*Journal of Industrial and Systems Engineering*
Vol. 10, No. 3, pp 72-91
Summer (July) 2017

# A new model for integrated lot sizing and scheduling in flexible job shop problem

**Mahmoud Fadavi[1], Rashed Sahraeian[1*], Mohammad Rohaninejad[1]**

[1] *Industrial Engineering Department, College of Engineering, Shahed University, Tehran, Iran*

*M.Fadavi@hotmail.com, sahraeian@shahed.ac.ir, rohaninejad.sm@gmail.com*

**Abstract**

In this paper an integrated lot-sizing and scheduling problem in a flexible job shop environment with machine-capacity-constraint is studied. The main objective is to minimize the total cost which includes the inventory costs, production costs and the costs of machine's idle times. First, a new mixed integer programming model, with small bucket time approach, based on Proportional Lot sizing and Scheduling Problems (PLSP), is proposed to formulate the problem. Since the problem under study is NP-hard, a modified harmony search algorithm, with a new built-in local search heuristic is proposed as solution technique. In this algorithm, it is improvised a New Harmony vector in two phases to enhance search ability. Additionally, Taguchi method is used to calibrate the parameters of the modified Harmony Search (HS) algorithm. Finally, comparative results demonstrate the effectiveness of the modified harmony search algorithm in solving the problem.It is also demonstrated that the proposed algorithm can find good quality solutions for all size problems. The objective values obtained by proposed algorithm are better from HS algorithm and exact method results.

**Keywords:** Lot-sizing, scheduling, flexible job shop, Harmony Search algorithm (HS)

## 1- Introduction

The Flexible Job Shop scheduling Problem (FJSP) is a generalization of the classical Job Shop Problem (JSP). In JSP, *n* jobs are processedon *m* unrelated machines, and the route each job takes on different machines is known. Moreover, processing times are fixed and all the machines are assumed to be available at time zero. FJSPextends JSP by assuming that, there is at least one instance of the machine type necessary to perform each given operation (Fattahi *et al.*, 2009). In other words, the main difference between JSP and FJSP is that the latter should choose a machine from a set of candidate machines to perform an operation. The FJSP is strongly NP-hard even with two machines where each job has at most three operations (Fattahi *et al.*, 2007). In turn, the problem under study which is a combination of FJSP and integrated lot sizing would be strongly NP-hard (Karimi *et al.*,2003).

Generally, production problems can be classified into three categories based on the time intervals; long term, medium term, and short term. Production planning is included in the medium-term level and scheduling is placed in the short-term level. Integrated production planning and scheduling problem,

*Corresponding author.
ISSN: 1735-8272, 

are quite important in the production planning. In order to find the optimal solutions, interrelationship between these two levelsshould be considered, and the planning decisions should be taken simultaneously (Ramezanian & Saidi-Mehrabad, 2013). In recent years, to meet the need for more precise and realistic production planning, many papers have been publishedabout simultaneous optimization of lot-sizing and scheduling (Rohaninejad *et al.*, 2015[b]).

Due to the NP-hardness of the problem under study, heuristic methods are used. In comparison with exact methods, these approaches are more reasonable in regard to the computational times. In recent years, the use of meta-heuristics such as Tabu Search (TS) and Genetic Algorithms (GAs) has led to better results than classical dispatching or greedy heuristic algorithms. Similarly, in the current study, a harmony search algorithm (HA) is modified and developed to solve the model.

In the following section, a brief review on integrated lot sizing and scheduling problems are given. In section 3, the mixed integer programmingformulation of the problem is presented. In section 4, the proposed solution techniques are presented. Computational results of solving the model, both with IBM ILOG CPLEX software andthe proposed algorithm are reported in Section 5. Section 6 concludes the paper and discussesits possible extensions.

## 2- Literature Review

Small bucket time models and big bucket timemodels are two basic modeling approaches for simultaneous optimization oflot-sizing and schedulingproblems.In the small bucket time models, one or at most two items may be producedin everyperiod. Three common types of models in small bucket time models are discrete lot-sizingscheduling problem (DLSP), continuous setup lot-sizing problem (CSLP) and proportionallot-sizing scheduling problem (PLSP). The PLSP model was first introduced by Drexl and Haase (1995). In PLSP model, the micro period is increased by increments in the computational complexity of the model (Rohaninejad *et al.*, 2015[c]). Integrated lot-sizing and scheduling problems studiedin a single machine environment can be found in Haase & Kimms (2000), Kovács (2009) and Weidenhiller&Jodlbauer(2009) among others. General models with parallel machines have been considered by Beraldi (2008) and Kaczmarczyk (2011). Flowshop problems are considered by Ramezanian & Saidi-Mehrabad (2013) and Ramezanian (2013) and flexible flow shop models are studied by Akrami (2006) and Mahdieh (2011).

In this paper, an integrated lot-sizing flexible flowshop scheduling problem in small bucket time is studied where each job has more than two operations and the number of machines is higher than two. Similar works in different environments can be found at Fandel and Stammen-Hegene (2006). They presented a mathematical model for integrated lot-sizing scheduling in job shop environment to minimize the total cost including the sequence-dependent setup costs, the storage costs, the production costs and the costs of maintaining the machines' setup conditions. Karimi-Nasab (2013) presented the same problemin a multi-level products environment with machines which had different working speed. They proposed a memetic algorithm to tackle the problem. Gómez Urrutia (2014) proposed a Lagrangian heuristic method to solve the lot-sizing problem with capacity constraints.

According to the literature, costs of holding, backorder, setup and overtime are considered as production system costs and minimization of makespan, average weighted tardiness, number of tardy jobs, total setup time, total flow time and total idle time of machines are considered as objective functions (Petrovic *et al.*, 2008). In this paper, the objective is to minimize the total costs which includes the holding, idle machine time and production costs.

Several solution methods have been developed to solve the problem under study. Telendo (2009) used GA to solve the integrated two-level lot sizing and scheduling problem. Almada-Lobo and James (2010) applied a neighborhood search meta-heuristics and Rohaninejad (2015[c]) applied a GA based on the ELECTRE method for multi-item capacitated lot-sizing and scheduling problem with sequence-dependent setup times and costs. Other meta-heuristics used in such problems are GA (Ponnambalam & Reddy, 2003), (Sikora *et al.*, 1996) and (Wang &Zheng, 2001), tabu search (TS) (Akrami*et al.*, 2006) and (Rohaninejad*et al.*, 2015[a]) simulated annealing (SA) (Ramezanian, & Saidi-Mehrabad, 2013), (Ponnambalam& Reddy, 2003) and (Wang, &Zheng, 2001).

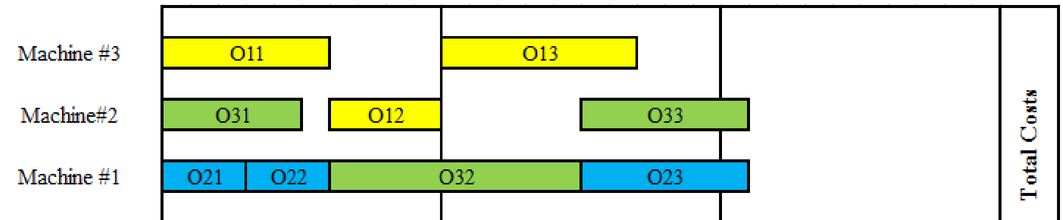
Harmony Search (HS) was first developed by Zong Woo Geem (2001), where they demonstrated its many applications. Wang (2010), presented three hybrid HA to minimize the total flow time in a flow shop system with blocking considerations. Next, for the same problem, Wang (2011) proposed local-best harmony search algorithm with dynamic sub-harmony memories. Finally, Gao (2014) presented discrete harmony search algorithm in a flexible job shop scheduling problem with multiple objectives.

To the best of our knowledge, there is no paper on harmony search for integrated lot-sizing and flexible job shop scheduling problem (in micro period case). In this paper we proposed a modified Harmony search algorithm (named *mHS*) on the base of local search approach. For small-size problems, we evaluate and compare exact solutions with those derived from *mHS*. For large problems, the results from *mHS* are compared to those of benchmark instances and the results indicate a good performance.

## 3- Mixed integer programming model

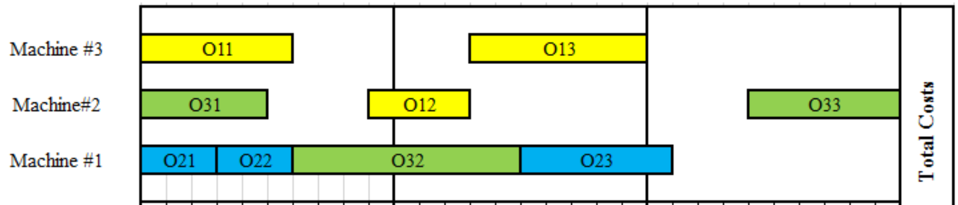
In this section, a new mixed integer mathematical programming model in the form of a PLSP is presented. The reason behind choosing PLSP model is that in such models, the production planning is performed in micro periods.

