A Novel SFLA for Re-entrant Hybrid Flow Shop Scheduling

Kun Wang, Lamei Xu, Cong Lv
School of Automation, Wuhan University of Technology, Wuhan, 430070, China.
2522466153@qq.com, 1729956248@qq.com, 757414298@qq.com

Abstract- In order to optimize makespan and total tardiness for re-entrant hybrid flow shop scheduling problems simultaneously, this paper proposes a novel Shuffled Frog Leaping Algorithm. The algorithm relies on the lexicographic method to process total tardiness and makespan. A suitable crossover operator is introduced. During the searching process of memeplex, instead of worst solution, multiple best solutions are used to generate new solutions. Further with that, the process of generating random solution is replaced with the neighborhood search. Then as indicated by the simulation based on a group of instances, the novel SFLA is superior in addressing hybrid flow shop scheduling problem.

1. Introduction
Regarding conventional HFSP (hybrid flow shop scheduling problem), it is usually assumed that each job can be processed only once on each machine. However, this assumption is not effective in some industrial processes, such as electronic industry because each job needs to be processed repeatedly on the same machine to reduce production costs. This sort of HFSP is known as re-entrant HFSP, namely RHFSP. RHFSP is widely used in semiconductor and manufacturing industry, such as the production of integrated circuits. RHFSP is highly complex compared to HFSP. It’s a NP-hard problem [1].

Heuristic algorithm and hybrid algorithm are the main effective approaches to solving RHFSP. Cho presented a parallel genetic algorithm with four different local search structures to solve RHFSP with minimum total tardiness and makespan [2]. Then proposed a novel TLBO algorithm, and designed a new decoding strategy, that is, the equivalent due date sorting method which transfers the individual to the feasible solution [3].

Lorenzo applied an effective heuristic algorithm to minimize the makespan and deduced the error bound [4]. Ying suggested an iterative Pareto greedy algorithm for solving the makespan and total tardiness of two-objective RHFSP [1]. Cho designed a two-level algorithm combining production planning algorithm and genetic algorithm [5].

SFLA was first proposed based on frog foraging behavior by Eusuff and Lansey in 2003[6]. It’s a swarm intelligence optimization algorithm combining meme algorithm and PSO algorithm. SFLA has the combining meme algorithm and PSO algorithm. SFLA exhibits the advantages of strong search ability, excellent robustness, fast convergence speed and ease to realize. SFLA is extensively used to solve engineering optimization problems, with many valuable results obtained. Alireza proposed a hybrid multi-objective SFLA and applied it to assemble line scheduling problem with pipeline balancing[7]. Alinia achieved a success in solving JSP by combining SFLA with local search algorithms based on PPX, GOX and SPX [8]. In this paper, a novel SFLA is proposed to optimize the makespan and total tardiness for cyclical RHFSP simultaneously.
2. Problem Description

Re-entrant hybrid flow shop scheduling problem includes \( m \) stages and \( n \) jobs. Stage \( i \) has \( \theta_i \) unrelated parallel machines (\( \theta_i \geq 1 \)), and more than one machine in a minimum of one stage, namely \( \exists i, \theta_i > 1 \). The processing sequence of every job is identical. Each job is processed in such a sequence: stage1, stage2, ..., stage \( m \) for \( L \) times (\( L > 1 \)). Machines in each stage of each times can be different. Each job is required to finish the processing of \( m \) stage for the first time completely, and only then the next time of processing is permitted. Finally, each job is processed for \( L \) times. The illustration of RHFSP is showed in Figure 1.

![Illustration of RHFSP](image)

Fig. 1 Illustration of RHFSP

RHFSP can be split into scheduling sub-problem and machine assignment sub-problem. Scheduling is purposed to make the specific machine for each job during each stage in every re-entrant. Machine assignment clarifies the processing sequence of job on each machine.

Objective functions are as follows.

\[
\begin{align*}
\text{Minimize } f_1 &= \sum_{h=1}^{n} \max\{C_h - d_h, 0\} \\
\text{Minimize } f_2 &= \max_{h=1,2,...,n} \{C_h\}
\end{align*}
\]

(1) (2)

where \( C_h \) — finishing time of job \( h \).
\( d_h \) — due dates of job \( h \).
\( f_1 \) — total tardiness.
\( f_2 \) — makespan.

