Mathematical models and an elephant herding optimization for multiprocessor-task flexible flow shop scheduling problems in the manufacturing resource planning (MRPII) system

Habib Reza Gholami\textsuperscript{a}, Esmaiel Mehdizadeh\textsuperscript{a,}\textsuperscript{*}, Bahman Naderi\textsuperscript{b}

\textsuperscript{a}Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
\textsuperscript{b}Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran

Abstract

Shop floor control (SFC) is one of the main concepts in manufacturing resource planning (MRPII) and production scheduling is a key element in SFC. This paper studies the hybrid flow shop scheduling problem where jobs are multiprocessor. The objective is to minimize total completion time. Although there are several papers considering hybrid flow-shop scheduling problem with multiprocessor tasks, but none propose a mathematical model for this problem. At first, the two problems (fixed and selective cases) are mathematically formulated by mixed integer linear programming models. Using commercial software, the model is used to solve the small instances of the problems. Moreover, an elephant herding optimization is developed to solve large instances of the problems. To numerically evaluate the proposed algorithm, it is compared with three available algorithms in the literature.

Keywords: Hybrid flow shop; Multiprocessor jobs, Mathematical modeling, Elephant herding optimization, manufacturing resource planning (MRPII).

\textsuperscript{*}Corresponding author mobile number: +989122812562
Email addresses: hr_ghlamy@yahoo.com (H. R. Gholami); emehdi@qiau.ac.ir (E. Mehdizadeh); bahman.naderi@aut.ac.ir (B. Naderi).
1. Introduction

Production is one of the most basic and important function of human activities in modern industrial societies, and production planning is selection of the future course of production actions. Manufacturing resource planning (MRPII) is a method for the effective planning of all resources of a manufacturing company. It is made up of a variety of interlinked functions. One of the main concepts in MRPII is shop floor control (SFC). The execution of the manufacturing portion of the planning order release schedule whereby factory orders are released, monitored and reported is called SFC. The SFC consists of factory coordination (FC) and production activity control (PAC). In PAC, more detailed planning is performed. At this point, the top-level plans of the company have been broken down into specific tasks. In firms using MRPII systems, execution of the detailed material and capacity plans involves the scheduling of machine and other work centers. Scheduling is a PAC functions in which we are concerned with ensuring the right tasks are conducted at the right times to produce the output. In scheduling we are looking to specify the start and finish time of the activities [1]. Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives and plays an important role in MRPII system [2]. Flow shop is one of the important problems in scheduling. For nearly several years, flow shop (FS) scheduling problems have been studied as an important subject in manufacturing researches. Some novel useful definitions and properties about flow shop scheduling problem were established by Aminnayeri and Naderi [3]. In the classical flow-shop problem, there are m stages in series with only one machine at each one [4]. In addition, there are n jobs each of which must be processed by the single machine at all the m stages, starting from stage 1, then stage 2 and till stage m [5].

There are some gaps between academic and practical aspects of this problem. One of these gaps in the flow shop is to assume one machine at each stage. While companies likely employ more than one machine at stages with more workload. In this case, they can reduce the impact of bottleneck stages or even more balance their production capacity. This generalized version of the problem is called hybrid flow shops (HFS). The hybrid flow shop has been well studied in the literature [6-8].
The hybrid flow shop classically assume that each job requires only one machine for each operation. Hence, it is assigned to only one machine, among machines at each stage. While another extension inspired from the realistic applications is assuming multiprocessor tasks. That is, an operation of a job may simultaneously need more than one machine for its process. A subset of machines among the machines available at that stage is required. In scheduling problems with multiprocessor tasks, the subset of machines required for processing of a job at a stage can be either a fixed or selective. If the subset is fixed, the machines are already determined due to technical limitations or machine eligibilities. The process of a job at a stage is started as soon as all the necessary machines are available. If the subset is selective, the machines are not predetermined and the selection is a decision to make. As soon as the required number of machines for processing of a job at a stage is idle, the process of that job can be started.

The applications of scheduling problems with multiprocessor tasks can be seen in some systems. For example, in semiconductor circuit design workforce planning [9], a design project is to be processed by \( m \) persons (a team of people). Other applications can be found in (i) the berth allocation problems, where a large vessel may occupy several berths for loading and/or unloading, (ii) diagnosable microprocessor systems, where a job must be performed on parallel processors in order to detect faults, (iii) manufacturing, where a job may need machines, tools, and people simultaneously, and (iv) scheduling a sequence of meetings where each meeting requires a certain group of people. In all the above examples, one job may need to be processed by several machines simultaneously [10-12].

