Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Two parameter-tuned metaheuristic algorithms for the multi-level lot sizing and scheduling problem

M. Babaei[a], **M. Mohammadi**[a], **S.M.T. Fatemi Ghomi**[b] and **M. A. Sobhanallahi**[a]

[a]*Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Karaj, Iran*
[b]*Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran*

A R T I C L E I N F O

Article history:
Received 18 July 2012
Received in revised format
14 August 2012
Accepted August 15 2012
Available online
15 August 2012

Keywords:
Genetic algorithm
Imperialist competitive algorithm
Taguchi methodology
Lot sizing and scheduling

A B S T R A C T

This paper addresses the problem of lot sizing and scheduling problem for n-products and m-machines in flow shop environment where setups among machines are sequence-dependent and can be carried over. Many products must be produced under capacity constraints and allowing backorders. Since lot sizing and scheduling problems are well-known strongly NP-hard, much attention has been given to heuristics and metaheuristics methods. This paper presents two metaheuristics algorithms namely, Genetic Algorithm (GA) and Imperialist Competitive Algorithm (ICA). Moreover, Taguchi robust design methodology is employed to calibrate the parameters of the algorithms for different size problems. In addition, the parameter-tuned algorithms are compared against a presented lower bound on randomly generated problems. At the end, comprehensive numerical examples are presented to demonstrate the effectiveness of the proposed algorithms. The results showed that the performance of both GA and ICA are very promising and ICA outperforms GA statistically.

1. Introduction

Lot sizing and scheduling are some of the most challenging subjects where tremendous efforts have been accomplished to propose efficient solutions. In lot sizing and scheduling, a set of activities is considered for the best use of production resources (such as machines, materials, etc.) to satisfy production goals under some conditions and over a certain period named the planning horizon. Therefore, optimization of lot sizing and scheduling with various conditions and restrictions is essential for improving production efficiency as well as system flexibility. Wagner and Within (1958) are believed to be this first who considered lot sizing and scheduling problems. Since then, several researchers have studied their model and many features and assumptions have been added to the original model.

The area of lot sizing and scheduling problems still remains interestingly researched as evident from the very recent works, Lang and Shen (2011) developed a mixed-integer programming (MIP)

* Corresponding author.
E-mail: mohammadi@tmu.ac.ir (M. Mohammadi)

doi: 10.5267/j.ijiec.2012.08.005

formulation for a capacitated single-level dynamic lot sizing problem with sequence-dependent setup costs and times, which includes product substitution options. Shim et al. (2011) considered a single machine capacitated lot sizing and scheduling problem to determine the lot sizes and the sequence of lots while satisfying the demand requirements and the machine capacity in each period of a planning horizon. They also considered setup carry over in their study. Mohammadi (2010) presented an integrated lotsizing, loading, and scheduling model for the capacitated flexible flow shops with sequence-dependent setups and to solve this problem, mixed integer programming-based heuristics based on iterative procedures was provided. There are a number of surveys and researches on lot sizing and scheduling problems Zhua et al. (2006), Buschkühl, et al. (2010), Quadt, et al. (2008) and Drexl, et al. (1997).

It is well-known that CLSP problems are *NP*-hard. Bitran and Yanasse (1982) showed that the CLSP with setup costs is an *NP*-hard problem, meaning that one cannot expect to find an efficient algorithm, which generates an optimal solution. On the other hand, when setup times are included, Maes et al. (1991) showed that even the feasibility problem becomes *NP*-complete. This implies that one cannot efficiently say whether a feasible solution exists at all. Consequently, heuristics and metaheuristics methods are needed to find a reasonably good solution in a moderate amount of computation time. Mohammadi, et al. (2010a, 2010b) employed fix-and-relax heuristic and MIP-based heuristic for simultaneous lot sizing and scheduling in capacitated flow shop with sequence-dependent setups. Araújo and Nagano (2010) proposed a new constructive heuristic based on a structural property for the problem of scheduling jobs in a no-wait flowshop problem with sequence-dependent setup times with the objective of minimizing makespan. Khanzadi et al. (2011) presented a heuristic approach and a new genetic algorithm approach for large scale multiple resource-constrained project-scheduling problems.

In addition, it has been reported that the evolutionary algorithms are capable of providing highly cost-efficient solutions within reasonable computing load (Xiao et al., 2012) so that these algorithms have been considered by many researchers. One of the most popular evolutionary algorithms is Genetic Algorithm (GA). GA generates solutions to optimization problems using techniques inspired by natural evolution, such as selection, crossover and mutation. GA has been extensively and successfully employed in lot sizing and scheduling problems. Sun et al. (2009) studied a problem, which scheduled productions of products on multiple identical machines. They developed a GA and compared it with a heuristic. Computational results showed that their genetic algorithm outperforms the heuristic. Mohammadi et al. (2011) employed a GA for simultaneous lot sizing and sequencing. Kimms (1999) used a GA to solve a new mixed-integer programming formulation for the multi-level, multi-machine proportional lot sizing and scheduling problem. Yao and Huang (2005) employed a hybrid genetic algorithm with a feasibility testing procedure and a binary search heuristic to efficiently solves the Economic Lot Size Scheduling Problem. The computational results show that the hybrid approach could be very helpful to derive the production scheduling and lot sizing strategies. Goren, et al.(2010) provided a worthy overview of recent advances in this field to highlight the many ways GAs can be applied to various lot sizing models.

Imperialist Competitive Algorithm (ICA) is a new socio-politically motivated global search strategy introduced by Atashpaz-Gargari and Lucas (2007) for dealing with different optimization problems. Past literature on the capacitated lot sizing and scheduling problem with complex setups and backlogging is reasonably sparse due to the problem's complexity. Keyvanfar and Zandieh (2012) studied an economic lot scheduling problem (ELSP) and employed an ICA to provide good solutions within reasonable computational times in order to minimize setup cost, holding cost and slack cost. Lian et al. (2012) investigated the optimization of process planning in which various flexibilities were considered to minimize total weighted sum of manufacturing costs. They proposed an imperialist competitive algorithm to find promising solutions with reasonable computational cost. The obtained results from computational experiments 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). Shokrollahpour et al. (2010) proposed an ICA to solve the two-stage assembly flow shop scheduling problem. They calibrated the parameters of the algorithm using the Taguchi method. In this study, they showed that the ICA indicated an improvement in comparison with the best algorithm proposed previously. Rajabioun et al. (2008), Khabbazi et al. (2009), Kaveh and Talatahari (2010), Lucaset al. (2010), Nazari-Shirkouhi et al. (2010) and Sarayloo and Tavakkoli-Moghaddam (2010) are other related study that employed ICA.