In this paper, tardiness is key objective while makespan is non-key objective. At present, there are two methods to resolve optimization problems with key objective. They’re weighted method and lexicographic order method. Weighted method assigns lower weights to non-key objective in order to increase the importance attached to key objective. Firstly, the latter algorithm evaluates two solutions by key objective. If the key objective of two solutions is no different, it evaluates them by non-key objective. That is to say, if \( f_1(x) < f_1(y) \) or \( f_1(x) = f_1(y) \) and \( f_2(x) < f_2(y) \), then \( x \) is better than \( y \). The lexicographic order method is chosen because it is more suited to SFLA.

3. A Novel SFLA for RHFSP

3.1. Coding and decoding strategy
RHFSP includes scheduling sub-problem and machine assignment sub-problem. Thus, the solution of scheduling and the solution of machine assignment are represented with two strings separately. Job strings \((\pi_1, \pi_2, \ldots, \pi_{mL})\) and machine assignment strings \((q_{11}, q_{12}, \ldots, q_{im}, q_{21}, q_{22}, \ldots, q_{im}, \ldots, q_{nL})\) are introduced to indicate the solutions for RHFSP with \(n\) jobs, \(m\) stages and \(L - 1\) re-entrant. \(\pi_i \in \{1, 2, \ldots, n\}\), length of job strings is \(L \times n\).

In job strings, job \(i\) will appear \(L\) times in total. The first appearance of job \(i\) in job strings indicates the first processing of job \(i\), the second appearance of job \(i\) in job strings indicates the second processing of job \(i\), and so on. In machine assignment strings, \(q_{ik}\) indicates the machine which is assigned to the job \(i\), stage \(k\) during processing \(l\). From the first machine in machine strings, \(m\) machines are assigned to job 1, processing 1 which contains \(m\) stages, next \(m\) machines are assigned to job 2, processing 1 which contains \(m\) stages, and so on.

Each machine is assigned to each job \(\pi_i\) in job strings. In case that \(\pi_i\) appears in strings for the first time, machine \(q_{ik}\) is assigned to each job from stage 1, \(k = 1, 2, \ldots, m\). In case that \(\pi_i\) appears in job strings for \(lh\) times, machines are also assigned to each job from first stage. What is essential is that earliest processing time of \(lh\) processing must be longer than the completion time of the \((l-1)th\) processing at corresponding stage.

For instance, job strings is (1, 7, 3, 10, 6, 2, 9, 8, 5, 3, 4, 7, 5, 1, 2, 9, 10, 6, 4, 8), machine strings is (1, 7, 3, 5, 1, 4, 1, 5, 2, 6, 3, 8, 3, 7, 2, 8, 1, 4, 1, 5, 3, 7, 3, 6, 1, 5, 2, 4, 3, 7, 3, 4, 3, 5, 2, 8, 1, 8, 2, 7). Where (1, 7) represents two machines for the first processing of job 1, (3, 5) represents two machines for the first processing of job 2. The first (1, 4) represents the two machines required for the first processing of job 3. The second (3, 7) represents the two machines required for the first re-entrant of job 1 and so on.

### 3.2. A new SFLA

In standard SFLA, searching process within memeplex has three main procedures as follows.

1. Conduct global search between \(x_g\) and \(x_w\) through eq. (3).

2. If the new solution is not better than the old one, conduct global search between \(x_b\) and \(x_g\) through eq. (4).

3. If the new solution is not better than the old one, \(x_w\) will be replaced with a new solution which is generated on a random basis.

\[
x_w = x_w + \text{Rand}() \ast (x_b - x_w) \tag{3}
\]

\[
x_w = x_w + \text{Rand}() \ast (x_g - x_w) \tag{4}
\]

Where \(x_b\) — best solution in memeplex.

\(x_g\) — globally optimal solution.

\(x_w\) — worst solution in memeplex.

\(\text{Rand}()\) — random numbers between [0, 1].

According to characteristics exhibited by SFLA, an improved search strategy is suggested. \(x_w\) is replaced with better solution for conducting global search on job strings and machine assignment strings. Perform VNS (variable neighborhood search) instead of procedure (3).

Global search on job strings uses GOX (generalized order crossover) [9]. The procedures of GOX are as follows.

1. Select \(x_i\) and \(x_2\) as parents. Mark each element in parents with \(\text{Index}\) to store the number of times of element appearance.

2. Select two position \(a, b\) \((a < b)\), pick out elements between \(a\) and \(b\) in \(x_2\) as substring. Delete elements in \(x_i\) which is identical to the elements in substring but has the same \(\text{Index}\).
(3) Find the element which is not only identical to the first element in substring. Position of this element is denoted by \( \text{pos} \). Insert substring into \( x_j \) at position \( \text{pos} \) to generate new solution \( x_{new} \).