In the PLSP model the planning horizon consists of a finite number of periods. Machine capacities in each period are limited, andthey are different from period to period. For best utilization of each machine capacities, the remaining capacity can be used by another operation in same period.  Inventory costs are different for each job and different from period to period. As an example for PLSP,in Figure 1, each job is assumed to have three operations and can be processed by three machines. Planning horizon consists of 3 periods and each period consists of 10 time units. As can be seen in Figure 1,solving the general FJSP with limited machine capacities leads to makespan of 21 unit time and total costs of 319. Adding inventory and idle machine costs in Figure 2, increases the makespan to 30 and the total costs decrease to 293.




| | Micro Period | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | Total Costs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MacroPeriod 1 | | | | | | | | | | MacroPeriod 2 | | | | | | | | | | MacroPeriod 3 | | | | | | | | | | |
| Machine 3 | Capacity | 10 | | | | | | | | | | 10 | | | | | | | | | | 10 | | | | | | | | | | |
| | Production Cost | 6 | 6 | 6 | 6 | 6 | 6 | | | | | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | | | | | | | | | | | | 68 |
| | IdleMachine Cost | | | | | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | |
| Machine 2 | Capacity | 10 | | | | | | | | | | 10 | | | | | | | | | | 10 | | | | | | | | | | |
| | Production Cost | 2 | 2 | 2 | 2 | 2 | | 3 | 3 | 3 | 3 | | | | | | 3 | 3 | 3 | 3 | 3 | 3 | | | | | | | | | | 51 |
| | IdleMachine Cost | | | | | | 1 | | | | | 2 | 2 | 2 | 2 | 2 | | | | | | | | | | | | | | | | |
| Machine 1 | Capacity | 10 | | | | | | | | | | 10 | | | | | | | | | | 10 | | | | | | | | | | |
| | Production Cost | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | | | | | | | | | | 69 |
| | IdleMachine Cost | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Inventory cost of job 3 | | | | | | | | | | | | | | | | | | | 3 | 3 | 3 | | | | | | | | | | | |
| Inventory cost of job 2 | | | | | | | 5 | | | | | | | | | | | | | | | | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 131 |
| Inventory cost of job 1 | | | | | | | | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | | | | | | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | |
| **Total Costs** | | 10 | 10 | 10 | 11 | 11 | 15 | 13 | 13 | 13 | 13 | 15 | 15 | 15 | 15 | 15 | 10 | 10 | 9 | 9 | 9 | 6 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | **319** |

**Figure 1.** PLSP Scheduling example whit out costs objective




| | Micro Period | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | Total Costs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MacroPeriod 1 | | | | | | | | | | MacroPeriod 2 | | | | | | | | | | MacroPeriod 3 | | | | | | | | | | |
| Machine 3 | Capacity | 10 | | | | | | | | | | 10 | | | | | | | | | | 10 | | | | | | | | | | 74 |
| | Production Cost | 6 | 6 | 6 | 6 | 6 | 6 | | | | | | | | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | | | | | | | | | |
| | IdleMachine Cost | | | | | | | 1 | 1 | 1 | 1 | 2 | 2 | 2 | | | | | | | | | | | | | | | | | | |
| Machine 2 | Capacity | 10 | | | | | | | | | | 10 | | | | | | | | | | 10 | | | | | | | | | | 55 |
| | Production Cost | 2 | 2 | 2 | 2 | 2 | | | | | 3 | 3 | 3 | 3 | | | | | | | | | | | | 3 | 3 | 3 | 3 | 3 | 3 | |
| | IdleMachine Cost | | | | | | 1 | 1 | 1 | 1 | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | |
| Machine 1 | Capacity | 10 | | | | | | | | | | 10 | | | | | | | | | | 10 | | | | | | | | | | 69 |
| | Production Cost | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | | | | | | | | | | |
| | IdleMachine Cost | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Inventory cost of job 3 | | | | | | | | 3 | 3 | 3 | | | | | | | | | | | | | | | | | | | | | | 95 |
| Inventory cost of job 2 | | | | | | | 5 | | | | | | | | | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | | | | | | | |
| Inventory cost of job 1 | | | | | | | | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | | | | | | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | |
| **Total Costs** | | 10 | 10 | 10 | 11 | 11 | 15 | 14 | 14 | 14 | 13 | 14 | 14 | 14 | 14 | 14 | 10 | 10 | 10 | 10 | 10 | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | **293** |

**Figure 2.** PLSP Scheduling example whit costs objective

In the proposed PLSP modelthe other assumptions are made following:

1) Planning horizon is constant and finite.
2) All machines are continuously available from time zero
3) Demand rates, production rates, Idle Machine cost rates, Production cost rates, and inventory holding costs rates are deterministic and not constant over the planning horizon.
4) A machine can only execute one operation at a time.
5) External demands occur only for end operation of jobs.
6) Processing time rates are different for different products and different machines.
7) Production may be continued indefinitely without interruption.
8) Setup time is connivance.
9) Objective is to minimize the total costs.
10) Shortage is not allowed.

The indices of the model are:

| | |
|---|---|
| $i$ | index of machine, $i = 1, \dots, m$ |
| $j$ | index of jobs, $j = 1, \dots, J$ |
| $h$ | index of operations, $h = 1, \dots, H$ |
| $t$ | index of period, $t = 1, \dots, T$ |

The parameters of the model are:

| | |
|---|---|
| $T$ | Sum of periods (Macro period*Micro period) |
| $m$ | Number of machines |
| $J$ | Number of jobs |
| $O_{j,h}$ | Operation $h$ of Job $j$ |
| $P_{i,j,h}$ | Processing time of operation$O_{j,h}$ on machine $i$ |
| $d_{j,t}$ | Demand of job $j$ in the end of period$t$ |
| $C_{i,t}$ | Available capacity of machine $i$ in period $t$ |
| $q_{j,h}$ | Production coefficient, which indicates how many units of item produced through operation $O_{j,h}$, are required to produce one unit of item produced through operation $O_{j,h+1}$ |

| | |
|---|---|
| $a_{i,j,h}$ | If operation $O_{j,h}$can be processed on machine $i$, is equal to one; otherwise, it is equal to zero |
| $S_{j,h,t}$ | Cost of one unit inventory holding for operation$O_{j,h}$in period$t$ |
| $b_{i,t}$ | Cost of unit idle time for machine $i$in period$t$ |
| $f_{ijh}$ | Cost of one unit production for operation$O_{j,h}$on machine$i$ |

The variables of the model are:

| | |
|---|---|
| $y_{i,j,h,t}$ | $\begin{cases} 1 : \text{If operation } O_{j,h} \text{ processed on machine } i \text{ in period } t \\ 0 : \text{otherwise} \end{cases}$ |
| $Q_{i,j,h,t}$ | Lot size of operation $O_{j,h}$ on machine $i$ in period $t$ |
| $I_{j,h,t}$ | Inventoryof operation $O_{j,h}$in the end of period $t$ |
| $B_{i,t}$ | Idle time of machine $i$ in period$t$ |

The problem's mixed integer programming model is as follows:

$$Minimize = \sum_{j=1}^{J}\sum_{h=1}^{H}\sum_{t=1}^{T} I_{j,h,t} * s_{jht} + \sum_{i=1}^{m}\sum_{j=1}^{J}\sum_{h=1}^{h}\sum_{t=1}^{T} Q_{i,j,h,t} * f_{i,j,h} + \sum_{i=1}^{m}\sum_{t=1}^{T} B_{i,t} * b_{i,t} \quad (1)$$