To the best of our knowledge, there is no result for employing a novel imperialist competitive algorithm (ICA) and a genetic algorithm (GA) as solution approaches for capacitated lot sizing and scheduling problem with complex setups and backlogging problems in the literature. In this paper, we present problem formulation and a lower bound. The parameters of the algorithms are calibrated using the Taguchi method. The calibrated algorithms are compared against the presented lower bound and a statistical method is used to comparing the results.

The paper is organized as follows: In section 2 problem formulation and lower bound are presented. In section 3 GA and ICA are developed. Computational experiments are presented in Section 4 and finally Section 5 describes conclusions.

2. Notation and problem formulation

In order to formulate the problem, let us introduce the following notations:

2.1. Notations

- Indices

i, j, k	Production type,
n, n', n''	Designation for a specific setup number,
m	Level of production,
t	Period,

- 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*in 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*.

- 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*,
  - $y^{n}_{i,j,m,t}$ Binary variable, which indicates whether the *n*th setup on machine *m*in period *t* is from product *i* to product *j* ($y^{n}_{i,j,m,t} = 1$) or not ($y^{n}_{i,j,m,t} = 0$),
  - $x^{n}_{i,j,m}$ Quantity of product *j* produced after *n*th setup on machine *m*in period *t*,
  - $q^{n}_{i,j,m}$ Shadow product: the gap (in quantity units) between *n*th setup (to product *j*) on machine *m*in period *t* and its related production in order to ensure that direct predecessor of this product (production of product *j* on machine *m*in period *t*) has been completed. In other words, idle time (in quantity units) before production of product *j* on machine *m*in period *t*in order to guarantee vertical interaction.

The mathematical model in this paper is described on the basis of the following:

2.2. Assumptions and formulation

Capacitated lot sizing focuses on how to make lot sizing planning and scheduling focuses on when each product should be produced to minimize total cost. In order 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 capacity of each machine is constrained
- 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 objective function is to find an optimal lot sizing and scheduling that minimize setup, inventory, production and backlogging costs.

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

subject to

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

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

$$bigM.\left(\sum_{i=1,i\neq j(for(n'>1))}^{N} y_{i,j,m,t}^{n'}-1\right)+\sum_{n=1}^{n'}\sum_{i=1}^{N}\sum_{k=1}^{N} y_{i,j,m,t}^{n}.S_{i,k,m}+\sum_{n=1}^{n'}\sum_{k=1}^{N} b_{k,m}.q_{k,m,t}^{n}$$
$$+\sum_{n=1}^{n'}\sum_{k=1}^{N} b_{k,m}.x_{k,m,t}^{n}$$
$$\leq bigM.\left(1-\sum_{i=1,i\neq j(for(n''>1))}^{N} y_{i,j,m+1,t}^{n''}\right)+\sum_{n=1}^{n''}\sum_{i=1}^{N}\sum_{k=1}^{N} y_{i,j,m+1,t}^{n}.S_{i,k,m+1} \tag{4}$$
$$+\sum_{n=1}^{n''}\sum_{k=1}^{N} b_{k,m+1}.q_{k,m+1,t}^{n}+\sum_{n=1}^{n''-1}\sum_{k=1}^{N} b_{k,m+1}.x_{k,m+1,t}^{n};\quad j=1,\ldots,N,$$
$$n'=1,\ldots,N,\qquad n''=1,\ldots,N,\qquad m=1,\ldots,M-1, t=1,\ldots,T$$

$$\sum_{n=1}^{n}\sum_{i=1}^{N}\sum_{j=1}^{N} y_{i,j,m,t}^{n}.S_{i,j,m}+\sum_{n=1}^{n}\sum_{j=1}^{N} b_{j,m}.x_{j,m,t}^{n}+\sum_{n=1}^{n}\sum_{j=1}^{N} b_{j,m}.q_{j,m,t}^{n}\leq C_{m,t;}m=1,\ldots,M, \quad t=1,\ldots,T \tag{5}$$

$$x_{j,m,t}^{n}\leq\left(\frac{C_{m,t}}{b_{m,t}}\right).\sum_{i=1,i\neq j(for(n>1))}^{N} y_{i,j,m,t}^{n};\quad n=1,\ldots,N,\qquad j=1,\ldots,N,\qquad m=1,\ldots,M, \quad t=1,\ldots,T \tag{6}$$

$$q_{j,m,t}^{n}\leq\left(\frac{C_{m,t}}{b_{m,t}}\right).\sum_{i=1}^{N} y_{i,j,m,t}^{n};\quad n=1,\ldots,N,\qquad j=1,\ldots,N,\qquad m=1,\ldots,M, \quad t=1,\ldots,T \tag{7}$$

$$y_{j,i,m,1}^{1}=0\ \ j\neq j_{0m};\quad i=1,\ldots,N,\qquad m=1,\ldots,M \tag{8}$$

$$\sum_{i=1}^{N} y_{j_{0m},i,m,1}^{1}=1;\quad m=1,\ldots,M \tag{9}$$

$$\sum_{j=1}^{N} y_{j,i,m,t}^{n}=\sum_{k=1}^{N} y_{i,k,m,t}^{n+1};\quad i=1,\ldots,N,\ n=1,\ldots,N,\qquad m=1,\ldots,M,\qquad t=1,\ldots,T \tag{10}$$

$$\sum_{j=1}^{N} y_{j,i,m,t-1}^{N}=\sum_{k=1}^{N} y_{i,k,m,t}^{1};\quad i=1,\ldots,N,\qquad n=1,\ldots,N,\qquad m=1,\ldots,M, \quad t=2,\ldots,T \tag{11}$$

