INTELLIGENT AUTOMATION & SOFT COMPUTING, 2017
https://doi.org/10.1080/10798587.2017.1302711



# On the Use of Genetic Algorithm for Solving Re-entrant Flowshop Scheduling with Sum-of-processing-times-based Learning Effect to Minimize Total Tardiness

Win-Chin Lin[a], Chin-Chia Wu[a], Kejian Yu[b], Yong-Han Zhuang[a] and Shang-Chia Liu[c]

[a]Department of Statistics, Feng Chia University, Taichung City, Taiwan; [b]Management School, Zhejiang Shuren University, Zhejiang, China; [c]Business Administration Department, Fu Jen Catholic University, New Taipei City, Taiwan

**ABSTRACT**

Most research studies on scheduling problems assume that a job visits certain machines only one time. However, this assumption is invalid in some real-life situations. For example, a job may be processed by the same machine more than once in semiconductor wafer manufacturing or in a printed circuit board manufacturing machine. Such a setting is known as the "re-entrant flowshop". On the other hand, the importance of learning effect present in many practical situations such as machine shop, in different branches of industry and for a variety of corporate activities, in shortening life cycles, and in an increasing diversity of products in the manufacturing environment. Inspired by these observations, this paper addresses a re-entrant *m*-machine flowshop scheduling problems with time-dependent learning effect to minimize the total tardiness. The complexity of the proposed problem is very difficult. Therefore, in this paper we first present four heuristic algorithms, which are modified from existing algorithms to solve the problem. Then, we use the solutions as four initials to a genetic algorithm. Finally, we report experimental performances of all the proposed methods for the small and big numbers of jobs, respectively.

**KEYWORDS**

Total tardiness; re-entrant flowshop; Genetic algorithm; Learning effect

## 1. Introduction

There is a common point in traditional scheduling problems, in which most researchers assume that each job can be processed by each machine only at one time. However, this constraint may not be valid in some real production applications. Applications can be seen in semiconductor wafer manufacturing (Vargas-Villamil & Rivera, 2001), in signal processing (Wang, Sethi, & van de Velde, 1997), and in printed circuit board manufacturing (Bengu, 1994; Bispo & Tayur, 2001; Kubiak, Lou, & Wang, 1996; Uzsoy, Lee, & Martin-Vega, 1992). In such situations, each job may be processed by the same machine several times. This is called the "re-entrant flowshop" in the literature.

In the re-entrant flowshop scheduling problem, researchers considered that jobs in the shop must be processed on all $m$ machines, every job follows the route of $M_1, M_2, \ldots, M_m$; $M_1, M_2, \ldots, M_m$; …; and $M_1, M_2, \ldots, M_m$, with $r$ re-entrants, and the job sequence is the same on any machine at each level (Pan and Chen (2003); Chen (2006); Chen, Pan, and Wu (2007); Xu, Li, Hu, and Li (2014)). Taking the minimization of the makespan criterion, Pan and Chen (2003) applied mixed binary integer program formulations, several heuristic algorithms, and the Lingo optimizer for this problem. In solving the same problem, Chen (2006) proposed five lower bounds to be used in a branch-and-bound algorithm to solve it. Chen et al. (2007) built a hybrid tabu algorithm for the problem. Recently Xu et al. (2014) adopted the CPLEX solver and a memetic algorithm (MA) for the problem. For more works of the re-entrant flowshop scheduling problem in different settings, the readers might refer to Choi and Kim (2009), Alfieri (2009), Rau and Cho (2009), Chu, Chu, and Desprez (2010), Liu (2010), and Boudhar and Meziani (2010). In addition, the reader might refer to three good surveys of the re-entrant scheduling problems by Bellman and Ernest (1982), Uzsoy et al. (1992), and Lin and Lee (2011).

Most of classical scheduling models considered that the job processing time was fixed and constant. However, it is invalid in many practical situations. For example, a steady decline in processing times usually takes place by performing the same task repeatedly (Biskup, 1999). Another example is each basic operation used in the manufacture of a product has its own learning function (Nadler & Smith, 1963). In such a situation, the processing time of a job maybe constantly occurring and the dynamic feature taken into consideration, or called "the learning effect".

The importance of the learning effect is present in many practical situations. Applications of the learning effect can be found in machine shop (Nadler & Smith, 1963), in different branches of industry and for a variety of corporate activities (Yelle, 1979), and in shortening life cycles and an increasing diversity of products in the manufacturing environment (Higgins, Le Roy, & Tierney, 1996). However, the notion of learning was not introduced to the field of scheduling until Biskup (1999), and Cheng and Wang (2000). Since then, scheduling with learning considerations has received more attention in the scheduling research community for only fifteen years. For details of research results on variants of the scheduling problem with different learning settings, the reader might refer to a good survey paper by Biskup (2008).

Existing literature released show that most scheduling with learning effects considered a single-machine case (Yang, Hsu,

**CONTACT** Shang-Chia Liu docchia@gmail.com

and Yang (2010), Yin, Xu, and Wang (2010), Yin and Wang (2011), Wang (2010), Yang and Yang (2012), Wang and Guo (2010), Wang, Zhou, Zhang, Ji, and Wang (2013)), yet the concept of learning process on scheduling flowshop are still relatively limited. For example, Wang and Xia (2005) provided the worst-case bound of the shortest processing time first (SPT) rule for the makespan and total completion time problems for the flowshop scheduling problem with a learning effect. Wu and Lee (2009) considered a permutation flowshop scheduling problem with a learning effect to minimize the sum of completion times. They proposed a dominance rule and several lower bounds used in the branch-and-bound algorithm to solve the problem. Wang and Wang (2011) considered the processing time of a job as an exponential function of its position in a processing permutation and discussed the worst cases for four regular performance criteria, namely; the total completion time, the total weighted completion time, the discounted total weighted completion time, and the sum of the quadratic job completion times for flowshop scheduling problems. Chung and Tong (2012) discussed a bi-criteria scheduling problem in an $m$-machine permutation flowshop environment with varied learning effects on different machines, where the objective is to minimize the weighted sum of the total completion time and makespan. They proposed a dominance criterion and a lower bound to accelerate the branch-and-bound algorithm for deriving the optimal solution. Kuo, Hsu, and Yang (2012) assumed that the time-dependent learning effect of a job was a function of the total normal processing time of jobs scheduled before the job and investigated worst-case analysis for the objective functions such as; the makespan, the total flowtime, the sum of weighted completion times, the sum of the $k$th power of completion times, and the maximum lateness on the flowshop setting. Cheng, Wu, Chen, Wu, and Cheng (2013) studied a two-machine flowshop scheduling problem with a truncated learning function where they considered the actual processing time of a job as a function of the job's position in a schedule and the learning truncation parameter, their objective function is to minimize the makespan. They proposed a branch-and-bound and three crossover-based genetic algorithms (GAs) to find the optimal and approximate solutions for the problem. Cheng (2013) considered a permutation flowshop scheduling problem with a position-dependent exponential learning effect to minimize the performance criteria of the makespan and total flow time. He showed Johnson's rule is not an optimal algorithm for the two-machine flow shop case and used the shortest total processing times first (STPT) rule to construct the worst-case performance ratios for the problems. For more papers with learning considerations, we refer the reader to the state of the art research concerning these problems by Janiak, Krysiak, and Trela (2011), and several recent learning studies by Yin, Xu, Sun, & Li, 2009; Yin, Xu, & Wang, 2010a, 2010b; (2011, 2012), and Zhang, Liu, Yin, and Wu (2016).

