International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times

F. Jolai [a], M. Rabiee [b] & H. Asefi [c]

[a] Department of Industrial Engineering, University of Tehran, Tehran, Iran

[b] Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran

[c] Department of Industrial Engineering, University of Tehran, Kish Int'l Campus, Iran

Available online: 13 Mar 2012

To cite this article: F. Jolai, M. Rabiee & H. Asefi (2012): A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times, International Journal of Production Research, DOI:10.1080/00207543.2011.653012

To link to this article: http://dx.doi.org/10.1080/00207543.2011.653012

International Journal of Production Research
2012, 1–20, iFirst

A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times

F. Jolai[a]*, M. Rabiee[b] and H. Asefi[c]

[a]*Department of Industrial Engineering, University of Tehran, Tehran, Iran;*
[b]*Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran;*
[c]*Department of Industrial Engineering, University of Tehran, Kish Int'l Campus, Iran*

(*Received 26 July 2011; final version received 21 December 2011*)

In this paper, we contemplate the problem of scheduling a set of n jobs in a no-wait flexible flow shop manufacturing system with sequence dependent setup times to minimising the maximum completion time. With respect to NP-hardness of the considered problem, there seems to be no avoiding application of metaheuristic approaches to achieve near-optimal solutions for this problem. For this reason, three novel metaheuristic algorithms, namely population based simulated annealing (PBSA), adapted imperialist competitive algorithm (AICA) and hybridisation of adapted imperialist competitive algorithm and population based simulated annealing (AICA + PBSA), are developed to solve the addressed problem. Because of the sensitivity of our proposed algorithm to parameter's values, we employed the Taguchi method as an optimisation technique to extensively tune different parameters of our algorithm to enhance solutions accuracy. These proposed algorithms were coded and tested on randomly generated instances, then to validate the effectiveness of them computational results are examined in terms of relative percentage deviation. Moreover, some sensitive analyses are carried out for appraising the behaviour of algorithms versus different conditions. The computational evaluations manifestly support the high performance of our proposed novel hybrid algorithm against other algorithms which were applied in literature for related production scheduling problems.

Keywords: no-wait; flexible flow shop; ICA; PBSA; Taguchi

1. Introduction

Scheduling is a decision-making process which may be explained as the allocation of resources to carry out certain tasks in an efficient manner. It is the process of making a sequence of operation on different machines in a manufacturing unit (Ponnambalam *et al*. 2001). Flexible flow shop is one of the most practical scheduling problems which have received considerable attention from researchers during the last two decades (Guinet *et al*. 1996, Kadipasaoglu 1997, Allaoui and Artiba 2004, Lee *et al*. 2004, Gholami and Zandieh 2009, Behnamian *et al*. 2010, Kia *et al*. 2010).

The flexible flow shop scheduling problem (FFSSP), also called multiprocessor flow shop or flow shop with parallel machines, consists of a set of two or more processing stages (or centres) with at least one stage having two or more parallel machines. The characteristic of a flexible flow shop is ubiquitously found in various industries such as textile and automobile manufacturing systems. The duplication of the number of machines in some stages can introduce additional flexibility, increase the overall capacities, and avoid bottlenecks if some operations are too long (Khalouli *et al*. 2010).

A distinguished class of scheduling problem is identified by a no-wait flexible flow shop. This manufacturing area can be modelled as a kind of flexible flow shop environment in which all of n jobs follow the same sequence at each stage, each of n jobs consist of k operations owning a predetermined processing order through machines. Each job is to be processed without pre-emption and interruption on k-stage or between them. That is, once a job is commenced on the first machine, it has to be continuously processed through the subsequent machine without interruption. In addition, each machine can handle no more than one job at a time and each job has to visit each machine exactly once. Therefore, it may mean that the starting time of a job on the first machine must be delayed in order to meet the no-wait requirement (Pan *et al*. 2008).

*Corresponding author. Email: fjolai@ut.ac.ir

ISSN 0020–7543 print/ISSN 1366–588X online

http://dx.doi.org/10.1080/00207543.2011.653012
http://www.tandfonline.com

There are three principal causes for 'no-waiting' in operations. The widespread reason comes back to the nature of the procedure. This situation occurs in chemical and petrochemical processing, plastic moulding and silverware production in which a series of processes must follow one another immediately to prevent degrading (Jolai *et al*. 2009). The second argument arises in industries faced with a lack of storage between intermediate machines (work stations). Finally, no-wait environments take place in service industries where customers should be served without any interruption between subsequent service stages. This kind of 'no-wait' environment is reasonable when the cost of waiting time is very high such as in emergency systems.

The no-wait flow shop scheduling problem has been studied over the last few decades, where the objective is proved to be strongly NP-hard when the number of machines is more than two (Rock 1984).

In some studies, the no-wait flow shop scheduling problem has been formulated as a travelling salesman problem (TSP). In this formulation, the waiting time in the operation intervals is converted into the distance matrix format of the TSP whereby the no-wait flow shop scheduling problem can be solved using the TSP technique. The original work conducted in this area refers to the research carried out by Gilmore and Gomory (1964). They studied a two-stage, single processor, no-wait flow shop problem using the TSP techniques. The results of the investigation revealed that a TSP-based, branch-and-bound algorithm obtained optimal solutions. In addition, in a two-stage no-wait flow shop with makespan performance Reddi and Ramamurty (1972) and Wismer (1972) have formulated the problem as an asymmetric TSP.

Salvador (1973) published one of the pioneer papers on no-wait hybrid flow shops by modelling the production system in the synthetic fibres industry as a no-wait hybrid flow shop. The proposed branch and bound explores only permutation sequences and jobs are assigned to the earliest available machine at each stage. The author employed dynamic programming for instances of a small size. Zhixin and Jiefang (2003) presented a heuristic algorithm named least deviation (LD) rule for two-stage no-wait hybrid flow shop scheduling with a single machine in each stage and makespan performance measure. The results of the study indicate that this algorithm is superior to other studied algorithms. Furthermore, the results showed that LD algorithm has low computational complexity and is easy to implement, thus it is the favoured application. Xie *et al*. (2004) proposed a new heuristic algorithm known as minimum deviation algorithm (MDA) to minimise makespan in a two-stage flexible flow shop with no-waiting time. The results of the study showed that MDA outperforms the partition method, the partition method with LPT, Johnson's and modified Johnson's algorithms. Xie and Wang (2005) consider the two-stage flexible flow shop scheduling problem with availability constraints. They discussed the complexity and the approximations of the problem and provide some approximation algorithms with worst case performance bounds for some special cases of the problem. Their results showed that the problems studied are more difficult to approximate than the case without availability constraints. Huang *et al*. (2009) considered a no-wait two-stage flexible flow shop with setup times and minimum total completion time performance measure. They proposed an integer programming model and ant colony optimisation (ACO) heuristic approach. The ACO results displayed that the efficiency of the proposed algorithm is superior to those solved by integer programming while having satisfactory solutions. Liu and Karimi (2008) proposed many mathematical models for the *m*-stage HFS with no-wait and/or limited storage, batching, identical as well as unrelated parallel machines and several optimisation criteria. Jolai *et al*. (2009) introduced no-wait flexible flow line scheduling problem with time windows and job rejection to maximising total profit. This is an extension of production and delivery scheduling problem with time windows. They also presented a mixed integer linear programming model and genetic algorithm (GA) procedures to solve their model efficiently. Comparison of the obtained results by GA with LINGO solutions and Tabu search showed that the proposed GA obtains better solutions in a very low computational time in comparison with the solutions obtained from LINGO optimisation software.

The majority of papers assume that the setup time is negligible or part of processing time. While this assumption simplifies the analysis and/or reflects certain applications, it adversely influences the solutions quality of many other applications of scheduling that require an explicit treatment of setup times (Allahverdi *et al*. 2008). Although machine setup time is a significant factor for production scheduling in all flow patterns, it may easily consume more than 20% of available machine capacity if not well handled (Pinedo 1995)

In many real-life situations such as chemical, printing, pharmaceutical, and automobile manufacturing, the setup operations, such as cleaning up or changing tools, are not only often required between jobs, but they are also strongly dependent on the immediately preceding process on the same machine (sequence-dependent) (Zandieh *et al*. 2006). In the literature, two types of sequence dependent setup times (SDST) were introduced. These are anticipatory sequence dependent setup time (ASDST) and non-anticipatory sequence dependent setup time (NSDST). For the case of anticipatory (or ASDST) the setup can begin even if the job is not available to be

processed and the corresponding machine is idle. Likewise, for the case of NSDST, the setup can begin only if both job and machine are available (Naderi *et al*. 2009).

Most scheduling problems are very difficult to solve (Blazewicz *et al*. 2001, Graham *et al*. 1979) and scheduling problems with SDSTs are among the most difficult classes of scheduling problems. A one-machine scheduling problem with SDST is NP-hard (Zandieh *et al*. 2006). Considering sequence-dependent setup times is gaining growing interest among researchers in recent years (Kurz and Askin 2004, Zhou *et al*. 2006, Ruiz and Stutzle 2008, Gholami and Zandie 2009, Naderi *et al*. 2009, Vinod and Sridharan 2009).