An example is provided in the Fig. 2 as follows.

\[
x_1 = 1 \ 7 \ 3 \ 1 \ 0 \ 6 \ 2 \ 9 \ 8 \ 5 \ 3 \ 4 \ 7 \ 5 \ 1 \ 2 \ 9 \ 10 \ 6 \ 4 \ 8 
\]

\[
\text{Index} = 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 
\]

\[
x_2 = 6 \ 8 \ 1 \ 0 \ 4 \ 2 \ 7 \ 6 \ 9 \ 3 \ 1 \ 3 \ 5 \ 4 \ 5 \ 8 \ 9 \ 7 \ 1 \ 2 \ 10 
\]

\[
\text{Index} = 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 
\]

\[
x_{new} = 1 \ 7 \ 3 \ 1 \ 0 \ 6 \ 2 \ 9 \ 3 \ 1 \ 3 \ 5 \ 8 \ 5 \ 3 \ 4 \ 7 \ 5 \ 1 \ 2 \ 9 \ 10 \ 6 \ 4 \ 8 
\]

\[
\text{Index} = 1 \ 7 \ 1 \ 0 \ 6 \ 2 \ 9 \ 3 \ 1 \ 3 \ 5 \ 8 \ 5 \ 4 \ 7 \ 2 \ 9 \ 10 \ 6 \ 4 \ 8 
\]

Fig. 2 GOX on scheduling strings

Global search conducted on machine strings uses two points crossover. We select two positions both on \( x_j \) and \( x_2 \), and let the elements between two positions on \( x_2 \) replace the elements between two positions on \( x_j \). In doing so, new solution \( x_{new} \) will be generated. An examples is given in the following Fig. 3.

\[
x_1 = 1 \ 7 \ 3 \ 5 \ 1 \ 4 \ 1 \ 5 \ 2 \ 6 \ 3 \ 8 \ 3 \ 7 \ 2 \ 8 \ 1 \ 4 \ 1 \ 5 \ 3 \ 7 \ 3 \ 6 \ 1 \ 5 \ 2 \ 4 \ 3 \ 7 \ 3 \ 4 \ 3 \ 5 \ 2 \ 8 \ 1 \ 8 \ 2 \ 7 
\]

\[
x_2 = 2 \ 4 \ 1 \ 6 \ 1 \ 8 \ 3 \ 6 \ 3 \ 4 \ 1 \ 7 \ 1 \ 5 \ 3 \ 6 \ 2 \ 4 \ 1 \ 6 \ 2 \ 4 \ 2 \ 5 \ 3 \ 4 \ 3 \ 7 \ 1 \ 8 \ 3 \ 5 \ 2 \ 6 \ 1 \ 7 \ 1 \ 7 \ 3 \ 4 
\]

\[
x_{new} = 1 \ 7 \ 3 \ 5 \ 1 \ 4 \ 1 \ 5 \ 2 \ 6 \ 3 \ 8 \ 3 \ 7 \ 2 \ 8 \ 1 \ 4 \ 1 \ 6 \ 2 \ 4 \ 3 \ 6 \ 1 \ 5 \ 2 \ 4 \ 3 \ 7 \ 3 \ 4 \ 3 \ 5 \ 2 \ 8 \ 1 \ 8 \ 2 \ 7 
\]

Fig. 3 two points crossover on machine assignment strings

We set a constant \( \beta \), and a random number \( \alpha \) satisfying \([0,1]\) uniform distribution is generated. If \( \alpha < \beta \), global search is conducted for job strings, and global search is performed for machine strings otherwise. Scheduling sub-problem is more complex than machine assignment problem, for which the former requires more calculation resource than the latter. After carrying out a large number of experiments, \( \beta = 0.8 \).

In standard SFLA, the worst solution will be replaced with a random solution when global search is incapable to obtain a better solution. In general, a solution generated randomly is invariably not good enough for RHFS, which suggests that previous searching process is in vain. In this case, we use VNS instead.

As for scheduling strings, neighborhood structure is swap. Two jobs \( \pi_i \) and \( \pi_j \) are selected randomly (\( \pi_i \neq \pi_j \)), before their position in scheduling strings is exchanged. For machine assignment strings, neighborhood structure is change. A machine \( q_{ik} \) is selected from machine assignment strings randomly. If \( q_{ik} \) is capable of processing \( o_{ik} \), the set of the machine which can do this processing is \( \Theta \). A different machine is chosen from \( \Theta \) to replace \( q_{ik} \).