Looking into the literature of multiprocessor tasks, the following papers can be found: Chen and Lee [10] considered the problem of parallel machines with multiprocessor tasks where the subset of machines is selective. They proposed a pseudo polynomial algorithm for two- and three-machine problems. Jansen and Porkolab [13] study the same problem, yet, the processing time of a job is a function of the subset selected. They proposed a fully polynomial time approximation scheme for the preemptive version of the problem and a polynomial time approximation scheme for its non-preemptive one. Many studies have applied heuristic and meta-heuristic algorithms to solve scheduling problems [14]. Since the problem of hybrid flow shops with multiprocessor tasks (HFS-
MT) is strongly NP-hard, the most effective algorithms of this problem are metaheuristics. Serifoğlu and Ulusoy [15] proposed a genetic algorithm for the problem. Oğuz et al. [16] and Oğuz and Ercan [17] investigated HFS-MT where the subset of machines required for processing of a job at a stage is selective and develop a tabu search and a genetic algorithm, respectively. Later, Ying and Lin [18] proposed an ant colony optimization algorithm. They compared the proposed algorithm with two available algorithms of genetic algorithm [17] and tabu search [16]. Next, Tseng and Liao [19] developed a particle swarm optimization and compared it with ant colony optimization by [18] for performance. Kahraman et al. [20] proposed a parallel greedy algorithm and show that this algorithm outperforms genetic algorithm [17] and tabu search [16]. Engin et al. [21] proposed another genetic algorithm shown to be better than the parallel greedy algorithm by [20]. Wang et al. [22] proposed a simulated annealing that shows high performance in the numerical experiments. Recently, Xu et al. [23] developed a shuffled frog-leaping algorithm and compare it with genetic algorithm by [15], ant colony optimization by [18], and particle swarm optimization by [19] and simulated annealing by [22]. Lahimer et al. [24] developed a limited discrepancy search heuristic to solve the problem and lower bound for the problem. An enhanced invasive weed optimization (EIWO) meta-heuristic algorithm presented in order to solve flexible flow shop scheduling problem with probable rework times, transportation times with a conveyor between two subsequent stages, different ready times and anticipatory sequence dependent setup times [25]. Very recently, Hidri et al. [26] developed another lower bound for HFS-MT.

Although there are several papers considering hybrid flow-shop scheduling problem with multiprocessor tasks, but none of them propose a mathematical model for this particular problem. The hybrid flow-shop is already formulated by Ziaeifar [27] by a non-linear model, and linearly by Behnamian and Fatemi Ghomi [28]. But they both fail to consider multiprocessor tasks. This paper studies the hybrid flow shop scheduling problem with multiprocessor tasks. Both cases of fixed and selective subsets of machines are considered. At first, the two problems are mathematically formulated by mixed integer linear programming models. Using commercial software, the model is used to solve the small instances of the problems. Moreover, an effective elephant herding optimization (EHO) is developed to solve large instances of the problems. Elephant
herding optimization is a population-based algorithm for continuous search space [29]. To numerically evaluate the proposed algorithm, it is compared with three available algorithms in the literature, simulated annealing by [22], shuffled frog-leaping algorithm (SFLA) by [23], and limited discrepancy search (LDS) by [24].

The rest of the paper is arranged as follows. Section 2 presents the mathematical models of the problems under consideration. Section 3 develops an elephant herding optimization. Section 4 numerically evaluates the proposed model and algorithms. Section 5 concludes the paper and introduces some future research directions.

2. Problem definition and formulation

The problem of scheduling hybrid flow shops with multiprocessor tasks can be described as follows. There is a set of $n$ jobs in form of multiprocessor tasks. Each job needs $m$ operations for completion. For each operation, there is one work stage and in each stage $i$, there is a set of $m_i$ machines. The process of a job at a stage requires $d_{j,i}$ machines simultaneously. The $d_{j,i}$ machines can be either fixed (say Case 1) or selective (say Case 2). The objective is to schedule jobs at each stage so as to minimize makespan, maximum completion time of jobs.

For each of two cases, one mixed integer linear programming model is developed. Before presenting the models, the following parameters and indexes are established.

