International Journal of Science and Engineering Applications
Volume 2 Issue 6, 2013, ISSN-2319-7560 (Online)

# Solving Multi-level, Multi-product and Multi-period Lot Sizing and Scheduling Problem in Permutation Flow Shop

M. Babaei
Department of Industrial Engineering
Kharazmi University
Karaj, Iran

M. Mohammadi
Department of Industrial Engineering,
Kharazmi University
Karaj, Iran

S. M. T. Fatemi Ghomi
Department of Industrial Engineering,
Amirkabir University, Iran

M. A. Sobhanallahi
Department of Industrial Engineering,
Kharazmi University
Karaj, Iran

**Abstract**: In this paper, a new model of capacitated lot sizing and scheduling in a permutation flow shop is developed. In this model demand can be totally backlogged. Setups can be carryover and are sequence-dependent. It is well-known from literatures that capacitated lot sizing problem in permutation flow shop systems are NP-hard. This means the model is solved in polynomial time and metaheuristics algorithms are capable of solving these problems within reasonable computing load. Metaheuristic algorithms find more applications in recent researches. On this concern this paper proposes two evolutionary algorithms, one of the most popular namely, Genetic Algorithm (GA) and one of the most powerful population base algorithms namely, Imperialist Competitive Algorithm (ICA). The proposed algorithms are calibrate by Taguchi method and be compared against a presented lower bound. Some numerical examples are solved by both the algorithms and the lower bound. The quality of solution obtained by the proposed algorithm showed superiority of ICA to GA.

**Keywords**: permutation flow shop; evolutionary algorithms; lot sizing and scheduling

## 1. INTRODUCTION

The multilevel lot sizing problem concerns how to determine the lot size for producing or procuring an item at each and sequencing is to determine job ordering on each level. The objective of lot sizing and sequencing generally is to minimize the sum of the total setup cost and inventory holding cost. Lot sizing and sequencing problem plays an important role in the efficient operation of modern manufacturing and assembly processes.

The flow shoplot sizing has been a very extensively researched area since the seminal paper of Johnson [1]. In the flowshop problem (FSP) a set ofunrelated jobs are to be processed on a set of machines. These machines are disposed in series and each job has to visit all of them in the same order. A special case of flow shop that assumes the same order of products in all machines is called permutation flow shop. In this paper we consider a permutation flow shop problem with setup carryover, setup sequence-dependent and backlogging.

In highly capacitated environments as well as in many real-life situations, the inclusion of back orders is crucial because otherwise, no feasible plan would exist and the respective result that no feasible solution can be found is of minor importance in practical settings. On the other hand, in many real-life manufacturing environment the capacity of the machines are limited, or for cost saving reasons, it might be useful to produce a product volume in a period other than its demand period to save setup time and costs. In traditional lot sizing models producing of a product in a period before its delivery to the customer is permitted. In this case, inventory cost occurs. In our case, it is also possible that the product cannot be delivered on time. It is then backlogging occurs and backlogging costs are incurred for every unit at period of the delay. While only few lot sizing approaches consider the possibility of back ordering, it is of great importance in practical settings: If capacity is limited, some productsmay have to be backlogged [2].

Quadt and Kuhn [3] investigated a capacitated lot sizing and scheduling problem with setup times, setup carryover, backorders, and parallel machines. They formulated a mixed integer formulation of the problem and a new solution procedure. The solution procedure was based on a novel “aggregate model” which uses integer instead of binary variables. Song and Chan [4] considered a single item lot sizing problem with backlogging on a single machine at a finite production rate. The objective function was to minimize the total cost of setup, stockholding and backlogging to satisfy a sequence of discrete demands.Other researchers have considered backlogging including Wolsey and Pochet [5], Cheng et al. [6]and Karimi et al.[7].

Graham et al. [8] showed that the permutation flow shop scheduling problem is strongly *NP*-complete. Since then many researches have been attracted to develop heuristic and metaheuristic algorithms for these problems. Among researchers that employed metaheuristic we can cite to Mohammadi et al. [9]. They employed a GA as a solution approach. Their proposed algorithm was used for a simultaneous lot sizing and sequencing problem in permutation flow shops involving sequence-dependent setups and capacity constraints. Ruiz et al. [10]proposed new genetic algorithms for solving the permutation flow shop scheduling. The minimization of the total completion time or makespanwas considered as theoptimization criterion. They used new genetic operators, advanced techniques like hybridization with local search and an efficient population initialization as well as a new generational scheme. The following is brief review of studies that have considered GA as a solution approach for a permutation flow shop problem. The authors employed their proposed GA for different optimization criteria. A GA was used inAllada and Ruiz [11]study with total tardiness minimization criterion. Tseng and Lin [12]have minimized makespan in a permutation flow shop problem.TavakkoliMoghaddam et al. [13]employed a GA to minimize the makespan. Goren [14]provided an overview of recent advances in the field in order to highlight the many ways GAs can be applied to various lot sizing models.