$s.t$:

$$I_{j,h,t} = I_{j,h,t-1} + \sum_{i=1}^{m} Q_{i,j,h,t} - \sum_{i=1}^{m} q_{j,h} * Q_{i,j,h+1,t} \qquad j = 1, \dots, J; h = 1, \dots, H-1; \; t = 1, .., T; I_{j,h,0} = 0 \quad (2)$$

$$I_{j,H,t} = I_{j,H,t-1} + \sum_{i=1}^{m} Q_{i,j,H,t} - d_{j,t} \qquad j = 1, \dots, J; t = 1, .., T \quad (3)$$

$$I_{j,h-1,t-1} + \sum_{i=1}^{m} Q_{i,j,h-1,t} \geq \sum_{i=1}^{m} q_{j,h-1} * Q_{i,j,h,t} \qquad i = 1, .., m; \; t = 1, \dots, T; \; h = 2, \dots, H; \; I_{j,h-1,0} = 0 \quad (4)$$

$$\sum_{i=1}^{m}\sum_{t=1}^{T} Q_{i,j,h-1,t} = \sum_{i=1}^{m}\sum_{t=1}^{T} q_{j,h-1} * Q_{i,j,h,t} \qquad j = 1, .., J; h = 2, .., H \quad (5)$$

$$\sum_{i=1}^{m}\sum_{t=1}^{T} Q_{i,j,H,t} = \sum_{t=1}^{T} d_{j,t} \qquad j = 1, .., J \quad (6)$$

$$Q_{i,j,h,t} * P_{i,j,h} \leq C_{i,t} * \left(y_{i,j,h,t-1} + y_{i,j,h,t}\right) \qquad i = 1, .., m; j = 1, \dots, J; \; h = 1, \dots, H; t = 1, .., T; \; y_{i,j,h,0} = 0 \quad (7)$$

$$\sum_{j=1}^{J}\sum_{h=1}^{H} Q_{i,j,h,t} * P_{i,j,h} \leq C_{i,t} \qquad i = 1, .., m; t = 1, .., T \quad (8)$$

$$y_{i,j,h,t} \leq a_{i,j,h} \qquad i = 1, .., m; j = 1, \dots, J; \; h = 1, \dots, H; t = 1, .., T \quad (9)$$

$$\sum_{j=1}^{J}\sum_{h=1}^{H} y_{i,j,h,t} \leq 1 \qquad i = 1, \dots, m; t = 1, \dots, T \quad (10)$$

$$\sum_{i=1}^{m} y_{i,j,h,t} \leq 1 \qquad j = 1, \dots, J; h = 1, \dots, H; \; t = 1, \dots, T \qquad (11)$$

$$B_{i,t} \geq C_{i,t} - \sum_{j=1}^{J} \sum_{h=1}^{H} Q_{i,j,h,t} * P_{i,j,h} \qquad i = 1, .., m, t = 1, \dots, T \qquad (12)$$

$$y_{i,j,h,t} \in \{0,1\}$$

$$Q_{i,j,h,t} \geq 0 \quad I_{j,h,t} \geq 0 \quad B_{i,t} \geq 0 \qquad i = 1, .., m; j = 1, \dots, J; \; h = 1, \dots, H; t = 1, .., T \qquad (13)$$

Equation (1) indicates the objective function, i.e. the total cost which includes the inventory cost, production cost and machine idle cost. Constraint (2) determines the inventory of operation $O_{j,h}$in the each period and shows the inventory balance between the micro periods, for all operations of each job except the last operation of each job. As for the last operation of each job, constraint (3) controls the inventory of job $j$ in the end of period $t$to meet the demand. Constraint (4), (5) controls to quantity of operation $O_{j,h}$in the each period. Constraint (6) ensures the quantity of the last operation of job $j$ in all machines and all periods meet demand job $j$ in all periods.

Constraint (7) ensures that if possible, unused capacity of machine $i$ in each period is used. This constraint allows production variables to take non-zero values only if a machine is set up to process a given product in the current or previous period. In the PLSP at most two items of different types of items can be produced in each period. The first item produced in period $t$ corresponds to the second item produced in period $t-1$. For the second item the machine is going to be set up in period $t$ after a number of time units proportional to the time used for producing the first item. The splitting of the machine capacity for the production of two products within one period proportional to the quantities needed motivates the name of the model. (For further details about the PLSP models, refer to Drexl & Haase (1995) and Kaczmarczyk (2011)).

Constraint (8) ensures that the sum of work processing times on one machine in a specified period should not exceed the capacity of the machine in that period. Constraint (9) determines the feasibility of assigning operation $O_{j,h}$to machine$i$, in other words, an operation can be processed on a machine only if that machine is capable of processing that operation. Constraint (10) assigns the operations to a machine in each period. Constraint (11) emphasizes that each operation can be performed only on one machine in each period. Constraint (12) determines the idle time of machine$i$. Constraint 13 describes the decision variables.

## 4- Proposed algorithm for solving the problem

In this section first we present the basic Harmony Search algorithm. Then the difference between *HS* and *mHs* algorithm is described. Next, the method used for coding and decoding of the chromosomes is explained. Since all meta-heuristics algorithm start with an initial population, the mechanism for its generatingis explained. Finally, we proposed a new method to improvise a new harmony that enhances accuracy and convergence rate of harmony search (*HS*) algorithm.

### 4-1- Basic Harmony Search algorithm

Harmony search (*HS*) algorithm is a recently developed algorithm, and it is based on natural musical performance processes that occur when amusician searches for a better state of harmony, such as during jazz improvisation.In the classic harmony search (*HS*), the common parameters are the harmonymemory size (*HMS*), harmony memory consideration rate (*HMCR*), pitch adjusting rate (*PAR*), distance bandwidth (*BW*) and a stopping criterion (Geem *et al.*, 2001).

**4-1-1- Initialize the harmony memory**

At first HM filled by HMS randomly generated harmony vectors. $HM = \{X_1, X_2, \dots, X_n\}$ Let $X_i = \{X_i(1), X_i(2), \dots, X_i(n)\}$ represent the $i^{th}$ harmony vector in the HM, which is generated asfollows:

$$X_i(k) = LB(k) + R * \big(UB(k) - LB(k)\big) for k = 1,2, \dots\dots, n and i = 1,2, \dots\dots, HMS \quad (14)$$

Where *LB(k)* is a lower bound and *UB(k)* is an upper bound for decision variable $X_i(k)$, and $R$ is a random number between 0 and 1(Wang *et al.*, 2011).

**4-1-2- Improvise a new harmony**

Improvisation is the process of generating a new harmony vector $X_{new} = \{X_{new}(1), X_{new}(2), \dots\dots, X_{new}(n)\}$ in HS algorithm. $X_{new}$ is produced based on the following rules (i.e. improvisation)(Wang *et al.*, 2011).

```
for k = 1 to n   do
if R < HMCR   then
X_new(k) = X_a(k)        where a ∈ {1,2, ..., HMS}
if R  < PAR   then
X_new(k) = X_new(k) ± R * BW
end if
else
X_new(k) = LB(k) + (UB(k) − LB(k)) * R
end if
end for
```

**4-1-3- Update harmony memory**

Harmony memory is updated by comparing the new harmony ($X_{new}$) with the worst harmony vector ($X_w$). If $X_{new}$ is better than the $X_w$, then $X_{new}$ replaces $X_w$, and becomes a new member of the *HM* (Wang *et al.*, 2011).

**4-1-4- Check stopping criterion**

If the stopping criterion (e.g. maximum number of improvisations, or maximum computational time) is reached, the computation is terminated. Otherwise, Sections 4.4.2 and 4.4.3 are repeated.

## 4-2- Harmony Search and Modified Harmony Search

The proposed approach for solving the problem is based on three major steps: **1)** Determining the lot size of each operation in the period(s). **2)** Assigning the operations to an appropriate machine. 3) Determining the sequence of operations in each period.

In the classic HS algorithm, generation of a New Harmony vector for all 3 steps above, is combined into a single step. This is wrong since it limits part of the solution space. In the proposed *mHS* algorithm, two vectors are improvised: the firstis Lot size-Machine vector and the second is sequence vector. This method will enhance the algorithm's exploration ability.

Thus, the *mHS* and the HS are different in two aspects as follows:

1) The *mHS* improvised a New Harmony vector in two phases to enhance search ability. In the first phase, a new harmonyvector lot size-machine assignment for each job and in the last phase, a New Harmony vector sequence is generated.
2) Furthermore, inspired by the swarm intelligence of particle swarm, *mHS* modifies the improvisation step of the *HS*by a matrix called the "Repetition Matrix " which is explained in details in section 4.6.3

Figure 3 shows the schematic of the proposed *mHS* for integrated lot size andscheduling in a flexible job shop problem. In the following, the proposed Harmony Search algorithm is fully described.

### 4-3- Coding of Chromosomes

For encoding of chromosomes, a matrix with three rows and $M.N.T$columns is used in which $M$ is the number of machines, $N$ is the number of production positionsfor each machine in every period, and $T$ is the number of periods throughout the planning horizon. This method is adapted and modified from Rohaninejad (2015[b]). In this matrix, the elements of the first row indicate the quantities of items that are produced through operation $(j, h)$ in period $T$ which are equal to $q$ units. The elements of the second row include two components $(j, h)$which indicate the operation $h$ of Job $j$. In other words, the second row shows the sequence of different operations. Third row of the matrix represents the machine assignments made to elements of the second row. As an example, Figure 4 shows a chromosome for two jobs, each consists of two operations which should be processed on two machines in a planning horizon that spans three periods.

| | **Period 1** | | | | **Period 2** | | | | **Period 3** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lot size | 30 | 30 | 20 | | 20 | 20 | 30 | 20 | 10 | 20 | 40 | |
| O(*j*,*h*) | 1-1 | 2-1 | 1-2 | | 2-1 | 1-1 | 2-2 | 1-2 | 1-2 | 2-1 | 2-2 | |
| Machine | 1 | 2 | 1 | | 1 | 2 | 2 | 1 | 2 | 1 | 2 | |