To the best of our knowledge, flexible flow shop scheduling problems considering no-wait constraint and sequence dependent setup times at the same time have not been investigated. Hence, in this research we proposed three advanced metaheuristic algorithms namely population-based simulated annealing (PBSA), adapted imperialist competitive algorithm (AICA) and a hybridisation version of AICA and PBSA (AICA + PBSA) for solving the no-wait flexible flow shop scheduling problem with SDST to minimising maximum completion time (i.e. makespan). The remainder of this paper is arranged as follows: in Section 2 we present the problem outline and assumptions. Section 3 elaborates the proposed algorithms for solving the considered problem. Section 4 addresses the computational evaluation including parameter calibration, data generation and experimental results. Finally, the conclusion and some further research suggestions are presented in Section 5.

2. Problem description

In this section, first the notations which are used to describe the problem are defined then the assumptions of the studied problem are elaborated with a schematic example.

2.1 *Notations*

The notations are defined as below:

n	The number of jobs to be scheduled ($j = 1, 2, \ldots, n$).
m^i	The number of parallel machines at stage i.
$p_{i,j}$	Processing time for job j at stage i ($i = 1, 2, \ldots, k$).
r_j	Ready time for job j.
s^i_{jl}	Sequence-dependent setup time from job l to job j at stage i.
π	Permutation of the given jobs ($\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$).
C_j	Completion time of job j.
$C_{\max}$	Maximum completion time of a schedule (Makespan) ($C_{\max} = \max(C_j)$.

2.2 *Assumptions*

The no-wait flexible flow shop scheduling with sequence dependent setup times can be described as follows: this problem consists of performing a set of n jobs, $J = \{j_1, j_2, \ldots, j_n\}$ which have to process through k subsequent stages. The number of machines in parallel at each stage k is shown by m_k. Processing time of job j in stage i is described by $p_{i,j}$. Sequence-dependent setup time between job j and job l at stage i is depicted by s^i_{jl}. The problem is to find a sequence so that the maximum completion time is minimised. The problem is shown by $\mathrm{FFS}|\mathrm{no}-\mathrm{wait},\ \mathrm{SDST}|C_{\max}$. With assumption of four jobs and two machines on three stages, the problem is schematically depicted in Figure 1 (assume that job sequences are 4–2-1–3).

The characteristics of the problem considered in this paper are as follows:

- All of the data in all problems which are used in this paper are known deterministically.
- Once began on the machine a job must be completed without interruption. That is, once a job is started on the first machine, it must be processed through all machines without any pre-emption and interruption.
- Each stage has at least one machine and at least one stage must have more than one machine.
- A machine can process only one job at a time.
- Each job has to visit each machine exactly once.

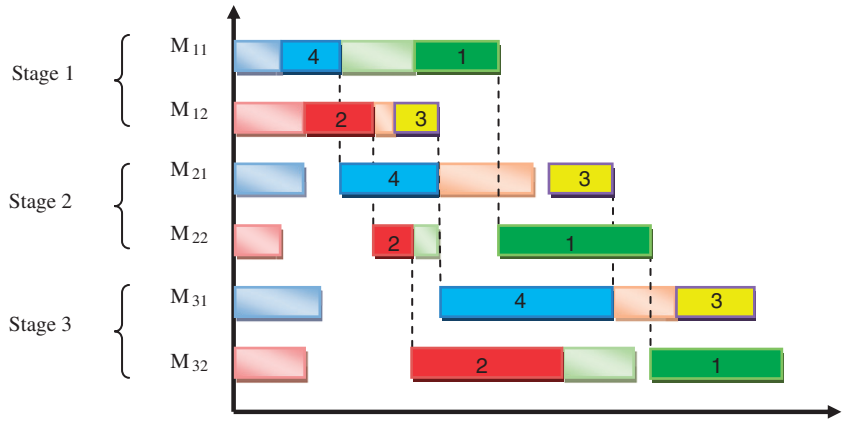

Figure 1. Schematic of the problem.

- All of the machines in all stages are available at all times, if they are not busy, with no breakdowns or scheduled or unscheduled maintenance.
- The release time of all jobs is zero. It means all jobs can be processed at the time 0.
- Setup times depend on sequencing of jobs which means the setup times are sequence dependent and the length of time required to do the setup depends on the prior and current jobs and the machine in mentioned stage to be processed (s^i_{jl}).

The no-wait, two-stage, flexible flow shop problems are extremely NP-hard (Sriscandarajah and Ladet 1986). So the no-wait k-stage flexible flow shop is NP-hard, too. Hence, all exact approaches for even simple problems will most likely have running times that increase exponentially with the problem size. In this paper three advance metaheuristic algorithms are suggested to the problem described above. The frameworks of these algorithms are explained in the next section.

3. Proposed algorithms

In this section, firstly the notations used to describe our proposed algorithms are defined, then three proposed algorithms, entitled population-based simulated annealing, adaptive imperialist competitive algorithm and hybridisation of population-based simulated annealing with adaptive imperialist competitive algorithm, are elaborated respectively for solving the addressed problem.

3.1 *Algorithms' notations*

3.1.1 *PBSA notations*

T_0	Start temperature.
T_f	Final temperature.
T_t	The temperature in tth iteration.
α	Cooling factor.
k	Boltzman constant.
n_{pop}	Number of initial solutions.
Max_{ipt}	Maximum iteration per temperature.
ΔE	Difference in fitness function between new and previous solution.
rand	A random number which are generated between zero and one.

3.1.2 *AICA notations*

Max*Dc*	Maximum decades.
PopSize	Number of initial countries.

P_i A socio-political specific of a country.
c_n The cost function for nth country.
N_{imp} Number of imperialists.
N_{col} Number of colonies.
k Boltzman constant.
C_n Normalised cost of nth country.
P_n Power of nth imperialist.
NC_n Difference in fitness function between new and previous solution.
P_{as} Percentage of assimilation.
ξ A positive constant for consideration of average power of colonies in each empire.
TC_n The total cost of the nth empire.
NTC_n The normalised total cost of nth empire.
PP_n Possession probability of each empire.
P_{ir} Percentage of imperialist revolution.
P_{cr} Percentage of colonies revolution.
P_r Probability of revolution.
I_{gw} Number of iterations for occurrence of global war.
N_{gw} Number of global war.

3.2 *Population based simulated annealing*

Simulated annealing (SA) is one of the well-known metaheuristic algorithms which are often used to solve many non-polynomial hard optimisation and operation research problems (Gaafar and Masoud 2005, Lin *et al*. 2009, Seyed-Alagheband *et al*. 2011, Yu *et al*. 2011). SA was introduced by Metropolis *et al*. (1953) and popularised by Kirkpatrick *et al*. (1983). In this paper, we applied population-based simulated annealing (PBSA) which has a similar structure to SA though had been developed by varying in number of initial generated solutions in order to achieve more accurate solutions by diversification. SA (also PBSA) is basically a simulation of the re-crystallisation of atoms in metal during its annealing. It begins with a starting temperature denoted by T_0 which decreases until it reaches its final temperature (T_f) using an annealing schedule to define how the temperature has changed during the annealing process.

The starting temperature must be hot enough to allow a move to almost any neighbourhood state. If this is not done then the final solution will be the same (or very close) to the original solution. However, if the temperature starts with a very high value, then the search can move to any neighbour and thus transform the search (at least in the early stages) into a random search. Effectively, the search will be random until the temperature is cool enough to start acting as a simulated annealing algorithm. It is usual to let the temperature decrease until it reaches zero. However, this can make the algorithm run for a lot longer, especially when a geometric cooling schedule is being used. In this paper we consider the geometric cooling approach (or linear decreasing scheme) which updates temperature at each time by the expression of $T_i = \alpha \times T_{i-1}$ where α represents a positive constant number less than one named cooling factor.

Algorithm proceeds by random initial solutions which the number of these solutions is considered as number of primary population (n_{pop}). Each solution for the addressed problem in SA is considered by an array in size of job numbers and also machine assignment in SA is based on the first available machine (FAM) rule. According to the fitness of the problem (i.e. makespan), the cost of each candidate is calculated in order to determine the best solution at each temperature. These best solutions are used to generate the next candidates using an approach such as neighbourhood generation. For the neighbourhood procedure three cases are applied: swap, insertion and reversion. The structures of these operators are described below:

- *Swap:* The positions of selected jobs (i.e. Jobs 5 and 6 in Figure 2) are exchanged (Figure 2(a)).
- *Reversion:* In this policy besides conducting swap, the jobs located in between the swapped jobs are reversed, too (Figure 2(b)).
- *Insertion:* In this case the job in the second position is located immediately after the job in the first location and the other jobs are shifted to the right hand side accordingly (see Figure 2(c)).

