A Bi-objective Simulation-optimization Approach for Solving a No-wait two Stages Flexible Flow Shop Scheduling Problem with Rework Ability

Mohammad Rahmanidoust¹, Jianguo Zheng¹

¹Glorious Sun School of Business and Management, DongHua University, Shanghai, China

Correspondence: Mohammad Rahmanidoust, Glorious Sun School of Business and Management, Donghua University, Shanghai, China.

Received: September 26, 2017    Accepted: November 13, 2017    Online Published: November 16, 2017

doi:10.5539/ibr.v10n12p197    URL: https://doi.org/10.5539/ibr.v10n12p197

Abstract

The paper suggests a new rule; called no-wait process. The rule has two stages, and is a flexible flow shop scheduling. The process is the subject to maximize tardiness while minimizing the makespan. This hybrid flow shop problem is known to be NP-hard. Therefore, we come to first, Non-dominated Sorting Genetic Algorithm (NSGA-II), then, Multi-Objective Imperialist Competitive Algorithm (MOICA) and finally, Pareto Archive Evolutionary Strategy (PAES) as three multi-objective Pareto based metaheuristic optimization methods. They are developed to solve the problem to approximately figure out optimal Pareto front. The method is investigated in several problems that differed in size and terms of relative percentage deviation of performance metrics. The conclusion, developed by this method is the most efficient and practicable algorithm at the end.

Keywords: multi-objective optimization, no wait, flexible flowshop, NSGA-II, MOICA, PAES

1. Introduction

Scheduling is arranging and planning sequence of event to complete the work. The goal is to utilize the resources optimally while reaching targets. This field has been attracted by many scholars during recent years (Mosheiov and Sidney 2010, Vallada and Ruiz 2011, Shafaei et al. 2011, Rabiee et al. 2012, Jolai et al. 2013, Ulrich 2013, Jolai et al. 2014, Yang et al. 2014, Tayebi Araghi et al. 2014, Xu et al. 2015, Rabiee et al. 2016, Nesello et al. 2017). The flow shop problem is an important subject in scheduling. One of the reasons is that most of the manufacturing systems follow batch shops, flow shop or semi flow shop routings (Baker 1974, Johnson 1954, Lin et al. 2006). Flexible flow shop or hybrid flow shop, is the one of the most important classes of scheduling which also defined as a flow shop with parallel machines and flexible flow line. For a literature review in this area, the readers are referred to those of (Richard and Zhang 1999, Ruiz and Vazquez-Rodriguez 2010, Ribas et al. 2010)

In a no-wait flow shop, the jobs are processed from one machine to the next one without waiting time (Huang et al. 2009). Suppose there are some sequences of jobs, some specified procedures in each that are processed by machine in disciplinary order. No pause or interruption is supposed to occur in the line. In other word, when the process starts, there is no stop for object before or after each machine. No repetition is allowed, one job at a given time by a machine. Therefore, when needed, the start of a job on the first machine must be delayed in order to meet the no-wait requirement (Tasgetiren et al. 2007). Nagano et al. (2013) examines the m machine no-wait flow shop problem with setup times of a job separated from its processing time. The performance measure considered is the makespan. The hybrid metaheuristic Evolutionary Cluster Search (ECS_NSIL) is employed to solve this scheduling problem. Nagano et al. (2015) addressed the problem of scheduling jobs in a no-wait flow shop with sequence-dependent setup times with the objective of minimizing the total flow time. As this problem is well-known for being NP-hard, they presented a new constructive heuristic, named QUARTS, in order to obtain good approximate solutions in a short CPU time. Samarghandi and ElMekkawy (2012) studied the problem of no-wait flow shop and proposed two frameworks based on genetic algorithm and particle swarm optimization to deal with the problem. Samarghandi and ElMekkawy (2014) proposed PSO algorithm to solve the problem of scheduling a no-wait flow-shop system with sequence-dependent set-up times. The application of this problem can be found in industries such as chemical industry, steel production, just-in-time manufacturing process,
Most of literatures about no wait hybrid flow shop scheduling problem have mainly focused on a single objective. For example, Liu et al. (2003) suggests an algorithm known as Least Deviation (LD) in which the focus is on only one machine in each station. The key performance indicator in this point of view is the makespan. The performance is thus high using this algorithm. Also the algorithm is easier in computation and implementation. Having created such values, this is considered favourable.

Xie et al. (2004) suggests a new heuristic algorithm named Minimum Deviation Algorithm (MDA) to minimize makespan in a similar method. MDA also performs better than partition method, partition method with LPT, Johnson’s and modified Johnson’s algorithms. Huang et al. (2009) considered a no-wait two stage flexible flow shop with setup times and with minimum total completion time key performance indicator. The author represents an enhanced programming as well as an Ant Colony approach. The solution was satisfactory with the approach and the results were efficient.

To best use of resources, Jolai et al. (2009) introduced no-wait flexible flow line scheduling problem with time windows and job rejection which is, in turn, an extension of production and delivery scheduling. He presented a similar method known as integer-linear programming model and genetic algorithm process as well.