The procedures of solving RHFS with new SFLA is shown as follows.

\text{Step 1: Initialization.} Initialize basic parameters of the algorithm: the number of frogs in population\((N)\), the number of memeplex\((s)\), the number of iteration in whole population\((T)\), the number of iteration in memeplex\((\mu)\).

\text{Step 2:} Generate initial population \( P. t=1. \) Calculate the total tardiness and makespan of each individual.

\text{Step 3:} Individual is sorted in ascending order by the total tardiness. For those individuals with the same tardiness, sort them in ascending order by makespan. Select the best individual as an optimal solution \( X_g \).
Step 4: Categorize individuals into groups: put the first individual into first group, put the second individual into second group…, put individual s into group s, put individual s+1 into first group and so on.

Step 5: Do local searching process of memeplex.

Step 5-1: Initialize parameters of memeplex search. \( m=1, n=1 \).

Step 5-2: Choose best three individuals from the population. Select an individual from them as \( x_j \) on a random basis. Then select an individual from the rest two individuals as \( x_2 \) at random. Conduct crossover operation between \( x_j \) and \( x_2 \).

Step 5-3: Generate a random number \( \text{rand} \) satisfying \([0,1]\) uniform distribution. If \( \text{rand} < 0.8 \), perform GOX for scheduling strings Otherwise, perform two points crossover for machine strings. If the new solution \( x_{\text{new}} \) generated by crossover operation satisfies the criteria of replacement, then replace \( x_j \) with \( x_{\text{new}} \).

Step 5-4: If the criteria is not satisfied, conduct crossover operation between \( x_j \) and \( X_g \) as described in Step 5-3.

Step 5-5: If the criteria fails to be satisfied, perform VNS on \( x_j \). Generate a random integer \( \alpha \) on interval \([0,1]\). If \( \alpha = 0 \), do operation swap on \( x_j \). Otherwise, do operation change. Do VNS for 10 times and obtain solution \( x_j' \). Then if the criteria is satisfied, replace \( x_j \) with \( x_j' \). Otherwise, \( x_j \) remains unchanged.

Step 5-6: \( m=m+1 \), if \( m \leq \mu \), go to Step 5-2.

Step 5-7: \( n=n+1 \), if \( n \leq s \), go to Step 5-2.

Step 6: Mix all individuals and reorder. Update the optimal solution \( X_g \).

Step 7: \( t=t+1 \), if \( t \leq T \), return step 4; otherwise, output \( X_g \).

The flowchart of the novel SFLA is presented in Figure 4.

```
Start

Initialization

Generate initial population

Sort individuals in ascending order

Divide individuals into groups

Do searching process of memeplex

Mix all individuals and reorder

Termination criteria satisfied?

N

Y

End
```

Fig. 4 Flowchart of the novel SFLA

4. Experiments And Results

In this section, experiments and results are presented and discussed. To demonstrate the advantage exhibited by the novel SFLA intuitively, GA [10] (genetic algorithm) and VNS [11] are selected for comparison purpose.
4.1. Description of instances

Considered of 30 instances: \( n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \) and \( m = 2, 4, 6 \). The number of re-entrant is 1. The processing time of each job on every machine is a random number on interval \([1,5]\). There are 6 stages, and the number of machines at each stages is 3, 5, 3, 2, 4, 3 in order. The formula to calculate the delivery time is

\[
d_i = \delta \sum_{j=1}^{m} \max_{k=1,2,...,\theta_k} \{p_{ijk}\}.
\]

Where \( p_{ijk} \) represents the processing time of job \( i \) processed by machine \( k \) on stage \( m \). \( \delta \) is a random number on interval \([1.6,1.7]\).

4.2. Parameters setting

SFLA applies the following values to the parameters: Population scale \( N = 50 \). The number of memeplex \( s = 5 \). The number of iteration in memeplex \( \mu = 2 \times 10^4 \). The number of iteration in the whole population \( T = 2 \times 10^4 \).

VNS uses the identical values with SFLA for the parameters. GA uses the following values for the parameters: Population scale = 100. Crossover probability = 0.6. Mutation probability = 0.1. Maximum iteration = 100.