Parameters and indices:

- $n$: The number of jobs
- $m$: The number of stages
- $j, k$: Indices for jobs where $\{1, 2, \ldots, n\}$
- $i$: Indices for stages where $\{1, 2, \ldots, m\}$
- $m_i$: The number of machines in stage $i$
- $l$: Indices for machines at stage $i$ where $\{1, 2, \ldots, m_i\}$
- $p_{j,i}$: Processing time of job $j$ at stage $i$
- $D_{j,i}$: The subset of machines required to process job $j$ at stage $i$ ($|D_{j,i}| = d_{j,i}$)
- $M$: A large positive number
Case 1. Multiprocessor tasks with fixed subset of machines

The following decision variables are defined for the model.

Variables:

\[ Y_{j,i,k} \] Binary variable taking value 1 if job \( k \) proceeds job \( j \) at stage \( i \), and 0 otherwise (where \( k > j \) and \( D_{j,i} \cap D_{k,i} \neq \emptyset \)) \hspace{1cm} (1)

\[ F_{j,i} \] Continuous variable for the completion time of job \( j \) at stage \( i \) \hspace{1cm} (2)

The problem can be formulated as follows.

\[
\text{Min } \sum_{j=1}^{n} F_{j,m} \\
\text{Subject to:}
\]

\[ F_{j,1} \geq p_{j,1} \quad \forall j \hspace{1cm} (3) \]
\[ F_{j,i} \geq F_{j,i-1} + p_{j,i} \quad \forall_{j,i>1} \hspace{1cm} (4) \]
\[ F_{j,i} - F_{k,i} + M \cdot (1 - Y_{j,i,k}) \geq p_{j,i} \quad \forall_{i,(j,k)|D_{j,i} \cap D_{k,i} \neq \emptyset} \hspace{1cm} (5) \]
\[ F_{k,i} - F_{j,i} + M \cdot (Y_{j,i,k}) \geq p_{k,i} \quad \forall_{i,(j,k)|D_{j,i} \cap D_{k,i} \neq \emptyset} \hspace{1cm} (6) \]
\[ F_{j,i} \geq 0 \quad \forall_{j,i} \hspace{1cm} (7) \]
\[ Y_{j,i,k} \in \{0, 1\} \quad \forall_{i,(j,k)|D_{j,i} \cap D_{k,i} \neq \emptyset} \hspace{1cm} (8) \]

Constraint set (3) assures that least possible completion time of each job the first stage. Constraint set (4) specifies that each job can be processed at most at one stage at a time. Constraint sets (5) and (6) are a pair of constraints to ensure each machine processes at most one job at a time. Finally, constraint sets (7) and (8) define the decision variables.

Case 2. Multiprocessor tasks with selective subset of machines

The following decision variables are defined for the model.

Variables:

Continuous variables in (2)

\[ Y_{j,i,k} \] Binary variable taking value 1 if job \( k \) proceeds job \( j \) at stage \( i \), and 0 \hspace{1cm} (9)
otherwise (where \( k > j \))

\[ Z_{j,i,l} \] Binary variable taking value 1 if job \( j \) is processed by machine \( l \) at stage \( i \), and 0 otherwise.

The problem can be formulated as follows.

Min \( \sum_{j=1}^{n} F_{j,m} \)

Subject to:

Constraint sets (3), (4) and (7)

\[ \sum_{l=1}^{m_i} Z_{j,i,l} = d_{j,i} \quad \forall j,i \] (11)

\[ F_{j,i} \geq F_{k,i} + p_{j,i} - M \cdot (3 - Y_{j,i,k} - Z_{j,i,l} - Z_{k,i,l}) \quad \forall j < k, i,l \] (12)

\[ F_{k,i} \geq F_{j,i} + p_{k,i} - M \cdot (2 - Y_{j,i,k} - Z_{j,i,l} - Z_{k,i,l}) \quad \forall j < k, i,l \] (13)

\[ Y_{j,i,k} \in \{0, 1\} \quad \forall j < k,i \] (14)

\[ Z_{j,i,l} \in \{0, 1\} \quad \forall j,i,l \] (15)

Constraint set (11) is to select machines to process each operation. Constraint sets (12) and (13) assure that a machine carries out at most one operation at a time. Constraint sets (14) and (15) define the decision variables.

4. Hybrid elephant herding optimization

Elephant Herding Optimization (EHO) is a novel swarm-based meta-heuristic algorithm that was inspired by the herding behavior of elephant group. This algorithm was proposed by Wang et al. [29], for solving continuous non-linear problems. In nature, elephants are social animals, and they live in several clans under the leadership of a matriarch, often the oldest cow, and the male elephants will leave their family group when they grow up. These two behaviors in EHO modeled into two phases: clan updating phase and separating phase.