Despite the fact that GA is one of the most popular algorithm,other metaheuristic algorithms have been used broadly by authors. Among metaheuristics, ICA is a novel population-based evolutionary algorithm proposed by Atashpaz-Gargari and Lucas [15].ICA is a novel socio-politically motivated metaheuristic algorithm inspired by imperialist competition. The results show that the algorithm performs significantly better than existing algorithms like genetic algorithm (GA), simulated annealing (SA), tabu search (TS), and particle swarm optimization (PSO) [16].So we propose a novel imperialist algorithm (ICA) that employed some genetic operators during local search.Many researchers have employed ICA as a solution approach for flow shops. As instances, Attar et al. [17]proposed a novel imperialist competitive algorithm to solve flexible flow shop scheduling problem the optimization criterion was minimization maximum completion time.Shokrollahpour et al. [18] used ICA for a two-stage assembly flow shop scheduling problem with minimization of weighted sum of makespan and mean completion time as the objective function. Rajabioun et al. [19],Khabbazi et al. [20],Kaveh and Talatahari [21] Lucaset al. [22],Nazari-Shirkouhi et al. [23] andSarayloo and Tavakkoli-Moghaddam[24] are other related study that employed ICA.

In literature, other metaheuristics have been employed by researchers. The following shows recent studies about using different kind of metaheuristics in permutation flow shop. Quan et al. [25]used PSO in permutation flow shop with makespan criterion. Ant Colony Optimization is employed byUdomsakdigool and Khachitvichyanukul [26].

As mentioned above, despite backlogging importance in practical settings, only few researchers have addressed capacitated lot sizing problems with back ordering especially in case of permutation flow shop. To the best of our knowledge, this is the first paper that deals with setup sequence-dependent, setup carryover and backlogging in a multi-level, multi-machine and multi-period permutation flow shop environment and proposes two metaheuristics to solve the model and develops a lower bound to compare algorithms.

This paper is organized as follows:in next section, notations used in the formulation are described and a lower bound is presented. In section 3 the genetic algorithm and in section 4 imperialist algorithm are proposed. In subsequence section, the algorithms are calibrated by Taguchi method and theperformance of proposed algorithms is evaluated. Finally, Section 6 is devoted to conclusions and recommendation for future studies.

## 2. PROBLEM FORMULATION

The following notations are used in the model:

### 2.1. Notations and assumptions

*2.1.1 Indices*

$i, j, k$ Index of production type

$n$Index of product type

$n'$ Designation for a specific setup number

$m$ Index of level of production

$t$ Index of period

*2.1.2 Parameters*

$T$ Planning horizon

$N$ Number of different products

$M$ Number of production levels/number of machines

$bigM$ A large real number

$C_{m,t}$ Available capacity of machine $m$in period $t$ (in time units)

$d_{j,t}$ External demand for product $j$ at the end of period $t$ (in units of quantity)

$h^{+}_{j,m}$ Storage costs unit rate for product $j$ in level $m$.

$h^{-}_{j,t}$ Shortage costs unit rate for product $j$ at the end of period $t$.

$b_{j,m}$ Capacity of machine $m$ required to produce a unit of product (or shadow product) $j$ (in time units per quantity units).

$P_{j,m,t}$ Production costs to produce one unit of product $j$ on machine $m$ at period $t$ (in money unit per quantity unit).

$S_{i,j,m}$ Sequence-dependent setup time for the setup of the machine $m$ from production of product $i$ to production of product $j$ (in time units); for$i \neq j, S_{i,j,m} \geq 0$and$i = j, S_{i.j.m} = 0$.

$W_{i,j,m}$ Sequence-dependent setup cost for the setup of the machine $m$ from production of product $i$ to production of product $j$ (in money units); for $i \neq j, W_{i,j,m} \geq 0$ and $i = j, W_{i.j.m} = 0$.

$j_{0m}$ The starting setup configuration on machine $m$.

*2.1.3 Decision variables*

$I^{+}_{j,m,t}$ Stock of product $j$ at level $m$ at the end of period $t$.

$I^{-}_{j,t}$ Shortage of product $j$ at the end of period $t$.

$y^{n}_{i,j,t}$ Binary variable, which indicates whether the $n$th setup on machines at period $t$ isfrom product $i$ to product $j$ ($y^{n}_{i,j,t} = 1$) or not ($y^{n}_{i,j,t} = 0$).

$x^{n}_{i,j,m}$ Quantity of product $j$ produced after $n$th setup on machine $m$ at period $t$.

$q^n_{i,j,m}$ Shadow product: the gap (in quantity units) between *n*th setup (to product *j*) on machine *m* at period *t* and its related production in order to ensure that direct predecessor of this product (production of product *j* on machine *m* at period *t*) has been completed.