Figure 2. Swap, reversion and insertion mutations.

```
Begin
Initialisation %
    Set Parameters (n_Pop, T_0, T_f, a, Max_ipt)
    Generating initial population (Randomly)
    Evaluate fitness of each solution
    Consider Max_ipt = 0, (x_1, x_2, ..., x_n) = (x_1,best, x_2,best, ..., x_n,best)
    (f(x_1), f(x_2), ..., f(x_n)) = (f(x_1,best), f(x_2,best), ..., f(x_n,best)) and
    f(x_Total best) = min((f(x_1,best), f(x_2,best), ..., f(x_n,best))
Main loop of algorithm %
  While T_i ≠ T_F do
    for it = 1 : Max_ipt
      %Generating new solutions by using one of neighborhood
        procedures (i.e. swap, insertion and reversion) randomly
        and evaluate fitness function for these solutions
        if Δ E < 0
          x_new = x_best and f(x_new) = f(x_best)
          else if rand < e^(-ΔE / KT_i)
          x_new = x_best and f(x_new) = f(x_best)
          end
        end
      end
    else report the best obtained solution
  end
end
```

Figure 3. The pseudo code of population based simulated annealing.

The algorithm accepts candidates if there is improvement in the fitness but, to avoid local optimum solutions, the algorithm also allows others to be kept with a probability obtained from Boltzman distribution as below:

$$P(\Delta E) = e^{\frac{-\Delta E}{KT_i}} \tag{1}$$

where ΔE is the difference in fitness between the old and new states and T_i denotes the temperature of the process and k is a constant parameter of the process (in this paper k equal to one is considered). In order to find best solutions this procedure executes at each temperature by a determined number which is denoted as Max_{ipt} (maximum iteration per temperature) until T_f. The pseudo code of the population-based simulated annealing algorithm which is applied in this investigation is presented in Figure 3.

3.3 *Adapted imperialist competitive algorithm*

Imperialist competitive algorithm (ICA) is a novel evolutionary algorithm for optimisation. ICA uses the socio-political evolution of humans as a source of inspiration for developing a strong optimisation strategy (Atashpas-Gargari and Lucas 2007). This algorithm starts with an initial population named 'Country' in which these countries are divided into two types of colonies and imperialists according to their power. Some of the best (most powerful) countries are selected to be the imperialist and the rest of the colonies are distributed among the imperialists. The more powerful imperialists have more colonies. The power of each country is a simile of its objective function value in the considering model. Following this approach, initial empires are created and competition begins. Imperialists try to achieve more colonies and colonies tend to move towards their imperialists. This competition leads to the collapse of weak empires and the extension of more powerful ones. At the end just one imperialist will remain which has the same position as its colonies.

Figure 4. The structure of one solution for a problem with six jobs.

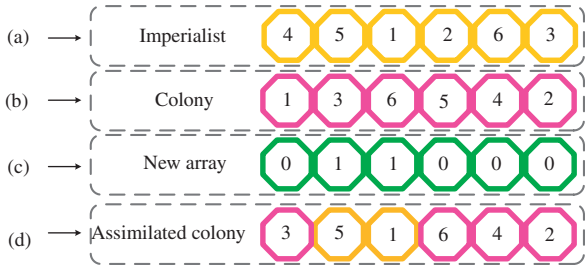

Figure 5. Imperialist and colony's array, new array and assimilated colony.

3.3.1 *Generating initial empires*

In this algorithm we denote solutions by a $1 \times N$ array called Country of which N is the dimension of the optimisation problem and is defined by:

$$\text{country} = [p_1, p_2, p_3, \ldots, p_N] \tag{2}$$

where P_i is a socio-political specific of a country and is considered to be optimised. In our model N represents the number of jobs which their sequence creates a $1 \times N$ array as a solution (country). Jobs are allocated to earliest available machine in order to their sequence at each stage. The structure of one solution for a problem with six jobs is shown in Figure 4.

To evaluate the cost of each solution, a cost function (f) (i.e. makespan) at the variables $(P_1, P_2, \ldots, P_N)$ is used as follows:

$$c_i = f(\text{country}_i) = f(p_{i1}, p_{i2}, \ldots, p_{iN}) \tag{3}$$

For creation of the initial empires, we select the number of N_{imp} countries which have the minimum cost as imperialists and consider all remaining countries as colonies with the number of N_{col}, where these colonies are distributed among the imperialists according to each imperialist normalised cost.

The normalised cost of each imperialist represents imperialist power to absorb colonies and is calculated as follows.

$$C_n = \max_i c_i - c_n \tag{4}$$

where c_n is the cost of nth imperialist and C_n is its normalised cost which is equal to deviation of the maximum total completion time ($C_{\max}$) from the nth imperialist cost.

To measure the proportional power for each imperialist (P_n) the following equation is used:

$$P_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \tag{5}$$

Then the initial number of colonies which could be belonged to nth imperialist is:

$$NC_n = \text{round}\{P_n.N_{Col}\} \tag{6}$$

We select NC_n of colonies randomly to allocate them to each imperialist in which the more powerful imperialists, have the more colonies.

3.3.2 *Assimilation*

In each empire, colonies tend to improve their power moving toward their imperialist which is called assimilating. The imperialist's and colony's array samples are shown in Figure 5.

Using this approach to execute assimilating, a percentage of job numbers in the colony's sequence array will be selected randomly to be the same as the imperialist's array. So we generate a new array the values of which could be one or zero in a random order by considering the number of ones which must be equal to the mentioned percentage named as a percentage of assimilation(P_{as}). A new random generated array is illustrated in Figure 5(c). Then the

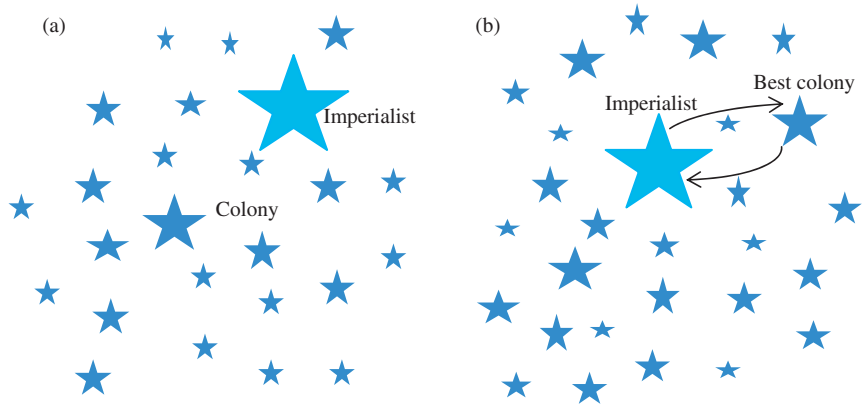

Figure 6. Exchanging positions of the imperialist and a colony.

subsequent job positions are determined base on the orders defined in the colonies (i.e. 2th sequence for job number 5, 3th sequence for job number 1). The obtained job sequence for our example is shown in Figure 5(d).

3.3.3 *Exchanging positions between imperialists and colonies*

During the competition and moving colonies towards the imperialists a colony might keep improving and achieve a cost less than the imperialist. It leads to swapped positions between the imperialist and the improved colony and creation of a new empire. This process will proceed during the algorithm (see Figure 6).

3.3.4 *Total power an empire*

In order to measure total power of an empire both the imperialist's power and cumulative power of colonies are considered, but the main affect caused by imperialist's power through the cumulative power of the colonies has a partial share with a positive constant factor between 0 and 1 called ξ. So, the equation of total cost is defined as follows:

$$TC_n = \text{cost}(\text{imperialist}_n) + \xi\,\text{mean}\{\text{cost}(\text{colonies of empire}_n)\} \tag{7}$$

where TC_n is the total cost of the nth empire and ξ is a constant, as explained, to adjust the effect of cumulative power of colonies in the equation.

3.3.5 *Imperialistic competition*

As all empires try to take the possession of the colonies of the other empires and control them, the imperialistic competition gradually brings about a decrease in the power of weaker empires and an increase in the power of the more powerful ones (Atashpas-Gargari and Lucas 2007). Imperialistic competition executes by picking one (or some) colony or colonies (Mohammadi *et al*. 2011). By this procedure, the most powerful empires will be more likely to possess weak colonies according to the possession probability of each empire. We will use this probability in the Roulette Wheel method for assigning the mentioned colonies to the empires. To determine possession probability of each empire, first the normalised total cost is computed as below:

$$NTC_n = \max\{TC_i\} - TC_n \tag{8}$$

where NTC_n is the normalised total cost of nth empire and TC_n is the total cost of nth empire. Having normalised the total cost, the possession probability of each empire is calculated as below:

$$pp_n = \left| \frac{NTC_n}{\sum_{i=1}^{N_{\text{Imp}}} NTC_i} \right| \tag{9}$$