As mentioned, EHO is developed to solve continuous function. But, TSAFSP is a combinatorial problem where its solution space is discrete. Therefore, this paper develops the discrete variant of EHO algorithm. This is done by defining a proper encoding scheme, crossing operators and moving operator in updating and separating...
phases. To further enhance EHO, a local search, based on simulated annealing, is applied. The stopping criterion is computational time of n (the number of jobs). In this case, more computational time is given to the algorithm to solve larger sizes. Figure 1 shows general outline of the proposed hybrid EHO.

{Please Insert Figure 1 here}

4.1. Encoding scheme and initialization

The first step in meta-heuristics is to determine the encoding scheme to make a solution recognizable for algorithms. The commonly used encoding scheme for HFS problems is job permutation representation [23]. In this representation, the permutation of job numbers shows the processing order of all jobs at the first stage. Assume a problem with 10 jobs. One possible permutation is \{5, 2, 10, 4, 1, 9, 7, 3, 8, 6\}.

We need a decoding method to obtain the complete schedule represented by an encoded solution. The following approach is used. The job sequence for the subsequent stages is based on "earliest completion time of jobs at previous stage". That is, the jobs are sorted according to ascending order of completion times are the previous stage. The machine assignment depends on the problem case. In case of the fixed subset of machines, the job at each stage is started as soon as all its corresponding machines are available. In case of the selective subset of machines, the job is started as soon as a necessary number of machines are available. Note that EHO starts with initial population that generated randomly.

4.2. Simulated annealing

Enhancing meta-heuristics with the local search is an effective idea to have better performance [30]. In this research, we use simulated annealing (SA) as a local search. SA is a local-search-based meta-heuristic algorithm with capability to leave the local optimum with conditional acceptance worse solutions. SA is inspired by the mechanism in the annealing of solids. It has been successfully applied to many combinatorial optimization problems.
This method operates over one incumbent solution $x$. Then generate a new solution $s$ using a move operator and evaluate its objective function. If the new solution is better, then it is accepted as the incumbent solution; otherwise, it is accepted with probability given by $\exp(\Delta / T)$ where $\Delta$ is the difference between objective values of the two solutions and $T$ is a control parameter referred as temperature. SA iterates until no improvement is found in 50 consecutive moves over the best solution found. Figure 2 shows the procedure of SA.

{Please Insert Figure 2 here}

As the move operator, we randomly use one of the four mechanisms below.

1) Swap: swapping operator is to swap the positions two randomly selected job numbers. Figure 3 shows a numerical example for a problem with 6 jobs. Suppose the selected positions are 3 and 5. Then the part numbers in these two positions are swapped and a new elephant is generated.

{Please Insert Figure 3 here}

2) Insertion: Insertion operation is to reallocate a job number into the sequence. That is, a job number of a randomly selected position, is reinserted into another randomly selected position. Figure 4 shows a numerical example for the above problem. Suppose the randomly selected position is position 1 and randomly selected position for reinsertion is position 4. Then, job number 1 is moved to position 4.

{Please Insert Figure 4 here}

3) Inversion: the inversion operator is to inverse the job numbers between two randomly selected positions. Figure 5 shows an example for above problem. Suppose the two randomly selected positions are 2 and 5. Then the part numbers between these two positions are inversed.

{Please Insert Figure 5 here}
4) Or-opt: The Or-opt operator is as follows. One position is randomly selected. Then the job number in that position and the next position are moved to another pair of two consecutive randomly selected positions. Figure 6 shows the numerical example for this operator over the above problem. Suppose the two selected neighboring job numbers are 3 and 2 in positions 2 and 3. They both are moved to positions 5 and 6.

{Please Insert Figure 6 here}

4.3. Clan updating phase

In this phase, matriarch of clan $C_i$ influences each elephant in clan $C_i$, in other words, the matriarch attempts to improve the elephants in its clan. Matriarch $C_i$ is the fittest elephant in clan $C_i$. The new elephants in each clan are generated by one-point crossover operator of each elephant and its matriarch. Note that matriarch at each clan cannot be updated by this operator. In this case, it can be updated by two-point crossover operator over best elephant of all the clans. $nC_i$ is the number of elephants in clan $C_i$. The one- and two-point crossover as follows:

1) One-point crossover: In this operator, the part numbers before a randomly selected position from matriarch are copied into the new elephant. The remaining part numbers from the other elephant are copied into the new elephant according to their relative order in the other elephant. Figure 7 shows a numerical example over above problem for this operator. Suppose the selected cut point is position 4. The part numbers from the beginning to this position are copied. Then, the copied part numbers are removed from the other elephant from left to right. The remaining part numbers (3-2) are copied into the empty positions in the new elephant.