As far as the authors know, there is only research on scheduling with the learning effects in the re-entrant flowshop setting. Xu et al. (2016) studied the re-entrant permutation flowshop scheduling problem with a position-based function to minimize the mean flow time. Wu et al. (2016) considered the re-entrant permutation flowshop scheduling problem with a sum-of-processing-times-based learning function to minimize the makespan. They developed a simulated annealing (SA) algorithm to find near-optimal solutions for the problem. In view of these observations, in this paper we study the re-entrant permutation flowshop scheduling problem with a sum-of-processing-times-based learning function to minimize the total tardiness, which is a criterion popularly considered in the field of scheduling.

The organization of this paper is as follows: In Section 2, we define some notations and describe the problem. In Section 3, we build a genetic algorithm by combining four well-known rules in solving the flowshop scheduling problem with learning considerations to search for approximate solutions. In Sections 4 and 5, we conduct extensive experimental results, which are included some small-job size and big-size jobs to test the performance of the proposed algorithms when changing the number of machines, the learning effects, and the number of re-entrant times. Finally, we provide some conclusions in Section 6.

## 2. Problem Statement

Consider an $m$-machine flowshop in which there are $n$ jobs to be processed on the $m$ machines (i.e., $M_1, M_2, \ldots M_m$) in the same order and the job sequence is the same on all machines. Consider that $n$ jobs are ready at time zero for all machines throughout the scheduling period and no preemption is allowed. Suppose that the waiting space between the machines is unlimited for the $n$ jobs to be processed. In the re-entrant permutation flowshop setting, each job visits the $m$ machines following the same order, that is; $M_1, M_2, \ldots, M_m$; $M_1, M_2, \ldots, M_m$; …; and $M_1, M_2, \ldots, M_m$, with up to $l$ re-entrant times (or levels), starting on machine $M_1$ and ending on $M_m$, and the job sequence is the same on any machine at each level. For clarification, we assume that each level has its own sum-of-processing-times-based learning effect, i.e., the actual processing time of a job $J_j$ to be scheduled in the $k$th position of a given schedule at level $l$ on machine $M_i$ is given by $p^l_{i[k]} = p^l_{ij} \cdot \left(1 + \sum_{t=1}^{k-1} p^l_{i[t]}\right)^a$, where $p^l_{ij}$ denotes the normal processing time of a job $J_j$ on machine $M_i$ at level $l$, $[k]$ denotes the job scheduled in the $k$th position of a sequence of jobs, $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$; $l = 1, 2, \ldots, r$, and $a < 0$ is the learning index. The objective of this paper is to minimize the total tardiness.

Let $d_j$ be the due date of a job $J_j$ and $C^l_{i[k]}$ denotes the completion time of a job scheduled in the $k$th position of a sequence of jobs on machine $M_i$ at level $l$. Therefore,

$$C^l_{i[k]} = \max\left(C^l_{i-1,[k]}, C^l_{i,[k-1]}\right) + p^l_{i[k]},$$

and the total tardiness is $\sum_{k=1}^{n} T^r_{m[k]}$, where $T^r_{m[k]} = \max\{0, C^r_{m[k]} - d_{[k]}\}$.

To illustrate the working of the above definition, we provide an example in the following.

**An illustrative example.** Table 1 shows the data of a problem instance with $n = 3$ jobs, $m = 2$ machines, and $r = 2$ re-entrant times, and the learning index is $a = -0.01$.

**Table 1.** The Illustrative Data.

| $l$ | $J_1$ | $J_2$ | $J_3$ |
|---|---|---|---|
| $M^1_1$ | 49 | 46 | 39 |
| $M^1_2$ | 20 | 52 | 56 |
| $M^2_1$ | 86 | 77 | 34 |
| $M^2_2$ | 49 | 67 | 45 |
| $d_j$ | 280 | 343 | 352 |

Consider a job sequence $(J_1, J_2, J_3)$ as the example. The completion times of three jobs on machine $M_2$ at level one are given as follows:

$$C^1_{2[1]} = C^1_{1[1]} + p^1_{2[1]} = 49 + 20 = 69$$

$$C^1_{2[2]} = \max\left(C^1_{1[2]}, C^1_{2[1]}\right) + p^1_{2[2]} = \max(93.2, 69) + 50.4 = 143.6$$

$$C^1_{2[3]} = \max\left(C^1_{1[3]}, C^1_{2[2]}\right) + p^1_{2[3]} = \max(129.5, 143.6) + 53.6 = 197.2$$

In what follows, the completion times of three jobs on machine $M_2$ at level two are given as follows:

$$C^2_{2[1]} = \max\left(C^2_{1[1]}, C^1_{1[3]}\right) + p^2_{2[2]} = \max(216.5\ ,\ 197.2) + 49 = 265.5$$

$$C^2_{2[2]} = \max\left(C^2_{1[2]}, C^2_{2[1]}\right) + p^2_{2[2]} = \max(290.1,\ 265.5)\ + 364.4 = 354.5$$

$$C^2_{2[3]} = \max\left(C^2_{1[3]}, C^2_{2[2]}\right) + p^2_{2[3]} = \max(322.4\ ,\ \ 354.5) + 42.9 = 397.4$$

In addition, Figure 1 represents the details of computing the completion times of three jobs on two machines at two levels. Therefore, total tardiness of the job sequence $(J_1, J_2, J_3)$ is $\sum_{j=1}^{3} \max\{0, C^2_{2[j]} - d_{[j]}\} = 56.9$.

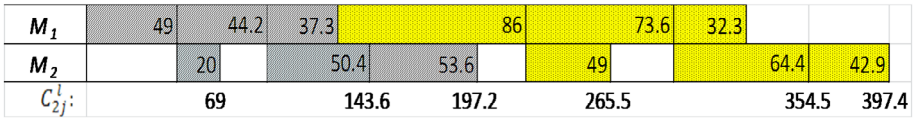



**Figure 1.** The flow of three jobs on two machines with two re-entrant times.

## 3. Four Heuristic-based Genetic Algorithms

The proposed problem without a learning effect is NP-hard (Koulamas (1994)). Therefore, it requires an advantage policy to consider an effective heuristic or metaheuristic for a good quality solution.

Genetic algorithm (GA) is one of the most popular metaheuristics. The procedures of the GA are composed of an initial population, a fitness function, crossover and mutation operators, a selection mechanical to choose next generation, and a stopping rule. In order to obtain the better initial solutions as the input of GA, we consider four heuristics. The first two heuristic rules (algorithms) shown below are priority-relevant rules, and the third and the fourth are the earliest due date heuristic and an (adapted) Johnson's algorithm.