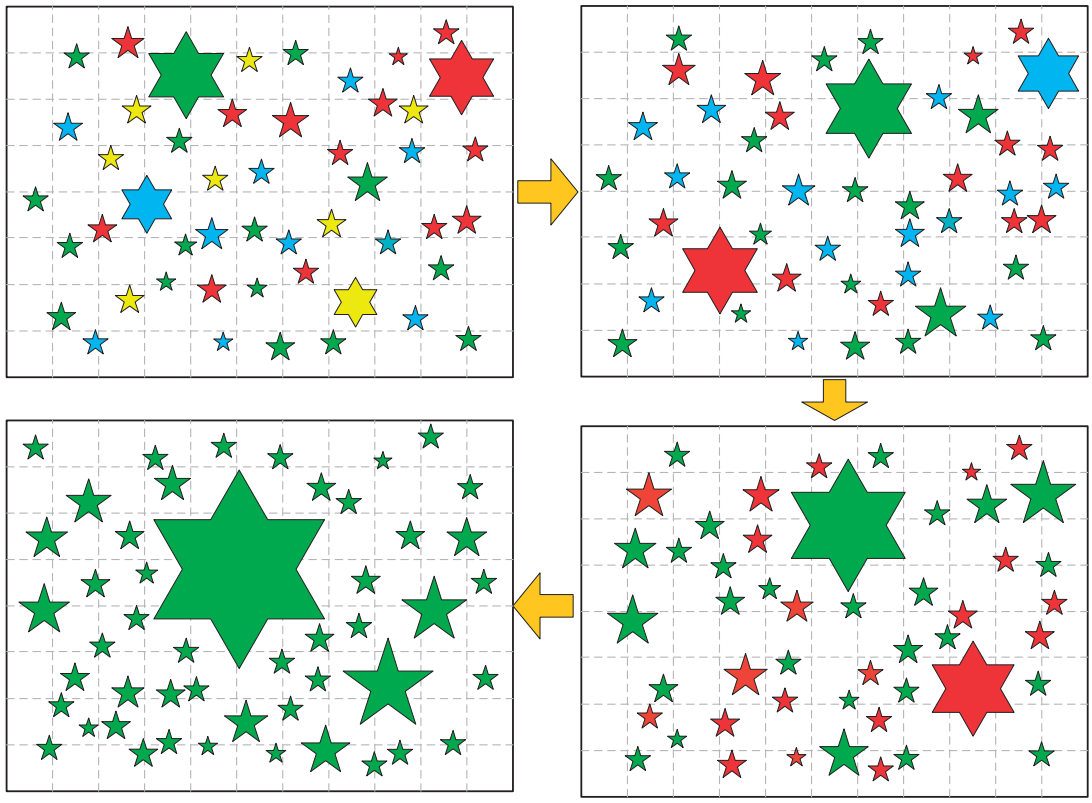

Figure 7. AICA imperialist competition.

Then we use the Roulette Wheel method for assigning the mentioned colonies to the empires. The structure of imperialist competition is schematically shown in Figure 7.

3.3.6 *Revolution*

Our revolution policy to produce new solutions proceeds by exchanging two positions of each imperialist array together and replacing newly obtained imperialist with the weakest imperialist colony. We will repeat it for a percentage of jobs for each imperialist named as percentage of imperialist revolution (P_{ir}) and will continue this process at iteration. We also adopt this revolution structure for colonies in which some colonies are selected to exchange two positions of their array together in a random selection and it is repeated for a percentage of jobs for each colony named percentage of colonies revolution (P_{cr}). We define the replacement ratio as the revolution rate and named revolution probability (P_r).

3.3.7 *Eliminating the weak empires*

During imperialistic competition, powerless empires will collapse so that their colonies will be allocated to other powerful empires. We define a collapsed empire as an empire which has lost all of its colonies.

3.3.8 *Global war*

To extend the best solutions search, we define a global war named I_{gw} which occurs after determined iterations of algorithm. When it happens, new countries equal to the number of the initial population will be generated and merged with the old population then the most powerful countries as the number of initial population will be selected according to sorted order of them based on their cost functions. This process is repeated a number of times which is denoted by N_{gw}.

```
Begin
Initialisation %
        – Set Parameters ((MaxDc, PopSize), N_imp, , P_as, I_gw, N_gw , P_ir, P_cr, P_r)
        – Generating initial Countries (Randomly)
        – Evaluate fitness of each country
Main loop of algorithm %
For it = 1 : MaxDc
 Form initial empires %
        – Choice most powerful countries as the imperialists
        – Assign other countries to imperialists based on imperialist power
  Assimilation %
       – Move the colonies of an empire toward the imperialist
      – Revolution among colonies and imperialist
       If cost of colony < own imperialist
        exchanging positions of the imperialist and related colony
       end
  Imperialistic competition %
       – Calculate Total power of the empires
      – Select the weakest colony of the weakest empire and assign this to
        one of the strange empires based on their powers
     – Eliminate the powerless empires (the imperialist with no colony)
  Global war %
    if global war condition is met
        – Form new countries as the same number as PopSize
        – Evaluate fitness of each country
        – Merge new and old countries
     end
 Selection for next iteration %
        Sort merged countries and choice the first countries with the size of the
        PopSize as new population
  end
end
```

Figure 8. The pseudo code of adapted imperialist competitive algorithm.

3.3.9 *Stopping criteria*

In this paper we assume imperialistic competition will be finished when an algorithm reaches the maximum decade. The pseudo code of adapted imperialist competitive algorithm is illustrated in Figure 8.

3.4 *Hybridisation of adapted imperialist competitive algorithm and population based simulated annealing*

A novel algorithm, which is hybridisation of adapted imperialist competitive algorithm and population based simulated annealing (AICA + PBSA), is described in this section. It has a similar base to AICA though applies a local search (PBSA) to improve imperialists in addition. In this algorithm, the numbers of PBSA initial solutions are equal to the number of imperialists and the number of outputs is the same as well. Note that solution representation in this algorithm is same as AICA and PBSA and also machine assignments are performed based on first available machine at each stage. Pseudo code for this proposed algorithm is depicted in Figure 9.

4. Experimental evaluation

4.1 *Generation of test data*

By the validation of proposed algorithms plus comparison between them, numbers of random problems are generated by considering five parameters as follows: number of jobs, number of stages, machine's distributions,

```
Begin
Initialisation %
        – Set Parameters ((MaxDc, PopSize), N_imp, P_as, I_gw, N_gw, P_ir, P_cr, P_r, n_Pop, T_0, T_f, a, Max_ipt)
        – Generating initial Countries (Randomly)
        – Evaluate fitness of each country
Main loop of algorithm %
 For it = 1 : MaxDc
 Formation of empires %
        – Choice most powerful countries as the imperialists
        – Use PBSA as a local search for reach to better imperialists
        – Assign other countries to imperialists based on imperialist power
 Assimilation %
       – Move the colonies of an empire toward the imperialist
       – Revolution among colonies and imperialist
        If cost of colony < own imperialist
         exchanging positions of the imperialist and related colony
        end
 Imperialistic competition %
       – Calculate Total power of the empires
       – Select the weakest colony of the weakest empire and assign this to
         one of the strange empires based on their powers
      – Eliminate the powerless empires (the imperialist with no colony)
 Global war %
    if global war condition is met
        – Form new countries as the same number as PopSize
        – Evaluate fitness of each country
        – Merge new and old countries
     end
 Selection for next iteration %
        – Sort merged countries and choice the first countries with the size of the
         PopSize as new population
 end
end
```

Figure 9. The pseudo code of hybridisation of AICA and PBSA.

Table 1. Factors and their levels.

Factors	Levels
Number of jobs	Small: 4, 5, 6, 7, 8 Large: 40, 60, 80, 100, 120
Number of stages	Small: 2, 3, 4 Large: 4, 6, 8
Machine distribution	Small: 1-constant (2) 2-variable (U(1, 2)) Large: 1-constant (3) 2-variable (U(1, 6))
Processing time	U(1, 99)
SDST	U(1, 25), U(1, 50)

processing times and sequence dependent setup times. Problems are categorised in two types: small and large. Numbers of jobs for each scale are considered separately. The value of the number of jobs in both scales is shown in Table 1. Moreover, numbers of machines in each stage are produced in two groups for each scale in first group number of machines at each stage are considered equal as are shown in Table 1.

Uniform distribution is used to generate processing times and sequence dependent setup times which processing times are generated by U(1, 99) also U(1, 25), U(1, 50) are used for sequence dependent setup times generation for both problem categories. In addition, machines are generated using a uniform distribution which is demonstrated

in Table 1. All parameters and their levels are shown in Table 1 briefly. Generally, with respect to parameters and their levels, 60 small size class problems and 60 large size are defined. Moreover, for each class, 10 random problems are generated. In other words, each proposed algorithm is run 1200 times.