{Please Insert Figure 7 here}

2) Two-point crossover: In this operator, two randomly selected cut positions from the best elephant are copied into the new elephant. Then, the remaining part numbers
from the matriarch are copied into the new elephant according to their relative order in
the matriarch. Figure 8 shows a numerical example over above problem for this
operator. Suppose the selected cut points are positions 3 and 5. The part numbers from
position 3 to position 5 are copied. Then, the copied part numbers are removed from the
matriarch from left to right. The remaining part numbers (1-3-1) are copied into the
empty positions in the new elephant.

{Please Insert Figure 8 here}

4.4. Separating phase
To further diversify EHO, it requires an operator like mutation in genetic algorithm.
In this phase, the worst elephant in each clan is replaced with a new elephant using the
insertion operator.

5. Numerical evaluation
This section evaluates the proposed models and algorithms in both cases. Let us remind
the in case 1, the subsets of machines required by each job at each stage is fixed; yet, in
the case 2, the subset is selective form all available machines in each stage. To this end,
for each case, three experiments are conducted, one for parameter tuning, one for model
evaluation and one for algorithm evaluation.

To compare the algorithms, the relative percentage deviation (RPD) is used. That is,
the objective function of the solution obtained by the algorithm is normalized as follows.
\[
\frac{TC - Min}{Min} \times 100
\]
where TC and Min are total completion time of the solution of the algorithm for a given
instance and the minimum makespan obtained for that instance, respectively. Note that
we also bring two effective algorithms into experiments to better position EHO in the
literature. These three algorithms are shuffled frog-leaping algorithm (SFLA) by [23],
simulated annealing (SA) by [22], and limited discrepancy search (LDS) by [24]. The
stopping criterion is set to 3nm seconds for all the algorithms.
5.1. Parameter tuning

Different levels of the control factors are shown in Table 1. $n_{clan}$ is the number of all clan, $nC_i$ is the number of elephants in clan $C_i$, $p_{MIE}$ is the probability to execute the multi-type individual enhancement scheme on each elephant in clan $C_i$, betta is temperature reduction factor, $Tf$ is final temperature, $prob$ is means the probability of executing the swapping, insertion, inversion, Or-opt movement scheme, respectively.

{Please Insert Table 1 here}

EHO algorithm has 6 factors and for each one 3 different levels are considered. The required number of tests for a full factorial experiment is $3^6$, but Taguchi method uses orthogonal array $L_{27}$ which includes only 27 tests. To implement the experiments, we consider a set of 10 instances with different combinations of the number of orders, parts and machines. The results are evaluated using Minitab16. According to the results, the parameters for the EHO algorithm are set as follows.

\[ n_{clan}=10, nC_i=10, p_{MIE}=0.3, betta=0.75, tf=0.3, prob=[0.1\ 0.1\ 0.4\ 0.4] \]

5.2. Evaluation in Case 1

In this section, at first, the algorithm is tuned. Then, the model and algorithm are evaluated over small sized instances. Finally, the algorithm is compared with two state-of-the-art algorithms (SFLA and SA) over large sized instances.

Now, a set of 16 instances are generated in the following sizes (two instances for each size).

\[ n = \{6,8,10,12\}, m = \{3,5\} \text{ and } m_i = \{U[2,5]\} \]

The processing times are randomly distributed over (1, 99). The model is implemented a maximum of 1000 seconds of computational time.

Table 2 shows the results obtained by the proposed model and algorithm. As clear, the model of case 1 cannot optimally solve instances larger than 12 jobs and 3 stages. All the instances with 8 jobs or less can be solved to optimality in less than one second.
Instances with 10 jobs can also be optimally solved in around 100 seconds. The proposed algorithm solves 10 instances out of 14 to optimality.

Now, the proposed algorithm is evaluated against SFLA and SA over large sizes. A set of 60 instances, 5 instances for each of the following 12 sizes:

\[ n = \{20, 50, 100\}, m = \{3, 6\}, m_i = \{2, U[1, 3]\} \]

Table 3 shows the results over large instances. As it can be seen, EHO outperforms the SFLA, SA and LDS with average RPF of 0.09%. The second best is LDS with average RPD of 0.53%.