In comparison with LINGO, studies show that the GA is a better solution in a computational time. Jolai et al. (2012) introduced a new hybrid algorithm with sequence-dependent setup times to minimize the total completion time. They suggest three algorithms. One, Population Based Simulated Annealing (PBSA), second, Adapted Imperialist Competitive Algorithm (AICA) and finally, hybridization of Adapted Imperialist Competitive and Population Based Simulated annealing (AICA+PBSA) for the problem. All the studies support the hybrid algorithm against the others which are applied in literature for related production scheduling problem. Rabiee et al. (2014) proposed the problem with respect to unrelated parallel machines, sequence-dependent setup times, probable reworks and different ready times to actualize the problem. What they proposed is based on imperialist competitive algorithm (ICA), simulated annealing (SA), variable neighborhood search (VNS) and genetic algorithm (GA) to solve the problem. The result revealed the advantages of our algorithm. To reduce makespan, Ramezani et al. (2013) suggested no-wait scheduling problem focused on set up time which is anticipatory and also sequence dependent in a flexible flow shop environment with two sets of same machines in parallel. They introduced it as a novel method since it was NP-hard. Their meta-heuristic method was about invasive weed optimization, adjustable neighborhood exploration and simulated strengthening to attack of the problem. The result showed, hybrid-metaheuristic outperformed in comparison with singular ones.

As said before, most of studies concentrated on single objective problem. But in reality, there is no single objective at all and we need to consider a batch of goals at once. However, there are also some studies working on multi-objective in no wait flow shop scheduling problem. Allahverdi and Aldowisan (2004) suggested a method in no wait flow shop scheduling problem in with sum of makespan is important and also the maximum delay as a measure. Their methods for comparison were hybrid simulated annealing and a hybrid genetic heuristics. Also they suggested a dominance relation (DR) and a branch-and-bound algorithm. Herein also, after computation, the heuristic method performs higher and better in comparison with existing heuristics if the makespan and maximum lateness is considered significant. Also the dominance relation and branch and bound algorithm were totally effective. Aiming to minimize average of tardiness and time, Rahimi-Vahed et al. (2008) offered a bi-criteria no-wait flow shop scheduling problem. What they suggest is a new method named multi-objective scatter search as a metaheuristic algorithm for finding near optimal Pareto frontier. They were looking for effectiveness of this approach by solving some experimental problems in comparison with SPEA-II. Here also the better performance was appeared in multi-objective scatter search.

Multi-objective immune algorithm also proposed by Tavakkoli-Moghaddam et al. (2007) to minimize two goals as weighted average completion time and weighted average tardiness for a no-wait flow shop scheduling problem. He compared the algorithm with a conventional multi-objective genetic algorithm, i.e., SPEA-II. The generic algorithm won the game especially for significant and big problems. Pan et al. (2009) offered another algorithm called discrete differential evolution (DDE) for solving the no-wait flow shop scheduling problems with makespan and maximum tardiness measures. The results based on the famous benchmarks and statistical performance comparisons showed that DDE algorithm is much more efficient in comparison to the hybrid differential evolution (HDE) algorithm proposed by Qian et al. (2009). For the same objective, i.e. to minimize makespan and tardiness, Khalili (2012) proposed a multi-objective no-wait hybrid flow shop scheduling problem.
and suggested a novel Multi-Objective Electromagnetism Algorithm (MOEA) to solve the problem. He formulated the problem with mixed integer-linear programming models and proposed an effective Multi-Objective Electromagnetism Algorithm (MOEA) to reach the goal.

As far as multi-objective approach is considered, there are a few studies about no wait flexible flow shop problem. The author presented three multi-objective based algorithms to discover a no wait two stage flexible flow shop scheduling problem with a number of machines in each stage. The goals were minimizing makespan (i.e. $C_{\text{max}}$) and maximum tardiness (i.e. $T_{\text{max}}$).

In the next section, section 2, the multi objective terms are discussed. Then the author wrote about the bi-criteria no wait two stages flexible flow shop. Then in section 3, the multi-objective optimization search techniques are considered. Investigating the proficiency of suggested multi-objective metaheuristic approaches is presented next section. And at last, the outcome of the research is induced and directions for further researches are depicted in section 5.

2. Multi-objective No-wait Two-stage Flexible Flow Shop Optimization

Now we turn to the problem statement. That is the concept of the multi objective optimization and also the structure of the problem.

2.1 Multi-objective Optimization

A multi-objective optimization problem formula is as below:

$$\min_{x \in X^n} f(x) = \{f_1(x), f_2(x), ..., f_M(x)\}$$ (1)

$$s.t. \quad g(x) \leq 0, \quad h(x) = 0$$ (2)

Wherein $g(x) \leq 0$, $h(x) = 0$ shows the possible solution in $n$ dimensional search space and $f(x)$ is a $M$ dimensional vector of objective values. Map between decision variables of $x \in X^n$ and objective space of $f \in F^M$ is determined by objective functions. In reality, the aim of a multi-objective optimization is to figure out the entire non-dominated solutions of the problem (any solution that is not able to develop an objective function with no effect on other objective).

If any of below conditions are met in an optimization problem with minimum objectives, solutions $x_1$ dominates solution $x_2$.

1. For every single objective $f(x_i) \leq f(x_j)$.
2. At least in one objective $f(x_i)$ has a lower value compared with $f(x_j)$.

Having defined the dominant solutions, the optimal solution of a multi-objective optimization problem is defined as set of non-dominated solutions known as Pareto-optimal set which forms the Pareto front (CoelloCoello et al. 2002).