**Figure4.** Manner of coding of chromosomes

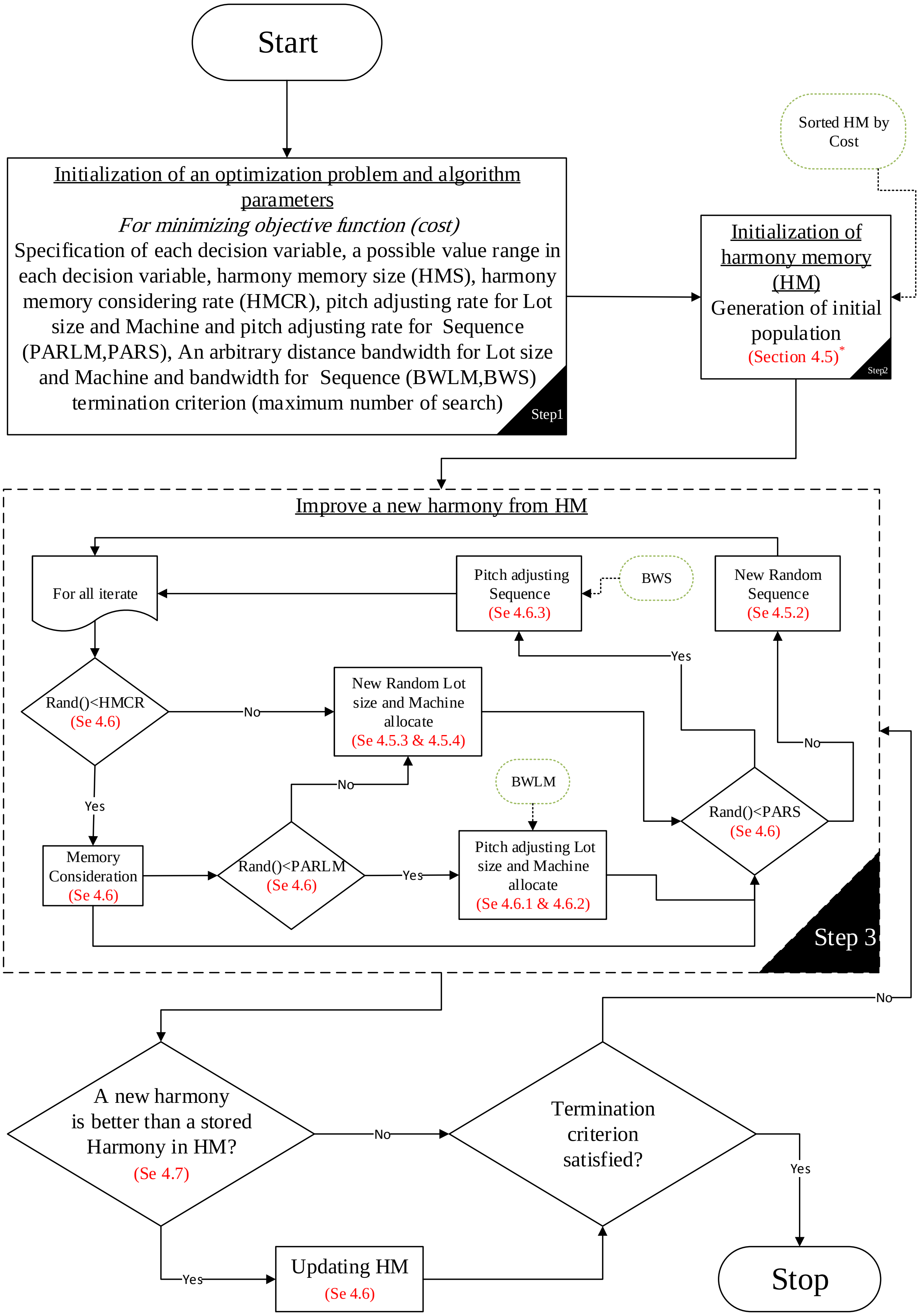



**Figure 3.** Schematic of the modified Harmony Search in PLSP

*. Described the corresponding section of each part

## 4-4-Decoding of chromosomes

In this section, the start time of an operation, the processing duration of that operation, and the machine availability should be considered. The start time for an operation is the maximum time of machine availability and the time when the required inventory is provided. Also, the processingtime duration of operation $O_{j,h}$ will be equal to $P_{j,h}^{i}.q$ , when operation $O_{j,h}$ has been assigned to machine$i$.Figure 5 shows the manner of decoding of chromosome.

| | Period 1 | | | | Period 2 | | | | Period 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lot size | 30 | 30 | 20 | | 20 | 20 | 30 | 20 | 10 | 20 | 40 | |
| O(*j*,*h*) | 1-1 | 2-1 | 1-2 | | 2-1 | 1-1 | 2-2 | 1-2 | 1-2 | 2-1 | 2-2 | |
| Machine | 1 | 2 | 1 | | 1 | 2 | 2 | 1 | 2 | 1 | 2 | |

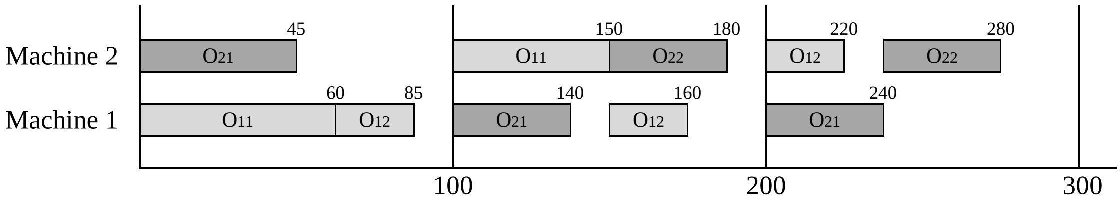



**Figure 5.**Manner of decoding of chromosome

In figure5, operation $O_{1,1}$is processed for 60 time units, because the value of$a_{1,1}^{1}$ is assumed to be equal to 2in the first period. Furthermore, it can be seen that the values of $a_{1,1}^{2}$ ,$a_{2,1}^{1}$ and $a_{2,1}^{2}$ are assumed as 2.5, 2.0 and 1.5, respectively. Also, the coefficient of utilization of product from operation $O_{21}$ in the product of operation $O_{22}$ is assumed as1.0 (Rohaninejad *et al.*, 2015[b]).

## 4-5- Generating the initial population of the algorithm

According to above, the generation of the initial population is based on the following four steps.

### 4-5-1- Determining the production periods for each item

In this step, the production periods for each item are selected randomly and all the periods have an equal chance ofbeing selected. In order to prevent shortage, the selection of the first production period for each item is a function of the first period where there is a demand for that item (Rohaninejad *et al.*, 2015[b]).

### 4-5-2- Determining the sequence of operations which are assigned to a period

For determining the sequences of operations assigned to a specificperiod, arandom method is borrowed form Rohaninejad (2015[b]). In this approach, from the operations assigned to a period, one operation is randomly selected and entered into the first unplanned sequence priority of that period.Moreover, in the first period, in order to prevent having a shortage, no operation is planned unless the previous operation is done.

### 4-5-3- Determining the lot size values for each operation

In this step, we use the procedure that Rohaninejad (2015[b]) presented for simultaneous lot-sizing and scheduling in flexible jobshop problems. In this procedure, the production quantity of operation $O_{j,\mathrm{h}}$ item in sequence $s_1$is equal to:

$$Q_{j,h}^{s_1} = I_{j,h} + Rand(D_{j,h} - \Delta_{j,h} + \sum_{s=1}^{s_1-1} a_{j,h}.Q_{j,h+1}^{s} - I_{j,h}). \quad (15)$$

Rand is random number between 0 and 1, and $D_{j,h}$ is the total demand for the item ($O_{j,h}$) of operation during the planning horizon and $\Delta_{j,h}$is difference between total production and total consumption of the item produced in operation $O_{j,h}$prior to sequence $s_1$.(Rohaninejad *et al.*, 2015[b]).

**4-5-4- Assigning machines to operations**

To assign a machine to an operation, one of the allowed machines is randomly assigned toeach operation.

## 4-6-Proposed New Harmony algorithm for PLSP

In this paper, an initial population of harmony vectors are generated according to section 4.3 and stored in the harmony memory (*HM*). There are four segments in the *HM*: lot size of operation $O_{j,h}$, machine assignment matrix, sequence of job operations and finally evaluation of total cost. Figure 6 presents the structure of the proposed harmony memory. It is noteworthy that *HM* should be sorted by cost.