4.2 *Parameter tuning*

It is known that the great choice of parameters has a striking impact on performance of algorithms. Furthermore, the suitable design parameter values highly depend on the type of problems. Most research has been conducted using evolutionary algorithms with fixed parameter values after some preliminary experiment or have been fixed with reference to values of the previous similar literature. The main motive of this behaviour is related to the large number of parameters and their levels; because a comprehensive calibration requires time and is resource-consuming. Calibration plays a prominent role in the improvement of performance of algorithms and in some cases it is a compulsory step in the development of algorithms. For the purpose of calibration of algorithms several methods were used in the literature. However, the most frequently used and exhaustive approach is a full factorial experiment (Montgomery 2000, Ruiz *et al*. 2006). This methodology is usually utilised when number of factors and their levels, plus CPU time of algorithm, is small or moderate. Using this approach is difficult for algorithms with numerous factors and levels and high CPU time. To diminish the number of required tests, fractional factorial experiment (FFE) was developed (Cochran and Cox 1992). FFEs permit only a portion of the total possible combinations to estimate the main effect of factors and some of their interactions (Naderi *et al*. 2010)

A family of matrices decreasing the number of experiments is established by Ross (1989). Taguchi developed a family of FFE matrices which eventually reduce the number of experiments, but still provide sufficient information. In the Taguchi method, the orthogonal arrays are used to study a large number of decision variables with a small number of experiments (Ross 1989)

The experimental design proposed by Taguchi involves using orthogonal arrays to organise the parameters affecting the process and the levels at which they should be varying. Instead of having to test all possible combinations like the factorial design, the Taguchi method tests pairs of combinations. This makes necessary collection of the data to determine which factors have most significant effects on product quality with a minimum amount of experiment, thus saving time and resources. An advantage of the Taguchi method is that it emphasises a mean performance characteristic value close to the target value rather than a value within certain specification limits, thus improving the final quality. Additionally, the Taguchi method for experimental design is straightforward and easy to apply for many engineering situations. This makes it a powerful yet simple tool. It can be used for various research projects or to identify problems in a manufacturing process. Also, it has a wide range of application in analysis of many different parameters without a prohibitively high amount of experimentation.

In the Taguchi approach, variables are classified in two groups: controllable and noise factors (uncontrollable). Noise factors are those over which we have no direct control. Since elimination of the noise factors is impractical and often impossible, the Taguchi method seeks to minimise the effect of noises and determines optimal levels of important controllable factors based on the concept of robustness (Phadke 1989).

Besides determining the optimal levels, Taguchi identifies the relative significance of individual factors in terms of their main effects on the objective function. Taguchi has created a transformation of the repetition data to another value which is the measure of variation. The transformation is the signal-to-noise (S/N) ratio which explains why this type of parameter design is called robust design (Phadke 1989, Al-Aomar 2006). Here, the term 'signal' denotes the desirable value (mean response variable) and 'noise' denotes the undesirable value (standard deviation); so the S/N ratio indicates the amount of variation present in the response variable. The aim is to maximise the signal-to-noise ratio.

Taguchi categorises objective functions into three groups: the smaller-the-better type, the larger-the-better type and nominal-is-best type. Since almost all objective functions in scheduling are categorised in the smaller-the-better type, its corresponding S/N ratio (Phadke 1989) is:

$$S/N \text{ ratio} = -\log_{10}\left(\frac{1}{n}\sum_{i=1}^{n}(\text{objective function})_i^2\right) \tag{10}$$

Before calibration of AICA + PBSA, the algorithm is subjected to some preliminary tests to obtain the effective parameters and proper parameter levels to be tested in the fine-tuning process. Some quick experiments showed that

P_{as}, P_{ir} and P_{cr} have not any significant impact on the performance of the algorithm. So we fixed them with $P_{as} = 0.3$, $P_{ir} = 0.2$ and $P_{cr} = 0.2$.

In order to achieve to more accurate and stable results for our proposed algorithm, we considered 11 parameters for tuning. These parameters are (Max_{Dc}, *PopSize*), N_{imp}, ξ, P_r, $I_{gw,}$ N_{gw}, n_{Pop}, T_0, T_f, α, Max_{ipt}; they are shown with their level in Table 2. The considered orthogonal array with eleven factors and three levels in Taguchi method is L_{27}. The orthogonal array L_{27} is presented in Table 3.

By calculating all of the experimental results in the Taguchi method, the average S/N ratio and average maximum completion time were obtained for both scales. Figures 10 and 11 display the average S/N ratio obtained for small and large scales at each level, respectively. In small scale as illustrated in Figure 10, optimal levels are A(1), B(2), C(1), D(1), E(3), F(3), G(3), H(3), J(3), K(2), L(1) also optimal levels for large scale problems are

Table 2. Parameters and their levels.

		Parameters										
		A	B	C	D	E	F	G	H	J	K	L
Scale	Level	(Max_{Dc}, *PopSize*)	N_{imp}	ξ	P_r	I_{gw}	N_{gw}	T_f	T_0	n_{Pop}	Max_{ipt}	α
Small	1	(200, 50)	3	0.10	0.2	0.20* Max_{Dc}	0	20	0.10	1	2	0.90
	2	(100, 100)	4	0.15	0.3	0.30* Max_{Dc}	1	30	0.20	2	3	0.95
	3	(50, 200)	5	0.20	0.4	0.40* Max_{Dc}	2	40	0.30	3	4	0.99
Large	1	(300, 100)	6	0.20	0.2	0.20* Max_{Dc}	1	40	0.01	3	4	0.90
	2	(200, 150)	8	0.25	0.3	0.25* Max_{Dc}	2	50	0.02	4	6	0.95
	3	(100, 300)	10	0.30	0.4	0.30* Max_{Dc}	3	60	0.03	5	8	0.99

Table 3. The orthogonal array L_{27}.

Experiments	A	B	C	D	E	F	G	H	J	K	L
1	A(1)	B(1)	C(1)	D(1)	E(1)	F(1)	G(1)	H(1)	J(1)	K(1)	L(1)
2	A(1)	B(1)	C(1)	D(1)	E(2)	F(2)	G(2)	H(2)	J(2)	K(2)	L(2)
3	A(1)	B(1)	C(1)	D(1)	E(3)	F(3)	G(3)	H(3)	J(3)	K(3)	L(3)
4	A(1)	B(2)	C(2)	D(2)	E(1)	F(1)	G(1)	H(2)	J(2)	K(2)	L(3)
5	A(1)	B(2)	C(2)	D(2)	E(2)	F(2)	G(2)	H(3)	J(3)	K(3)	L(1)
6	A(1)	B(2)	C(2)	D(2)	E(3)	F(3)	G(3)	H(1)	J(1)	K(1)	L(2)
7	A(1)	B(3)	C(3)	D(3)	E(1)	F(1)	G(1)	H(3)	J(3)	K(3)	L(2)
8	A(1)	B(3)	C(3)	D(3)	E(2)	F(2)	G(2)	H(1)	J(1)	K(1)	L(3)
9	A(1)	B(3)	C(3)	D(3)	E(3)	F(3)	G(3)	H(2)	J(2)	K(2)	L(1)
10	A(2)	B(1)	C(2)	D(3)	E(1)	F(2)	G(3)	H(1)	J(2)	K(3)	L(1)
11	A(2)	B(1)	C(2)	D(3)	E(2)	F(3)	G(1)	H(2)	J(3)	K(1)	L(2)
12	A(2)	B(1)	C(2)	D(3)	E(3)	F(1)	G(2)	H(3)	J(1)	K(2)	L(3)
13	A(2)	B(2)	C(3)	D(1)	E(1)	F(2)	G(3)	H(2)	J(3)	K(1)	L(3)
14	A(2)	B(2)	C(3)	D(1)	E(2)	F(3)	G(1)	H(3)	J(1)	K(2)	L(1)
15	A(2)	B(2)	C(3)	D(1)	E(3)	F(1)	G(2)	H(1)	J(2)	K(3)	L(2)
16	A(2)	B(3)	C(1)	D(2)	E(1)	F(2)	G(3)	H(3)	J(1)	K(2)	L(2)
17	A(2)	B(3)	C(1)	D(2)	E(2)	F(3)	G(1)	H(1)	J(2)	K(3)	L(3)
18	A(2)	B(3)	C(1)	D(2)	E(3)	F(1)	G(2)	H(2)	J(3)	K(1)	L(1)
19	A(3)	B(1)	C(3)	D(2)	E(1)	F(3)	G(2)	H(1)	J(3)	K(2)	L(1)
20	A(3)	B(1)	C(3)	D(2)	E(2)	F(1)	G(3)	H(2)	J(1)	K(3)	L(2)
21	A(3)	B(1)	C(3)	D(2)	E(3)	F(2)	G(1)	H(3)	J(2)	K(1)	L(3)
22	A(3)	B(2)	C(1)	D(3)	E(1)	F(3)	G(2)	H(2)	J(1)	K(3)	L(3)
23	A(3)	B(2)	C(1)	D(3)	E(2)	F(1)	G(3)	H(3)	J(2)	K(1)	L(1)
24	A(3)	B(2)	C(1)	D(3)	E(3)	F(2)	G(1)	H(1)	J(3)	K(2)	L(2)
25	A(3)	B(3)	C(2)	D(1)	E(1)	F(3)	G(2)	H(3)	J(2)	K(1)	L(2)
26	A(3)	B(3)	C(2)	D(1)	E(2)	F(1)	G(3)	H(1)	J(3)	K(2)	L(3)
27	A(3)	B(3)	C(2)	D(1)	E(3)	F(2)	G(1)	H(2)	J(1)	K(3)	L(1)