5.2. Evaluation for case 2

In this section, like case 1, at first, the algorithm is tuned. Then, the model and algorithm are evaluated over small sized instances. Finally, the algorithm is compared with two algorithms (SFLA and SA) in literature over large sized instances.

Now, the MILP model and algorithm of case 2 is evaluated to solve the problem. A set of 12 instances are generated as follows (2 instances for each size).

\[ n = \{6, 8, 10\}, m = \{3, 5\} \text{ and } m_i = \{U[2, 5]\}\)
The processing times are randomly distributed over (1, 99). Table 4 shows the results obtained by the model (within 1000s of computational time) and EHO. As can be seen, the model of case 2 cannot optimally solve instances larger than 8 jobs and 3 stages. It gives the average optimally gap of 14.43%. As of EHO for case 2, it optimally solves 4 instances out of 6 ones that are already solved by the model.

{Please Insert Table 4 here}

The proposed algorithm is now compared with the three other algorithms from the literature (i.e., SFLA, SA, and LDS) over large sizes. A set of 60 instances, 5 instances for each of the following 12 sizes:

\[ n = \{20, 50, 100\}, \quad m = \{3, 6\}, \quad m_i = \{2, U[1, 3]\} \]

Table 5 shows the results over large instances. As it can be seen, EHO outperforms SFLA and SA with average RPF of 0.16%. The second best is LDS with average RPD of 0.67%.

{Please Insert Table 5 here}

{Please Insert Figure 11 here}

In this part, we also show the performance of the algorithms versus the problem sizes. In Figure 11, the average RPDs and 95% confidence level of these four tested algorithms are shown. Moreover, Figure 12 shows the average RPDs of the algorithms versus the number of jobs. As shown, in all three problem sizes, the proposed EHO keeps its better performance, especially for larger size instances.

{Please Insert Figure 12 here}

5. Conclusion and future research

The hybrid flow shop scheduling problem with multiprocessor tasks in MRPII system were studied. Two cases of fixed and selective subsets of machines were considered. At first, the two problems are mathematically formulated by mixed integer linear
programming models, one for each case. Then, using commercial software, the models were used to solve the small instances of the problems to optimality. The model of case 1 solves instances up to 12 jobs while the model of case 2 solves instances with at most 10 jobs within 1000 seconds.

Moreover, a novel algorithm, called elephant herding optimization, was proposed to solve large instances of the problems. The proposed algorithm was compared with two available algorithms (simulated annealing and shuffled frog-leaping algorithm) in the literature for performance. In both cases, the proposed algorithm outperforms the other algorithms.

One interesting future research direction is to include backorder possibility like [31] where the author predicts backorder aging and unfilled backorders. Another future research is incorporating setup times into the model.

References


Habib Reza Gholami is currently a Ph.D. student at the Department of Industrial Engineering, Islamic Azad University, Qazvin Branch, Iran. He obtained MSc degree from Islamic Azad University, South Tehran Branch in 2004, and his BSc degree in Computer science from Shiraz University in Iran (1991). His research interests are in the areas of operation research, such as production planning, production scheduling.

Esmaeil Mehdizadeh is currently an Associate Professor at the Department of Industrial engineering, Islamic Azad University, Qazvin Branch, Iran. He received his Ph.D. degree from Islamic Azad University, Science and Research Branch, Tehran in 2009, MSc degree from Islamic Azad University, South Tehran Branch in 1999, and BSc degree from Islamic Azad University, Qazvin Branch in 1996 all in Industrial Engineering. His research interests are in the areas of operation research, such as production planning, production scheduling, fuzzy sets, and meta-heuristic algorithms. He has several papers in journals and conference proceedings. Also, he is a Managing Editor of the International Journal of Optimization in Industrial Engineering.

Bahman Naderi completed his BSc degree in Industrial Engineering from Mazandaran University of Science and Technology (Iran), MSc and Ph.D. form Amirkabir University of Technology Tehran, Iran, and a post-doctoral program at University of Windsor, Canada.
Currently, he is an Assistant Professor at Kharazmi University. His research interests are applied operations research (mathematical modeling and solution methods).