$$y_{i,j,m,t}^{n}=0\ or\ 1 \tag{12}$$

$$I_{j,m,t}^{+}, I_{j,t}^{-}, x_{j,m,t}^{n}, q_{j,m,t}^{n}\geq 0 \tag{13}$$

$$I_{j,m,0}^{+}=0,\ \ j=1,\ldots,N,\ \ m=1,\ldots,M \tag{14}$$

In this model, the objective function is Eq. (1). The backlogging or storage at the end of each period is considered by Eq. (2). Constrain (3) guarantees total of in-flows to each node is equal to of out-flows from that node. Eq. (4) ensures within one period each typical product *j* one machine *m* is produced before its direct successor. The left side of Eq. (4) is equal to the time between the beginning of period *t* and the end of production of product j on machine *m* if n'th setup in machine *m* and period *t* is from every product i to product j (for $n' > 1, i \neq j$), else it is a negative number. In other words, if $x_{j,m,t}^{n'}$ cannot get a positive value when the left side of Eq. (4) would get a negative value. The right side of Eq. (4) is equal to the time between the beginning of period *t* and the beginning of production of product *j* on machine *m+1*if n''th setup in machine *m+1* and period *t* is from every product *i* to product *j* (for $n'' > 1, i \neq j$), else it is a big number. In fact, if $x_{j,m+1,t}^{n''}$ cannot get a positive value, the right side of Eq. (4) would get a big value. The capacity constraints of machine are considered by Eq. (5). Eq. (6) respects setups in production process. Eq. (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. Eq. (10) and Eq. (11) represent the relationship between successive setups. Eq. (8) to Eq. (11) ensure that for each triple (*n*,*m*,*t*) there is exactly one pair (*i*,*j*) which $y_{i,j,m,t}^{n} = 1$. The type of variables is defined by Eq. (12) and Eq. (13) and finally Eq. (14) indicates that at the end of planning horizon there is no on-hand inventory.

2.3. Lower bounds

In this section, we present a lower bound. The lower bound is developed by adding a new equation and solving a new model. We add following equation to the relaxed model and consider $y_{i,j,m,t}^{n}$ are continuous variable between 0 and 1.

$$\sum_{i=1}^{N} y_{i,j,m,t}^{1} + \sum_{i=1,i\neq j}^{N} \sum_{n=2}^{N} y_{i,j,m,t}^{n} = a_{j,m,t} \tag{15}$$

In this Equation $a_{j,m,t}$ is binary variable. Eq. (15) was proved that is valid to model. We refer the proof of this Eq. to Mohammadi et al. (2010a).

3. The proposed algorithms

As mentioned above, the CLSP problem is strongly *NP*-hard and developing heuristics or metaheuristics method is essential. In this section, we propose two metaheuristic algorithms, namely, Genetic Algorithm (GA) and Imperialist Competitive Algorithm (ICA).

3.1. The proposed Genetic Algorithm

One of the most popular evolutionary metaheuristics is genetic algorithms, which have been applied to various optimization problems with promising results (Goren, et al., 2010). GA is probabilistic search optimization algorithm inspired by the process of natural evolution and the principles of 'survival of the fittest' (Holland, 1975). In this section, we discuss the way a GA can be customized to address a lot sizing and scheduling problem. Like other population-based metaheuristics, GA starts with an initial population. Since the quality of the initial population can help the algorithm reach better solutions, so we first present a simple and effective heuristic to generate initial solution and we discuss the issue of encoding in our case a binary matrix. Finally, the GA stages and operators according to our model are designed. The components of GA which are used in this article are described as follows.

3.1.1 Initial Population and chromosome representation

To generate initial solution, values of continuous variables are generated randomly by uniform distribution function. Quantity of product (*x*), Shadow product (*q*), Stock of product (I^{+}) and Shortage of product (I^{-}) are continuous variables and the binary variables are sequencing of job at planning horizon (y). The binary variables are coded by determination of sequencing of product. We use a simple and effective heuristic which has been presented by Mohammadi et al. (2011).

This heuristic is used for $t = 1$ to T and is described as follows:

(1) The products are sorted in the decreasing order of $W_{j,m} = \sum_{i=1}^{N} W_{i,j,m}; j = 1, \ldots, N$.
(2) Let $[i]$ indicate the ith 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.

By using this method, M different initial populations is produced (for $m = 1, \ldots, M$), the remaining initial populations have been generated randomly. Chromosomes are represented in the form of matrices with $N \times T \times M$ dimensions to show the value of binary variables.
In Fig. 1, a sample chromosome with T =2, N =3 and M=2 is depicted.

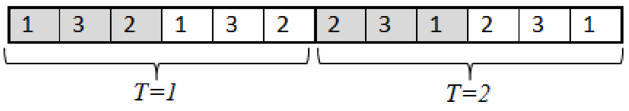


Fig. 1. A sample chromosome

According to Fig. 1, an encoded binary variables have been shown. The corresponding binary variables (decoded)in this chromosome during period T=1 are, $y_{1,1,1,1}^{1} = y_{1,3,1,1}^{2} = y_{3,2,1,1}^{3} = y_{1,1,2,1}^{1} = y_{1,3,2,1}^{2} = y_{3,2,2,1}^{3} = 1$ and the corresponding binary variables to this chromosome during period T=2 are, $y_{2,2,1,2}^{1} = y_{2,3,1,2}^{2} = y_{3,1,1,2}^{3} = y_{2,2,2,2}^{1} = y_{2,3,2,2}^{2} = y_{3,1,2,2}^{3} = 1$ and other binary variables are equal to 0.

3.1.2 Fitness function

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

3.1.3 Selection operator

In order to select of appropriate parents from literature, we consider Rank selection (A), Random selection (B), Tournament selection (C) and Roulette wheel (D) as selection approaches and must be chosen in the process of the parameter calibration.

3.1.4 Crossover operation and mutation operator

Nagano et al. (2008) have proposed several crossover operators and we employed the similar job two-point crossover method. In order to produce small perturbations on chromosomes to promote diversity of the population, a shift mutation operator has been used in this article. The probability of crossover and mutation must be determined by parameters calibration.

3.1.5 Population replacement

Chromosomes for the next generation are selected from the enlarged population. The best pop_size chromosomes of the enlarged population have been selected for the next generation.

3.1.6. Termination criterion

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

3.2. Hybrid Imperialist Competitive algorithm

3.2.1. Frame work of imperialist competitive algorithm

ICA is a novel population-based evolutionary algorithm proposed by Atashpaz-Gargari and Lucas (2007). ICA is one of the most powerful evolutionary algorithms and it has been used extensively to solve different kinds of optimization problems. This method is based on socio-political process of imperialistic competition. Fig. 2 expresses an overview to ICA. We discuss each step of the algorithm in turn and focus on the developing every step to propose the algorithm.

