Group scheduling in flexible flow shops: a hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism

N. Karimi, M. Zandieh & A.A. Najafi

Faculty of Industrial and Mechanical Engineering, Qazvin Islamic Azad University, Qazvin, Iran
Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C., Tehran, Iran
Faculty of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran

Available online: 29 Oct 2010


To link to this article: http://dx.doi.org/10.1080/00207543.2010.481644
demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
Group scheduling in flexible flow shops: a hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism

N. Karimia, M. Zandiehb* and A.A. Najafić

Faculty of Industrial and Mechanical Engineering, Qazvin Islamic Azad University, Qazvin, Iran; bDepartment of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C., Tehran, Iran; cFaculty of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran

(Received 8 December 2009; final version received 15 March 2010)

This paper applied a novel evolutionary algorithm, imperialist competitive algorithm (ICA), for a group scheduling problem in a hybrid flexible flow shop with sequence-dependent setup times by minimising maximum completion time. This algorithm simulates a social-economical procedure, imperialistic competition. Initial population is generated randomly and evolution is carried out during the algorithm. Firstly individuals, countries, are divided into two categories: imperialists and colonies. Imperialist competition will occur among these empires. This competition will increase some empires authority by ruining a weak empire and dividing its colonies among others. Electromagnetic-like mechanism concepts are employed here to model the influence of the imperialist on their colonies. The algorithm will continue until one imperialist exists and possesses all countries. In order to prevent carrying out extensive experiments to find optimum parameters of the algorithm, we apply the Taguchi approach. The computational results are compared with the outstanding benchmark on the flow shop scheduling problem, random key genetic algorithms (RKGA), and it shows superiority of the ICA.

Keywords: flexible flow shop; group scheduling; imperialist competitive algorithm; electromagnetic-like mechanism; Taguchi method

1. Introduction

The scheduling flow shop problem is a class of scheduling problem which has been widely used in manufacturing systems. Here we consider that our problem is made up of stages with multiple identical machines which should process any group of jobs. Each machine needs a specific setup to process a group. Dividing jobs into groups helps us to reduce setup time, throughput time and work-in-process inventories, and simplifies the flow of parts. These setup times of a machine for each group depend on the last group that was processed on the same machine. A problem with these characteristics is flexible flow shop sequence dependent group scheduling. The objective of this problem is finding a schedule which minimises the maximum completion time, makespan.

Flow shop scheduling has a significant role in a manufacturing system, so it has been used in a wide range of research. Work on sequence dependent set up times in flow shop
scheduling have been done by many researchers. Gupta (1986), Kochhar and Morris (1987), Rios-Mercado and Bard (1999) proposed some heuristic approaches for optimisation of a problem; while Reeves (1995), Ruiz et al. (2005) implemented a metaheuristic algorithm to solve this problem.

Mitrofanov (1966) and Burbidge (1975) firstly presented group scheduling in order to reduce setup times. More information can be found in Allahverdi et al. (1999) and Cheng et al. (2000) who prepared a review on researches that involve set up times. They presented a cell scheduling problem with sequence-dependent setup time. This technique combines the flexibility of job shop with a flow shop production. First, jobs of each group should be sequenced, and then a sequence of groups should be determined. For changing parts from one group to another group, set-up time must be considered. But, for changing between parts in a group, no set-up time should be considered because setup time in each group is negligible, and this is because of similarities between group parts which are an important difference between group scheduling problem and flow shop scheduling problem. There is a wide literature in this field but we focus on literature related to our problem.

Flow shop problems schedule a number of jobs which should be processed on a number of stages to optimise problem’s performance measures. If some of these stages (at least one) consist of multiple identical machines, the problem would change to a hybrid flexible flow shop in which jobs should be processed on at most one machine in each stage. In these kinds of problem, jobs cannot come back to stages they visited before. One of the most important features of the hybrid flexible flow shop is the jobs capability to skip from stages, which means that a job does not need to be processed in that stage. Because of problems with flexibility characteristics there would be more than one machine in some stages. So jobs of different groups may be processed in the same stage at the same time. For groups of jobs only a setup time is required and these setup times are sequence-dependent.