4.3. Results and discussion

Table 1 indicates that novel SFLA obtains optimal solutions of 28 instances. VNS obtains optimal solutions of 2 instances. It demonstrates that best solutions of SFLA are noticeably superior to VNS’s and GA’s on large-scale. Table 2 reveals that the values of SFLA are superior for worst solutions.

| \( n \times m \) | SFLA | SFLA | GA | \( n \times m \) | SFLA | SFLA | GA |
|-------------|------|------|----|-------------|------|------|----|
| 10\(^2\)  | 3    | 13   | 8  | 13  | 119  | 58   | 247 | 75  | 218 | 64  | 70\(^6\) | 70277 | 5196 | 172 | 5554 | 203 | 7817 | 206 |
| 10\(^4\)  | 33   | 31   | 35 | 39  | 34   | 33   | 60\(^2\) | 4136 | 165 | 4322 | 192 | 6231 | 174 |
| 10\(^6\)  | 6    | 38   | 10 | 45  | 10   | 46   | 60\(^4\) | 3759 | 153 | 4142 | 183 | 5405 | 170 |
| 20\(^2\)  | 56   | 26   | 112| 31  | 257  | 36   | 70\(^2\) | 2277 | 93  | 2652 | 104 | 5850 | 142 |
| 20\(^4\)  | 302  | 61   | 383| 78  | 379  | 58   | 70\(^4\) | 5653 | 187 | 5835 | 206 | 8928 | 202 |
| 20\(^6\)  | 119  | 58   | 247| 75  | 218  | 64   | 70\(^6\) | 5396 | 172 | 5554 | 203 | 7817 | 206 |
| 30\(^2\)  | 259  | 41   | 319| 47  | 774  | 61   | 80\(^2\) | 3303 | 109 | 3513 | 123 | 7545 | 149 |
| 30\(^4\)  | 883  | 92   | 1005| 102 | 1247 | 88   | 80\(^4\) | 7636 | 215 | 8173 | 254 | 11689| 234 |
| 30\(^6\)  | 626  | 81   | 754 | 93  | 853  | 90   | 80\(^6\) | 7467 | 207 | 7646 | 241 | 10392| 223 |
| 40\(^2\)  | 572  | 54   | 797 | 64  | 1481 | 77   | 90\(^2\) | 4290 | 119 | 4646 | 133 | 10033| 176 |
| 40\(^4\)  | 1715 | 117  | 1862| 140 | 2350 | 112  | 90\(^4\) | 102225| 252 | 10909 | 315 | 15531| 267 |
| 40\(^6\)  | 1454 | 108  | 1497| 135 | 1998 | 118  | 90\(^6\) | 9878 | 237 | 9816 | 276 | 13296| 245 |
| 50\(^2\)  | 1108 | 71   | 1311| 76  | 2631 | 93   | 100\(^2\) | 5478 | 136 | 6512 | 158 | 12348| 191 |
| 50\(^4\)  | 2777 | 138  | 2980| 165 | 4050 | 143  | 100\(^4\) | 12296| 271 | 13173| 336 | 19346| 295 |
| 50\(^6\)  | 2496 | 132  | 2571| 167 | 3498 | 148  | 100\(^6\) | 12086| 265 | 11720| 297 | 16964| 275 |
Table 2 comparison of the worst solutions