3.2.1. Generating of Initial countries

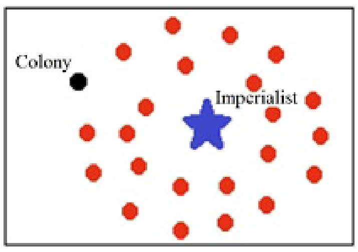
The ICA initiates with an initial population. Each individual of the population is called a ‘country’ equivalent ‘chromosome’ in GA. We use the generating initial population method that has been described in the proposed GA.

3.2.2. Generating of Initial imperials

The countries (solutions) are divided into two categories: imperialists and colonies. This classification is based on power of each country. Some of the most powerful countries in the initial population are Imperialists, and the rest weaker countries in initial solution are the colonies of imperialists. A set of one imperialist and their colonies forms one empire. The total power of an empire is set equal to the power of the imperialist country plus a percentage of the mean power of its colonies. After forming the initial empires, Assimilation of colonies starts.

3.2.3. Assimilation of colonies

Imperialist states change socio-political characteristics of colonies in such a way that they become similar to them (increase their power). Through this movement, some parts of the structure of a colony will be similar to the structure of the empire to enhance their empire. The aim of assimilation procedure is to assimilate the colonies’ characteristic toward their imperialist such as culture, social structure, language, etc. This fact has been modeled by moving all the colonies toward the imperialist. The assimilating operator is shown in Fig. 2.



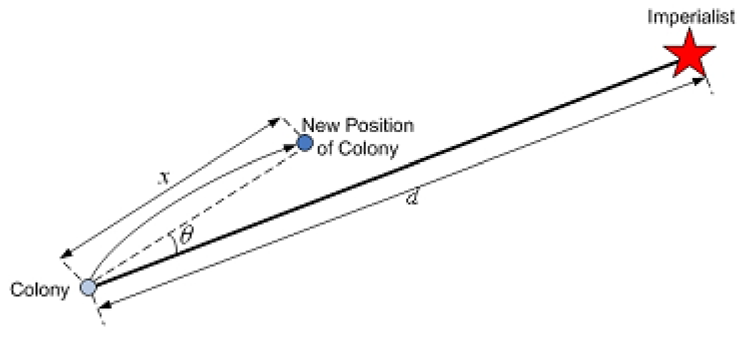


Fig. 2. A sample of empire **Fig. 3**. Moving colonies toward their related imperialist.

As shown in Fig. 3, each colony moves toward its imperialist. The important point is the method of colonies moving towards the imperialist. Each country (colony) has different socio-political characteristics (variables), so each socio-political characteristic (variables) moves toward the related socio-political of imperialist. For this reason, depend on types of variables, we proposed two different ways to assimilation variables.

Continuous variables of colonies move toward related continuous variables of its imperialist and binary variables move toward binary variables of its imperialist. So, the continuous variables of each colony move toward the imperialist by x units. x is a random number with uniform distribution, $x\sim U(0,\beta \times d)$. Where $\beta > 1$ and d is distance between colony and the its imperialist. d is named the vector of movement for colony toward imperialist. Parameter βcauses the colony to get closer to imperialist from both sides. To search wider area around current solution we add a random amount of deviation θ to the direction of movement where $\theta\sim U(-\gamma,\gamma)$. The movement of binary variables is accomplished by crossover operation, as same as a genetic algorithm. Crossover allows exchanging information between different solutions (chromosomes) so it is useful to simulate the assimilation of binary variables.

3.2.4.Revolution

Revolution is a fundamental change in power or organizational structures that takes place in a relatively short period of time. The colony randomly changes its position in the socio-political axis. Fig. 4 shows the colonies revolution. The revolution increases the exploration of the algorithm and prevents the early convergence of countries to local minimums. The revolution rate in the algorithm indicates the percentage of colonies in each colony, which will randomly change their position. A very high value of revolution decreases the exploitation power of algorithm and can reduce its convergence rate (Abdi et al., 2011). This mechanism is similar to mutation process in GA for creating diversification in solutions. Mutation increases the variety in the population, so this operator is used for creating a revolution in binary variables the same as genetic algorithm.

3.2.5.Exchange the colony with imperialist

Meanwhile moving toward the imperialist, a colony may get to a situation with lower cost than the imperialist. In this case, the imperialist and the colony change their positions. Thereafter, the algorithm will keep on by the imperialist in the new position and the colonies will be assimilated by the imperialist in its new position. Fig. 5 shows the empire after exchanging position between the imperialist and the colony.

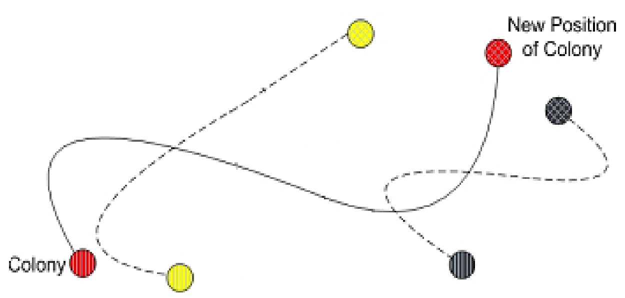


Fig. 4. Revolution

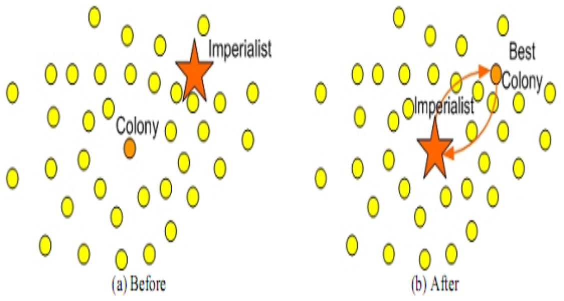


Fig. 5. Exchanging the positions of a colony and the imperialist

3.2.6. Imperialistic competition

In ICA, all empires compete to take possession of more colonies besides their current colonies. By keeping on the imperialistic competition, the power of weaker empires will decrease and the power of more powerful ones will reinforce. To model this competition among imperialists, the weakest colony of the weakest empire is freed from its current imperialist and waited to be possessed by all empires. Each of the empires (based on its total power) will have a likelihood of taking possession of the

mentioned colonies. It means, these colonies will not definitely be possessed by the most powerful empires, but these empires will be more likely to possess them each imperialist attempt to gain the colonies of other empires. The most powerful empires have a more chance to gain the colonies from the weakest empires. Fig. 6 shows how each empires taking possession of the weakest colonies.
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. However, the power of the colonies of an empire has a negligible effect, on the total power of that empire. This fact is modeled by defining the total cost of an empire:

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

where $TC_n$the total is cost of the j^{th} empire and ρ is a positive small number. If values of ρ are small, the total power of the empire will be determined by approximately only the imperialist and increasing it will increase the role of the colonies in determining the total power of an empire. 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

$$N.T.C._n = T.C._n - \max\{T.C._i\},$$

where, $N.T.C._n$and $T.C._n$ are the total cost and the normalized total cost of n^{th} empire, respectively. Having the normalized total cost, the possession probability of each empire is given by

$$p_{p_n} = \left|\frac{N.T.C._n}{\sum_{i=1}^{N_{imp}} N.T.C._i}\right|.$$

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

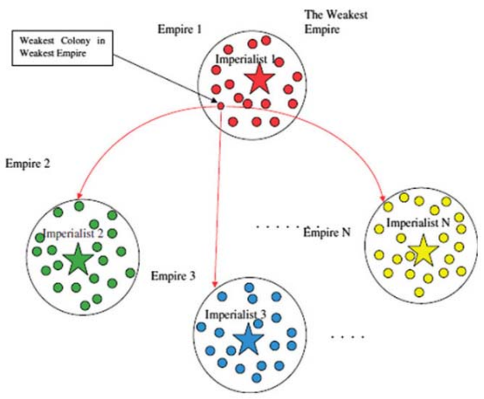


Fig. 6.The more powerful an empire is, the more likely it will possess the weakest colony of the weakest empire (Imperialistic competition)

3.2.7.Elimination of powerless empires.

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

3.2.8. Stop criterion

By keeping on the algorithm and spending the time, all of the empires will collapse except the most powerful one and all the colonies will be subjected to this empire. In such an ideal new world, all the colonies will have the same positions and the 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 (Kayvanfar et al., 2012). Stopping criterion in proposed algorithm is to get the maximum decades.

4. Experiments

4.1. Parameters tuning based on Taguchi method

One important decision to make when implementing a metaheuristic is how to set the parameter values. Modifying the parameter tuning of a metaheuristic could greatly impact the algorithm's performance. In order to calibrate the algorithms, there are several ways to statistically design the experimental investigation, but the most frequently used and exhaustive approach is a full factorial experiment (Montgomery, 2000). This approach cannot be always effective since it becomes increasingly difficult to carry out investigations when the number of factors becomes considerably large. In order to reduce the number of required tests, a fractional factorial experiment (FFE) was developed (Kayvanfar et al., 2012). Taguchi (1986) developed a family of FFE matrices that ultimately lessens the number of experiments, but still provides adequate information so that in this paper Taguchi method is employed to calibrate the parameters of algorithms. As universalization, we study three different categories of problems, i.e., small problems, medium problems, and large problems. For each instance, because of different size of problems, the factors level differs. From literatures and preliminary experimental we determine the parameters and the levels. For the proposed GA we consider selection type, crossover probability, mutation probability, number of population and maximum iteration. And for the proposed ICA we consider number of population, number of imperialist, maximum iteration, revolution probability andρ. After preliminary experiment we considered the levels of the factors for GA and ICA that are reported in Table 1.

Table 1
Factor levels for small, medium, and large problems

	Problem Size	Parameter	Level 1	Level 2	Level 3	Level 4
GA	Small	Selection Type	A	B	C	D
		Crossover Probability	0.4	0.5	0.6	0.7
		Mutation Probability	0.1	0.2	0.3	0.4
		Number of Population	300	400	500	600
		Maximum Iteration	300	400	500	600
	Medium	Selection Type	A	B	C	D
		Crossover Probability	0.4	0.5	0.6	0.7
		Mutation Probability	0.1	0.2	0.3	0.4
		Number of Population	100	200	300	400
		Maximum Iteration	100	200	300	400
	Large	Selection Type	A	B	C	D
		Crossover Probability	0.4	0.5	0.6	0.7
		Mutation Probability	0.1	0.2	0.3	0.4
		Number of Population	50	100	150	200
		Maximum Iteration	50	100	150	200
ICA	Small	Number of Population	300	400	500	600
		Number of Imperialist	5	10	15	20
		Maximum Iteration	300	400	500	600
		Revolution Probability	0.1	0.2	0.3	0.4
		ρ	0.25	0.75	1.25	1.75
	Medium	Number of Population	100	200	300	400
		Number of Imperialist	5	10	15	20
		Maximum Iteration	100	200	300	400
		Revolution Probability	0.1	0.2	0.3	0.4
		$\boldsymbol{\rho}$	0.25	0.75	1.25	1.75
	Large	Number of Population	50	100	150	200
		Number of Imperialist	5	10	15	20
		Maximum Iteration	50	100	150	200
		Revolution Probability	0.1	0.2	0.3	0.4
		$\boldsymbol{\rho}$	0.25	0.75	1.25	1.75

Taguchi splits the factors into two main groups: controllable and noise factors. Those factors that we cannot directly control them are noise factors. Since removal of the noise factors is often impossible, the Taguchi method looks for to minimize the effect of noise and to determine the optimal level of the important controllable factors based on the concept of robustness (Tsai, et al.). A transformation of the repetition data to another value is employed in Taguchi method, which is the measure of variation. The transformation is named the signal-to-noise (S/N) ratio. The term "signal" indicates the desirable value (response variable) and "noise" signifies the undesirable value (standard deviation). Therefore, the S/N ratio specifies the amount of variation present in the response variable. Here, maximization of the signal-to-noise ratio is addressed (i.e., is the goal) (Kayvanfar et al., 2012). Taguchi categorizes objective functions into three groups: the smaller-the-better type, the larger-the-better type, and the nominal-is-best type. Since our objective functions in minimization so the smaller-the-better type is suitable, the corresponding S/N ratio (Phadke, 1989) is:

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

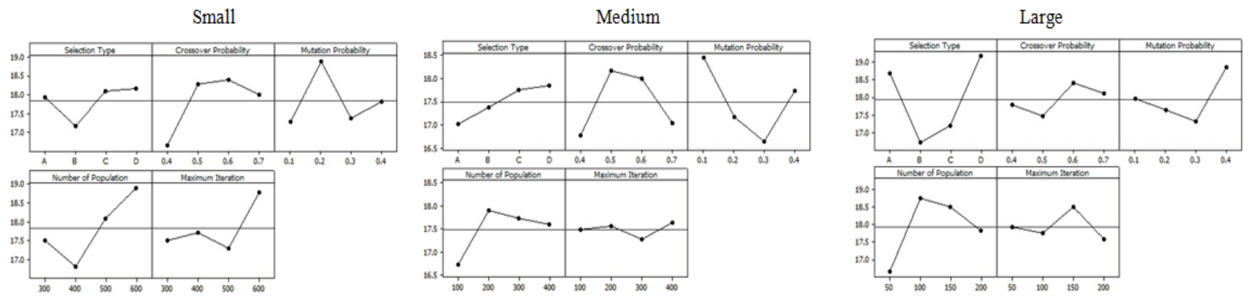


Fig. 7. Main effect plot for S/N ratios for GA in different problem size

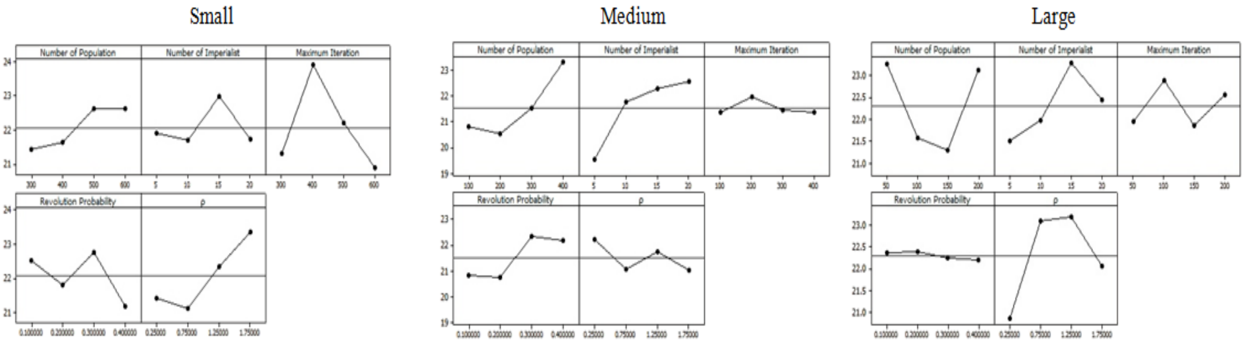


Fig. 8. Main effect plot for S/N ratios for ICA in different problem size

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. Fig. 7 and Fig. 8 show the average S/N ratio obtained at each level for ICA and GA on small, medium and large problems. According to Fig. 7 and Fig. 8 the optimal levels of factors are easily determined.

Table 2
Optimal level of the parameters for GA and ICA

	Problem size	Optimal level of the parameters
GA	Small	Select Type = D, Crossover Probability = 0.6, Mutation Probability = 0.2, Number of Population = 600, Maximum Iteration = 600
	Medium	Select Type = D, Crossover Probability = 0.5, Mutation Probability = 0.1, Number of Population = 200, Maximum Iteration = 400
	Large	Select Type = D, Crossover Probability = 0.6, Mutation Probability = 0.3, Number of Population = 100, Maximum Iteration = 150
ICA	Small	Number of Population=500, Number of Imperialist = 15, Maximum Iteration = 400, Revolution Probability =0.3, ρ = 1.75
	Medium	Number of Population=400, Number of Imperialist = 20, Maximum Iteration = 200, Revolution Probability =0.3, ρ = 0.25
	Large	Number of Population=50, Number of Imperialist = 15, Maximum Iteration = 100, Revolution Probability=0.1, ρ = 1.25

The optimal factors for each algorithm in different problem size are shown in Table 2. Based on optimal parameters, we design some experiments and compare the tuned-algorithms in each category.

4.2. Computational and statistical evaluation

In this section, in order to evaluate and compare the performance of two proposed algorithms, we consider different problem sizes and define a comparison measurement to determine which algorithm is superior to the other one. We design an experiment that is intended to extensively test of the computational efficiency of the proposed search algorithms. The primary objective is to solve problems classified into three different sizes, small, medium, and large (Logendran et al., 2006). We conducted 20 problems for each size (small medium and large) and also for lower bound. For each of the 20 instances, 15 independent runs are carried out. The response variable for each algorithm is considered on the following performance measure: % increase over the lower bound:

$$\frac{1}{15}\left[\left(\sum_{i=1}^{15}\frac{\text{Metaheuristic}_{\text{sol}_i} - LB_i}{LB_i}\right)\times 100\right],$$

where $Metaheuristic_{sol_i}$ is obtained solution by the algorithm (GA or ICA) and LB_i is the solution obtained by the lower bound. It is obvious that the smaller value of the performance measurement shows the metaheuristic is more efficient. On the other hand, the algorithm with smaller performance measurement compared with the one with higher value has better performance and outperforms the second one. The required parameters for these problems are extracted from the following uniform distributions:

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

The GA and the ICA algorithms are coded in MATLAB programming language and are run on a PC with a 2.27 GHz Intel Core i5 processor and 3 GB RAM memory processor running at 2 GHz with elapse CPU time = 7200 (Mohammadi, et al., 2011). All of the classified problem instances are solved by the GA and the ICA algorithm and finally by lower bound. The comparing measurement is calculated. The computational results are reported in Table 3, Table 4 and Table 5.