- All jobs are available at zero time.
- Job processing cannot be interrupted.
- Machines are always available, with no breakdowns or scheduled or unscheduled maintenance.
- Infinite buffer exists between stages, before the first and after the last stage.
- Jobs are available for processing at a stage immediately after processing completion at the previous stage.
- Machines in parallel are identical in capability and processing rate.
- Each machine can process only one job at the same time.
- The ready time for each job is larger than 0 and the time it completes processing on the previous stage.
- A job cannot be processed on more than one machine at the same time.

Ham et al. (1985) first presented a method for optimising a two-machine sequence-independent group scheduling problem. A heuristic algorithm for flow shop sequence dependent group scheduling problem was proposed by Schaller et al. (2000). Logendran et al. (2006a) proposed a tabu search (TS) algorithm for minimising makespan in a two machine flow shop sequence dependent group scheduling problem. Vickson and Alfredson (1992), Cetinkaya (1992), Yang and Chern (2000), and some other researchers, also worked on this problem but with two machines.

But Logendran et al. (2005) applied heuristics for solving multi-machine group scheduling in hybrid flexible flow shops. Group scheduling within the context of sequence
dependent setup times in hybrid flexible flow shops is considered in the paper by Logendran et al. (2006b). They presented a heuristic to solve it. Reddy and Narendran (2003) presented heuristics for solving the sequence dependent flow shop group scheduling problem by considering different situations like non-availability of all jobs at the beginning. Leu and Nazemetz (1995) analysed different heuristic on group technology in flow shop manufacturing problems. A heuristic for the case of multiple setups per group is developed and compared with a single setup per group at each stage, and integrated into a genetic algorithm for scheduling non-similar groups on a flow line by Wilson et al. (2004).

A genetic algorithm and a memetic algorithm with local search are proposed and empirically evaluated for scheduling a flow shop manufacturing cell with sequence-dependent family setups by Franca et al. (2005). The same algorithm as Logendran et al. (2006a) was solved by Hendisadeh et al. (2008) through a tabu search-based meta-heuristic for this problem. Previous research by Zandieh and Karimi (2009) has shown the good performance of a multi-population genetic algorithm for solving similar scheduling problems. Karimi et al. (2010) also presented a multi-phase algorithm for the same problem.

This paper develops a novel algorithm for solving flexible flow shop sequence dependent group scheduling problem. Our group scheduling problem can be shown to be NP-hard by its simplifying to the multi-stage flow shop job-scheduling problem which is proved to be NP-hard by Garey et al. (1979). Since this problem is NP-hard, we investigate a metaheuristic algorithm, the imperialist competitive algorithm (ICA), to solve it. This paper describes and models the competition among imperialists and also tries to apply it to a scheduling problem.

The rest of the paper is organised as follows: first, we briefly introduce ICA. We then describe the Taguchi approach in the next section. Experimental evaluation is reported in Section 4. Finally, Section 5 gives some conclusions and future research.

2. Imperialist competitive algorithm

Most of the algorithms in optimisation fields focus on biological evolution of human or other living things, and pay no attention to their social and historical evolution as one of the most complicated and successful evolutionary processes. Atashpaz-Gargari et al. (2008) first proposed imperialist competitive algorithm as a social-political based optimisation method that has a high capability and speed.

As with other evolutionary algorithms, this algorithm starts with an initial population. Individuals which are from the named country will be divided into two categories: imperialists and colonies. Each colony should be in the possession of one of the imperialists according to imperialists’ power. Assimilation strategy and imperialistic competition are the bases of this algorithm. Like the assimilation strategy in the real world that tries to abolish the language and culture of colonies, in the proposed algorithm colonies are affected by their imperialists through a mathematical relationship. Through imperialistic competition, weak imperialists gradually lose their authority and finally will be eliminated. This competition eventually leads the algorithm to have only one imperialist.