To formulate this model the following assumptions are considered:

- Several products are produced in a flow shop environment and each product can be produced only on one machine at the same time,
- Inventory cost incurred when a product unit is hold between a particular period,
- If the product cannot be delivered on time shortage cost is incurred,
- Setup times reducing machine capacity and each machine is constrained in capacity
- Setups are sequence-dependent and must be complete in a period.
- There must be precisely N (number of products) setups in each period on each machine, even if a setup is just from a product to itself, with respect to this issue that setup time (and cost) from a product to itself is zero

The mathematical model in this paper is described on the basis of the above assumptions and notations

## 2.1. Mathematical formulation

Capacitated lot sizing focuses on how to make lot sizing planning and sequencing focuses on the order of each product should be produced to minimize total cost. The objective function is to find an optimal lot sizing and sequencing that minimize setup, inventory, production and backlogging costs.

$$min \sum_{n=1}^{N}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T} W_{i,j,m}.y^n_{i.j,t} + \sum_{n=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T} P_{j,m,t}.x^n_{j,m,t} + \sum_{j=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T} h^+_{j,m}.I^+_{j,m,t} + \sum_{j=1}^{M}\sum_{t=1}^{T} h^-_{j,t}.I^-_{j,t} \quad (1)$$

*Subject to*

$$d_{j,t} = I^+_{j,M,t-1} + \sum_{n=1}^{N} x^n_{j,M,t} - I^+_{j,M,t} - I^-_{j,t-1} + I^-_{j,t};\quad j = 1,\dots,N,\quad t = 1,\dots,T \quad (2)$$

$$I^+_{j,m,t-1} + \sum_{n=1}^{N} x^n_{j,m,t} = I^+_{j,m,t} + \sum_{n=1}^{N} x^n_{j,m+1,t};\quad j = 1,\dots,N,\quad m = 1,\dots,M-1,\quad t = 1,\dots,T \quad (3)$$

$$\sum_{n=1}^{n'}\sum_{i=j0}^{N}\sum_{j=1}^{N} y^n_{i,j,t}.S_{i,j,m} + \sum_{n=1}^{n'}\sum_{j=1}^{N} b_{j,m}.q^n_{j,m,t} + \sum_{n=1}^{n'}\sum_{j=1}^{N} b_{j,m}.x^n_{j,m,t} \le$$
$$\sum_{n=1}^{n'}\sum_{i=j0}^{N}\sum_{j=1}^{N} y^n_{i,j,t}.S_{i,j,m+1} + \sum_{n=1}^{n'}\sum_{j=1}^{N} b_{j,m+1}.q^n_{j,m+1,t} + \sum_{n=1}^{n'-1}\sum_{j=1}^{N} b_{j,m+1}.x^n_{j,m+1,t};\quad n' = 1,\dots,N,\quad m = 1,\dots,M-1,$$
$$t = 1,\dots,T \quad (4)$$

$$\sum_{n=1}^{N}\sum_{i=j0}^{N}\sum_{j=1}^{N} y^n_{i,j,t}.S_{i,j,m} + \sum_{n=1}^{N}\sum_{j=1}^{N} b_{j,m}.x^n_{j,m,t} + \sum_{n=1}^{N}\sum_{j=1}^{N} b_{j,m}.q^n_{j,m,t} \le C_{m,t};\quad m = 1,\dots,M, t = 1,\dots,T \quad (5)$$

$$x^n_{j,m,t} \le \left(\frac{C_{m,t}}{b_{j,m}}\right). \sum_{i=j0, i\neq j(for(n>1)}^{N} y^n_{i,j,t},\quad n = 1,\dots,N,\quad j = 1,\dots,N, m = 1,\dots,M,\quad t = 1,\dots,T \quad (6)$$

$$q^n_{j,m,t} \le \left(\frac{C_{m,t}}{b_{j,m}}\right). \sum_{i=j0}^{N} y^n_{i,j,t},\quad n = 1,\dots,N,\quad j = 1,\dots,N,\quad m = 1,\dots,M,\quad t = 1,\dots,T \quad (7)$$

$$y^1_{j,i,1} = 0\ \ j \neq j_{0m},\quad i = 1,\dots,N \quad (8)$$

$$\sum_{i=1}^{N} y^1_{j_{0m},i,1} = 1, \quad (9)$$

$$\sum_{j=j0}^{N} y^n_{j,i,t} = \sum_{k=1}^{N} y^{n+1}_{i,k,t}\quad i = 1,\dots,N,\quad n = 1,\dots,N-1,\quad m = 1,\dots,M,\quad t = 1,\dots,T \quad (10)$$

$$y^n_{i,j,t} = 0\ or\ 1 \quad (11)$$

$$I_{j,m,t}^{+}, I_{j,t}^{-}, x_{j,m,t}^{n}, q_{j,m,t}^{n} \geq 0 \tag{12}$$

$$I_{j,m,0}^{+} = 0, \quad j = 1, \dots, N, \quad m = 1, \dots, M \tag{13}$$

In this model, the objective function is Equation (1). The backlogging or storage at the end of each period is considered by Equation (2). Constrain (3) ensures total of in-flows to each node is equal to of out-flows from that node. Equation (4) ensures within one period each typical product $j$ one machine $m$ is produced before its direct successor. The capacity constraints of machines are considered by Equation (5). Equation (6) respects setups in production process. Equation (7) indicates the relationship between shadow products and setups. Constraints (8) and (9) ensure that for each machine, the first setup at the beginning of the planning horizon is from a defined product. Equation (10) represents the relationship between successive setups. The type of variables is defined by Equations (11) and (12) and finally Equation (13) indicates that at the end of planning horizon there is no on-hand inventory.