According to above, a New Harmony vector $X_{new}$ has three parts: harmony vector lot size for each operation($Q_{new}$), harmony vector machine assignment($M_{new}$), and harmony vector sequence ($S_{new}$).

| | | **Period 1** | | | | **Period 2** | | | | **Period 3** | | | | **Cost** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HM(1) | Lot size | 30 | 30 | 20 | | 20 | 20 | 30 | 20 | 10 | 20 | 40 | | 1024 |
| | $O^1(j,h)$ | 1-1 | 2-1 | 1-2 | | 2-1 | 1-1 | 2-2 | 1-2 | 1-2 | 2-1 | 2-2 | | |
| | Machine | 1 | 2 | 1 | | 1 | 2 | 2 | 1 | 2 | 1 | 2 | | |
| HM(2) | Lot size | 30 | 30 | 20 | 30 | 20 | 20 | 20 | | 10 | 20 | 40 | | 1157 |
| | $O^2(j,h)$ | 2-1 | 1-1 | 1-2 | 2-2 | 2-1 | 1-1 | 1-2 | | 1-2 | 2-1 | 2-2 | | |
| | Machine | 2 | 1 | 1 | 2 | 1 | 2 | 1 | | 2 | 1 | 2 | | |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| HM(n) | Lot size | 30 | 30 | 20 | 10 | 20 | 10 | 30 | 20 | 10 | 10 | 20 | 30 | 1298 |
| | $O^n(j,h)$ | 1-1 | 2-1 | 1-2 | 2-2 | 2-1 | 1-1 | 2-2 | 1-2 | 1-1 | 1-2 | 2-1 | 2-2 | |
| | Machine | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | |

**Figure 6.** Scheme of harmony memory structure

A New Harmony vector, $X_{new}$ generated from the *HM* based on memory considerations, pitch adjustments, and randomization, is shown in Figure 3. Computational procedure for the *mHS* can be summarized as follows:

**Step 1:** Set the parameters, $HMS, HMCR, PARLM, PARS, BWLM, BWS$ and the termination criterion.

**Step 2:** Initialize the HM and calculate the objective function value for each harmony (section4.3).

**Step 3:** Improvise a New Harmony vector $X_{new}$ : We improvise a New Harmony vector in two phases to enhance searching ability. In the first phase, generate a New Harmony vector lot size($Q_{new}$), and machine assignment($M_{new}$) for each job. In the second phase, generate a New Harmony vector sequence($S_{new}$). In other words, either the lot size for a job does not change so it comes straight from the harmony memory, or a new vector is generated for each job. A similar procedure is done for the machine assignment as well. Once all the lot sizes and machine assignments are determined, a new sequence vector could be generated. Thus, a New Harmony vector lot size($Q_{new}$), and machine assignment($M_{new}$) are proposed in the following pseudo code:

```
For(all job)  do
For(k = 1 to n)do
if(r1 < HMCR)then
    Q_new(k) = Q_a(k) where a ∈ (1,2, … HMS)
    M_new(k) = M_a(k) where a ∈ (1,2, … HMS)
if(r2 < PARLM)then
        Q_new(k) = Q_new(k) ± r3 * BW_Q(k)
        M_new(k) = M_new(k) ± r4 * BW_M(k)
else if
        Q_new(k) = LB_Q(k) + r5 * (UB_Q(k) − LB_Q(k))
        M_new(k) = LB_M(k) + r6 * (UB_M(k) − LB_M(k))
 end if
 else if
    Q_new(k) = LB_Q(k) + r7 * (UB_Q(k) − LB_Q(k))
    M_new(k) = LB_M(k) + r8 * (UB_M(k) − LB_M(k))
end if
 end for
end for
```

And, improvise a New Harmony vector sequence($S_{new}$) with following pseudo code:

```
For(k = 1 to n)do
if(r9 < PARS)then
    S_new(k) = S_new(k) ± r9 * BW_S(k)
 else if
    S_new(k) = LB_Q(k) + r10 * (UB_Q(k) − LB_Q(k))
end if
end for
```

$r1, r2, \dots, r10$ is random number between 0 and 1.

**Step 4:** Update the *HM* as, $X_w = X_{new}$ ; if $Cost(X_w) < Cost(X_{new})$

**Step 5:** If the given termination criterion is satisfied, return the best harmony vector in the *HM*; otherwise go to Step 3.

We can easily add a local search to the *mHS* algorithm in order to enhance its exploration ability. This can be done by improving each candidate harmony vector generated in the improvisation phase. There are three types of neighborhoods that can be employed in the local search: the first is based on lot size, the second is based on machine assignment and the third is based on a job permutation.

### 4-6-1- Pitch adjusting lot size

Pitch adjusting of a lot size can be obtained by increasing or decreasing the quantity of lot size. Since the converting process is needed for fitness evaluation, lower and upper bounds for lot sizes of each job should be calculated. In other words, to prevent the shortage, the band width of the lot size is obtained from the following procedure:

According to section 4-5-3, this procedure starts from the first period and continues to the last. The lot size upper bound is the maximum quantity of theitem produced during the operation $O_{j,\mathrm{h}}$, that is $r_{j,h}$. Similarly, thelot size lower bound is the minimum quantity of theitem produced during the operation$O_{j,\mathrm{h}}$, that is $I_{j,h}$. Finally, a band width of lot size in sequence $s_1$is equal to:

$$BWQ_{j,h}^{s_1} = D_{j,h} - \Delta_{j,h} + \sum_{s=1}^{s_1-1} a_{j,h}.Q_{j,h+1}^{s} - I_{j,h} \quad (16)$$

Thus, A Pitch adjusting of a lot sizein sequence $s_1$is equal to:

$$Q_{new}(k) = Q_{new}(k) \pm Rand * BWIQ * BWQ_{j,h}^{s_1}(k), \quad (17)$$

Where $BWIQ$ is Bandwidth Quantity Index.In this equation Rand is random number between 0 and 1, and $D_{j,h}$ is the total demand for the item $(O_{j,h})$ of operation during the planning horizon and $\Delta_{j,h}$is difference between total production and total consumption of the item produced in operation $O_{j,h}$prior to sequence $s_1$.

### 4-6-2- Pitch adjusting machine assignment

A pitch adjusting of a machine assignment is performed by assigning/not assigning a machine to an operation$O_{j,\mathrm{h}}$in each period. Thus, a band width of machine assignment for job $j$ is the number of machines that can be used in period$t$.

A pitch adjusting of a machine assignment is equal to:

$$M_{new}(k) = M_{new}(k) \pm Rand * BWIM * BWM(k) \quad , \quad (18)$$

Where$BWIM$ is Bandwidth Machine Index.

### 4-6-3- Pitch adjusting sequence

A neighbor for a job permutationcan be obtained by *INSERT*, *INVERSE* or *SWAP* operators (Pinedo, 2012).The local search in the *mHS* algorithm is not directly applied to the position of the candidate harmony vector but to the job permutation implied by the candidate real vector. In this method the candidate real vector is achieved by maximum repeat rate of operation $O_{j,h}$ in harmony memory. To determine the new sequence, the number of operation $(O_{j,h})$ which is achieved after operation $(O_{j',h'})$ in all harmony memory sequence vector is considered, and stored in the Repetition Matrix. Thus, $w_{p,q}$ in Repetition Matrix is number of $q$ that performed after $p$, where$p$ $and$ $q$ $\in O_{j,h}$.

Thus a New Harmony sequence vector, with following pseudo code is improvised:

$\boldsymbol{for}$ $each\ Period$
$\boldsymbol{while}$ $Max \neq 0$
$Max = Maximum\ w_{p,q}\ ;\ where\ w_{p,q}\ \in Matrix\ Repeat$
$u_{p,q} = 1\ , where\ u_{p,q}\ \in Matrix\ Index$
$w_{p,q} = w_{q,p} = 0$
$\boldsymbol{if}\ u_{i,p} = 1; then\ u_{i,q} = 1\ and\ w_{i,q} = w_{q,i} = 0\ ; \boldsymbol{end};\ \ for\ all\ i \in p$
$\boldsymbol{if}\ u_{q,i} = 1; then\ u_{p,i} = 1\ and\ w_{p,i} = w_{i,p} = 0\ ; \boldsymbol{end};\ \ for\ all\ i \in p$
$\boldsymbol{end\ while}$

$$S_q = \sum_{i=1}^{p} u_{i,q}$$

$new\ sequnce = sort\ S_q$

Furthermore, figure 7 presents the scheme of scheduling in harmony memory. In this case *HMS* is 10 and harmony sequence vectors are composed of two periods, two jobs and three operations. RepetitionMatrix is created by the number of repetition a pairwise comparison of operation with each other yields (Figure 8). Matrix Index array in Figure 8 is calculated by the above pseudo code. The sum of column array in Matrix Index is rate of$(O_{j,h})$, and it seeks to find the best sequence. To find the new sequence, sum of column array in matrix index should be sorted, to find the best sequence in period 1. And this procedure should be continued for each period. See Figure 8(c)