In the following we describe all steps of the algorithm according to our problem, in detail.
2.1 Representation

We represent the problem variable as a country which is similar to a chromosome in genetic algorithms (GA). From a historical-cultural point of view, these variables are the countries culture, language, economical rules, etc. In our mentioned problem, each job is assigned a random real number whose integer part is the machine number to which the job is assigned and whose fractional part is used to sort the jobs assigned to each machine. So we use this concept in order to assign a group to machines. Because of considering one setup time for each group, job assignment is the same as group assignment. For the first stage, we should produce a real number for each group between 1 and number of machines plus 1. To define the sequence of jobs in each group random numbers should be produced and the job with the smallest number would be processed first. In other stages, groups sequence would be defined based on their completion time.

We choose the matrix representation form to show our solution, in which rows represent groups and each digit in the rows is the job of that group, except the first row which indicates the group assignment and sequence as described above (see Figure 1).

2.2 Initial population

The ICA starts with a randomly generated set of countries called the initial population. The size of the initial population would be given to the algorithm.

2.3 Generation initial empires

All countries should be evaluated by the problem’s cost function. Our aim in this group scheduling in a hybrid flexible flow shop with sequence-dependent setup times is to minimise the total completion time (i.e. makespan).

\[ C_{\text{max}} = \max \{C_j\}, \quad \text{where } C_j \text{ is completion time of job } j. \]

We can choose a specific number of best countries \((N_{\text{imp}})\) with respect to \(C_{\text{max}}\) as imperialists. Other countries are colonies \((N_{\text{col}})\) that should be assigned to these imperialists. Each imperialist depending upon its power has authority on a number of countries. To estimate the number of countries, first the normalised cost of each imperialist should be calculated:

\[ F_n = \max_i \{|f_i| - f_n\} \quad (1) \]

\(f_n\) is the cost of the \(n\)th imperialist, \(\max_i \{|f_i|\}\) is the maximum cost among imperialists and \(F_n\) is a normalised cost of this imperialist. The relative power of each imperialist can be

<table>
<thead>
<tr>
<th>Groups sequence</th>
<th>2.33</th>
<th>1.67</th>
<th>2.53</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>0.23</td>
<td>0.10</td>
<td>0.50</td>
</tr>
<tr>
<td>Group 2</td>
<td>0.23</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>Group 3</td>
<td>0.70</td>
<td>0.90</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Figure 1. Solution representation.
estimated as follows:

\[ p_n = \left[ \frac{F_n}{\sum_{i=1}^{N_{imp}} F_i} \right] \]  

(2)

Now the number of colonies each imperialist achieves is as follows:

\[ NC_n = \text{round}\{ p_n \times N_{col} \} \]  

(3)

\( NC_n \) is an initial number of colonies of \( n \)th empire. Thus empires can be formed by each imperialist and its \( NC_n \) number of colonies are selected randomly from the initial population.

2.4 Assimilation strategy

As mentioned earlier each imperialist tries to make its colonies approach it in different social-political aspects. Imperialists try hard to perform their assimilation strategy but through this strategy some deviations may occur. These deviations may be caused by the effect of other imperialists’ assimilation strategy or influence of countries of that empire.

In order to model this movement, we propose an electromagnetism-like mechanism (EM) (Birbil and Fang 2003). This method uses an attraction–repulsion mechanism to improve colonies towards the optimum, which is the significant feature of this algorithm. The attraction mechanism leads to movements in better imperialists direction and the repulsion mechanism prevents a move in subordinate imperialists track. This approach is implemented on individuals of each empire, so the movement is toward the best colony (imperialist) but it is not a direct movement because of attraction-repulsion of other imperialists.