## 2.3. Lower bound

In this section we present a lower bound that developed by Mohammadi et al. [27]. We first relax binary variables to continuous variables that fall in [0, 1].

Then we add following equation to relaxed model:

$$\sum_{i=j0}^{N} y_{i,j,t}^{1} + \sum_{i=1,i\neq j}^{N} \sum_{n=2}^{N} y_{i,j,t}^{n} = a_{j,t} \tag{14}$$

In this equation $a_{j,t}$ is binary variable.

Equation (14) was proved that is valid to model. We refer the proof of this equation to Mohammadi et al [27].

## 3. GENETIC ALGORITHM

GAs are probabilistic search optimization algorithms that were inspired by the process of natural evolution and the principles of survival of the fittest [28].Genetic Algorithms (GAs) can be employed to find a near-optimal solution for *NP*-hard problems. GAs evolves a population of individuals according to the progress of algorithm to reach a good solution. Genetic operators (such as natural selection, mutation, and cross over) manipulate individuals in a population of solution over several iterations to improve their fitness.The algorithm generates a new candidate pool of solutions iteratively from the presently available solutions and replaces some or all of the existing members of the current solution tool with the newly created feasible solutions.

We first present a simple and effective heuristic to generate initial solution and then discuss the issue of encoding in our case aninteger array representation. We then turn to the evolutionary stages of the algorithm and the specific genetic operators that have been designed to increase search efficiency.

*Initial Population*

A genetic algorithm starts with popularly an initial population of feasible solutions. Initial population can affectsthe performance of GAs. So that a simple and effect heuristic is used for $t = 1$to $T$and is described as follows:

(1) The products are sorted in the decreasing order of $W_{j,m} = \sum_{i=1}^{N} W_{i,j,m}; j = 1, \dots, N$.

(2) Let $[i]$ indicate the $i$th product in an ordered sequence in this heuristic.

For $[i] = 1$ to $N$:

(a) Consider inserting product $[i]$ into every position.

(b) Calculate the sum of setup costs for all products scheduled so far using the actual setup costs.

(c) Place product $i$ in the position with the lowest resultant sum of setup costs.

With using this method, $M$ different initial populations is produced (for $m = 1, \dots, M$), the remaining initial populations have been generated randomly.In this way, binary variables are coded in the form of matrices with $N \times T$ dimensions.

In Figure1 a sample chromosome with T =3, N =3 is depicted.

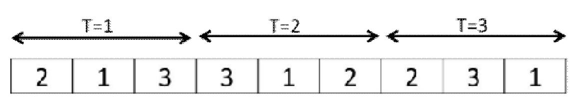


Figure 1. A chromosome representation

In Figure 1 an encoded binary variables have been shown. The corresponding decoded binary variablesin this chromosomeduring period T=1 are, $y_{0,2,1}^{1} = y_{2,1,1}^{2} = y_{2,3,1}^{3} = 1$ and the corresponding decoded binary variables to this chromosome during period T=2 are, $y_{0,3,2}^{1} = y_{3,1,2}^{2} = y_{1,2,2}^{3} = 1$and finally for period T=3, $y_{0,2,3}^{1} = y_{2,3,3}^{2} = y_{3,1,3}^{3} = 1$and other binary variables would get value 0.With encoding of the binary variables, we are able to employ crossover and mutation operators more efficientlyand more effectively than noncoding chromosomes.

*Fitness function*

The fitness value of each chromosome has been calculated by solving the corresponding problem.

*Selection operator*

The requirement parents for using of crossover have been obtained by one of the five selection method, Deterministic Sampling (A), Random Sampling (B), Roulette Wheel (C), Ranking (D) andTournament(E).

*Crossover operation and mutation operator*

Several crossover operators have been proposed in reference [29]. Similar job two point crossover has been used in this research. In order to produce small perturbations on chromosomes to promote diversity of the population, a shift mutation operator has been used in this article.Crossover and mutation probability must be determined during parameters calibration.

*Population replacement*

Chromosomes for the next generation are selectedfrom the enlarged population. The best pop_sizechromosomes of the enlarged population have beenselected for the next generation.

*Termination criterion*

The algorithm must terminate according to a criterion. This criterion is specified by reaching to maximum number of iteration *it_max*.

## 4. IMPERIALIST COMPETITIVE ALGORITHM

ICA is a novel population-based evolutionary algorithm proposed by Atashpaz-Gargari and Lucas [15]. The ICA initiates with an initial population, like most evolutionary algorithms. Each individual of the population is called a ‘country’ equivalent ‘chromosome’ in GA. Some of the most powerful countries are chosen to be the imperialiststates and the other countries constitute the colonies of theseimperialists. All the colonies of initial countries are partitioned among the mentioned imperialists based on theirpower. Equivalent of fitness value in the GA, the power ofeach country, is conversely proportional to its cost. Anempire is constituted from the imperialist states with theircolonies [30].