| | | ← | | Period 1 | | | → | ← | | Period 2 | | | → |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HM(1) | $O^1(j,h)$ | 2-1 | 1-1 | 2-2 | 1-2 | 1-3 | 2-3 | 1-3 | 2-3 | | | | |
| HM(2) | $O^2(j,h)$ | 2-1 | 2-2 | 1-1 | 2-3 | 1-2 | 1-3 | 2-2 | 2-3 | | | | |
| HM(3) | $O^3(j,h)$ | 1-1 | 2-1 | 1-2 | 1-3 | 2-2 | 2-3 | 2-2 | 2-3 | | | | |
| HM(4) | $O^4(j,h)$ | 1-1 | 1-2 | 2-1 | 2-2 | 1-3 | 2-3 | 2-1 | 2-2 | 1-3 | 2-3 | | |
| HM(5) | $O^5(j,h)$ | 2-1 | 2-2 | 1-1 | 2-3 | 1-2 | 1-3 | 2-1 | 2-2 | 2-3 | | | |
| HM(6) | $O^6(j,h)$ | 2-1 | 2-2 | 1-1 | 2-3 | 1-2 | 1-3 | 2-2 | 2-3 | | | | |
| HM(7) | $O^7(j,h)$ | 2-1 | 1-1 | 1-2 | 2-2 | 2-3 | 1-3 | 1-1 | 1-2 | 1-3 | 2-3 | | |
| HM(8) | $O^8(j,h)$ | 2-1 | 2-2 | 1-1 | 1-2 | 2-3 | 1-3 | 2-2 | 2-3 | 1-3 | | | |
| HM(9) | $O^9(j,h)$ | 1-1 | 1-2 | 1-3 | 2-1 | 2-2 | 2-3 | 2-1 | 2-2 | 2-3 | | | |
| HM(10) | $O^{10}(j,h)$ | 1-1 | 1-2 | 2-1 | 2-2 | 1-3 | 2-3 | 1-1 | 1-2 | 1-3 | | | |

**Figure7.** Scheme of Scheduling in harmony memory

| | $O_{1,1}$ | $O_{1,2}$ | $O_{1,3}$ | $O_{2,1}$ | $O_{2,2}$ | $O_{2,3}$ |
|---|---|---|---|---|---|---|
| $O_{1,1}$ | | 10 | 10 | 4 | 6 | 0 |
| $O_{1,2}$ | 0 | | 10 | 3 | 5 | 0 |
| $O_{1,3}$ | 0 | 0 | | 1 | 2 | 0 |
| $O_{2,1}$ | 6 | 7 | 9 | | 10 | 6 |
| $O_{2,2}$ | 4 | 5 | 8 | 0 | | 4 |
| $O_{2,3}$ | 0 | 3 | 5 | 0 | 0 | |

**(a)- RepetitionMatrix**

| | $O_{1,1}$ | $O_{1,2}$ | $O_{1,3}$ | $O_{2,1}$ | $O_{2,2}$ | $O_{2,3}$ |
|---|---|---|---|---|---|---|
| $O_{1,1}$ | | 1 | 1 | 0 | 1 | 1 |
| $O_{1,2}$ | 0 | | 1 | 0 | 0 | 1 |
| $O_{1,3}$ | 0 | 0 | | 0 | 0 | 0 |
| $O_{2,1}$ | 1 | 1 | 1 | | 1 | 1 |
| $O_{2,2}$ | 0 | 1 | 1 | 0 | | 1 |
| $O_{2,3}$ | 0 | 0 | 1 | 0 | 0 | |
| Sum | 1 | 3 | 5 | 0 | 2 | 4 |

**(b)-Matrix Index**

| Sort(Sum) | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Sequence | $O_{2,1}$ | $O_{1,1}$ | $O_{2,2}$ | $O_{1,2}$ | $O_{2,3}$ | $O_{1,3}$ |

**(c)-New Sequence**

**Figure 8.** New Sequence base on harmony memory (for period 1)

## 4-7- Calculate the objective function

In this paper, the objective function is to minimize the total cost which includes the inventory cost, production cost and idle costs ofmachines. Production cost, can easilybe calculated according to equation 1. Idle costs and Inventory costs are related to start and finish time of operations in the sequence. According to section 4.2, the start time of an operation depends on the time the required inventory is provided to start processingthe operation. Machine availability time and sufficient inventory are necessaryfor the start of operation, thus the maximum of these two factors is the start time for the operation. Hence, the start time, finish time, and inventory of operation($O_{j,h}$), can be obtained with the following pseudo code:

$\boldsymbol{If}(In_{j,h-1} > Q_{j,h})\boldsymbol{then}$; *with given consumption rate*
  $ST_{j,h} = At\ (M_i)$
  $FT_{j,h} = ST_{j,h} + P^{i}_{j,h} * Q_{j,h}$
$\boldsymbol{else\ if}$
  $\boldsymbol{if} M_{i'}(O_{j,h-1}) = M_i(O_{j,h})$ $\boldsymbol{then}$
    $ST_{j,h} = At\ (M_i)$
    $FT_{j,h} = ST_{j,h} + P^{i}_{j,h} * Q_{j,h}$
  $\boldsymbol{else\ if}$
    $l = \max(FT_{j',h'-1}, ST_{j,h-1})$ ; $where O_{j',h'-1} is pre operation O_{j,h} in machine i$
    $\boldsymbol{if}(l = FT_{j',h'-1})\boldsymbol{then}$
      $ST_{j,h} = l$
    $\boldsymbol{else\ if}$
      $ST_{j,h} = ST_{j,h-1} + 1$
    $\boldsymbol{end\ if}$
    $v_{j,h-1} = In_{j,h-1} + \min((FT_{j,h-1} - ST_{j,h-1}), (ST_{j,h} - ST_{j,h-1}))/\ P^{i'}_{j,h-1}$ ; $i' \in M(O_{j,h-1})$
    $\boldsymbol{if}\ v_{j,h-1} \geq Q_{j,h}\boldsymbol{then}$
      $FT_{j,h} = ST_{j,h} + Q_{j,h} * P^{i}_{j,h}$ ; $i \in M(O_{j,h})$
    $\boldsymbol{else\ if}$
      $\boldsymbol{if}\ P^{i'}_{j,h-1} \leq P^{i}_{j,h}\boldsymbol{then}$
        $FT_{j,h} = ST_{j,h} + Q_{j,h} * P^{i}_{j,h}$
      $\boldsymbol{else\ if}$
        $\boldsymbol{if}\ v_{j,h-1} \leq Q_{j,h} * (1 - P^{i}_{j,h}/P^{i'}_{j,h-1})\boldsymbol{then}$
          $FT_{j,h} = FT_{j,h-1} + P^{i}_{j,h}$
        $\boldsymbol{else\ if}$
          $FT_{j,h} = ST_{j,h} + Q_{j,h} * P^{i}_{j,h}$
        $\boldsymbol{end\ if}$
      $\boldsymbol{end\ if}$
    $\boldsymbol{end\ if}$
  $\boldsymbol{end\ if}$
$\boldsymbol{end\ if}$

Here,$P_{j,h}^{i}$ is the duration of processing time of$O_{j,h}$ on machine $i$ and $Q_{j,h}$ is the lot size $O_{j,h}$ , $In_{j,h}$ is the inventory $O_{j,h}$, $ST_{j,h}$ is the start time of $O_{j,h}$ ,$FT_{j,h}$ is the finish time of $O_{j,h}$, and At is the availably time. Thus, inventory costs and idle costs can be calculated.

## 5-Computation results

### 5-1- Generating random instances for the problem

Since there is no dataset for this problem (lot size and scheduling of jobs in a flexible environment), it is necessary to generate random instances for the problem in order to evaluate and compare the performance of the proposed algorithm, with the optimal method (Rohaninejad *et al.*, 2015[b]). We consider different problem sizes as is shown in Table 1.

Finally, 18 instances with the Table 1 conditions are generated in small, mediumand large sizesand each instance is labeled with $(\alpha, \beta, \gamma, \delta)$, which $\alpha$ indicates the number of jobs,$\beta$ indicates the total number of operations,$\gamma$ indicates the number of machines and $\delta$is the number of periods in the planning horizon (Rohaninejad *et al.*, 2015[b]).

**Table 1.** Manner of generating random instances

| Parameter | Notation | Generated by |
|---|---|---|
| Demand quantities of every job | $d_{j,t}$ | Normal distribution with the average round ($2.NT$) and variance 2. ($NT$ is the number of macro period). |
| Length of each period | $L_t$ | Normal distribution with the average ($3.O.J/2$) and variance M. ($O$ denotes the total number of operations, $M$ is the number of machines, and J is the number of jobs) |
| Fixed loss of time for production unit of items | $P_{i,j,h}$ | Uniform distribution between [0.1-0.5] |
| Production costs | $f_{i,j,h}$ | Uniform distribution between [0.2-1] |
| Idle costs | $b_{i,t}$ | Uniform distribution between [0.1-0.5] |
| Holding costs | $S_{j,h,t}$ | Uniform distribution between[1-4] |
| Production coefficient | $q_{j,h}$ | 1 |

### 5-2- Parameters calibration

In this paper, the Taguchi method is applied for calibration the parameters of the proposed harmony search algorithm. Taguchi in early 1960s has proposedthis method. The concept of Taguchimethod is based on maximizing a performance measure called signal-to-noise ratio by running a partial set ofexperiments using the orthogonal arrays (Wu & Hamada, 2011). S/N ratio is calculated as follows:

$$\frac{S}{N} ratio = -10\log\left(\frac{1}{N}\sum_{i=1}^{n}\frac{1}{y_i^2}\right) (19)$$

Where$y_i$and $n$ denote the response variables and number of replications, respectively. We consider 4 factors that can have significant effect on the algorithm. These factors and their levels are shows in table 2.