2.2 The Statement of Bi Criteria No-wait two Stage Flexible Flow Shop Problem

The no-wait two stage flexible flow shop scheduling problem is shown below: Given the processing time $P_{ij}$ of job $j$ on stage $i$($i=1, 2$), each of $n$ job will be sequentially processed in stage $1, 2$ respectively. At each stage there are $m_i$ machines. Also at a given time, each machine can process maximum a single job. Likewise, each action needs to be processed on one machine. Once the order of the action at the first stage is cleared the similar order is done for the second stage. To meet the no-wait boundaries, the end time of a job on a machine must be similar to the start time of the job on the next machine. This way, there is no elapse time in the entire operation. The aim here is, reduction of makespan ($C_{\text{max}}$). The matching fitness function is considered at below:

$$C_j = \text{Completion time of job } j$$ (3)

$$\text{Makespan} = C_{\text{max}} = \max(C_j)$$ (4)

$$\min \ z_1 = C_{\text{max}}$$ (5)

Next aim is reduction of maximum tardiness which is calculated below:

$$\min \ z_2 = T_{\text{max}}$$ (6)

Wherein $T_i$ is the tardiness of job $i$, equal to $\max(0, C_i - d_i)$, and $d_i$ is the due date of job $i$. 
3. Possible Approaches to Find Solution

There are many classical approaches to solve multi-objective problems among them includes embracing goal programming, integer programming, e-constraint method and weighted sum method. The main features of the classical methods can be described as follows (Deb 2001, CoelloCoello et al. 2002):

1. Changing the problem from multi-objective to single objective
2. Experimentally, the methods might be applied randomly to find out the best solution
3. Each typical method includes some user-defined parameters that are not easy to set in an arbitrary problem. Some meta-heuristics have been developed to eliminate such deficiencies. They are genetic algorithms and evolutionary computation. The ability to figure out a reasonable estimate of Pareto frontier in one operation and good computational time, is one of the benefit of this method.

This paper offers three multi-objective metaheuristic methods to solve the problem; NSGA-II (Non-dominated Sorting Genetic Algorithm), MOICA (Multi-Objective Imperialist Competitive Algorithm) and PAES (Pareto Archive Evolutionary Strategy). They are to examine the output of the algorithms in solving the no-wait two stages flexible flow shop problem.

Metaheuristic algorithms are generally, based on a searching system which is random. Here, the problem altered from a phenotype into a genotype that is informally called chromosome. To discover the best solution, it uses intensification as well as diversification where the first intends to use local search area and the second explores the optimal solution globally. The chromosome, fitness evaluation, related operators and structures of applied system are described elaborated respectively:

3.1 Solution Representation and Fitness Evaluation

Some random values that are equals to the number of jobs in length is generated from 0 to 1. This is to show the chromosome. The jobs then are tossed by finding the increasing order of values in vector. See figure 1.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>0.45</th>
<th>0.63</th>
<th>0.13</th>
<th>0.33</th>
<th>0.77</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job sequence</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 1. An example of chromosome representation and its job sequence

After calculation of jobs, procedure of machine assignments is done using a heuristic method. It means in each stage, to assign a job to the machines, the earliest available time machine is chosen and the job with the highest importance is assigned to that machine. If there are two important jobs at the same time, one is chosen randomly. The pseudo code of heuristic procedure is shown in Figure 2.

3.1 Genetic Operators

3.1.1 Crossover

The process in which tow chromosomes are prevented from coupling making progeny is called crossover. The process aims to find a better solution by mixing the chromosomes. The study uses a uniform crossover that first creates a random binary mask with the similar extend as the chromosomes and then substitutes relative gene material of parent chromosome based on created binary mask. This crossover results in a good exploitation of solution space (Syswerda 1989).

3.1.2 Mutation

After crossover, we perform an exchange mutation. For exchange mutation, two different arbitrary genes of the parent chromosome choose and swap the allele values. (Eiben and Smith. 2003).
3.3. Multi-objective Algorithms