For a job $J_j$, let $TP_{j[k]} = \sum_{l=1}^{r}\sum_{i=1}^{m} p^l_{ik}, k = 1, 2, \cdots, n$, be the total processing time without considering a learning effect if job $J_j$ to be scheduled in the position $k$, and $TT$ be the total of all process times, i.e. $TT = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n} p^l_{ij}$, moreover, let $ST_k$ be the starting time of a job to be scheduled in a position $k$ of a job sequence, and $ST_{1=}0$.

### 3.1. Cost Over Time (COVERT) Rule

There are three steps for the COVERT rule.

Step 1: Calculate a priority order to be scheduled in the position $k$ for jobs. (i) If $d_j < \left(ST_k + TP_{j[k]}\right)$, define$PR_j = 1$. (ii) if $d_j > (ST_k + TP_{j[k]})$ and $d_j < TT$, define $/, PR_j = (TT - d_j)/\left(RT - TP_{j[k]}\right)$, where $RT$ is the sum of the yet unscheduled jobs. (iii) If $d_j \geq TT$, define $PR_j = 0$.

Step 2: Calculate $CF_j = PR_j/TP_{j[k]}$ for unscheduled jobs.

Step 3: Select the job with the maximum $CF$ and place this job to position $k$.

### 3.2. Critic Ratio (CR) Rule

Define $CR_j = (d_j - ST_j)/TP_{j[k]}$. The critical ratio rule is to select the job with the smallest value of $CR$ and place this job in the first position of a sequence of jobs, where $ST_{1=}0$. This job is marked as being scheduled. Then select the smallest value of $CR$ among the unscheduled jobs and place this job in the second position, where $ST_2$ is the starting time of the second job. Continue this process until all the $n$ jobs are arranged in a sequence of jobs.

**Table 2.** The Performance of Four Heuristic Algorithms for $n = 8$.

| | | COVERT+GA | | | CR+GA | | | EDD+GA | | | JOHNSON+GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| parameters | **labels** | Best | Mean | Worst | Best | Mean | Worst | Best | Mean | Worst | Best | Mean | Worst |
| τ | 0.25 | 0.0010 | 1.4054 | 8.8850 | 0.0011 | 1.3598 | 8.5064 | 0.0012 | 1.3479 | 8.0603 | 0.0010 | 1.2606 | 7.6429 |
| | 0.5 | 0.0007 | 1.1893 | 7.1717 | 0.0009 | 1.2230 | 7.2748 | 0.0008 | 1.1972 | 7.3350 | 0.0009 | 1.1086 | 6.5060 |
| R | 0.25 | 0.0010 | 1.5178 | 9.6502 | 0.0010 | 1.5053 | 9.3008 | 0.0012 | 1.4820 | 9.1667 | 0.0013 | 1.3829 | 8.2719 |
| | 0.5 | 0.0008 | 1.3123 | 7.9651 | 0.0012 | 1.3271 | 8.4307 | 0.0010 | 1.2755 | 7.8353 | 0.0009 | 1.2148 | 7.3848 |
| | 0.75 | 0.0008 | 1.0619 | 6.4698 | 0.0009 | 1.0418 | 5.9403 | 0.0008 | 1.0601 | 6.0910 | 0.0007 | 0.9559 | 5.5666 |
| a | −0.1 | 0.0010 | 1.3632 | 10.254 | 0.0013 | 1.3166 | 10.012 | 0.0012 | 1.3563 | 8.7541 | 0.0014 | 1.2813 | 8.9501 |
| | −0.01 | 0.0007 | 1.2742 | 7.268 | 0.0008 | 1.2913 | 7.0298 | 0.0008 | 1.2452 | 6.6141 | 0.0007 | 1.1554 | 6.2256 |
| | −0.001 | 0.0008 | 1.2547 | 6.5634 | 0.0010 | 1.2663 | 6.6300 | 0.0009 | 1.2157 | 6.6687 | 0.0008 | 1.1169 | 6.0476 |
| m | 2 | 0.0006 | 0.6784 | 6.4481 | 0.0007 | 0.6403 | 5.6444 | 0.0008 | 0.6080 | 5.6731 | 0.0008 | 0.5940 | 5.0481 |
| | 3 | 0.0007 | 1.2583 | 7.6422 | 0.0011 | 1.2578 | 8.4391 | 0.0011 | 1.2053 | 7.7831 | 0.0008 | 1.1011 | 6.6990 |
| | 5 | 0.0012 | 1.9554 | 9.9947 | 0.0013 | 1.9761 | 9.5883 | 0.0011 | 2.0042 | 9.6368 | 0.0014 | 1.8586 | 9.4763 |
| *l* | 2 | 0.0012 | 1.7036 | 11.129 | 0.0018 | 1.6798 | 10.827 | 0.0015 | 1.6501 | 10.852 | 0.0016 | 1.4493 | 9.2993 |
| | 3 | 0.0008 | 1.2263 | 7.4186 | 0.0007 | 1.2258 | 7.0745 | 0.0009 | 1.2271 | 6.932 | 0.0009 | 1.1572 | 6.8069 |
| | 4 | 0.0005 | 0.9623 | 5.5371 | 0.0006 | 0.9686 | 5.7704 | 0.0006 | 0.9405 | 5.309 | 0.0005 | 0.9472 | 5.1171 |
| Average | | | 1.2974 | | | 1.2914 | | | 1.2725 | | | 1.1846 | |