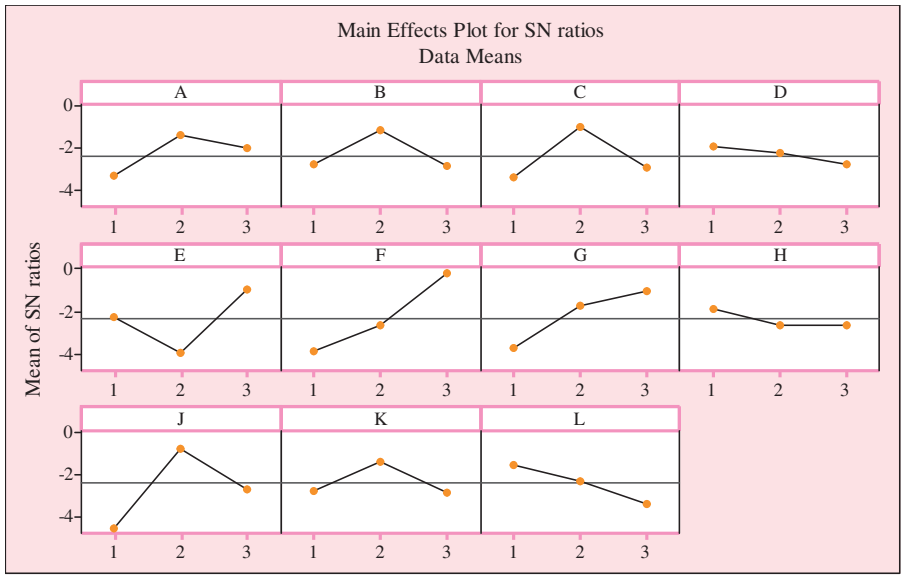

Figure 10. The S/N ratio plot for small scale in Taguchi methodology.

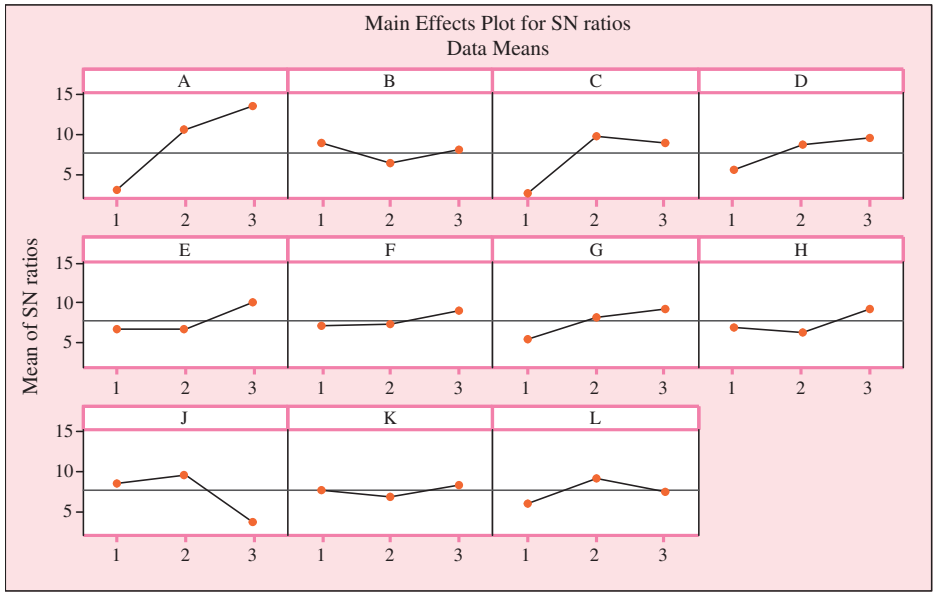

Figure 11. The S/N ratio plot for large scale in Taguchi methodology.

demonstrated in Figure 11 which those are A(3), B(3), C(2), D(3), E(3), F(3), G(3), H(2), J(2), K(2), L(2). Furthermore, computed results in terms of RPD by considering mean makespan in Taguchi experimental analysis confirmed the achieved optimal levels using S/N ratio for both scales (see Figures 12 and 13). Moreover, Tables 4 and 5 exhibit the order of factor in minimising makespan in rank row for small and large scale, respectively.

4.3 *Computational evaluation*

In this section, the performance of suggested PBSA, AICA and AICA + PBSA are evaluated. All proposed algorithms were implemented in MATLAB 7.9 and run on 24 parallel PC with Pentium IV and 2.4 GHZ processor and 512 RAM memory.

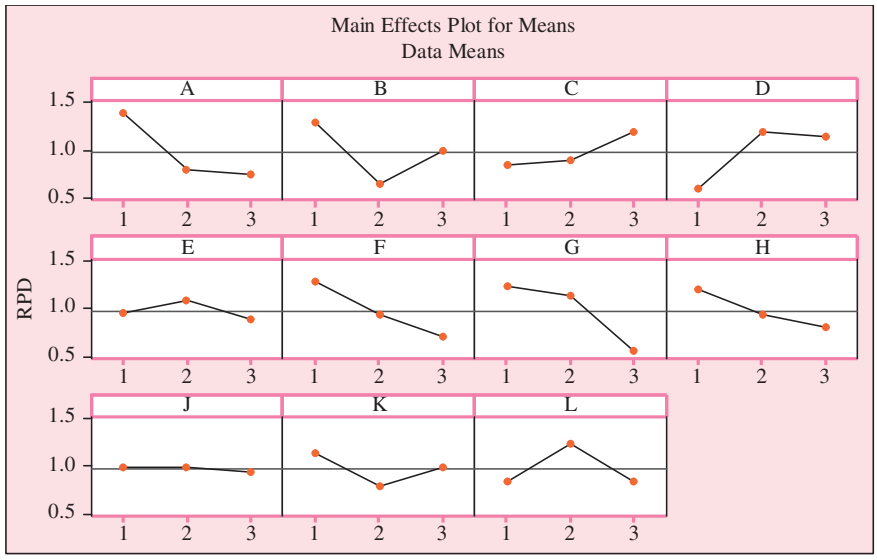

Figure 12. The RPD plot for small scale in Taguchi methodology.

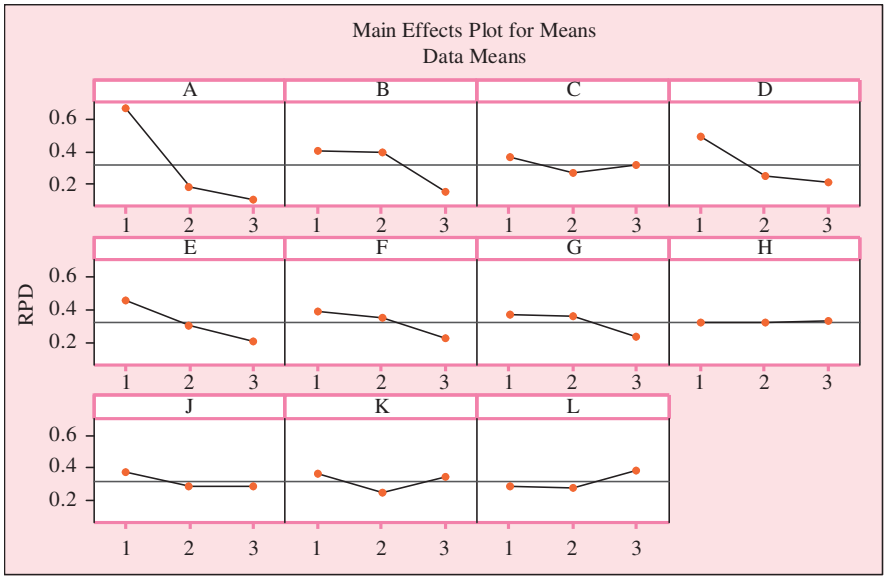

Figure 13. The RPD plot for large scale in Taguchi methodology.

Table 4. S/N ratio table for small scale.

Level	A	B	C	D	E	F	G	H	J	K	L
1	1.391	1.290	0.842	0.597	0.945	1.292	1.239	1.193	0.995	1.142	0.844
2	0.792	0.644	0.892	1.189	1.091	0.941	1.141	0.941	0.991	0.792	1.242
3	0.745	0.994	1.195	1.142	0.892	0.695	0.548	0.793	0.941	0.994	0.842
Delta	0.646	0.646	0.353	0.593	0.199	0.597	0.691	0.400	0.054	0.350	0.400
Rank	2.5	2.5	8	5	10	4	1	6	11	9	7

Table 5. S/N ratio table for large scale.

Level	A	B	C	D	E	F	G	H	J	K	L
1	2.975	8.813	2.616	5.586	6.710	7.067	5.469	6.982	8.425	7.588	6.020
2	10.540	6.398	9.729	8.663	6.565	7.241	8.127	6.304	9.511	6.738	9.024
3	13.545	8.066	8.834	9.652	10.176	8.965	9.288	9.178	3.659	8.314	7.493
Delta	10.571	2.415	7.112	4.067	3.610	1.898	3.819	2.874	5.852	1.576	3.003
Rank	1	9	2	4	6	10	5	8	3	11	7

Table 6. ARPD for the algorithms in small scale.