3.3.1 Non-dominated Sorting Genetic Algorithm (NSGA-II)

Probability of gaining Pareto-optimal solution using GAs are high, since they work with body of points. This makes it a strong tool for MOOPs too. The Non-dominated Sorting Genetic Algorithm (NSGA-II) is a well-known and extensively used algorithm based on its predecessor NSGA and proposed by Deb et al (2002). Essentially, NSGA-II differs from non-dominated sorting Genetic Algorithm (NSGA) implementation in a number of ways. First, NSGA-II uses an elite-preserving mechanism, thereby assuring the preservation of previously found good solutions. Second, NSGA-II uses a fast non-dominated sorting procedure. Third, NSGA-II does not require any tunable parameter thereby making the algorithm independent of the user (Sivakumar et al, 2011). NSGA-II is a fast and very efficient Multi-objective evolutionary algorithm (MOEA), which incorporates the features of an elitist archive and a rule for adaptation assignment that takes into account both the rank and the distance of each solution regarding others. Salazar and Kishor have applied and compared the efficiency of NSGA-II with existing methods for reliability optimization problems (Kishor et al, 2008).
NSGA-II is an elitist multi-objective evolutionary algorithm which carries out an approximation of the Pareto front, based on the non-dominance concept. For achieving different Pareto fronts, a ranking procedure is performed at each generation. Also, this algorithm takes advantage of an operator called crowding operator for its diversification. NSGA-II starts from a randomly generated population of chromosomes (solutions), P_0 of size N. The population is sorted based on non-dominance. Each solution is assigned a fitness (or rank) equal to its non-dominance level (1 is the best level) the minimization of the fitness being assumed. A children population (Q_0) of size N is then created by applying the genetic operators: binary tournament selection, recombination, and mutation (Furtuna, et al, 2011). NSGA-II has been used in several prior studies like: Minella et al. (2008), Behnamian et al (2009), Zandieh and Karimi (20111) and Rabiee et al (2012), Asefi et al. (2014). The crossover and mutation operators in this algorithm are as mention in NSGAI. The framework of the proposed NSGAII is generally illustrated in the following pseudo code (Figure 3).

```
Begin
    Input : N, P, P_n, Max_Gen;
    Generate Init_Pop(P);
    Repair Init_Pop(P);
    Evaluate fitness values of the Init_Pop;
    Assign rank base on pareto dominance sort;
    for i = 1 to Max_Gen do
        for j = 1 to round([P_n \times N] / 2]
            Select two individuals: (X_1, X_2);
            Select one scenario for crossover operation;
            perform one point crossover: (X_1, X_2) \rightarrow (X_1^\prime, X_2^\prime);
        endfor
        for j = 1 to round([P_n \times N] / 2]
            Select an individual: X;
            Select one scenario for mutation/neighborhood operation;
            Select one case between (swap, reversion, insertion);
            perform mutation: X \rightarrow X^\prime;
        endfor
    Repair New_Pop(P);
    Combine offsprings and parents \{P \cup Q\}
    Assign rank based on pareto dominance sorting algorithm;
    Calculate the crowded distance of individuals in each front;
    Select the best N individual base on rank and crowded distance;
end for
Output : Extract the best pareto front;
End
```

Figure 3. Proposed NSGA-II algorithm in pseudo code

3.3.2 PAES Algorithm

The study suggests a simple multi-purpose metaheuristic algorithm called PAES. Offered by Knowles and Corne (1999, 2000), the algorithm applies a local search development strategy to a non-dominated solution in a pool of solutions that already applied. Hereunder, the most alternates of PAES which is (1+1) development strategy is discussed.

A solution is randomly generated, evaluated and saved in archived. Here is when the algorithm starts. Procedure at iteration t is continued with creating a new solution by transform current solution and compare it to current solution for dominance. The one with more dominance is accepted. If both solutions have same dominance priority, the new one is compared to archived solution which is archived. The accepted solution in the archive
added there and removes the rejected one. Otherwise the new solution replaced by one of the archive member. At last, if there is no dominate member in the archive and archive has still vacancy, the new solution is added to archive. If there is no space in solution pool, the solution in the busiest area is removed and the new one is added. In when considering intricacy of problem to find non-dominated solutions, we consider a limitless archive scope to maintain non-dominated solutions. This is to obtain more Pareto solutions. The structure of proposed PAES is shown in Figure 4.

\begin{algorithm}
\textbf{Begin}
\textbf{Input}: \textit{MaxIteration};
Generate \textit{Init}_\textit{Sol} and set it as \textit{Current}_\textit{Sol};
Evaluate fitness value of the \textit{Current}_\textit{Sol};
Add \textit{Current}_\textit{Sol} to archive;
\textbf{for} \textit{i} = 1 to \textit{MaxIteration} \textbf{do}
\hspace{1em} Generate \textit{New}_\textit{Sol} by mutation of \textit{Current}_\textit{Sol};
\hspace{1em} Evaluate fitness value of the \textit{New}_\textit{Sol};
\hspace{1em} \textbf{if} \textit{New}_\textit{Sol} dominates \textit{Current}_\textit{Sol}
\hspace{2em} Set \textit{New}_\textit{Sol} as \textit{Current}_\textit{Sol};
\hspace{2em} Update \_\textit{Archive};
\hspace{1em} \textbf{elseif} \textit{Current}_\textit{Sol} dominates \textit{New}_\textit{Sol}
\hspace{2em} Discard \textit{New}_\textit{Sol};
\hspace{1em} \textbf{else} \textit{Current}_\textit{Sol} and \textit{New}_\textit{Sol} don't dominate each other;
\hspace{2em} Update \_\textit{Archive} using \textit{New}_\textit{Sol};
\hspace{2em} Randomly select next \textit{Current}_\textit{Sol} between \textit{New}_\textit{sol} and \textit{Current}_\textit{Sol};
\hspace{1em} \textbf{end if}
\textbf{end for}
\textbf{Output}: Extract nondominated solution as pareto front;
\textbf{end}
\end{algorithm}

Figure 4. PAES algorithm

3.3.3 Multi-objective Imperialist Competitive Algorithm (MOICA)

3.3.3.1 Creating Primary Empires

Every single solution in the imperialist competitive algorithm simulates as an array. The arrays include different values that need to be adjusted. What is called chromosome in GA terminology, is named country here. In an N-dimensional optimization problem, a country is a 1×N array. This array is defined by: 

\[ \text{\textit{country}} = [p_1, p_2, p_3, ..., p_N], \]

where \( p_i \) is the variable to be optimized. Each variable in a country denotes a socio-political characteristic of a country. From this point of view, the algorithm searches for the best country that is the country with the best combination of socio-political characteristics, such as culture, language and economic policy (Atashpaz-Gargari and Lucas 2007). After country development, a non-dominance technique and a crowding distance are used to shape the fronts and rank member of each front, respectively. At that point, the members of front one are archived. Non-dominance technique and crowding distance described as below:

1: Non-dominance technique: imagine that there are \( r \) objective functions. When the following conditions are satisfied, the solution \( x_1 \) dominates another solution \( x_2 \). If \( x_1 \) and \( x_2 \) do not dominate each other, they are placed in the same front.