**Table 3.** The Performance of Four Heuristic Algorithms for $n = 10$.

| | | COV-ERT+GA | | | CR+GA | | | EDD+GA | | | JOHN-SON+GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| parame-ters | labels | Best | Mean | Worst | Best | Mean | Worst | Best | Mean | Worst | Best | Mean | Worst |
| **τ** | 0.25 | 0.0023 | 1.8777 | 9.0275 | 0.0016 | 1.8461 | 8.7031 | 0.0024 | 1.8392 | 8.6195 | 0.0021 | 1.7178 | 7.8363 |
| | 0.5 | 0.0018 | 1.6248 | 7.4445 | 0.0066 | 1.6462 | 7.5387 | 0.0016 | 1.6001 | 7.2441 | 0.0018 | 1.4819 | 6.5870 |
| **R** | 0.25 | 0.0021 | 2.0466 | 9.7134 | 0.0068 | 2.0598 | 9.6683 | 0.0025 | 2.0579 | 9.3983 | 0.0026 | 1.8810 | 8.6161 |
| | 0.5 | 0.0018 | 1.7835 | 8.2012 | 0.0013 | 1.7570 | 8.2469 | 0.0019 | 1.6777 | 7.7791 | 0.0021 | 1.6141 | 7.0621 |
| | 0.75 | 0.0023 | 1.4236 | 6.7933 | 0.0042 | 1.4217 | 6.4475 | 0.0016 | 1.4233 | 6.6180 | 0.0012 | 1.3044 | 5.9567 |
| **a** | −0.1 | 0.0020 | 1.8923 | 10.8835 | 0.0023 | 1.9188 | 10.6919 | 0.0027 | 1.9317 | 10.4050 | 0.0028 | 1.8727 | 9.6576 |
| | −0.01 | 0.0023 | 1.6703 | 6.9335 | 0.0067 | 1.6558 | 6.7822 | 0.0016 | 1.6219 | 6.6000 | 0.0015 | 1.4778 | 6.0367 |
| | −0.001 | 0.0019 | 1.6910 | 6.8908 | 0.0034 | 1.6638 | 6.8886 | 0.0017 | 1.6054 | 6.7904 | 0.0016 | 1.4490 | 5.9406 |
| **m** | 2 | 0.0006 | 0.9656 | 6.1769 | 0.0008 | 0.9204 | 6.2213 | 0.0008 | 0.8986 | 5.6733 | 0.0009 | 0.8297 | 5.1972 |
| | 3 | 0.0019 | 1.6958 | 8.2449 | 0.0016 | 1.6740 | 8.1852 | 0.0021 | 1.6402 | 8.3790 | 0.0015 | 1.5119 | 7.1392 |
| | 5 | 0.0037 | 2.5923 | 10.2861 | 0.0099 | 2.6440 | 9.9562 | 0.0031 | 2.6201 | 9.7431 | 0.0035 | 2.4580 | 9.2985 |
| ***l*** | 2 | 0.0033 | 2.4007 | 12.055 | 0.0020 | 2.3551 | 11.6770 | 0.0028 | 2.3108 | 11.0434 | 0.0029 | 2.1679 | 9.8663 |
| | 3 | 0.0014 | 1.5968 | 7.3753 | 0.0044 | 1.6099 | 7.4313 | 0.0020 | 1.5702 | 7.1434 | 0.0014 | 1.4595 | 6.7141 |
| | 4 | 0.0015 | 1.2563 | 5.2779 | 0.0060 | 1.2734 | 5.2544 | 0.0012 | 1.2780 | 5.6086 | 0.0016 | 1.1721 | 5.0546 |
| Average | | | 1.7512 | | | 1.7461 | | | 1.7196 | | | 1.5999 | |

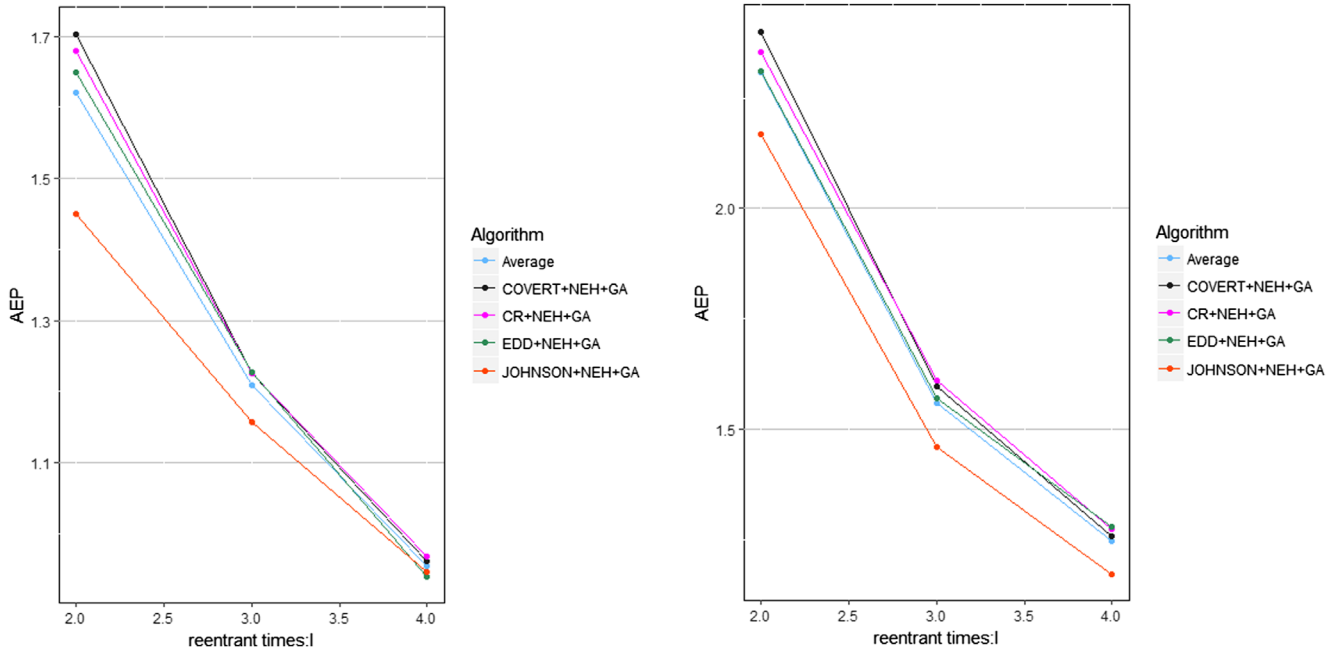



**Figure 2.** The performance of four algorithms when *l* changes (n = 8, left, and n = 10).

Note that the details of Cost Over Time (COVERT) Rule and Critic Ratio (CR) Rule, readers can refer to a book by Sule (1997) on pages 14–17.

### 3.3. Earliest Due Date (EDD) Rule

The earliest due date (EDD) is a very popular algorithm for due date scheduling problems. We thus employ EDD as the third rule.

Select the job with the earliest due date among the $n$ jobs and place this job in the first position. Then select the earliest due date among the jobs not yet to be scheduled and place it in the second position. Continue this process until the jobs are arranged in a sequence of jobs. The resulting sequence of jobs will be in an ascending order of their due dates.

### 3.4. (Adapted) Johnson's Rule

The Johnson's rule is a powerful algorithm to generate a minimum makespan for the two-machine flowshop scheduling makespan minimization problem. For simplification, we only applied the Johnson's algorithm to the first and the last machines, and took re-entrant times into consideration. The steps of adapted Johnson rule are as follows:

Step 1: For a job $J_j$, calculate $A_{1j} = \sum_{l=1}^{r} p_{1j}^{l}$, and $A_{mj} = \sum_{l=1}^{r} p_{mj}^{l}$, for $j \in \{1, 2, \cdots, n\}$.

Step 2: Find the minimum among the pool of $A_{1j}$'s and $A_{mj}$'s. If the minimum is from $A_{1j}$'s, then place the corresponding job in the earliest position. If the minimum is from $A_{mj}$'s, then place the corresponding job in the latest position.

Step 3: Indicate the job as being scheduled and efface the associated $A_{1j}$'s and $A_{mj}$'s.

Step 4: If all $n$ jobs are place in the sequence, we output a complete sequence of jobs; otherwise, go to Step 2.