**Table 2.** Factors and their levels for New *HS* algorithm

| Factor | Levels | Number of levels |
|---|---|---|
| HMCR (Harmony Memory Consideration Rate) | {0.1;0.3;0.5;0.7} | 4 |
| PARLM (Pitch Adjustment Rate for Lot size and Machine assignment) | {0.2;0.3;0.4;0.5} | 4 |
| PARS (Pitch Adjustment Rate for Sequence) | {0.2;0.3;0.4;0.5} | 4 |
| BWLM (Bandwidth Lot size and Machine index) | {0.2;0.4;0.6;0.8} | 4 |

The lower the better, the higher the better, and the nominal the better are three type of quality characteristics in the analysis of the S/N ratio (Wu & Hamada, 2011).In this paper, the S/N ratio for the larger the better characteristic is calculated.

The Taguchi design of *mHS* algorithm is$L_{16}(4^4)$. The selected Taguchi design has 16 different combinations of parameter levels. For every trial, three random instances (two small size andone medium size) are considered and each of themis replicated six times in order to obtain more reliable results. The mean response variable is calculated as follows:

$$Objective_{i,j} = \frac{(\sum_{k=1}^{6} objective_{i,j,k}/6) - Min_i}{Min_i} \quad (20)$$

Where $i$ indicate the instance index ($i = 1, \dots ,3$)and $j$ indicate the trial number ($j = 1, \dots ,16$) and $k$ the reapplication number ($k = 1, \dots ,6$).The $objective$ value is converted in to S/N ratio. Figure 9 show the mean S/N ratio for each level of control factors in *mHS* algorithm.

For the analysis of variance (ANOVA), factors are considered as control factors and thelevels with largest values of S/N ratio are selected as the optimal value for each of them. Theoptimal level for factors in Table2 are 0.3, 0.2, 0.5 and 0.4 from top to bottom. Other parameters are shown in table3.

Table 3. The parameter of *mHS* algorithm

| Problem size | Number of Iteration | HMS(harmony memory size) | nNew (number of New Harmony) |
|---|---|---|---|
| Small size | 50-100 | 10 | 10-20 |
| Medium size | 100-200 | 15 | 20-40 |
| Large size | 200-300 | 20 | 40 |

## 5-3- Comparison and Evaluation of the Proposed Solution Methods

In this section, the performance of *HS* and *mHs* for solving the considered problem with Matlab 8.3 software is investigated andthe results are compared with the exact solutions found by CPLEX 12.6 solver. The experiment is run on a *PC* with 2.2*GHz* processor and 8*GB* of *RAM*. The obtained results of solving the small,medium and large size instances are listed in Tables 4 to 6 respectively.

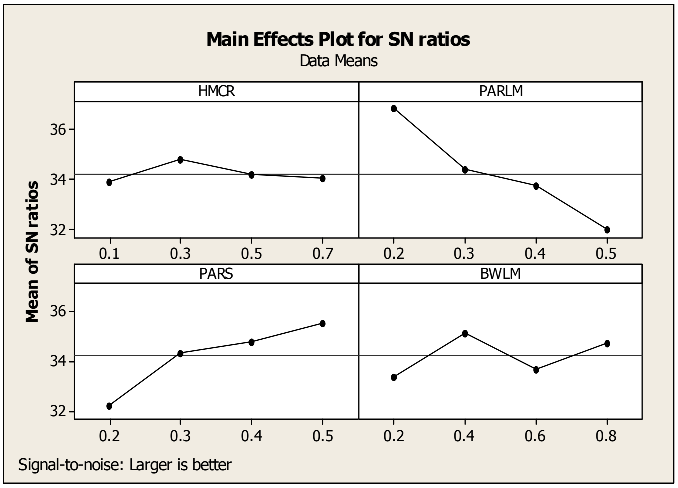



**Figure9.** The mean S/N ratio plot for each levelof the factors in *mHS* algorithm

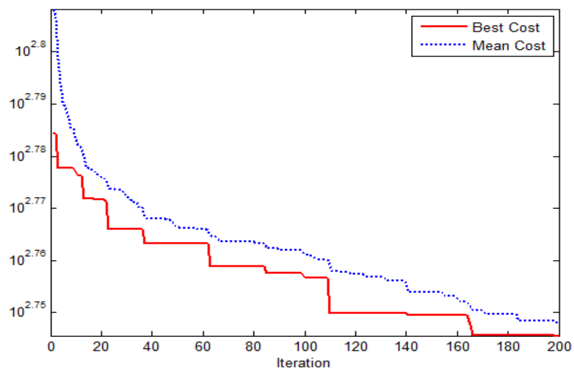



**Figure10.** Decreasing trend of costs, in 200 iterations, solved by *mHS* algorithm.

To evaluate the performance, this criterion will be used: $Gap_a = \frac{Obcetive_a - \mathrm{Min}(Obcetive)}{\mathrm{Min}(Obcetive)}$ (21)

The computation time of the *HS* and *mHS* are smaller than the exact method. Also, in cases where the CPLEX software has not reached the optimal solution after3600 s of computation time, *mHS* achieves a better solution. The total cost for *mHS* algorithm is smaller than those *HS* and exact methods. Thus, the *mHS* has a better performance for all problem sizes.

A general review of the results in Table 4,5 and 6 shows that:

- ✓ The *HS* method can only solve small size problems and also needs more computational time to solve these test problems.
- ✓ The *mHS* algorithm can solve all the test problems.
- ✓ The *mHS* has the ability to obtain solution for all test problems.
- ✓ The *mHS* can find good quality solutions for all size problems. The objective values obtained by *mHS* are better from *HS* and exact method results.

**Table 4.** Details of computational results for small size instance

| Test Problems | Iteration | Time(s) | | | Objective function | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | MIP | HS | mHS | MIP | HS | mHS | %Gap -MIP | %Gap -HS | %Gap -mHS |
| (2,2,2,2) | 50 | 0.11 | 1.15 | 1.94 | 17.03 | 17.03 | 17.03 | 0.0% | 0.0% | 0.0% |
| (2,3,2,3) | 50 | 1 | 3.12 | 4.93 | 59.33 | 59.33 | 59.33 | 0.0% | 0.0% | 0.0% |
| (3,2,2,4) | 100 | 41.48 | 7.8 | 16.5 | 62.3 | 62.3 | 62.3 | 0.0% | 0.0% | 0.0% |
| (3,3,2,3) | 100 | 74.88 | 7.8 | 19.22 | 81.63 | 83.95 | 81.63 | 0.0% | 2.8% | 0.0% |
| (2,3,2,4) | 100 | 243.32 | 7.36 | 16.07 | 91.29 | 93.26 | 91.3 | 0.0% | 2.2% | 0.0% |
| (2,2,2,5) | 100 | 1167.46 | 8.11 | 18.22 | 130.87 | 133 | 130.87 | 0.0% | 1.6% | 0.0% |
| (2,4,2,4) | 100 | >3600 | 9.46 | 22.02 | 130.56 | 133.83 | 126.17 | 3.5% | 6.1% | 0.0% |
| (3,4,2,3) | 100 | >3600 | 9.99 | 28.67 | 101.41 | 99.2 | 97.61 | 3.9% | 1.6% | 0.0% |
| (3,3,2,4) | 100 | >3600 | 9.6 | 25.66 | 133.06 | 128.46 | 125.11 | 6.4% | 2.7% | 0.0% |
| (3,4,2,4) | 100 | >3600 | 12.62 | 35.02 | 160.26 | 154.27 | 146.77 | 9.2% | 5.1% | 0.0% |

**Table 5.** Details of computational results for medium size instance

| Test Problems | Iteration | Time(s) | | | | Objective function | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MIP | | HS | mHS | MIP | HS | mHS | %Gap -MIP | %Gap -HS | %Gap -mHS |
| | | R | B | | | | | | | | |
| (4,4,3,4) | 100 | >3600 | 1170 | 24.32 | 79.62 | 291.79 | 302 | 280 | 4% | 8% | 0% |
| (4,4,3,6) | 100 | >3600 | 576 | 23.48 | 75.81 | 1141.2 | 634.08 | 589.79 | 93% | 8% | 0% |
| (4,5,3,5) | 150 | >3600 | 1564 | 57.93 | 250 | --- | 580.34 | 550.14 | --- | 5% | 0% |
| (4,6,4,5) | 150 | >3600 | 2536 | 76.39 | 335 | 602.76 | 611.06 | 570.6 | 6% | 7% | 0% |
| (4,7,4,5) | 150 | >3600 | 700 | 94.26 | 580 | 1079.6 | 991.53 | 876.43 | 23% | 13% | 0% |

**Table 6.** Details of computational results for large size instance

| Test Problems | Iteration | Time(s) | | | | Objective function | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MIP | | HS | mHS | MIP | HS | mHS | %Gap -MIP | %Gap -HS | %Gap -mHS |
| | | R | B | | | | | | | | |
| (5,8,5,6) | 200 | >3600 | 2978 | 231.2 | 1224 | 1817.2 | 1726.4 | 1640.8 | 11% | 5% | 0% |
| (8,9,6,6) | 200 | >3600 | --- | 583.38 | 3083 | --- | 1902.8 | 1782.2 | --- | 7% | 0% |
| (9,9,7,6) | 200 | >3600 | --- | 824.37 | 4585 | --- | 2243.7 | 2004.3 | --- | 12% | 0% |

In tables 5 and 6, R is real time needed to solve MIP Problem with Cplex solver and B is the best time to find the best feasible solution in PC with 2.2 *GHz* Processor and 8*Gb* of *RAM. (*Because this problem is NP-Hardand Cplex cannot solve this size problem).