| n * m | SFLA | SFLA | GA | n * m | SFLA | SFLA | GA |
|-------|------|------|----|-------|------|------|----|
|       | \( f_1 \) | \( f_2 \) | \( f_1 \) | \( f_2 \) | \( f_1 \) | \( f_2 \) | \( f_1 \) | \( f_2 \) |
| 10*2  | 10   | 15   | 35  | 23  | 42   | 20   | 2039 | 83   |
| 10*4  | 35   | 31   | 85  | 47  | 38   | 34   | 4801 | 169  |
| 10*6  | 13   | 36   | 40  | 50  | 11   | 45   | 4244 | 156  |
| 20*2  | 83   | 26   | 198 | 33  | 326  | 49   | 2848 | 97   |
| 20*4  | 447  | 63   | 501 | 77  | 463  | 60   | 6644 | 195  |
| 20*6  | 193  | 62   | 386 | 89  | 256  | 64   | 6061 | 182  |
| 30*2  | 1070 | 90   | 1307| 113 | 1333 | 90   | 3829 | 111  |
| 30*4  | 765  | 85   | 954 | 105 | 959  | 91   | 6844 | 210  |
| 30*6  | 744  | 54   | 989 | 71  | 1713 | 74   | 5414 | 127  |
| 40*4  | 1933 | 118  | 2163| 151 | 2517 | 113  | 10976| 251  |
| 40*6  | 1684 | 103  | 1878| 126 | 2220 | 120  | 10367| 233  |
| 50*4  | 1369 | 71   | 1782| 90  | 2869 | 95   | 6213 | 140  |
| 50*6  | 3086 | 139  | 3619| 183 | 4217 | 144  | 13798| 268  |
| 50*6  | 2867 | 131  | 3126| 159 | 3703 | 147  | 13215| 264  |
| 60*2  | 2039 | 83   | 2626| 104 | 4308 | 114  |      |      |
| 60*4  | 6644 | 195  | 2626| 104 | 4308 | 114  |      |      |
| 60*6  | 6061 | 182  | 6379| 185 | 5789 | 172  |      |      |
| 70*2  | 2848 | 97   | 3233| 113 | 6323 | 140  |      |      |
| 70*4  | 6644 | 195  | 7016 |233  | 9181 | 204  |      |      |
| 70*6  | 6061 | 182  | 6379| 185 | 5789 | 172  |      |      |
| 80*2  | 3829 | 111  | 4252| 125 | 8141 | 158  |      |      |
| 80*4  | 6844 | 210  | 6582| 231 | 10905| 226  |      |      |
| 80*6  | 5414 | 127  | 5626| 139 | 10961| 180  |      |      |
| 90*2  | 10976| 251  | 11512|315  | 16371| 270  |      |      |
| 90*4  | 10976| 251  | 11512|315  | 16371| 270  |      |      |
| 90*6  | 10367| 233  | 10300|261  | 13881| 252  |      |      |
| 100*2 | 6213 | 140  | 7088| 162 | 13387| 197  |      |      |
| 100*4 | 13798| 268  | 14160|315  | 20086| 301  |      |      |
| 100*6 | 13215| 264  | 13249|305  | 17895| 282  |      |      |

Figure 5 and Figure 6 present the distribution graph of solutions for three algorithms working on instance \( 80 \times 6 \) and instance \( 100 \times 6 \). The three algorithms are shown to have gap on astringency. SFLA displays excellent performance because it coordinates with VNS. In contrast, GA and VNS fail to balance global search and local search. Therefore, this novel SFLA is a competitive method to solve RHFSP.

5. Conclusion
After encoding machine assignment sub-problem and scheduling sub-problem, a new SFLA was introduced to make makespan and total tardiness optimal for RHFSP at the same time. Searching
process in memeplex is not premised on worst solution, and it replaces the procedures of generating a random new solution with VNS, which was effective in making global search and local search work in a collaborative way in the memeplex searching process. As revealed by the experiments with totally 30 instances, the new SFLA is superior to GA and VNS in solving RHFSP.

The next stage of work includes making improvement to other intelligent optimization algorithms such as artificial bee algorithms (ABC) for solving RHFSP. Besides, we will focus on developing the new strategy of combining global search and local search by using a balance method.

References
[1] Ying K C, Lin S W, Wan S Y 2014 Int J of Production Research vol 52 (Great Britan: Institution of Production Engineers) pp 5735–5747
[2] Hang-Min Cho, Suk-Joo Bae, Jungwuk Kim, In-Jae Jeong 2011 Computers & Industrial Engineering vol 61 (Amsterdam: Elsevier Ltd)
[3] Shen J N, Wang L, Zheng H Y 2016 Int J of Production Research vol 54 (Great Britan: Institution of Production Engineers) pp 3622-3639
[4] Lorenzo L 2017 Lecture Notes in Engineering and Computer Science vol 2229 (Denmark: Newswood Ltd) pp 50-56
[5] Cho H M, Jeong I J 2017 Computers & Industrial Engineering vol 106 (Amsterdam: Elsevier Ltd) pp 174-181
[6] Eusuff M M , Lansey K E 2003 J of Water Resources Planning and Management vol 129 (American: American Society of Civil Engineers, Water Resources Planning and Management Division, American Society of Civil Engineers ) pp 210-225
[7] Alireza R V, Ali H M 2007 Computers & Industrial Engineering vol 53 (Amsterdam: Elsevier) pp 642-666
[8] Alinia Ahandani M, Pourqorban Shirjoposht N, Banimahd R 2009 14th Int CSI Computer Conf (New York: IEEE Press) pp 670-675
[9] Mattfeld D C 1995 Evolutionary search and the job shop: investigations on genetic algorithms for production scheduling (Germany: Physica-Verlag Heidelberg)
[10] Holland J H 1975 Control and Artificial Intelligence vol 6 (Michigan: University of Michigan Press) pp 126-137
[11] Mladenovid N, Hansen P 1997 Computers and Operations Research vol 24 (Amsterdam: Elsevier Ltd) pp 1097-1100