In order to speed the search quality, we adopted a local improvement process. The idea of this process is similar to the NEH algorithm (Nawaz, Enscore, and Ham (1983)), yet the jobs were not sorted by their total processing times at the beginning, instead a feasible sequence of jobs constructed by the above four rules was used. Let $S$ be a feasible sequence of

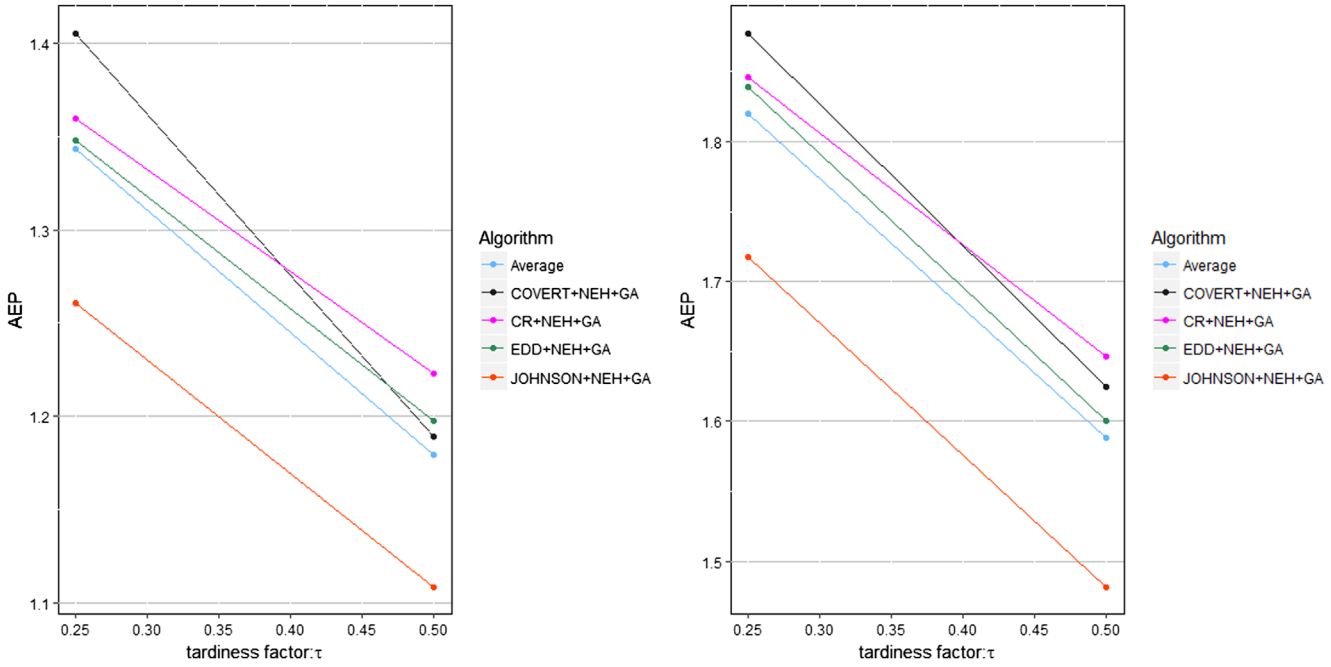



**Figure 3.** The performance of four algorithms when $\tau$ changes (n = 8, left, and n = 10).

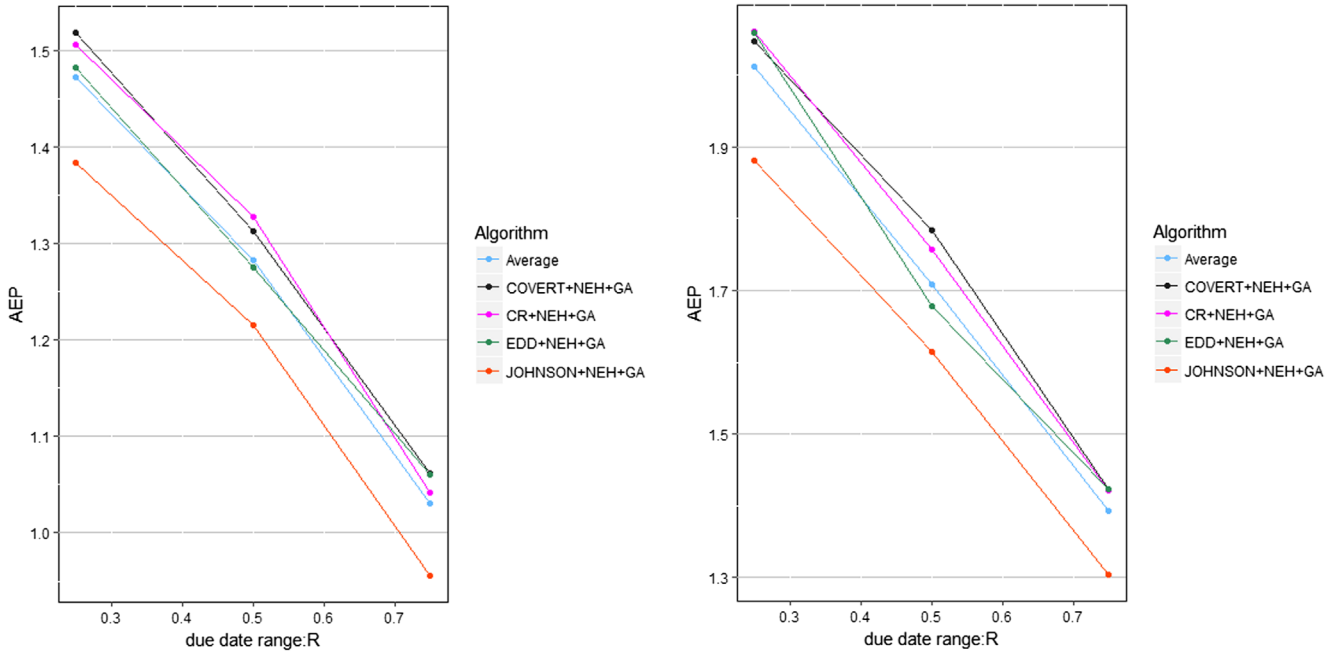



**Figure 4.** The performance of four algorithms when $R$ changes (n = 8, left, and n = 10).

jobs constructed from the above four rules. The following 3 steps are then used. First, find the best tardiness of reallocation of the first and the second jobs in $S$ without changing the position for the other $n$-2 jobs in $S$. The relative positions of these two jobs with respect to each are then fixed in the later process. Second, pick the job in the third position of $S$ and insert it into all possible positions of the scheduled subsequence, calculate the resulting value of the tardiness. Find the best position and place this job in the best place. Third, repeat Step 2 for the job in the next position, and insert it into all possible positions of the scheduled subsequence, calculate the resulting value of the tardiness, find the best position and place this job in the best place. Repeat this process until a job is placed in the last position. A new sequence of jobs is thus constructed. We will denote this local improvement as NEH-type method.