Row	Job × Stage	PBSA	AICA	AICA + PBSA
1	4 × 2	**0.0000**	**0.0000**	**0.0000**
2	4 × 3	**0.0000**	**0.0000**	**0.0000**
3	4 × 4	**0.0000**	**0.0000**	**0.0000**
4	5 × 2	**0.0000**	**0.0000**	**0.0000**
5	5 × 3	**0.0000**	**0.0000**	**0.0000**
6	5 × 4	**0.0000**	**0.0000**	**0.0000**
7	6 × 2	0.4299	**0.0856**	**0.0856**
8	6 × 3	0.7527	**0.3237**	0.4084
9	6 × 4	1.1422	**0.1858**	0.2408
10	7 × 2	0.8517	0.1945	**0.1898**
11	7 × 3	1.2101	**0.1692**	0.2652
12	7 × 4	1.2196	0.3130	**0.2977**
13	8 × 2	1.7491	0.5125	**0.3394**
14	8 × 3	1.7065	0.9090	**0.4013**
15	8 × 4	1.0924	0.5471	**0.1947**
Total	Average	0.6770	0.2160	**0.1615**

Table 7. ARPD for the algorithms in large scale.

Row	Job × Stage	PBSA	AICA	AICA + PBSA
1	40 × 4	0.5978	0.3667	**0.0342**
2	40 × 6	0.9906	0.7484	**0.0536**
3	40 × 8	1.0464	0.6247	**0.0200**
4	60 × 4	1.2111	0.7238	**0.0273**
5	60 × 6	0.6956	0.4246	**0.0122**
6	60 × 8	0.7695	0.4290	**0.0290**
7	80 × 4	1.0411	0.7628	**0.0752**
8	80 × 6	0.9381	0.9621	**0.0174**
9	80 × 8	0.9364	0.6687	**0.0252**
10	100 × 4	0.9892	0.5010	**0.0550**
11	100 × 6	1.1810	0.6820	**0.0126**
12	100 × 8	1.2774	0.6594	**0.0264**
13	120 × 4	0.7936	0.4856	**0.0053**
14	120 × 6	1.0586	0.5661	**0.0098**
15	120 × 8	1.1263	0.6423	**0.0219**
Total	Average	0.9768	0.6165	**0.0283**

Algorithm comparison is performed using a usual performance measure which is known as RPD (relative percentage deviation) to evaluate them. The best obtained solutions for each instance are calculated. *RPD* is computed by the given formula as below:

$$RPD = \frac{alg_{sol} - \min_{sol}}{\min_{sol}} \times 100 \tag{11}$$

where alg_{sol} is the obtained makespan value for a given algorithm in an instance and $\min_{sol}$ is the best value through algorithms in a related problem. It is clear that for each algorithm, closer obtained limits to zero and no-overlapping with other algorithms in their upper and lower limits, shows which has yielded more proper solutions.

As mentioned before in order to evaluate performance of suggested algorithms, a number of random problems in two types of small and large size are generated. After running algorithms, we converted raw data to *RPD*. *RPDs* average values (i.e. average relative percentage deviation or *ARPD*) for small and large problems are shown in Tables 6 and 7 respectively. In each type of problem, values of the algorithm which produces the better result are shown in bold. The average of total results in small and large size problems implies transcendence of AICA + PBSA though it is more sensible in large size problems.

Also for significant statistical analysis of difference among algorithms, the 95% confidence interval for computed values in RPD is calculated in both types of problems. Figures 14 and 15 illustrate the 95% confidence interval for related RPD to each algorithm in small and large size respectively. As it can be seen in Figure 14, there is statistical significant difference between PBSA and AICA, AICA + PBSA where they treat by better performance, whereas there is not statistical significant difference between AICA and PBSA + AICA because of overlapping between their upper and lower limits. AICA + PBSA has tangible transcendence to other both algorithms in large size problems (see Figure 15).

In order to scrutinise algorithms a sensitivity analysis for ARPD values by considering variations of the number of jobs is shown in Figure 16. According to Figure 16, AICA + PBSA has more correct behaviour and produces

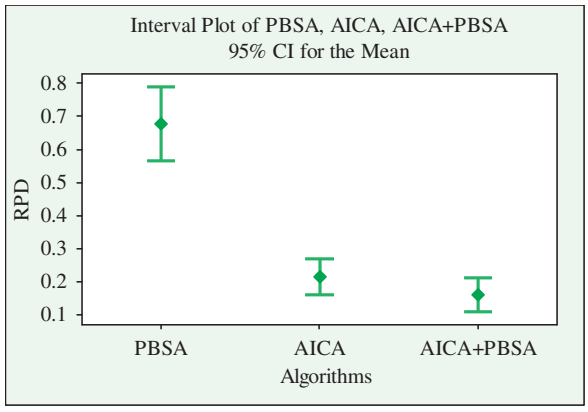

Figure 14. Means plot and LSD intervals (at the 95% confidence level) for algorithms in small scale.

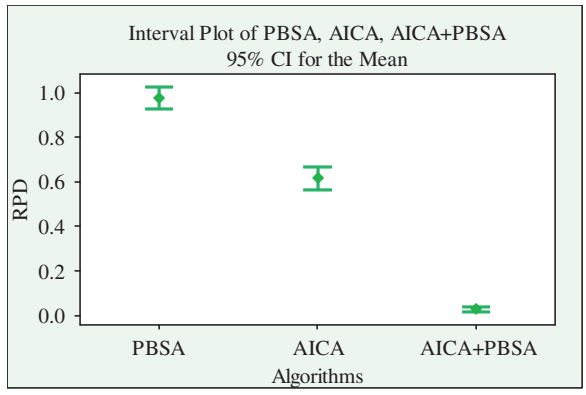

Figure 15. Means plot and LSD intervals (at the 95% confidence level) for algorithms in large scale.

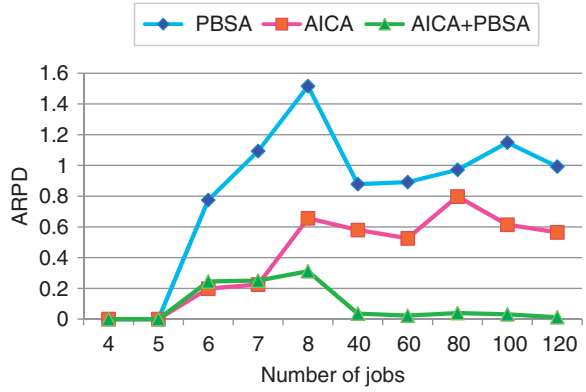

Figure 16. Interaction between performance of algorithms and number of jobs in terms of ARPD.

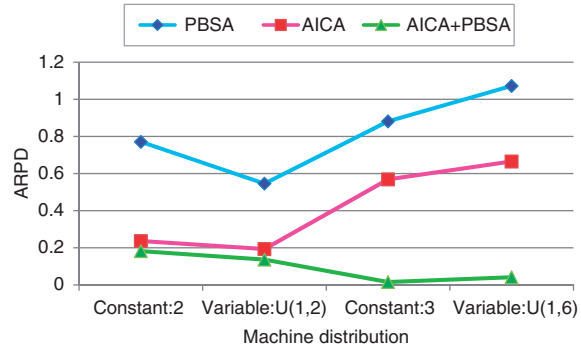

Figure 17. Interaction between performance of algorithms and number of machines in each stage in terms of ARPD.

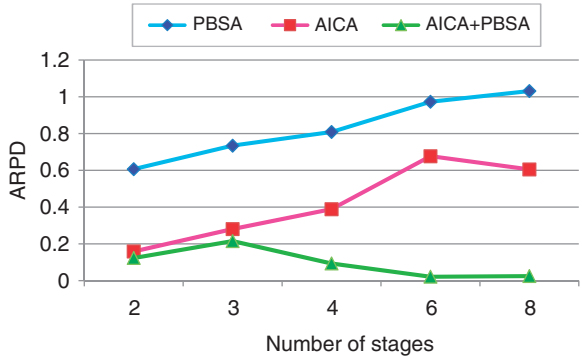

Figure 18. Interaction between performance of algorithms and number of stages in terms of ARPD.

maximum job completion times with minimum values in problems with eight and more jobs. It could be seen for large scale problems, difference between AICA + PBSA is significant but in small scale problems AICA and AICA + PBSA in most cases are near to each other. In addition, interaction between performance of algorithms versus machine distribution and number of stages are demonstrated in Figures 17 and 18, respectively. Overall, it could be concluded that AICA + PBSA is the best algorithm among the other approaches which we applied in this study.