After all empires were formed, the competition between countries starts. First, the colonies in each of empires start moving toward their imperialist. During this movement, if the colony gets better cost function than its imperialist does, they will exchange their positions and the algorithm will continue with the new imperialist. The power of each empire is calculated by imperialist cost function and colonies. The empire which is weaker than the others loses its colonies. Each imperialist attempts to gain the colonies of other empires. The most powerful empires have a more chance to gain the colonies from the weakest empires. The more powerful an empire is, the more likely it will possess the weakest colony of the weakest empire (Imperialistic competition)

During the competition weak imperialists will lose their weakest colony gradually. When an empire loses all of its colonies, it will be eliminated from the population. In fact the empire collapses. The final level of imperialist rivalry is when there is only one empire in the world. The main steps of ICA are described as follows:

*Step 1 Generating of Initial countries*

Each individual of the population is called a ‘country’ equivalent ‘chromosome’ in GA. Each country denotes a socio-political characteristic in that country such as culture, language, business, economic policy and etc. The socio-political characteristic in countries is the same different type of variables. There are two different types of variables, continuous variables $(x, q, I^{+}, I^{-})$ and binary variables $(y)$. Each country consists of five variables, $x, q, I^{+}, I^{-}$and $y$ where all of these variables must be optimized.

Initial values of continuous variables are generated randomly by uniform distribution function. To generate initial value for binary variable, we use a simple and effective heuristic which has been presented by Mohammadi et al. [27].

*Step 2 Generating of Initial imperials*

A set of the most powerful countries form imperialists and the rest weaker countries are colonies of imperialists. The power of each country is calculated based on the objective function.

*Step 3 Assimilation of colonies*

Assimilation has been modeled by moving all the colonies toward the imperialist. Each country (colony) has different socio-political characteristics (variables), so every socio-political characteristic (variables) could moves toward the related socio-political of imperialist in different ways. Continuous variables of colonies move toward related continuous variables of its imperialist and binary variables move toward binary variables of its imperialist.

The assimilation of continuous variables is modeled by moving the colony toward the imperialist by $x$units $x \sim U(0, \beta \times d)$. Where $\beta > 1$and$\theta \sim U(-\gamma, \gamma)$. $d$is distance between colony and the its imperialist.

The movement of binary variables is accomplished by crossover operation, like crossover operator in genetic algorithms. Crossover allows exchanging information between different solutions (chromosomes) so it is useful to assimilate binary variables.

*Step 4 Revolution*

The revolution increases the exploration of the algorithm and prevents the early convergence of countries to local minimums. A very high value of revolution decreases the exploitation power of algorithm and can reduce its convergence rate [31].In each iteration, some of the colonies are chosen and their positions are exchanged. This mechanism is similar to mutation process in genetic algorithm for creatingdiversification in solutions. Mutation increases the variety in the population, so this operator is used for creating a revolution in binary variables.

*Step 5 Exchange the colony with imperialist*

During assimilation and revolution, a colony may get to a situation with lower cost than the imperialist. In this case, the imperialist and the colony change their positions.

*Step 6 Imperialistic competition*

To start the competition, after selecting the weakest colony, the possession probability of each empire must be found. The normalized total cost of an empire is simply obtained by

$$NTC_n = TC_n - \max\{TC_i\}$$

Where, $NTC_n$and $TC_n$ are the total cost and the normalized total cost of $n$th empire, respectively.

The total power of an empire is mainly contributed by the power of imperialist country. It is clear that the power of an empire includes the imperialist power and their colonies.

$$TC_n = cost\{imperialist_n\}C + \rho * mean\{cost(colonies\ of\ empire_n)\}$$

Where $\rho$ is a positive small number.The possession probability of each empire is given by

$$p_{p_n} = \left|\frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i}\right|$$

Roulette wheel method was used for assigning the mentioned colony to empires.

*Step 7 Elimination of powerless empires.*

During the competition weak imperialists will lose their weakest colony gradually. When an empire loses all of its colonies, it collapses. At the end just one imperialist will remain. This is the optimum point.

*Step 8 Stop criterion*

In such an ideal new world, all the colonies will have the same positions and same costs and they will be controlled by an imperialist with the same position and cost as themselves. In such a world, there is no difference not only among colonies, but also between colonies and imperialist [32]in this situation; the algorithm has reached the global solution.

Stopping criterion in proposed algorithm is to get the maximum decades (maximum iteration).

## 5.PARAMETER CALIBRATION AND COMPUTATIONAL TESTING

Parametercalibration andcomputationalexperiments are key steps in the developmentof any algorithm.Conventionally, setting parameters relies on a trial and-error procedure. However, this procedure cannot determine optimal parameter settings and consumes considerable time [33]. In this paper, we employed the Taguchi Methodology to optimize the parameters of the algorithms via systematic experiments.

### 5.1. Parameter calibration

Taguchi [34] developed a family of fractional factorial experiment (FFE) matrices that ultimately lessens the number of experiments, but still provides adequate information. Full fractional experiment is not an effective approach when the number of factors becomes large, so in our case because of the large number of factors and few levels for each factor we use Taguchi method to calibrate the parameters.