In this study, representation of structure - a sequence of the jobs in the problem is followed by (Etiler, Toklu, Atak, and Wilson (2004)). We ran four times a genetic algorithm (GA). In the first one, the sequence of jobs constructed by COVERT algorithm and locally improved by the NEH-type method, as an initial solution, was used in a GA, which is designated as COVERT+GA. As for the second, third, and fourth genetic algorithms, denoted as CR+GA, EDD+GA, and JOHNSON+GA, were constructed by the CR, EDD, and Johnson's rules and locally improved by NEH-type rule, respectively. The steps for the genetic algorithm are as follows:

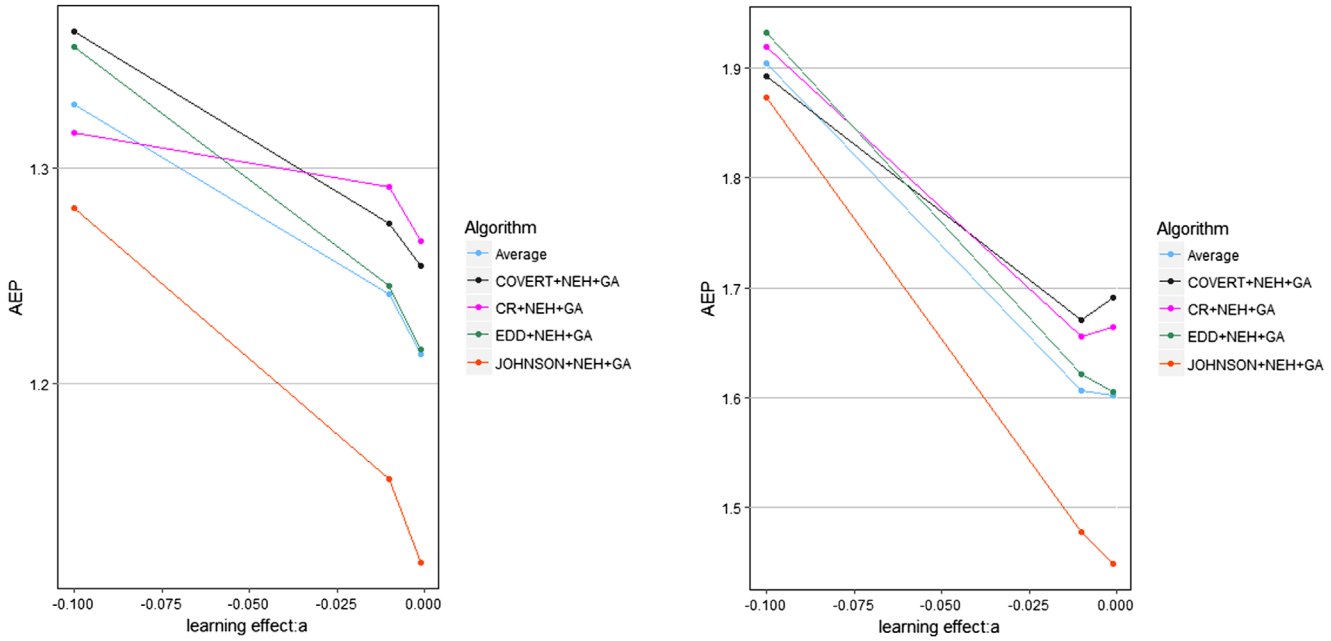



**Figure 5.** The performance of four algorithms when a changes (n = 8, left, and n = 10).

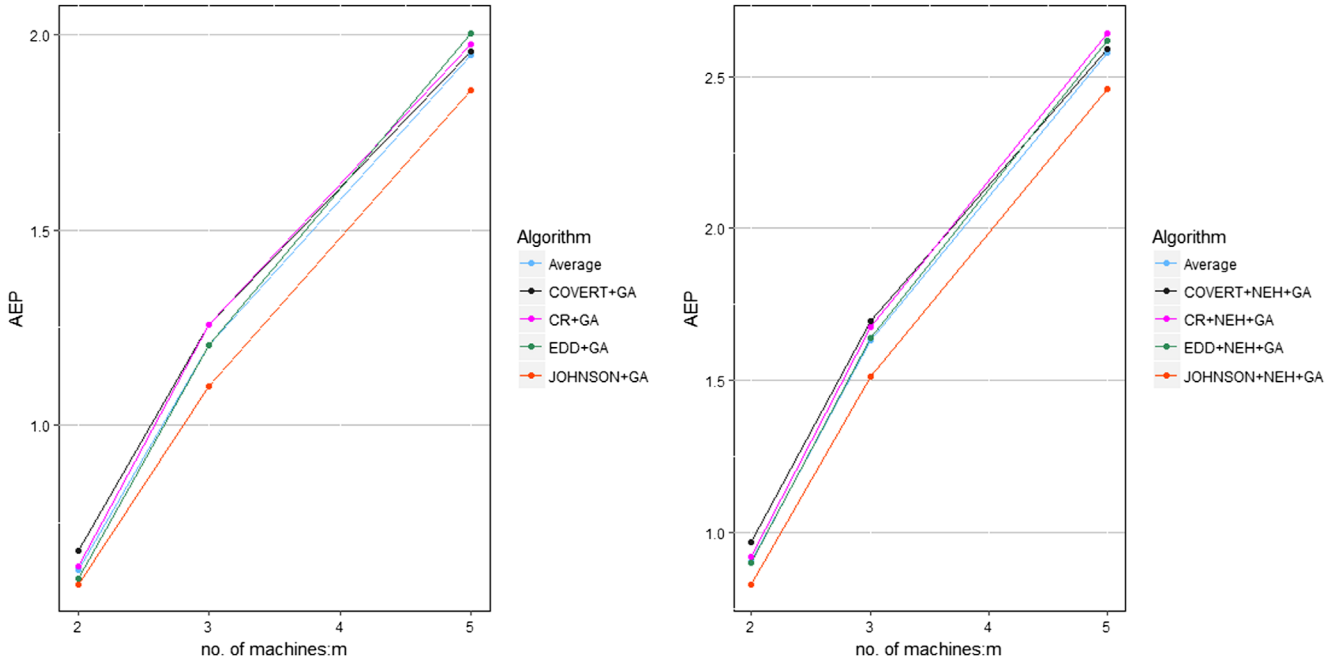



**Figure 6.** The performance of four algorithms when $m$ changes (n = 8, left, and n = 10).

### 3.5. Genetic algorithm process

Step 1: Initial population- The sequences of jobs constructed by the aforementioned four heuristics, and improved by the NEH-type method mentioned in the above, as initial sequences in GA.

Step 2: Population size- We adopt an initial population and create other members by using a pairwise interchange operator up to population size (Chen, Vempati, & Aljaber, (1995)). In a preliminary trial, the population size $N$ is set at 20 in our computational experiment.

Step 3: Fitness function- The fitness function of a schedule, a sequence of jobs, can be calculated as: $g(S_i(v)) = \max_{1\le k\le N}\left\{T_j(S_k(v))\right\} - T_j(S_i(v))$, where $S_i(v)$ is the $i$th string chromosome in the $v$-th generation, $T_j(S_i(v))$ is the total tardiness of $S_i(v)$, and $g(S_i(v))$ is the fitness value of $S_i(v)$. The probability, $P(S_i(v))$, of selection for a sequence of jobs can be calculated as: $P(S_i(v)) = g(S_i(v))/\sum_{k=1}^{N} g(S_k(v))$. This is also the criterion used for the selection of parents for the reproduction of children.