The consequent force applied on each colony by its imperialist and all other imperialists would be calculated by considering charge of colonies. Colony \( i \) in empire \( j \) has a charge value \( Q_{ij} \) that can be estimated by the following formulas:

\[ Q_{ij} = \exp \left( -N \frac{f_{ij} - bestf_j}{\sum_{j=1}^{N_{imp}} f_{ij} - bestf_j} \right) \]

(4)

where \( f_{ij} \) is cost of colony \( i \) in empire \( j \) and \( bestf_j \) is cost of the best colony in empire \( j \) (imperialist), so the charge value for the imperialist itself is equal to one that would be shown by \( Q_j \).

The total force exerted on each individual can be defined by the following equation:

\[ F_{ij} = \sum_{j=1}^{N_{imp}} \begin{cases} 
(x_i - x_j) \frac{Q_i Q_j}{\|x_i - x_j\|^2}, & \text{if } f_j < f_i \\
(x_i - x_j) \frac{Q_i Q_j}{\|x_i - x_j\|^2}, & \text{if } f_j > f_i \\
do nothing, & \text{otherwise}
\end{cases} \]

\[ i = 1, \ldots, NC_n. \]

\( x_{ij} \) is the current status of colony \( i \) in empire \( j \), and \( x_j \) is the status of empire \( j \).

The movement is in the direction of the resultant force on each colony. This move would be applied on all colonies but not on imperialists. The length of each movement is
determined by a randomly chosen parameter (α) from a uniform distribution $U(0, 1)$. The imposed force is normalised firstly to avoid producing infeasible solutions.

$$x_{ij}^k = \begin{cases} x_{ij}^k + \alpha \frac{F^k_{ij}}{P_{ij}} (u^k - x_{ij}^k), & \text{if } F^k_{ij} > 0 \\ x_{ij}^k + \alpha \frac{F^k_{ij}}{P_{ij}} (x_{ij}^k - t^k), & \text{if } F^k_{ij} \leq 0 \end{cases}, \quad i = 1, \ldots, NC_n, \ j = 1, \ldots, N_{\text{imp}}, \ k = 1, 2, \ldots, t.$$  

(6)

Also $t^k$ and $u^k$ are the lower and upper bound of each $k$th variable individual, respectively. $t$ is the length of each individual. This strategy will be implemented on all individuals of empires separately.

2.5 Exchanging position of imperialist and a colony

In a movement toward imperialist, a colony may get to a better position than its imperialist. In other words, through this approach some colonies may become so powerful that they can get out of their imperialist authority. Thus the position of imperialist and colony should exchange and the algorithm should continue with the new imperialist.

2.6 Imperialistic competition

As described before, all empires try to increase their power by possessing more colonies. As the algorithm proceeds, one of the more powerful empires would become the most powerful one because of the imperialistic competition. In the algorithm we show this competition by giving the possession of the weakest colony of the weakest empire not to the most powerful one, but to the empire that is more likely to possess them. This likelihood for possession depends on the total cost of an empire.

$$TC_n = \text{Cost(imperialist}_n) + \varepsilon \times \text{mean(Cost(colonies of empire}_n)}$$  

(7)

$TC_n$ is the total cost of $n$th empire and $\varepsilon$ is a number between $(0, 1)$ and near 0. Enlargement in $\varepsilon$ values will increase colonies role in determining the total cost of the empire. Then, normalisation should be done.

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

(8)

$NTC_n$ is the normalised total cost of $n$th empire. Now we can calculate the possession likelihood of empires (see Figure 2).

$$P_{pn} = \frac{NTC_n}{\sum_{i=1}^{N_{\text{imp}}} NTC_i}$$  

(9)

We show these values in a vector:

$$p = [P_{p1}, P_{p2}, P_{p3}, \ldots, P_{pN_{\text{imp}}}]$$  

(10)

Since imperialists can possess the mentioned colonies randomly and also, depending on the probability which is based on possession likelihood of each imperialist, another vector with the same size should be formed with uniformly distributed random numbers, $U(0, 1)$:

$$R = [r_1, r_2, r_3, \ldots, r_{N_{\text{imp}}}]$$  

(11)
By subtracting $R$ from $P$ the possession probability vector can be reached:

$$D = [P - R] = [P_1 - r_1, P_2 - r_2, P_3 - r_3, \ldots, P_{imp} - r_{imp}]$$

(12)

2.7 Eliminating the powerless empires

Imperialistic competition will gradually divide all colonies of a weak empire. This powerless empire will be eliminated. There are different conditions for elimination of an empire. We here consider a number of colonies of an empire as a criterion, if it is equal to zero that empire would be eliminated.