**Procedure:** Hybrid EHO Algorithm

**Initialization**

**While** Computational_time < n **do**

- Simulated annealing
- clan updating phase
- separating phase

**Endwhile**

Figure 1. The general outline of hybrid EHO

**Procedure:** Simulated annealing

**While** stopping criterion is not met **do**

- move operator
- acceptance criterion

**Endwhile**

Figure 2. The general outline of SA

The current sequence

| 2 | 3 | 1 | 4 | 6 | 5 |

The new sequence

| 2 | 3 | 6 | 4 | 1 | 5 |

Figure 3. The numerical example for swapping operator.
The current elephant

2 4 6 1 3 5

The new elephant

4 6 1 2 3 5

Figure 4. The numerical example for insertion operator.

The current elephant

3 6 1 2 4 5

The new elephant

3 4 2 1 6 5

Figure 5. The numerical example for inversion operator.

The current elephant

5 3 2 4 6 1

The new elephant

5 4 6 1 3 2

Figure 6. The numerical example for Or-opt operator.

The matriarch

4 3 6 5 1 2

The current elephant

2 6 4 5 1 3

The new elephant

4 3 6 2 5 1

Figure 7. The numerical example for one-point crossover.

The best elephant

4 3 6 5 1 2

The matriarch

2 6 4 5 1 3

New elephant

2 4 6 5 1 3

Figure 8. The numerical example for two-point crossover.
Figure 9. The average RPD and 95% confidence level of the tested algorithms in case 1.

Figure 10. The average RPD of the tested algorithms versus the number of jobs in case 1.
Figure 11. The average RPD and 95% confidence level of the tested algorithms in case 2.

Figure 12. The average RPD of the tested algorithms versus the number of jobs in case 2.
Table 1. Factors and their levels for EHO.

<table>
<thead>
<tr>
<th>Level</th>
<th>(n_{clan})</th>
<th>(nC_i)</th>
<th>(P_{MIE})</th>
<th>(\beta)</th>
<th>(T f)</th>
<th>(prob)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>0.1</td>
<td>0.55</td>
<td>0.1</td>
<td>[0.1 0.1 0.4 0.4]</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
<td>0.3</td>
<td>0.75</td>
<td>0.2</td>
<td>[0.2 0.2 0.3 0.3]</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>20</td>
<td>0.5</td>
<td>0.9</td>
<td>0.3</td>
<td>[0.4 0.4 0.1 0.1]</td>
</tr>
</tbody>
</table>

Table 2. Model’s and algorithm’s results (computational time in seconds) in case 2.

<table>
<thead>
<tr>
<th>(n \times m)</th>
<th>Model</th>
<th>Algorithm (EHO)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(TC)</td>
<td>Time</td>
</tr>
<tr>
<td>6\times3</td>
<td>154</td>
<td>0.05</td>
</tr>
<tr>
<td>6\times3</td>
<td>290</td>
<td>0.06</td>
</tr>
<tr>
<td>6\times5</td>
<td>228</td>
<td>0.08</td>
</tr>
<tr>
<td>6\times5</td>
<td>360</td>
<td>0.09</td>
</tr>
<tr>
<td>8\times3</td>
<td>270</td>
<td>0.36</td>
</tr>
<tr>
<td>8\times3</td>
<td>748</td>
<td>0.31</td>
</tr>
<tr>
<td>8\times5</td>
<td>780</td>
<td>0.48</td>
</tr>
<tr>
<td>8\times5</td>
<td>1740</td>
<td>0.51</td>
</tr>
<tr>
<td>10\times3</td>
<td>858</td>
<td>143.83</td>
</tr>
<tr>
<td>10\times3</td>
<td>1422</td>
<td>26.86</td>
</tr>
<tr>
<td>10\times5</td>
<td>1160</td>
<td>119.56</td>
</tr>
<tr>
<td>10\times5</td>
<td>1814</td>
<td>74.08</td>
</tr>
<tr>
<td>12\times3</td>
<td>1540</td>
<td>178.92</td>
</tr>
<tr>
<td>12\times3</td>
<td>1430</td>
<td>132.05</td>
</tr>
<tr>
<td>12\times5</td>
<td>1306</td>
<td>1000</td>
</tr>
<tr>
<td>12\times5</td>
<td>1458</td>
<td>1000</td>
</tr>
</tbody>
</table>
Table 3. The average RPD of the algorithms over large instances for case 1