Step 4: Crossover- Followed by (Falkenauer and Bouffoix (1991), Etiler and Toklu (2001), Etiler et al. (2004)), we adopt linear order crossover (LOX) method in each GA. In order to protect the best schedule, which has the minimum total tardiness at each generation, we transfer this schedule to the next population with no change, or the crossover rate $P_{c=}$ 1.

Step 5: Mutation- The mutation rates ($P_m$) are set at 0.25 based on our preliminary experiment.

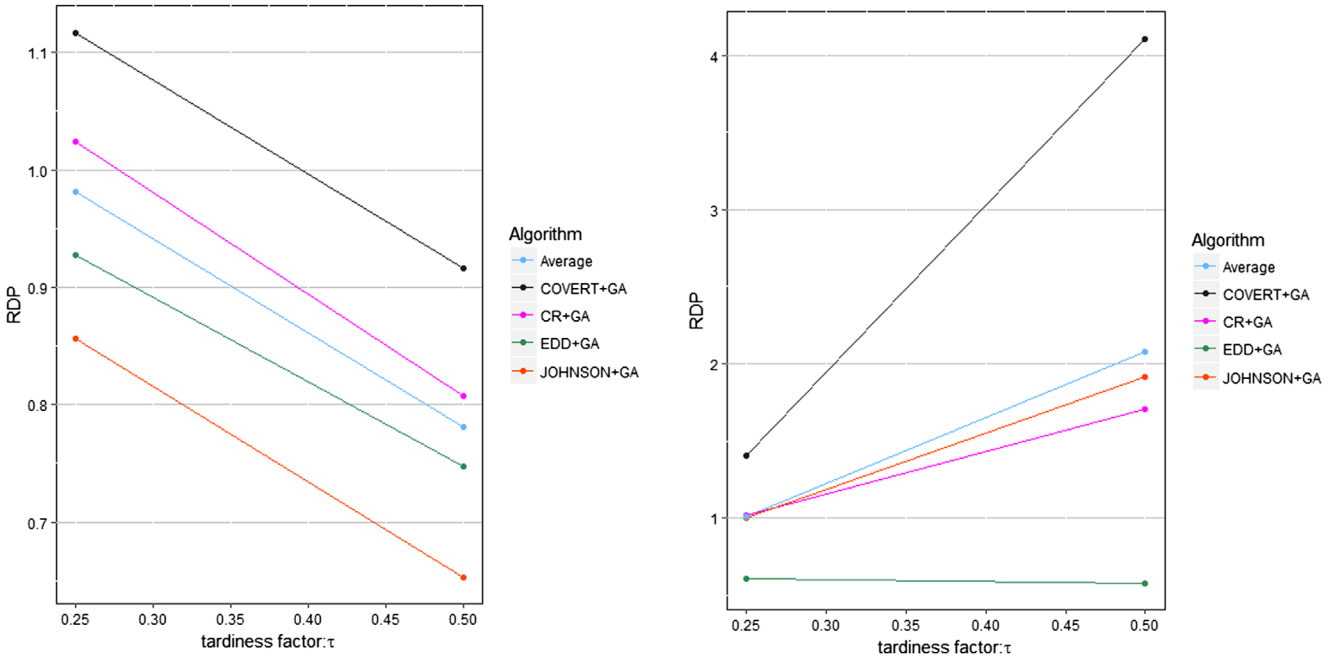



**Figure 7.** The performance of Four algorithms when $\tau$ changes (n = 40, left, and n = 80).

Step 6: Selection- In each GA, the population sizes are set to 20 from generation to generation. Excluding the best schedule, which has the minimum total tardiness, the other off-springs are generated randomly from the parent chromosomes by the roulette wheel method.

Step 7: Stopping rule- Each GA is terminated after 2000 generations in our preliminary experiment.

## 4. The Performances of Four Algorithms on the Small Number of Jobs

The algorithms were coded in Fortran and executed on Compaq Visual Fortran version 6.6 with a 2.66-GHz Intel(R) Core(TM)2 Duo E8200 CPU, and 1.99-GB RAM on Windows XP. We conducted computational experiments to test the performance of the four heuristics COVERT+GA, CR+GA, EDD+GA and JOHNSON+GA in solving the re-entrant permutation flowshop scheduling problem by using the same data.

The processing times were generated randomly from uniformly distributed integers over 1 to 100. The due dates were generated randomly from another discrete uniform distribution $\{0.5 \times TT \times (1 - \tau - \frac{R}{2}, 0.5 \times TT \times (1 - \tau + \frac{R}{2})\}$, where $TT$ the total of all process times, defined in section 3, $\tau$ is the tardiness factor and $R$ is the due date range. The factor 0.5 in the above formula was used to avoid too large due date and no tardiness to happen. The values of $\tau$ were set at 0.25 and 0.5, and the levels of $R$ were designed at 0.25, 0.5 and 0.75. The learning effects, $a$, were set at -0.1, -0.01 and -0.001. The test problem instances were generated for each combination of $\tau$, $R$ and $a$. Hence, 18 experimental situations were conducted and 100 replicates were generated randomly. The number of jobs was tested at 8 and 10 for small jobs, while the number of jobs was used at 40 and 80 for the large of jobs. As for the number of machines, 2, 3 and 5 were used to examine the effect of size of machines. The total test problem instances were then 64800 ($18 \times 100 \times 3 \times 2 \times 2 \times 3$).

The criterion used for evaluating the four heuristic algorithms was the average error percentage (AEP), AEP = $\{(V_i\text{-}V^*)/V^*\} \times 100\%$, where $V_i$ is the objective value obtained by a heuristic, and V* the optimal solution, which was obtained by an enumerative method, for small-sized problem, $n = 8$ and 10. All results are summarized in Tables 2 and 3.

**Table 4.** Results of Fisher's LSD for Difference between AEP Means for 4 GA Algorithms for n = 8, 10.

| Pairwise Comparison | Pairwise Difference | LSD(α =0 .05) = 0.0647 |
|---|---|---|
| Between Algorithms | $\lvert\overline{AEP_I} - \overline{AEP_J}\rvert$ | Difference > LSD? |
| COVERT+GA vs. CR+GA | \|1.5243–1.5188\| | No |
| COVERT+GA vs. EDD+GA | \|1.5243–1.4961\| | No |
| COVERT+GA vs. JOHNSON+GA | \|1.5243–1.3922\| | Yes |
| CR+GA vs. EDD+GA | \|1.5188–1.4961\| | No |
| CR+GA vs. JOHNSON+GA | \|1.5188–1.3922\| | Yes |
| EDD+GA vs. JOHNSON+GA | \|1.4961–1.3922\| | Yes |

As shown in Figure 2, Tables 2 and 3, the algorithm JOHNSON+GA generated the lowest average error percentage (AEP) among the four algorithms on the reentrant times (l) for $n = 8$ and 10. The average error percentage declines as the value of r increases, from 2 to 4, for all four algorithms. The overall average error percentage of the four algorithms, COVERT+GA, CR+GA, EDD+GA and JOHNSON+GA, were 1.2974, 1.2914, 1.2725, and 1.1846 for $n = 8$, and were 1.7512, 1.7461, 1.7196, and 1.5999 for $n = 10$, respectively. The trend is also shown in Figure 2.