1. For all the objective functions, solution \( x_1 \) is not poorer than another solution \( x_2 \).
2. For at least one of the \( r \) objective functions \( x_1 \) is exactly better than \( x_2 \).

Solutions that are not dominated by others, constitute in front number 1. Meanwhile, the solutions that are only dominated by solutions in front number 1, organize front number 2. The same order applies to crate the other fronts which is shown in Figure 5.
2. Crowding distance: this is a tool to show the quantity of solutions in each step. See below figure. This is an estimate of the solution mass around a given solution.

![Crowding distance diagram]

**Figure 6. Crowding distance**

The crowding distance measure which is used in MOICA is shown in equation (7). The solutions having a lower value of the crowding distance are prioritized over solutions with upper value of the crowding distance.

\[
CD_i = \sum_{k=1}^{n} \frac{f_{k,\max}^P - f_{k,\min}^P}{f_{k,\max}^P - f_{k,\min}^P}
\]

(7)

Where:

- the number of objective functions, \( f_{k,i+1}^P \) is the k-th objective function of the (i+1)-th solution
- \( f_{k,i}^P \) is the k-th objective function of the (i-1)-th solution after sorting the population according to crowding distance of the k-th objective function
- \( f_{k,\text{max}}^P \) and \( f_{k,\text{min}}^P \) are the maximum and minimum value of objective function k, respectively.

Next, the prioritized solution is selected in the mass in terms of non-dominance and crowding distance. The selected solution is imperialists and the rests are colonies.

In order to compute the cost of prioritized solution (the imperialist), each target function is calculated. After that each target function is calculated:
Where:

- $\text{cost}_{i,n}$ is the normalized value of objective function $i$ for imperialist $n$
- $f_{i,n}^p$ is the value of the objective function $i$ for imperialist $n$
- $f_{i,n}^{p,\text{best}}, f_{i,n}^{p,\text{max}}, f_{i,n}^{p,\text{min}}$ are the best, maximum and minimum values of objective function $i$ in each iteration, respectively.

At last, the total value of each imperialist is calculated through:

$$Total\ Cost_n = \sum_{i=1}^{r} \text{cost}_{i,n}$$

Where:

- $r$ is the quantity of target function

After calculation the cost, the strength of each imperialist is obtained as well as the colonies distributed among the imperialist according to power of each imperialist country.

$$P_n = \left| \frac{Total\ Cost_n}{\sum_{i=1}^{N_c} Total\ Cost_n} \right|$$

At this point, the primary quantity of colonies is calculated as below:

$$NC_n = \text{round}\{P_n \cdot N_{\text{col}}\}$$

Where:

- $NC_n$ is the primary quantity of colonies of the $n$-th imperialist
- $N_{\text{col}}$ is the number of all colonies

$NC_n$ colonies are selected randomly and assigned to one imperialist. Apparently, the greater quantity of colonies, the stronger imperialist and the less quantity of colonies, the poorer imperialist.

3.3.3.2 Total Strength of an Empire

Imperialist country has the major impact on the total strength of an empire. But the strength of its colonies does not have such effect. Therefore, the equation of the total power of an empire is shown below. (Karimi et al. 2010, Shokrollahpour et al. 2011).

$$TP_{\text{Emp}_n} = (Total\ Cost(\text{imperialist}_n) + \xi \text{mean}(Total\ Cost(\text{colonies\ of\ empire}_n)))(1 - QE_n)$$

Where:

- $TP_{\text{Emp}_n}$ is the total power of the $n$th empire
- $\xi$ is a positive number which is considered to be less than 1
- Total cost of imperialists and colonies are calculated by Eq.8 and Eq.9.
- $QE_n$ is the quality of empire $n$th

$QE_n$ is determined as below:

First: all of the imperialists and colonies are accumulated and then the non-dominated solutions are chosen. The percentage of the non-dominated solution belonging to each empire is calculated as $QE_n$. 

205
Remember that, the total strength of the empire to be determined by just the imperialist when the value of $\zeta$ is small and increasing it will increase the role of the colonies in determining the total power of an empire.

3.3.3.3 Moving the Colonies of an Empire toward the Imperialist (Assimilating)

Having distributed colonies among imperialists, the imperialist and relevant colonies go together. Figure 7 shows the movement. There, $d$ is the distance between imperialist and colony. $X$ is a random variable with a uniform (or any proper) distribution between 0 and $\beta \times d$ and $\beta$ is a number greater than 1. Direction of the movement is shown by $\theta$, which is a uniform distribution between $-\gamma$ and $\gamma$.