Comparing the CPU times of exact solution confirms that computation timegrows exponentially by increasing the dimension of the problem. The presented heuristics and meta-heuristicalgorithms can find the optimal solution for small size problems.The computation time of the *HS* and*mHS* are smaller than the exact method.By comparing the proposed algorithms, it is concluded that the *mHS* algorithm has a moreconvincing performance than other solutions.

## 6- Conclusion and future research directions

This paper proposed an integrated lot sizing and flexible job shop scheduling problem with machine capacity constraints to minimize the total cost; including the inventory, idle machine and production costs.A new mixed integer programming model, with small bucket time approach, based on Proportional Lot sizing and Scheduling Problems (PLSP), is proposed to formulate the problem. In order to solve the problem, a New Harmony search algorithm had been developed.In this method, we improved a New Harmony vector in two phases to enhance search ability. In the first phase, a New Harmony vector lot size-machine assignment for each job and in the last phase, a New Harmony vector sequence is generated.The performance of the proposed algorithm was compared with the exact method and the classic harmony search algorithm in small, medium and large size problems. Computational results and comparisons demonstrated the effectiveness and efficiency of the proposed algorithm.The proposed algorithm can find good quality solutions for all size problems. The objective values obtained by proposed algorithm are better from HS algorithm and exact method results.

This approach can be used in the solving of other problemsof simultaneous optimization of lot-size and scheduling. As for the future work, the model can be extended to include the maintenance periods, the remaining work in the sequence and thelatest time of machines' availability in machine assignment.

### References


Adetunji, O. A. B., &Yadavalli, V. (2012). An integrated utilisation, scheduling and lot-sizing algorithm for pull production.*International Journal of Industrial Engineering: Theory, Applications and Practice*, 19(3).

Akrami, B., Karimi, B., &Hosseini, S. M. (2006). Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: The finite horizon case. *Applied Mathematics and Computation*, 183(1), 634-645.

Almada-Lobo, B., & James, R. J. (2010). Neighbourhood search meta-heuristics for capacitated lot-sizing with sequence-dependent setups.*International Journal of Production Research*, 48(3), 861-878.

Beraldi, P., Ghiani, G., Grieco, A., &Guerriero, E. (2008). Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs.*Computers & Operations Research*, 35(11), 3644-3656.

Drexl, A., &Haase, K. (1995).Proportional lotsizing and scheduling.*International Journal of Production Economics*, 40(1), 73-87.

Fandel, G., &Stammen-Hegene, C. (2006).Simultaneous lot sizing and scheduling for multi-product multi-level production.*International Journal of Production Economics*, 104(2), 308-316.

Fattahi, P., Jolai, F., &Arkat, J. (2009).Flexible job shop scheduling with overlapping in operations.*Applied Mathematical Modelling*, 33(7), 3076-3087.

Fattahi, P., Mehrabad, M. S., &Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3), 331-342.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. S. (2014). Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives.*Journal of*

*Intelligent Manufacturing*, 1-12.

Geem, Z. W., Kim, J. H., &Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2), 60-68.

Gómez Urrutia, E. D., Aggoune, R., &Dauzère-Pérès, S. (2014). Solving the integrated lot-sizing and job-shop scheduling problem.*International Journal of Production Research*, 52(17), 5236-5254.

Haase, K., &Kimms, A. (2000).Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities.*International Journal of Production Economics*, 66(2), 159-169.

Kaczmarczyk, W. (2011).Proportional lot-sizing and scheduling problem with identical parallel machines.*International Journal of Production Research*, 49(9), 2605-2623.

Karimi, B., Ghomi, S. F., & Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31(5), 365-378.

Karimi-Nasab, M., Seyedhoseini, S. M., Modarres, M., &Heidari, M. (2013). Multi-period lot sizing and job shop scheduling with compressible process times for multilevel product structures. *International Journal of Production Research*, 51(20), 6229-6246.

Kovács, A., Brown, K. N., &Tarim, S. A. (2009).An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups.*International Journal of Production Economics*, 118(1), 282-291.

Mahdieh, M., Bijari, M., & Clark, A. (2011).Simultaneous Lot Sizing and Scheduling in a Flexible Flow Line.*Journal of Industrial and Systems Engineering*, 5(2), 107-119.

Modrak, V. (2012). Alternative Constructive Heuristic Algorithm for Permutation Flow-Shop Scheduling Problem with Make-Span Criterion. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 19(7).

Morais, M. D. F., GodinhoFilho, M., &Boiko, T. J. P. (2014). Hybrid flow shop scheduling problems involving setup considerations: a literature review and analysis. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 20(11-12).

Petrovic, S., Fayad, C., Petrovic, D., Burke, E., & Kendall, G. (2008).Fuzzy job shop scheduling with lot-sizing.*Annals of Operations Research*, 159(1), 275-292.

Pinedo, M. L. (2012). Scheduling: theory, algorithms, and systems.Springer Science & Business Media.

Ponnambalam, S. G., & Reddy, M. (2003). A GA-SA multiobjective hybrid search algorithm for integrating lot sizing and sequencing in flow-line scheduling.*The International Journal of Advanced Manufacturing Technology*, 21(2), 126-137.

Ramezanian, R., &Saidi-Mehrabad, M. (2013). Hybrid simulated annealing and MIP-based heuristics for stochastic lot-sizing and scheduling problem in capacitated multi-stage production system. *Applied Mathematical Modelling*, 37(7), 5134-5147.

Ramezanian, R., Saidi-Mehrabad, M., &Fattahi, P. (2013). MIP formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints. *Journal of Manufacturing Systems*, 32(2), 392-401.

Rohaninejad, M., Kheirkhah, A. S., VahediNouri, B., &Fattahi, P. (2015[a]). Two hybrid tabu search–firefly algorithms for the capacitated job shop scheduling problem with sequence-dependent setup cost. *International Journal of Computer Integrated* Manufacturing, 28(5), 470-487.

Rohaninejad, M., Kheirkhah, A., &Fattahi, P. (2015[b]). Simultaneous lot-sizing and scheduling in flexible job shop problems. *The International Journal of Advanced Manufacturing Technology*,78(1-4), 1-18.

Rohaninejad, M., Kheirkhah, A., Fattahi, P., &Vahedi-Nouri, B. (2015[c]).A hybrid multi-objective genetic algorithm based on the ELECTRE method for a capacitated flexible job shop scheduling problem.*The International Journal of Advanced Manufacturing Technology*, 77(1-4), 51-66.

Sikora, R., Chhajed, D., & Shaw, M. J. (1996).Integrating the lot-sizing and sequencing decisions for scheduling a capacitated flow line.*Computers & Industrial Engineering*, 30(4), 659-679.

Toledo, C. F. M., França, P. M., Morabito, R., &Kimms, A. (2009). Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem.*International Journal of Production Research*, 47(11), 3097-3119.

Wang, L., &Zheng, D. Z. (2001).An effective hybrid optimization strategy for job-shop scheduling problems.*Computers & Operations Research*, 28(6), 585-596.

Wang, L., Pan, Q. K., &Tasgetiren, M. F. (2010).Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms.*Expert Systems with Applications*, 37(12), 7929-7936.

Wang, L., Pan, Q. K., &Tasgetiren, M. F. (2011).A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem.*Computers & Industrial Engineering*, 61(1), 76-83.

Weidenhiller, A., &Jodlbauer, H. (2009). Equivalence classes of problem instances for a continuous-time lot sizing and scheduling problem. *European Journal of Operational Research*, 199(1), 139-149.

Wu, C. J., & Hamada, M. S. (2011). *Experiments: planning, analysis, and optimization* (Vol. 552). John Wiley & Sons.

Zhu, C. (2012). Applying Genetic Local Search Algorithm to Solve the Job-Shop Scheduling Problem.*International Journal of Industrial Engineering: Theory, Applications and Practice*, 19(9).