<table>
<thead>
<tr>
<th>n x m</th>
<th>Algorithms</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SA</td>
<td>SFLA</td>
<td>EHO</td>
<td>LDS</td>
</tr>
<tr>
<td>20x3</td>
<td>1.43</td>
<td>0.51</td>
<td>0.00</td>
<td>0.12</td>
</tr>
<tr>
<td>20x6</td>
<td>1.53</td>
<td>0.70</td>
<td>0.00</td>
<td>0.24</td>
</tr>
<tr>
<td>50x3</td>
<td>1.05</td>
<td>0.65</td>
<td>0.32</td>
<td>0.45</td>
</tr>
<tr>
<td>50x6</td>
<td>1.33</td>
<td>1.02</td>
<td>0.16</td>
<td>0.53</td>
</tr>
<tr>
<td>100x3</td>
<td>1.82</td>
<td>1.32</td>
<td>0.09</td>
<td>0.95</td>
</tr>
<tr>
<td>100x6</td>
<td>1.57</td>
<td>1.16</td>
<td>0.00</td>
<td>0.88</td>
</tr>
<tr>
<td>Average</td>
<td>1.46</td>
<td>0.89</td>
<td>0.09</td>
<td>0.53</td>
</tr>
</tbody>
</table>

Table 4. Model’s and algorithm’s results (computational time in seconds) in case 2.

<table>
<thead>
<tr>
<th>n x m</th>
<th>Model</th>
<th></th>
<th></th>
<th>Algorithm (EHO)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>Time</td>
<td>Optimality Gap</td>
<td>C_max</td>
<td>Time</td>
</tr>
<tr>
<td>6x3</td>
<td>130</td>
<td>1.25</td>
<td>0</td>
<td>130*</td>
<td>54</td>
</tr>
<tr>
<td>6x3</td>
<td>148</td>
<td>0.23</td>
<td>0</td>
<td>148*</td>
<td>54</td>
</tr>
<tr>
<td>6x5</td>
<td>200</td>
<td>231.08</td>
<td>0</td>
<td>204</td>
<td>90</td>
</tr>
<tr>
<td>6x5</td>
<td>250</td>
<td>413.87</td>
<td>0</td>
<td>250*</td>
<td>90</td>
</tr>
<tr>
<td>8x3</td>
<td>336</td>
<td>5.32</td>
<td>0</td>
<td>340</td>
<td>72</td>
</tr>
<tr>
<td>8x3</td>
<td>448</td>
<td>99.73</td>
<td>0</td>
<td>448*</td>
<td>72</td>
</tr>
<tr>
<td>8x5</td>
<td>636</td>
<td>1005.47</td>
<td>5.33%</td>
<td>636*</td>
<td>120</td>
</tr>
<tr>
<td>8x5</td>
<td>1156</td>
<td>1005.80</td>
<td>11.25%</td>
<td>1158</td>
<td>120</td>
</tr>
<tr>
<td>10x3</td>
<td>1186</td>
<td>1122.27</td>
<td>19.83%</td>
<td>1186</td>
<td>90</td>
</tr>
<tr>
<td>10x3</td>
<td>1214</td>
<td>1021.60</td>
<td>15.82%</td>
<td>1214</td>
<td>90</td>
</tr>
<tr>
<td>10x5</td>
<td>1568</td>
<td>1026.83</td>
<td>28.34%</td>
<td>1552</td>
<td>150</td>
</tr>
<tr>
<td>10x5</td>
<td>1394</td>
<td>1017.03</td>
<td>6.03%</td>
<td>1402</td>
<td>150</td>
</tr>
</tbody>
</table>

Table 5. The average RPD of the algorithms over large instances for case 2

<table>
<thead>
<tr>
<th>n x m</th>
<th>Algorithms</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SA</td>
<td>SFLA</td>
<td>EHO</td>
<td>LDS</td>
</tr>
<tr>
<td>20x3</td>
<td>1.82</td>
<td>0.55</td>
<td>0.21</td>
<td>0.19</td>
</tr>
<tr>
<td>20x6</td>
<td>1.66</td>
<td>0.79</td>
<td>0.04</td>
<td>0.39</td>
</tr>
<tr>
<td>50x3</td>
<td>1.95</td>
<td>0.66</td>
<td>0.07</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>50×6</td>
<td>2.75</td>
<td>1.03</td>
<td>0.62</td>
<td>0.88</td>
</tr>
<tr>
<td>100×3</td>
<td>2.67</td>
<td>1.16</td>
<td>0.02</td>
<td>1.16</td>
</tr>
<tr>
<td>100×6</td>
<td>2.14</td>
<td>1.56</td>
<td>0.01</td>
<td>0.92</td>
</tr>
<tr>
<td>Average</td>
<td>2.17</td>
<td>0.96</td>
<td>0.16</td>
<td>0.67</td>
</tr>
</tbody>
</table>