![Figure 7. Moving colonies toward the imperialist with a random angle](image)

3.3.3.4 Information Sharing between Colonies

To improve their position, the colonies share their information. To do so, in this part, one of these operators including one-point, two-point and continuous uniform crossover shown in Figure 8 are selected randomly. The mass part that are sharing information are shown by $Pc$. Those colonies with stronger position have more opportunity to share their information since the selection here is by a competition that is describe below:

![Figure 8. Operators for information sharing between colonies](image)

How come a selection is done?

A binary competition process is used to find out the best solution for both crossover and mutation operators in this way:

Step 1: choose two solutions with the same size

Step 2: the lowest front number needs to be chosen if both populations are from different fronts.

If both of them are in same front, choose the solution with highest crowding distance.

3.3.3.5 Revolution

Some revolutions have been done during last decades on the colonies. To do so, one of the famous policies, swap, reversion and insertion, exchange and local search is randomly chosen. The operators’ structures are elaborated
as follows: (Pris the revolution rate)

- **Swap:** a colony’s initial numbers are chosen randomly (numbers 1.46 & 2.27 in Figure 9) and their substitute each other.

![Figure 9. Swap operator](image)

- **Reversion:** in the current strategy, in addition to conducting substitution, the number that is placed between the substituted numbers are also changed.

![Figure 10. Reversion](image)

- **Insertion:** for the insertion strategy, like substitution, two numbers of a colony is chosen by chance (Numbers 1.46 & 2.27 in Figure 11). After that, the next number is places approximate to the number in the first position. The other number is moved right side consequently.

![Figure 11. Insertion](image)

- **Perturbation:** in perturbation strategy, one number is chosen by chance, and another number is generated by chance. Then these two are substituted.

![Figure 12. Perturbation](image)

**Local search:** the strategy refers to randomly selection of the iteration. After that all of the two-point exchanges are investigated.

### 3.3.3.6 Improve Imperialist

Using the rules describe earlier, the step involves in producing some neighborhoods around each imperialist considering its total power. The less imperialist power, the more neighborhood generation. Similarly, the more
imperialist power, the less neighborhood generation. In other word, the quantity of neighborhoods directly depends on the power of the given imperialist. This is a leaner formula ranged between \( N_{\text{min}} \) and \( N_{\text{max}} \) that is illustrated below:

\[
N_{\text{e}} = \text{floor}(N_{\text{e}_{\text{min}}} + \frac{(N_{\text{e}_{\text{max}}} - N_{\text{e}_{\text{min}}})(TP \ Emp_{\text{best}} - TP \ Emp_{\text{worst}})}{(TP \ Emp_{\text{worst}} - TP \ Emp_{\text{best}})})
\]

(13)

Where:
- \( TP \ Emp_{\text{best}} \) is the value of total power the most power empire
- \( TP \ Emp_{\text{worst}} \) is the value of total power of the weakest power empire.

How the number of neighborhood and power are related are shown below:

Figure 13. The relationship between power of each imperialist and number of neighborhoods

3.3.3.7 Colonies Updated

The primary mass of colonies, assimilating, information sharing among colonies, revolution and improve imperialist are combined all together in each decades to shape the empire that is called combined mass. Then, based on combined mass, the archive is updated. Then, for any of the imperialists, the best colony is chosen according to non-domination sorting and crowding distance by size of mass of colonies for a given empire \( NC(i) \).

3.3.3.8 Archive Adaption

For the combined mass, the classification is done using non-dominated and crowding distance. To archive, the front one members are chosen. At last, these members are retained and after classifying the solutions in archive, the other members are removed. Meanwhile, the size of archive equals \( n \ Archive \).

3.3.3.9 Exchanging Positions of the Imperialist and a Colony

In this step, the total cost of each imperialist is updated. Next, the best imperialist and colony are combined. Then, this mass is arranged by the non-dominated sorting and crowding distance. At last, the best mass is chosen as imperialist. The step is illustrated below.

Figure 14. Exchanging positions of the imperialist and a colony
3.3.3.10. Imperialistic Competition

This term refers to a match among imperialist in which the weaker the territories, the more reduction of power would be and the more powerful the territories, the more power it gains. The competition is to take the hegemony of the weakest colony of the weakest territory. The competition is started by first choosing one or more colonies that are the weakest. Then the hegemony of these weak colonies is taken by a stronger territory through the competition. So far, this does not necessarily mean that the strongest territory is the winner. This means that those that are stronger keep more hegemony. This competition is modeled by just selecting one of the weakest colonies of the weakest territory to formulate the hegemony of each territory first is obtained the normalized total cost as follows.

\[ NTP\ Emp_n = \max \{ TP\ Emp_i \} - TP\ Emp_n. \]  

(14)

Where:

- \( NTP_n \) is the normalized total power of nth empire
- \( TP_n \) is the total power of nth empire

After calculating normalized total power, the hegemony probability of each territory is obtained by:

\[ P_{Pn} = \frac{NTP\ Emp_n}{\sum_{i=1}^{N_{imp}} NTP\ Emp_i} \]  

(15)

Next, to allocate the abovementioned colony to a territory, a so called roulette wheel method is used that is shown below:

3.3.3.11 Eliminating the Powerless Empires

Through the competition, weak territories will ruin and their members dispense among other territories. The study refers to this ruined territory as collapses as shown below:
3.3.3.12 Stopping Criteria

The stopping point in competition, in this study, refers to the situation where there is only one territory remained among all countries. The process of territories purification is shown in three spectrums below:
The process that the study is offered is concisely illustrated below:

![Flowchart of MOICA](image)

4. Computational Experiments

4.1 Problem Design

The paper investigates the influence of some approaches for 36 test problems. Data are classified in three segments: quantity of operations, quantity of machines in first and second stages and, the dissemination of operation time in those stages. On the other hand, the abovementioned problems has been classified in two major segments; small and large problems. Table 1 depicts the quantity of operations and machine in small and large scale.
Table 1. Factors and their levels

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of jobs ( N )</td>
<td>( \text{Small}: 8, 10, 14, 16, 20, 24 )</td>
</tr>
<tr>
<td>No. Machines (in both stages)</td>
<td>( \text{Large}: 72, 80, 88, 108, 120, 132 )</td>
</tr>
<tr>
<td>Processing times ( p_{ij} )</td>
<td>( U(4, 40) )</td>
</tr>
<tr>
<td>Sequence dependent setup times ( S_{jki} )</td>
<td>( U(4, 40) )</td>
</tr>
<tr>
<td>Probability of rework ( P_{re} )</td>
<td>Exponential distribution ( \lambda e^{-t} ) with mean equal to 0.05</td>
</tr>
<tr>
<td>Rework times ( RT_{ij} )</td>
<td>Round ( U(0.3, 0.6) )</td>
</tr>
</tbody>
</table>

In addition the due dates are generated using the following formula:

\[
U(0, \sum_{j=1}^{n} (p_{1j} + p_{2j})) + \text{round} \left( \frac{(p_{1j} + p_{2j})}{(m_1 + m_2)} \right)
\] (16)

4.2 Parameter Setting

To evaluate the performance the suggested process, we need to set some key success factors. To come to this conclusion, some operations are simulated for both sizes of problems. Table 2 shows the tuned values of the proposed algorithms’ values.

Table 2. Tuned values of the parameters of the algorithms

<table>
<thead>
<tr>
<th>algorithm</th>
<th>parameter</th>
<th>small</th>
<th>large</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGAII</td>
<td>population size</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>max generation</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>crossover rate</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>mutation rate</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>Pop Size</td>
<td>150</td>
<td>300</td>
</tr>
<tr>
<td>MOICA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N_{imp}</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Maxdc</td>
<td>250</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>( \xi )</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>P_{AS}</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>P_{R}</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>PAES</td>
<td>max iteration</td>
<td>50000</td>
<td>200000</td>
</tr>
</tbody>
</table>

4.3 Performance Measures

The so-called Pareto based multi-objective optimization algorithm aims to find an estimate of non-dominated front. The key performance indicators of these methods are different from the single-purpose method. Hence, another method is needed to evaluate the key performance indicators of this algorithm. Then to evaluate the output of multi-purpose algorithms quantitatively, below evaluation methods are applied:

Number of Pareto solutions (NPS): in this method, the number of non-dominated solutions resulted by an algorithm is computed.

Mean ideal distance (MID): The approximation between Pareto solutions and ideal point is defined. The calculation of MID is:

\[
\text{MID} = \frac{\sum_{i=1}^{n} c_i}{n}
\] (17)

Where \( n \) is the quantity of non-dominated solutions and \( c_i = \sqrt{f_{1i}^2 + f_{2i}^2} \). The lower the value, the better performance the MID has.

- **Diversification metric (DM):** This performance metric shows the range of solutions resulted by algorithms and is calculated as follows:

\[
\text{DM} = \sqrt{(\text{max } f_{1i} - \text{max } f_{2i})^2 + (\text{max } f_{2i} - \text{max } f_{2i})^2}
\] (18)
• **The spread of non-dominance solutions (SNS):** Regarding to MID, this computes the range of non-dominated solutions. The SNS is defined as follows:

\[
SNS = \sqrt{\frac{\sum_{i=1}^{n} (MID - c_i)^2}{n-1}}
\]  

(19)

• **% Domination:** this key performance indicator applies a constructed Pareto combination set. Then the percentage of the solution belonging to each algorithm is calculated.

### 4.4 Experimental Results

This section refers to the result of experiments that are done through all the algorithms. The effectiveness of each algorithm is presented and compared in terms of key performance indicators. All algorithms were coded using MATLAB 2013a and run on personal computer with a 2.66 GHz CPU and 4 GB main memory.

The efficiency of the algorithms was stated by solving 36 variant problems of which 18 are small and 18 are large in scale. The outputs of three algorithms regarding the five key performance indicator for both size are compared and shown in Table 3 to 7 respectively.

Relative Percentage Deviation (RPD) is applied for the best solutions in terms of the key performance indicator. The calculation is shown below:

\[
RPD = \left| \frac{Method_{sol} - Best_{sol}}{Best_{sol}} \right| \times 100
\]  

(20)

Where

- **Method**<sub>sol</sub> is value of method
- **Best**<sub>sol</sub> is the best value between the algorithms

Table 4, 5 and 6 show the output with 95% sureness for the percentage of domination, DM and MID key performance indicator for small size respectively. Deep analysis shows that MOICA beats the others in terms of domination percent. Concerning DM, there is no considerable difference between NSGAII and MOICA. Yet the MOICA still beats PAES. The presented facts disclose that for MID key performance indicator, the three algorithms are similar and there is no considerable variance.