A transformation of the repetition data is created toanother value by Taguchi which is the measure of variation. The transformation is the signal-to-noise (S/N) ratio. The S/N ratio is obtained by following equation:

$$\frac{S}{N}\ ratio = -10\log(objective\ fnction)^2$$

Where “signal” describes the desirable value, in our study we use the following performance measure as desirable value.

$$\left(\left(\sum_{i=1}^{15} Metaheuristic_{sol_i} - LB_i \Big/ LB_i\right) \times 100\right) \Big/ 15$$

Where $Metaheuristic_{sol_i}$ is the obtained solution by the algorithm (GA or ICA) and $LB_i$ is the solution obtained by the lower bound. Note that each problem runs 15 times and the average of this runs consider as corresponding desirable value for the problem. It is obvious that the smaller value of the performance measurement shows that the metaheuristic is more efficient.

The term “noise” specifies the undesirable value (standard deviation). The S/N ratioindicates the amount of variation present in the response variable. Here, maximization of the signal-to-noise ratio isdesirable (i.e., is the goal).

In order to calibrate the parameters of the algorithms we determine the level of each parameter. Table 1 shows the level of each parameter for two metaheuristics. Since increase in number of products (N) leads to polynomial increase in computational timeso that to make a fair calibration, we divide number of population and maximum iteration bynumber of products (N).

**Table 1. Factor levels**

| | Parameter Level | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |
|---|---|---|---|---|---|---|
| | Selection Type | A | B | C | D | E |
| | Crossover Probability | 0.1 | 0.2 | 0.3 | 0.4 | .5 |
| GA | Mutation Probability | 0.05 | .10 | .15 | .20 | .25 |
| | Number of Population | 500/N | 600/N | 750/N | 900/N | 1000/N |
| | Maximum Iteration | 500/N | 600/N | 750/N | 900/N | 1000/N |
| | Number of Population | 500/N | 600/N | 750/N | 900/N | 1000/N |
| | Number of Imperialist | 5 | 7 | 10 | 12 | 15 |
| ICA | Maximum Iteration | 0.5 | 0.7 | 1 | 1.2 | 1.5 |
| | Revolution Probability | 500/N | 600/N | 750/N | 900/N | 1000/N |
| | $\rho$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |

In this case, Taguchi method needs 25 experiments for each algorithm.As mentioned above, for each 25 experiment 15 independent runs are carried. The termination criterion of maximum elapsed CPU time ist=7200S [9]The required parameters for these problems are extracted from the following uniform distributions: $c \approx U(5,10), d \approx U(0.5,1), h^{+} \approx U(0.05,0.1), h^{-} \approx U(1,5), b \approx U(0.02,0.04), p \approx U(0.02,0.04), s \approx U(100,1100)$

In order to conduct the experiments, we implemented GA and ICA examples in MATLAB run on a PC with a 2.27 GHz Intel Core i5 processor and 3 GB RAM memory and analyzed the result by Minitab 16 software.Figure2 and 3 show the average S/N ratio obtained at each level for ICA and GA.

According to Figure 2 and Figure 3 the optimal levels of factors have been indicated in Table 2.

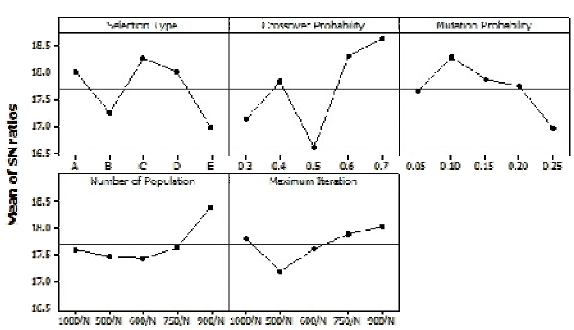


Figure 2. Main effect plot for S/N ratios for GA

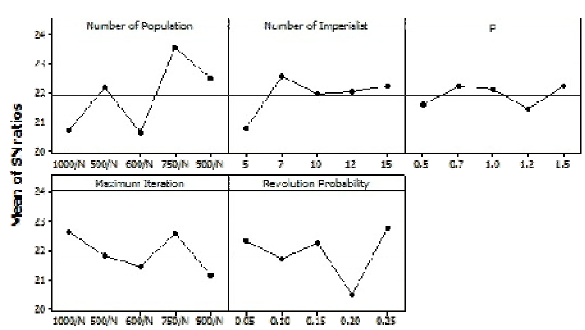


Figure 3. Main effect plot for S/N ratios for ICA

**Table 2. The optimal levels of factors**

| | Parameter Level | Optimum |
|---|---|---|
| GA | Selection Type | C |
| | Crossover Probability | 0.7 |
| | Mutation Probability | 0.10 |
| | Number of Population | 900/N |
| | Maximum Iteration | 900/N |
| ICA | Number of Population | 750/N |
| | Number of Imperialist | 7 |
| | Maximum Iteration | 750/N |
| | Revolution Probability | 0.25 |
| | $\rho$ | 0.7 |

## 5.2. Comparison of the algorithms

In this section, in order to evaluate and compare the performance of two proposed, we consider different problem sizes.