Table 3
Computation results of the algorithms for small size problems

Problem Size ($N \times M \times T$)	GA	ICA
$2 \times 2 \times 2$	10.65%	5.75%
$2 \times 2 \times 3$	9.49%	6.42%
$2 \times 3 \times 2$	11.92%	4.63%
$3 \times 2 \times 2$	7.59%	6.08%
$2 \times 3 \times 3$	1015%	4.14%
$3 \times 2 \times 3$	10.43%	6.67%
$3 \times 3 \times 2$	12.00%	3.76%
$3 \times 3 \times 3$	9.38%	4.64%
$4 \times 3 \times 3$	11.58%	5.58%
$3 \times 4 \times 3$	10.35%	3.15%
$3 \times 3 \times 4$	9.86%	2.94%
$4 \times 3 \times 3$	6.95%	3.93%
$3 \times 4 \times 3$	13.05%	5.55%
$3 \times 3 \times 4$	11.81%	3.63%
$4 \times 4 \times 4$	10.57%	4.87%
$4 \times 4 \times 5$	14.54%	7.68%
$4 \times 5 \times 4$	9.73%	3.91%
$5 \times 4 \times 4$	10.30	4.96%
$5 \times 5 \times 4$	11.43%	5.48%
$5 \times 4 \times 5$	10.12%	6.39%
mean	**%10.60**	**%5.01**

Table 4
Computation results of the algorithms for medium size problems

Problem Size ($N \times M \times T$)	GA	ICA
$5 \times 5 \times 5$	13.35%	4.42%
$5 \times 5 \times 6$	14.72%	6.82%
$5 \times 6 \times 5$	12.09%	4.40%
$6 \times 5 \times 5$	13.38%	5.08%
$5 \times 6 \times 6$	13.17%	5.38%
$6 \times 5 \times 6$	10.77%	4.75%
$6 \times 6 \times 5$	9.08%	5.87%
$6 \times 6 \times 6$	10.50%	5.44%
$7 \times 6 \times 6$	10.30%	5.07%
$6 \times 7 \times 6$	13.19%	5.89%
$6 \times 6 \times 7$	7.80%	5.31%
$7 \times 7 \times 7$	10.18%	5.70%
$7 \times 7 \times 8$	10.99%	4.96%
$7 \times 8 \times 7$	15.81%	3.94%
$8 \times 7 \times 7$	13.52%	6.09%
$7 \times 8 \times 8$	13.04%	5.06%
$8 \times 7 \times 8$	10.64%	5.04%
$8 \times 8 \times 7$	10.09%	5.35%
$8 \times 8 \times 8$	11.46%	6.18%
$8 \times 8 \times 9$	6.65%	4.83%
mean	**%11.54**	**%5.28**

Table 5
Computation results of the algorithms for large size problems

Problem Size ($N \times M \times T$)	GA	ICA
$9 \times 9 \times 9$	12.13%	5.11%
$9 \times 9 \times 10$	12.94%	6.43%
$9 \times 10 \times 9$	10.67%	2.92%
$10 \times 9 \times 9$	6.68%	3.40%
$9 \times 10 \times 10$	10.61%	5.72%
$10 \times 9 \times 10$	8.42%	5.52%
$10 \times 10 \times 9$	11.15%	4.82%
$10 \times 10 \times 10$	10.73%	4.64%
$10 \times 10 \times 11$	7.85%	4.63%
$10 \times 11 \times 10$	10.54%	6.07%
$11 \times 10 \times 10$	9.37%	4.83%
$11 \times 11 \times 10$	10.74%	3.35%
$11 \times 10 \times 11$	10.94%	3.34%
$10 \times 11 \times 11$	13.10%	5.79%
$11 \times 11 \times 11$	9.26%	5.71%
$12 \times 12 \times 12$	11.63%	3.93%
$13 \times 13 \times 13$	13.05%	6.02%
$14 \times 14 \times 14$	9.19%	5.45%
$15 \times 15 \times 15$	-	-
$16 \times 16 \times 16$	-	-
Mean	**%10.50**	**%4.87**

Finding the optimum value for the second lower bound requires more than 7200 s and the objective value at this time has been considered.

From the computational results, the mean of performance measurement in small, medium and large problems shows that the proposed ICA outperforms the proposed GA. In the ICA, we have used some advanced techniques such as compound assimilation policy, compound revolution policy, designed improvisation scheme and hybridization continuous variables and binary variables. Despite the fact that the mean of measurement could be used to comparing two algorithms, we have used a well-known methodology to compare two algorithms statistically. A statistic experiment is carried out by means of a multi-factor analysis. Tukey Honest Significant Difference (HSD) confidence Intervals is employed to compare algorithms.

The results are showed in Fig. 9. According to this Fig., it is clear that the proposed ICA that are statistically better than the proposed GA in small, medium and large problems consequently in every size problems.

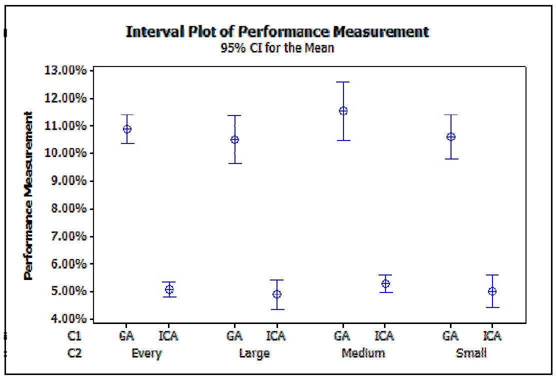


Fig. 9. Means and 95% LSD Intervals for the compared algorithms

5. Conclusions

To the best of our knowledge, this was the first reported comparison of the imperialist competitive algorithm and genetic algorithm for solving the capacitated lot sizing and scheduling problem with backlogging in flow shop environment. We proposed two metaheuristics algorithm (ICA and GA). As universalization, we have studied the developed models in three different categories (small problems, medium problems, and large problems). For each instance, because of different size of problems, the factors level differs and we have employed Taguchi method to calibrate each category separately. In addition, we have presented a lower bound to evaluate the performance of the algorithms. Some problems were designed in each category and were solved by the algorithms. The computational results and statistical comparisons demonstrated the superiority of the proposed ICA to GA in terms of solution quality. As the future work, employing other metaheuristics algorithm, such as Particle Swarm Optimization (PSO) and Harmony Search (HS) is proposed.

References

Abdi, B., Mozafari, H., Ayob, A., & Kohandel, R. (2011). Imperialist Competitive Algorithm and its Application in Optimization of Laminated Composite Structures. *European Journal of Scientific Research*, 55 (2), 174-187.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, *IEEE Congress on Evolutionary Computation, Singapore*, 4661–4667.

Araújo, C. D., Nagano, M. S. (2010). A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem. *International Journal of Industrial Engineering Computations*, 2, 155–166.