2.8 Stopping criteria

An algorithm would continue until one or all of its criteria are met. We consider a condition with only one empire as a stopping criterion. In this situation all countries are under control of only one empire. Thus, there is no difference between empire and colonies.

3. Taguchi experimental design

Clearly the imperialist competitive algorithm, like most other searching algorithms, is mainly influenced by values of parameters. These parameters can be set manually or by using different setting approaches such as full factorial experiment. This is a comprehensive approach but it would lose its efficiency by increasing the number of parameters (Montgomery 2000), while in Taguchi’s proposed approach, a large number of decision variables would be tuned through a small number of experiments.

This method divides factors into two controllable and noise factors to minimise the effect of noise and determine optimal level of important controllable factors based on the concept of robustness.
Taguchi’s method applies two major tools which are the orthogonal array and the signal-to-noise ratio. An orthogonal array is a fractional factorial matrix, which creates a comparison between levels of any factor or interaction of factors. The array is called orthogonal because all columns can be evaluated independently. Controllable factors will be placed in the inner orthogonal array, and noise factors will be placed in the outer orthogonal array. The measured values that are obtained through the experiments will be transformed into signal-to-noise ratio. In this ratio, desirable values are called signal and undesirable values are stated as noise. Actually this ratio is the amount of variation in the response variable. Signal-to-noise ratio can be categorised in different sets according to its characteristics: continuous or discrete; nominal-is-best, smaller-the-better, or larger-the-better.

Based on our scheduling problem features, we apply the smaller-the-better:

\[ S/N \text{ ratio } = -10 \log_{10}(\text{objective function})^2 \] (13)

### 3.1 Data generation

Required data for the considered problem consist of the processing times, range of the sequence dependent setup times (SDST) and skipping probability.

Processing times are made from a uniform distribution of \( U(5, 75) \); setup times are uniformly generated in the interval \((5, 25)\). Skipping probability is considered in this problem equal to 0.2; uniformly distributed random numbers generated between 0 and 1 for each job. If this number is less than one minus skipping probability value, the job has positive processing times, otherwise processing times are zero.

Because of the flexible characteristic of this problem, we should define the measure of flexibility for it. By multiplying the number of stages and measure of flexibility we can calculate the number of stages which have parallel machines and if this is a real number we should round it to a greater number. We consider the value 2/3 for flexibility. To know which stages have more than one machine, we produce random permutation of the number of stages \((p)\), and then first \(p\) numbers are stages that have a parallel machine. These stages have 2 or 3 machines by probability value equal to 0.5. Numbers of jobs for each group are between (3, 12).

### 3.2 Parameters tuning

Control factors of our algorithm are: population size, number of empires, and the value of \( \varepsilon \) for calculating total cost. Factors with their levels are shown in Table 1. The fittest design for this algorithm is \( L_9 (3^3) \) as shown in Table 2.

As mentioned earlier besides controllable factors, the Taguchi method requires a noise factor. Here we consider a number of iterations as three level noise factors. Each set of experiments shown in Table 2 has three noise factor combinations.

We implement these experiments in Borland C++ and run on a PC with 2.33 GHz Intel Core 2 Duo and 2 GB of RAM memory. Table 3 illustrates obtained data which is transformed into S/N ratio. Table 3 shows the order of factor in minimising makespan in \textit{Rank row}. Figure 3 plots the S/N value against each control factor.
Because of the smaller-the-better characteristic of S/N ratio, the optimum value of each parameter can easily be obtained using Figure 3 as follows: population size, 50; number of empire in population, 8; epsilon value, 0.1.