For each problem set, 15 independent instances are randomly generated (225 problems) and the required parameters for these problems are extracted from the following uniform distributions:

$$c \approx U(5,10), d \approx U(0.5,1), h^{+} \approx U(0.05,0.1), h^{-} \approx U(1,5), b \approx U(0.02,0.04), p \approx U(0.02,0.04), s \approx U(100,1100)$$

For each of the 15 instances, 15 independent runs are carried out for each algorithm within a reasonable CPU time, 7200 s.We obtain the mean of 15 instances as the response variable of each instance. This response is used to compare two algorithms. Problems have been solved in MATLAB run on a PC with a 2.27 GHz Intel Core i5 processor and 3 GB RAM memory.The computed results are reported in Table 3.

**Table 3. Computational results**

| Problem set | Dimension of problems ($N \times M \times T$) | GA | ICA |
|---|---|---|---|
| 1 | $2 \times 2 \times 2$ | 16.72% | 9.66% |
| 2 | $3 \times 3 \times 3$ | 16.76% | 9.42% |
| 3 | $4 \times 4 \times 4$ | 9.38% | 6.46% |
| 4 | $5 \times 5 \times 5$ | 15.01% | 11.69% |
| 5 | $6 \times 6 \times 6$ | 15.02% | 8.55% |
| 6 | $7 \times 7 \times 7$ | 15.84% | 7.48% |

| | | | |
|---|---|---|---|
| 7 | $8 \times 8 \times 8$ | 14.85% | 10.07% |
| 8 | $9 \times 9 \times 9$ | 13.71% | 9.20% |
| 9 | $10 \times 10 \times 10$ | 14.04% | 10.12% |
| 10 | $11 \times 11 \times 11$ | 10.95% | 6.97% |
| 11 | $12 \times 12 \times 12$ | 10.32% | 6.60% |
| 12 | $13 \times 13 \times 13$ | 12.17% | 6.92% |
| 13 | $14 \times 14 \times 14$ | 11.76% | 8.99% |
| 14 | $15 \times 15 \times 15$ | 13.41% | 8.58% |
| 15 | $16 \times 16 \times 16$ | 15.08% | 8.07% |

We are now employing a 95% confidence level and we are using Tukey HSD confidence intervals to compare algorithms with Minitab 16 Software. The result has been showed in Figure 4. From the Figure 4 it is clear that the proposed ICA algorithm is statistically better than the proposed GA algorithm.

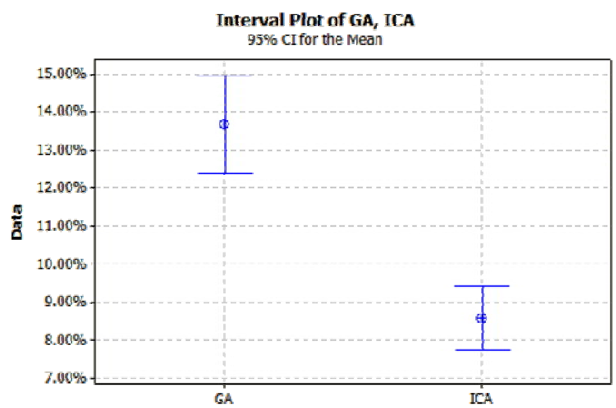


Figure 4:Tukey HSD confidence intervals

## 6. CONCLUSION

This paper studies the permutation flow shop lo sizing and scheduling problem and developed a new model for the problem under sequence-dependent and carryover setups with considering backlogging.

To solve the problem, two metaheuristics was proposed namely, GA and ICA. Since the parameters of any algorithm has significant effect onalgorithms performance, we use a fractional factorial experiment namely, Taguchi method. In order to evaluate the effectiveness and robustness of the proposed GA and ICA, we carried out a comparison between the algorithms. In this context, we presenteda lower bound and compared the algorithms against it. The distance betweenthe algorithms and the lower bound was calculated. Base on the results, Tukey HSD confidence intervals was employed to determine which algorithm is statistically superior to the other one. The results showed the ICA outperforms the GA.

As a direction for future research, it would be interesting to develop other metaheuristic, like Particle Swarm Optimization, Harmony Search and Honey Bee Algorithm. As an additional contribution, developing of the single objective into multi objective models can be considered.

## REFERENCES

[1] Johnson SM. Optimal two- and three-stage production

schedules with setup times included. 1954;1(61-68).

[2] Quadt D, Kuhn H. Capacitated lot-sizing with extensions: a review. 2008;6(61–83).

[3] Quadt D, Kuhn H. Capacitated Lot-Sizing and Scheduling with Parallel Machines, Back-Orders, and Setup Carry-Over. 2009;56.

[4] Song Y, Chan GH. Single item lot-sizing problems with backlogging on a single machine at a finite production rate. 2005;161 (191–202).

[5] Wolsey Y, Pochet L. Lot size models with back-logging: Strong reformulations and cutting planes. 1988;40(317–335).