Bitran, G., &Yanasse, H. (1982).Computational complexity of the capacitated. *Management Science*, 28(10), 1174–1186.

Buschkühl, L., Sahling, F., Helber, S., &Tempelmeier, H. (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum*, 32, 231–261.

Drexl, A., & Kimms, A. (1997). Lot sizing and scheduling - Survey and extensions. *European Journal of Operational Research*, 99, 221-235.

Goren, H. G., Tunali, S., & Jans, R. (2010). A review of applications of genetic algorithms in lot sizing. *Journal of Intelligence Manufacturing*, 21, 575–590.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: The U. of Michigan Press

Kaveh, A., & Talatahari, S. (2010). Optimum design of skeletal structures using imperialist competitive algorithm. *Computer Structure*, 88(21–22), 1220–1229.

Kayvanfar, V., & Zandieh, M. (2012). The economic lot scheduling problem with deteriorating items and shortage: an imperialist competitive algorithm. *The International Journal of Advanced Manufacturing Technology*, doi:10.1007/s00170-011-3820-6

Khabbazi, A., Gargari, E., & Lucas, C. (2009). Imperialist competitive algorithm for minimum bit error rate beamforming. *International Journal of Bio-Inspiration Computing*, 1(1/2), 125–133.

Khanzadi, M., Soufipour R., & Rostami, M. (2011).A new improved genetic algorithm approach and a competitive heuristic method for large-scale multiple resource-constrained project-scheduling problems. *International Journal of Industrial Engineering Computations*, 2, 737–748.

Kimms, A. (1999). A genetic algorithm for multi-level, multi-machine lot sizing and scheduling. *Computers & Operations Research*, 26, 829-848.

Lang, J. C., &Shen, Z.-J.M. (2011).Fix-and-optimize heuristics for capacitated lot-sizing with sequence-dependent setups and substitutions. *Production, Manufacturing and Logistics*, 3, 214.

Lian, K., Zhang, C., Shao, X., & Gao, L. (2012). Optimization of process planning with various flexibilities using an imperialist competitive algorithm. *The International Journal of Advanced Manufacturing Technology*, 59, 815-828.

Logendran, R., Salmasi, N., &Sriskandarajah, C. (2006).Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. *ChelliahSriskandarajah*, 33, 158 – 180.

Lucas, C., Nasiri-Gheidari, Z., & Tootoonchian, F. (2010).Application of an imperialist competitive algorithm to the design of a linear induction motor. *Energy Conversion Management*, 51(7), 1407–1411.

Maes, J., McClain, J., & Van Wassenhove, L. (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research*, 53, 131-148.

Mohammadi, M. (2010).Integrating lotsizing, loading, and scheduling decisions in flexible flow shops.*The International Journal of Advanced Manufacturing Technology*, 50, 1165–1174.

Mohammadi, M., FatemiGhomi, S.-M.-T., Karimi, B., & Torabi, S-A. (2010a). MIP-based heuristics for lotsizing in capacitated pure flow shop with sequence-dependent setups. *International Journal of Production Research*, 48 (10), 2957–2973.

Mohammadi, M., FatemiGhomi, S.-M.-T., Karimi, B., & Torabi, S-A. (2010b). Rolling-horizon and fix-and-relax heuristics for the multi-product multi-level capacitated lotsizing problem with sequence-dependent setups. *Journal of Intelligent Manufacturing*, 21(4), 501–510.

Mohammadi, M., Ghomi, S. F., & Jafar, N. (2011). A genetic algorithm for simultaneous lotsizing and sequencing of the permutation flow shops with sequence-dependent setups. *Expert Systems with Applications*, 24, 87-93.

Montgomery, D. (2000). *Design and Analysis of Experiments*. New York: Wiley.

Nagano, M., Ruiz, R., & Lorena, L. (2008).A constructive genetic algorithm for permutation flowshop scheduling. *Computers & Industrial Engineering*, 55 (1), 195–207.

Nazari-Shirkouhi, S., Eivazy, H., Ghodsi, R., Rezaie, K., &Atashpaz- Gargari, E. (2010).*Computers and Industrial Engineering*, 37(12), 7615–7626.

Phadke, M. (1989).*Quality Engineering using Robust Design*. Engelwood Cliffs: Prentice-Hall.

Quadt, D., & Kuhn, H. (2008). Capacitated lot-sizing with extensions: a review. *4OR*, 6(1), 61–83.

Rajabioun, R., Atashpaz-Gargari, E., & Lucas, C. (2008).Colonial competitive algorithm as a tool for Nash equilibrium point achievement. *International Journal of Intelligent Computing and Cybernetics*, 49 (11), 680–695.

Sarayloo, F., &Tavakkoli-Moghaddam, R. (2010).Imperialistic competitive algorithm for solving a dynamic cell formation problem with production planning. *Advanced Intelligent Computing Theories and Applications*, 6215, 266–276.

Shim, I.-S., Kim, H.-C., Doh, H.-H., & Lee, D.-H.(2011). A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs. *Computers & Industrial Engineering*, 61(4), 920-929.

Shokrollahpour, E., Zandieh, M., & Dorri, B. (2010).A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. *International Journal of Production Research*, 49(11), 3087–3103.

Sun, H., Huang, H.-C., & Jaruphongsa, W. (2009).Genetic algorithms for the multiple-machine economic lot scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 43, 1251-1260.

Taguchi, G. (1986). Introduction to quality engineering. White Plains: Asian Productivity.

Tsai, J.T., Ho, W.H., Liu, T.K., & Chou, J.H. (2007). Improved immune algorithm for global numerical optimization and job-shop scheduling problems. *Applied Mathematics and Computation*, 194, 406–424.

Wagner, H.-M., & Whithin, T.-M.(1958). Dynamic version of the economic lot size model. *Management Science*, 5, (89–96).

Xiao, Y., Kaku, I., Zhao, Q., & Zhang, R. (2012).Neighborhood search techniques for solving uncapacitated multilevel lot-sizing problems. *Computers & Operations Research*, 39, 647–658.

Yao, M. J., & Huang, J. X. (2005).Solving the economic lot scheduling problem with deteriorating items using genetic algorithms. *Journal of Food Engineering*, 70, 309–322.

Zhua, X., & Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, 38, 987–1007.