Table 7 presents the output of large scale problems. Alike the small scale problems, MOICA beats NSGAII and PEAS. The more size of the problem, the more considerable the advantages are.

Table 8, 9 and 10 also shows the outputs of algorithms in terms of domination percent in which DM and MID are shown consequently. By a glance at the table 8, MOICA's superiority in terms of performance in comparison to the next two. Outputs of DM are also illustrated in table 9 and disclose that the performance of NSGAII and MOICA identical and both of them outperform PAES. Moreover, when MID is considered, MOICA beats NSGAII and PEAS as shown in table 10.
Table 3. The simulation results for small size problems

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>MOICA</th>
<th>NSGAII</th>
<th>PAES</th>
<th>% Domination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>15.28</td>
<td>35.50</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>54.97</td>
<td>41.66</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>98.17</td>
<td>7.78</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>14</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4. 95% confidence interval for % domination in small size problems

<table>
<thead>
<tr>
<th>Individual 95% CIs For Mean Based on Pooled StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
</tr>
<tr>
<td>MOICA</td>
</tr>
<tr>
<td>NSGAII</td>
</tr>
<tr>
<td>PAES</td>
</tr>
</tbody>
</table>
Table 5. 95% confidence interval for DM in small size problems

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOICA</td>
<td>17</td>
<td>14.34</td>
<td>23.68</td>
<td>(-------*-------)</td>
</tr>
<tr>
<td>NSGAII</td>
<td>17</td>
<td>33.37</td>
<td>38.95</td>
<td>(--------*--------)</td>
</tr>
<tr>
<td>PAES</td>
<td>17</td>
<td>48.48</td>
<td>33.74</td>
<td>(--------*--------)</td>
</tr>
</tbody>
</table>

Table 6. 95% confidence interval for MID in small size problems

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOICA</td>
<td>18</td>
<td>4.33</td>
<td>13.35</td>
<td>(------------*------------)</td>
</tr>
<tr>
<td>NSGAII</td>
<td>18</td>
<td>3.72</td>
<td>12.05</td>
<td>(------------*------------)</td>
</tr>
<tr>
<td>PAES</td>
<td>18</td>
<td>10.06</td>
<td>14.67</td>
<td>(------------*------------)</td>
</tr>
</tbody>
</table>

0.0  5.0  10.0  15.0
Table 7. The simulation results for large size problems

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>95% CI Mean</th>
<th>95% CI StDev</th>
<th>95% CI Mean</th>
<th>95% CI StDev</th>
<th>95% CI Mean</th>
<th>95% CI StDev</th>
<th>95% CI Mean</th>
<th>95% CI StDev</th>
<th>95% CI Mean</th>
<th>95% CI StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>0.00</td>
<td>0.00</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>92.47</td>
<td>22.04</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>95.00</td>
<td>22.36</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
<td>(--*---)</td>
</tr>
</tbody>
</table>

Table 8. 95% confidence interval for % domination in large size problems

<table>
<thead>
<tr>
<th>Individual</th>
<th>95% CIs For Mean Based on Pooled StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOICA</td>
<td>20</td>
</tr>
<tr>
<td>NSGAII</td>
<td>20</td>
</tr>
<tr>
<td>PAES</td>
<td>20</td>
</tr>
</tbody>
</table>

216
Table 9. 95% confidence interval for DM in large size problems

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOICA</td>
<td>18</td>
<td>21.76</td>
<td>32.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSGAII</td>
<td>18</td>
<td>22.41</td>
<td>30.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAES</td>
<td>18</td>
<td>77.65</td>
<td>29.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10. 95% confidence interval for MID in large size problems

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>0.0</th>
<th>4.0</th>
<th>8.0</th>
<th>12.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOICA</td>
<td>18</td>
<td>0.362</td>
<td>1.536</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSGAII</td>
<td>18</td>
<td>6.411</td>
<td>3.650</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAES</td>
<td>18</td>
<td>10.692</td>
<td>5.756</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Conclusion and Further Researches

The study presents two criteria flow shop that is called two stage flexible flow shop. It aims to shorten the makespan and increase lateness of operations. The paper modeled Pareto optimal solutions and based the process on its similarity. NSGAII, MOICA and PEAS were suggested as the main three metaheuristic Pareto based multi-purpose algorithms. To estimate the efficiency of them, 36 problems, large and small, were answered. In multi-purpose norms, five key performance indicators, NPS, MID, DM, SNS and percentage domination were suggested to disclose the algorithms' efficiency. With similar efficiency, NSGAII beats PAES in both small and large scale of problem, in terms of DM. hence, MOICA is the best algorithm in case of efficiency for all the studied problems.

To guidance for researchers in similar cases, using other effective metaheuristic algorithm like multi-purpose and colony optimization or multi-purpose invasive weed optimization are suggested to work on. Also there would be valuable result if the stages increase for more than two.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (no. 70971020).

References


**Copyrights**
Copyright for this article is retained by the author(s), with first publication rights granted to the journal. This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).