[6] Cheng H, Madan MS, Gupta Y, So S. Solving the capacitated lot-sizing problem with backorder consideration. 2001;52(952–959).

[7] Karimi B, Fathemi Ghomi SMT, Wilson JM. A tabu search heuristic for solving the CLSP with backlogging and set-up carry-over. 2006;57(140–147).

[8] Graham RL, Lawler E L, Lenstra J K, Kan AHGR. Optimisation and approximation in deterministic sequencing and scheduling: a survey. A. 1979;5(287-326).

[9] Mohammadi M, Fatemi Ghomi SMT, Jafar N. A genetic algorithm for simultaneous lotsizing and sequencing of the permutation flow shops with sequence-dependent setups. 2011;24(87-93).

[10] Ruiz R, Maroto C, Alcaraz J. Two new robust genetic algorithms for the flowshop scheduling problem. 2006;34(461-476).

[11] Allada E, Ruiz R. Genetic algorithms with path relinking for the minimum tardi-ness permutation flow shop problem. 2010;38(57 - 67).

[12] Tseng LY, Lin YT. A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. 2009;198(84 – 92).

[13] Tavakkoli-Moghaddam R, Gholipour-Kanani Y, Cheraghalizadeh R. A genetic algorithm and memetic algorithm to sequencing and scheduling of cellular manufacturing systems. 2008;3(2).

[14] Goren HG, Tunali S, Jans R. A review of applications of genetic algorithms in lot sizing. 2010;21(575–590).

[15] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. 2007(4661–4667).

[16] Gao KL, Chaoyong Z, Xinyu S, Liang. Optimization of process planning with various flexibilities using an imperialist competitive algorithm. 2012; 59 (5-8)(815-828).

[17] Attar SF, Mohammadi M, Tavakkoli-Moghaddam R. A Novel Imperialist Competitive Algorithm to Solve Flexible Flow Shop Scheduling Problem in Order to Minimize Maximum Completion Time. 2011;28.

[18] Shokrollahpour E, Zandieh M, Dorri B. A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. 2010;49(11).

[19] Rajabioun R, Atashpaz-Gargari E, Lucas C. Colonial competitive algorithm as a tool for Nash equilibrium point achievement. 2008;49 (11)(680–695).

[20] Khabbazi A, Gargari E, Lucas C. Imperialist competitive algorithm for minimum bit error rate beamforming. 2009;1(125–133).

[21] Kaveh A, Talatahari S. Optimum design of skeletal structures using imperialist competitive algorithm. 2010;88(21–22).

[22] Lucas C, Nasiri-Gheidari Z, Tootoonchian F. Application of an imperialist competitive algorithm to the design of a linear induction motor. 2010;51(7)(1407–1411).

[23] Nazari-Shirkouhi S, H E, Ghodsi R, Rezaie K, Atashpaz-Gargari E. Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm. 2010;37(12).

[24] Sarayloo F, Tavakkoli-Moghaddam R. Imperialistic competitive algorithm for solving a dynamic cell formation problem with production planning. 2010(266–276).

[25] Quan-Ke P, Tasgetiren MF, Yun-Chia L. A Discrete Particle Swarm Optimization Algorithm for the Permutation Flowshop Sequencing Problem with Makespan Criterion. 2006.

[26] Udomsakdigool A, Khachitvichyanukul V. Ant colony algorithm for multi-criteria job shop scheduling to minimize makespan, mean flow time and mean tardiness. *International Journal of Management Science and Engineering Management*. 2011; 6(2)(117-123):6(2):

117-123.

[27] Mohammadi M, Fatemi Ghomi SMT. Genetic algorithm-based heuristic for capacitated lotsizing problem in flow shops with sequence-dependent setups. 2011;38(7201–7207).

[28] Holland JH. Adaptation in natural and artificial systems. 1975.

[29] Nagano MS, Ruiz, R, Lorena LAN. A constructive genetic algorithm for permutation flowshop scheduling. 2008;55(195–207).

[30] Kayvanfar V, Zandieh M. The economic lot scheduling problem with deteriorating items and shortage: an imperialist competitive algorithm.

[31] Abdi B, Mozafari H, Ayob A, Kohandel R. Imperialist Competitive Algorithm and its Application in Optimization of Laminated Composite Structures. 2011;55(2).

[32] Kayvanfar V, Zandieh M. The economic lot scheduling problem with deteriorating items and shortage: an imperialist competitive algorithm. 2012.

[33] Pasandideh SHR, Niaki STA, Yeganeh JA. A parameter-tuned genetic algorithm for multi-product economic production quantity model with space constraint, discrete delivery orders and shortages. 2010;41(306–314).

[34] Taguchi G. Introduction to quality engineering. 1986.

[35] Taguchi G. *Introduction to quality engineering.* White Plains: Asian Productivit; 1986.

[36] Grabowski J, Wodecki M. A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion. 2004;31(11).

[37] Onwubolu GC, Davendra D. Scheduling flow shops using differential evolution algorithm. 2006;171(2).