Large-size	80	2	1	17.4	14.5	18.3	20.1
	80	2	2	17.5	14.6	18.4	20.7
	80	2	3	17.6	14.8	18.5	20.8
	80	3	1	17.1	14.1	17.8	19.6
	80	3	2	17.2	14.3	17.9	19.8
	80	3	3	17.2	14.2	17.8	19.9
	80	4	1	16.5	13.7	16.8	19.4
	80	4	2	16.5	13.8	16.9	19.6
	80	4	3	16.6	13.9	17.0	19.7
	100	2	1	10.2	8.0	10.4	12.4
	100	2	2	10.3	8.1	10.5	12.5
	100	2	3	10.3	8.1	10.5	12.7
	100	3	1	9.9	7.9	10.2	12.2
	100	3	2	10.0	7.9	10.2	12.2
	100	3	3	10.1	8.0	10.3	12.3
	100	4	1	9.5	7.7	9.9	11.9
	100	4	2	9.6	7.8	9.9	12.0
	100	4	3	9.6	7.8	10.0	12.0