As mentioned earlier we adapted the RKGA, which is a well known benchmark in flow shop scheduling problems, to measure the efficiency, effectiveness and viability of ICA and compared them. For a better comparison, we also tuned the GA parameters through Taguchi method. These values are set as follows: population size, 150; crossover probability, 0.75; mutation probability, 0.15.

4. Experimental evaluation

In this section, we intend to evaluate the effectiveness and performance of the proposed algorithm through computational experiments. With respect to the concept of the
proposed algorithm, the stopping criterion is reaching a condition in which only one empire can exist. To investigate the algorithm performance, a comparison with RKGA is carried out. We examine the behaviour of the algorithm under different numbers of groups and stages.

We make use of some performance measures such as \((CV)^2\), GAP and improvement to compare algorithms.

We use the \((CV)^2\), to evaluate the algorithms convergence:

\[
(CV)^2 = \frac{\sum_{i=1}^{R} (C_{\text{max}}(i) - \bar{C}_{\text{max}})^2}{(n-1)(\bar{C}_{\text{max}})^2},
\]

where \(R\) is number of replications for each instance, \(C_{\text{max}}(i)\) is the \(i\)th individual makespan value and \(\bar{C}_{\text{max}}\) is the average value of makespan in replications.

The gap is percentage deviation of the feasible schedule from the corresponding lower bound. If \(\bar{C}_{\text{max}}\) is the best makespan in the replication, GAP can be defined as:

\[
GAP = \frac{(C_{\text{max}}(i) - \bar{C}_{\text{max}})}{\bar{C}_{\text{max}}},
\]

We evaluate the proposed algorithm by relative percentage improvement with respect to the mentioned RKGA. This metric can be calculated as below (Vilcot and Billaut 2008):

\[
\text{improvement} = \frac{C_{\text{max}}(GA) - C_{\text{max}}(ICA)}{C_{\text{max}}(GA)} \times 100,
\]

Table 4 addresses the average of convergence speed of the algorithm in different sizes according to number of stages \((n)\) and groups \((m)\). It can be easily seen that as the number of groups increases in the same number of stage, number of iterations for the algorithm termination will decrease.

To analyse the interaction between quality of the algorithms and different values of number of groups and stages, we present Table 5. It reports the results of experiments.

Figure 3. The S/N ratio plot.
In Table 5, the average of $(CV)^2$, GAP and improvement for different combinations of $s$ and $g$ through replication of experiments are represented.

$(CV)^2$ column proves that there is low variation in results which leads to high convergence of the proposed algorithm. Table 5 gives the differences between the best and the worst values obtained over 10 runs in the GAP column.

Regarding $(CV)^2$, GAP and improvement values in Table 5 we can conclude that ICA is superior to RKGA in most instances and also in average values of each index. As could be seen there is a clear trend, increase in number of groups in the same number of stages results in a better performance of ICA.

## 5. Conclusion and future work

This paper dealt with a new combinatorial algorithm, imperialist competitive algorithm (ICA), to solve the problem of group scheduling in a hybrid flexible flow shop with sequence-dependent setup times to minimise makespan. The proposed algorithm is inspired from a socio political process, the competition among imperialists and colonies.
It formulated this procedure using discrete variables and utilising the electromagnetic-like mechanism. In order to reinforce the algorithm, we bring robustness into it by making parameters of the algorithm calibrated. Calibration is carried out through extensive experiments by means of Taguchi method. It is also tested against an outstanding benchmark in flow shop scheduling problems, RKGA, using some performance measures. Results verified that the proposed algorithm dominated RKGA in terms of solutions quality and performance measures.

Furthermore, for future study the ICA can be studied in other scheduling problems or in other optimisation problems. It is worth considering other objectives of this problem. Some other realistic assumptions for research, such as transportation times, unrelated machines and machine availability constraints, would certainly be worthy of consideration.

References