5. Conclusion

This study has considered solving a no-wait flexible flow shop scheduling problem with sequence dependent setup times. In order to find proper schedules that minimise makespan, we studied adaptive imperialist competitive (AICA) and population-based simulated annealing (PBSA) algorithms for mentioned problem. We then suggested a novel hybrid meta-heuristic algorithm which has been formed by combining them (AICA + PBSA). In our suggested method the majority of exploitation procedure is in charge of an adaptive imperialist competitive algorithm and population-based simulated annealing is responsible for exploration procedure. In order to reinforce the algorithm and achieve reliable results, also to prevent carrying out extensive experiments to find optimum parameters of the algorithm, the Taguchi method was applied for these goals. A comprehensive and precise literature review on no-wait flexible flow shop scheduling, with the special attributes proposed here, shows that we have no problem with considering no-wait multi-stage flexible flow shop (more than two-stage) and sequence dependent setup times. To assess the effectiveness of our proposed algorithm some random problems in two scales were generated and results obtained by algorithms were analysed and compared with each other. Simulation results indicated that our suggested novel hybrid algorithm (AICA + PBSA) statistically outperformed the other studied algorithms. Furthermore, sensitive analysis of the performance of the proposed algorithms versus number of jobs demonstrated that AICA + PBSA in all cases was the best solution among the algorithms. As an interesting future research we can add some practical assumptions, including unrelated machine ready time, unrelated parallel machines and machine eligibility for solving the mentioned problem. Furthermore, applying our proposed hybrid metaheuristic in the other combinatorial optimisation problems such as the other production scheduling problem can be considered. Moreover, hybridisation of AICA with some meta-heuristic algorithms such as variable neighbourhood search and hill climbing algorithm could be addressed in future. Also, consideration of PBSA as a local search for colonies could be considered by researchers.

Acknowledgements

We express our warmest thanks to John Middle, Editor-in-Chief, for communicating the paper, and his useful suggestions. We also are grateful for the valuable comments and suggestions from the anonymous reviewers which have enhanced the quality of our paper. The authors would like to acknowledge the financial support of the University of Tehran for this research under grant number 8109003/1/06.

References

Al-Aomar, R., 2006. Incorporating robustness into genetic algorithm search of stochastic simulation outputs. *Simulation Modeling Practice and Theory*, 14 (3), 201–223.

Allahverdi, A., *et al*., 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187 (3), 985–1032.

Allaoui, H. and Artiba, A., 2004. Integrating simulation and optimisation to schedule a hybrid flow shop with maintenance constraints. *Computers and Industrial Engineering*, 47 (4), 431–450.

Behnamian, J., Fatemi Ghomi, S.M.T., and Zandieh, M., 2010. Development of a hybrid metaheuristic to minimise earliness and tardiness in a hybrid flowshop with sequence-dependent setup times. *International Journal of Production Research*, 48 (5), 1415–1438.

Blazewicz, J., *et al*., 2001. *Scheduling computer and manufacturing processes*. Berlin: Springer.

Cochran, W.G. and Cox, G.M., 1992. *Experimental designs*. 2nd ed. USA: Wiley.

Gaafar, L.K. and Masoud, S.A., 2005. Genetic algorithms and simulated annealing for scheduling in agile manufacturing. *International Journal of Production Research*, 43 (14), 3069–3085.

Gholami, M. and Zandieh, M., 2009. An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *International Journal of Production Research*, 47 (24), 6999–7027.

Gilmore, P.C. and Gomory, E., 1964. Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Operation Research*, 12 (5), 655–679.

Graham, R.L., Lawler, E.L., and Rinnooy Kan, A.H.G., 1979. Optimisation and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5 (1), 287–326.

Guinet, A., *et al*., 1996. A computational study of heuristics for two-stage flexible flowshops. *International Journal of Production Research*, 34 (5), 1399–1415.

Huang, R.H., Yang, C.L., and Huang, Y., 2009. No-wait two-stage multiprocessor flow shop scheduling with unit setup. *International Journal of Advanced Manufacturing Technology*, 44 (9–10), 921–927.

Jolai, F., *et al*., 2009. A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. *International Journal of Advanced Manufacturing Technology*, 42 (5), 523–532.

Kadipasaoglu, S.N., 1997. A note on scheduling hybrid flow systems. *International Journal of Production Research*, 35 (5), 1491–1494.

Khalouli, S., Ghedjati, F., and Hamzaoui, A., 2010. A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop. *Engineering Applications of Artificial Intelligence*, 23 (5), 765–771.

Kia, H.R., Davoudpour, H., and Zandieh, M., 2010. Scheduling a dynamic flexible flow line with sequence-dependent setup times: a simulation analysis. *International Journal of Production Research*, 48 (14), 4019–4042.

Kurz, M.E. and Askin, R.G., 2004. Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, 159 (1), 66–82.

Lee, G.C., Kim, Y.D., and Choi, S.W., 2004. Bottleneck-focused scheduling for a hybrid flowshop. *International Journal of Production Research*, 42 (1), 165–181.

Lin, S.W., *et al*., 2009. Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times. *International Journal of Production Research*, 47 (12), 3205–3217.

Liu, Y. and Karimi, I.A., 2008. Scheduling multistage batch plants with parallel units and no interstage storage. *Computers and Chemical Engineering*, 32 (5), 671–693.

Mohammadi, M., Tavakkoli-Moghaddam, R., and Rostami, H., 2011. A multi-objective imperialist competitive algorithm for a capacitated hub covering location problem. *International Journal of Industrial Engineering Computations*, 2 (3), 671–688.

Montgomery, D.C., 2000. *Design and analysis of experiments*. 5th ed. New York: Wiley.

Naderi, B., Fatemi Ghomi, S.M.T., and Aminnayeri, M., 2010. A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. *Applied Soft Computing*, 10 (3), 703–710.

Naderi, B., Zandieh, M., and Shirazi, M.A.H.A., 2009. Modeling and scheduling a case of flexible flowshops: total weighted tardiness minimisation. *Computers & Industrial Engineering*, 57 (4), 1258–1267.

Pan, Q.K., Wang, L., and Qian, B., 2009. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. *Computers & Operations Research*, 36 (8), 2498–2511.

Phadke, M.S., 1989. *Quality engineering using robust design*. Upper Saddle River: Prentice-Hall.

Pinedo, M., 1995. *Scheduling theory, algorithms, and systems*. Englewood Cliffs, USA: Prentice-Hall.

Ponnambalam, S.G., Aravindan, P., and Chandrasekaran, S., 2001. Constructive and improvement flow shop scheduling heuristics: an extensive evaluation. *Production Planning & Control*, 12 (4), 335–344.

Reddi, S.S. and Ramamoorthy, C.V., 1972. On the flow shop sequencing problem with no-wait in process. *Operetion Research Quarterly*, 23 (3), 323–331.

Rock, H., 1984. The three-machine no-wait flow shop problem is NP-complete. *Journal of Associate Computer Machinery*, 31 (2), 336–45.

Ruiz, R., Maroto, C., and Alcaraz, J., 2006. Two new robust genetic algorithms for the flow shop scheduling problem. *Omega*, 34 (5), 461–476.

Ruiz, R. and Stutzle, T., 2008. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187 (3), 1143–1159.

Salvador, M.S., 1973. A solution to a special case of flow shop scheduling problems. *In*: S.E. Elmaghraby, ed. *Symposium of the theory of scheduling and applications*. Berlin/Heidelberg: Springer-Verlag, 83–91.

Seyed-Alagheband, S.A., Fatemi Ghomi, S.M.T., and Zandieh, M., 2011. A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. *International Journal of Production Research*, 49 (3), 805–825.

Shafaei, R., Rabiee, M., and Mirzaeyan, M., 2011. An adaptive neuro fuzzy inference system for makespan estimation in multiprocessor no-wait two stage flow shop. *International Journal of Computer Integrated Manufacturing*, 24 (10), 888–899.

Sriscandarajah, C. and Ladet, P., 1986. Some no-wait shops scheduling problems: complexity aspects. *European Journal of Operational Research*, 24 (3), 424–445.

Vinod, V. and Sridharan, R., 2009. Simulation-based metamodels for scheduling a dynamic job shop with sequence-dependent setup times. *International Journal of Production Research*, 47 (6), 1425–1447.

Wismer, D.A., 1972. Solution of the flow shop sequencing problem with no intermediate queues. *Operation Research*, 20 (3), 689–697.

Xie, J., *et al*., 2004. Minimum deviation algorithm for two-stage no-wait flowshops with parallel machines. *Computers & Mathematics with Applications*, 47 (12), 1857–1863.

Xie, J. and Wang, X., 2005. Complexity and algorithms for two-stage flexible flow shop scheduling with availability constraints. *Computer and Mathematics with Application*, 50 (10–12), 1629–1638.

Yu, J.M., Kim, J.S., and Lee, D.H., 2011. Scheduling algorithms to minimise the total family flow time for job shops with job families. *International Journal of Production Research*, 49 (22), 6885–6903.

Zandieh, M., Fatemi Ghomi, S.M.T., and Moattar Husseini, S.M., 2006. An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Journal of Applied Mathematics Computation*, 180 (1), 111–127.

Zhixin, L. and Jiefang, D., 2003. A heuristic for two-stage no-wait hybrid flowshop scheduling with a single machine in either stage. *Tsinghua Science and Technology*, 8 (1), 43–48.

Zhou, Y., Beizhi, Li., and Yang, J., 2006. Study on job shop scheduling with sequence-dependent setup times using biological immune algorithm. *International Journal of Advanced Manufacturing Technology*, 30 (1–2), 105–111.