The same pattern of the performance results was easily seen for the four algorithms on the parameters tardiness factor ($\tau$), due date range ($R$) and learning effect ($a$), i.e. the average error percentage decreases as the value of parameters increases. Figures 3, 4, 5 and 7 showed that the algorithm JOHNSON+GA consistently generated the lowest average error percentages (AEP) among the four algorithms on parameters $\tau$, $R$ and $a$ for both job sizes 8 and 10.

As regards the number of machines ($m$), the AEP increases as the $m$ increases for all four algorithms and for both sizes of jobs, shown in Figure 6. The more the number of machines, the more complexity for the problems, therefore, searches for good schedules are more difficult for the algorithms.

**Table 5.** The Performance of Four Heuristic Algorithms for $n = 40$ and 80.

| | | n = 40 | | | | n = 80 | | | |
|---|---|---|---|---|---|---|---|---|---|
| parameters | Labels | COVERT+GA | CR+GA | EDD+GA | JOHNSON+GA | COVERT+GA | CR+GA | EDD+GA | JOHNSON+GA |
| τ | 0.25 | 1.1169 | 1.0244 | 0.9269 | 0.8564 | 1.4071 | 1.0197 | 0.6046 | 1.0007 |
| | 0.5 | 0.9161 | 0.8074 | 0.7476 | 0.6533 | 4.1109 | 1.7032 | 0.5781 | 1.9170 |
| R | 0.25 | 1.5162 | 1.2524 | 1.1399 | 1.1074 | 6.8837 | 2.8762 | 0.8479 | 3.4063 |
| | 0.5 | 0.9237 | 0.8478 | 0.7344 | 0.7601 | 0.9631 | 0.7650 | 0.4934 | 0.6859 |
| | 0.75 | 0.6095 | 0.6475 | 0.6374 | 0.3970 | 0.4302 | 0.4431 | 0.4328 | 0.2843 |
| a | −0.1 | 1.8845 | 1.4698 | 1.3531 | 1.5238 | 7.4897 | 3.2230 | 1.0275 | 3.9135 |
| | −0.01 | 0.5756 | 0.6206 | 0.5917 | 0.3839 | 0.4029 | 0.4422 | 0.3867 | 0.2402 |
| | −0.001 | 0.5894 | 0.6574 | 0.5669 | 0.3568 | 0.3844 | 0.4192 | 0.3600 | 0.2227 |
| m | 2 | 0.8102 | 0.6726 | 0.4872 | 0.6010 | 4.2268 | 1.2837 | 0.3898 | 1.6702 |
| | 3 | 0.9990 | 0.8922 | 0.8468 | 0.7366 | 2.3756 | 1.4491 | 0.5636 | 1.3984 |
| | 5 | 1.2402 | 1.1829 | 1.1778 | 0.9269 | 1.6746 | 1.3516 | 0.8207 | 1.3079 |
| *l* | 2 | 1.6590 | 1.4131 | 1.1739 | 1.3755 | 7.0705 | 3.0022 | 0.7990 | 3.6155 |
| | 3 | 0.8330 | 0.7925 | 0.8009 | 0.5037 | 0.7594 | 0.6564 | 0.5652 | 0.4980 |
| | 4 | 0.5575 | 0.5421 | 0.5370 | 0.3853 | 0.4471 | 0.4258 | 0.4100 | 0.2630 |
| Average | | 1.0165 | 0.9159 | 0.8372 | 0.7548 | 2.7590 | 1.3615 | 0.5914 | 1.4588 |

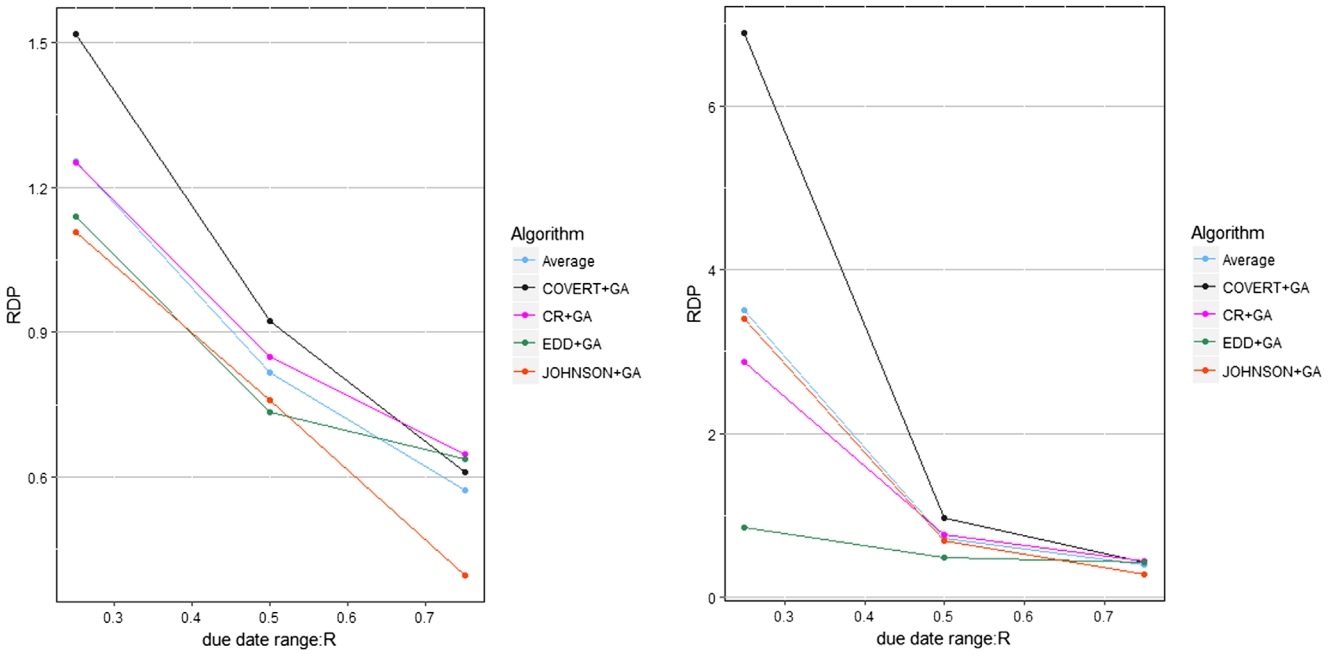



**Figure 8.** The performance of four algorithms when R changes (n = 40, left, and n = 80).

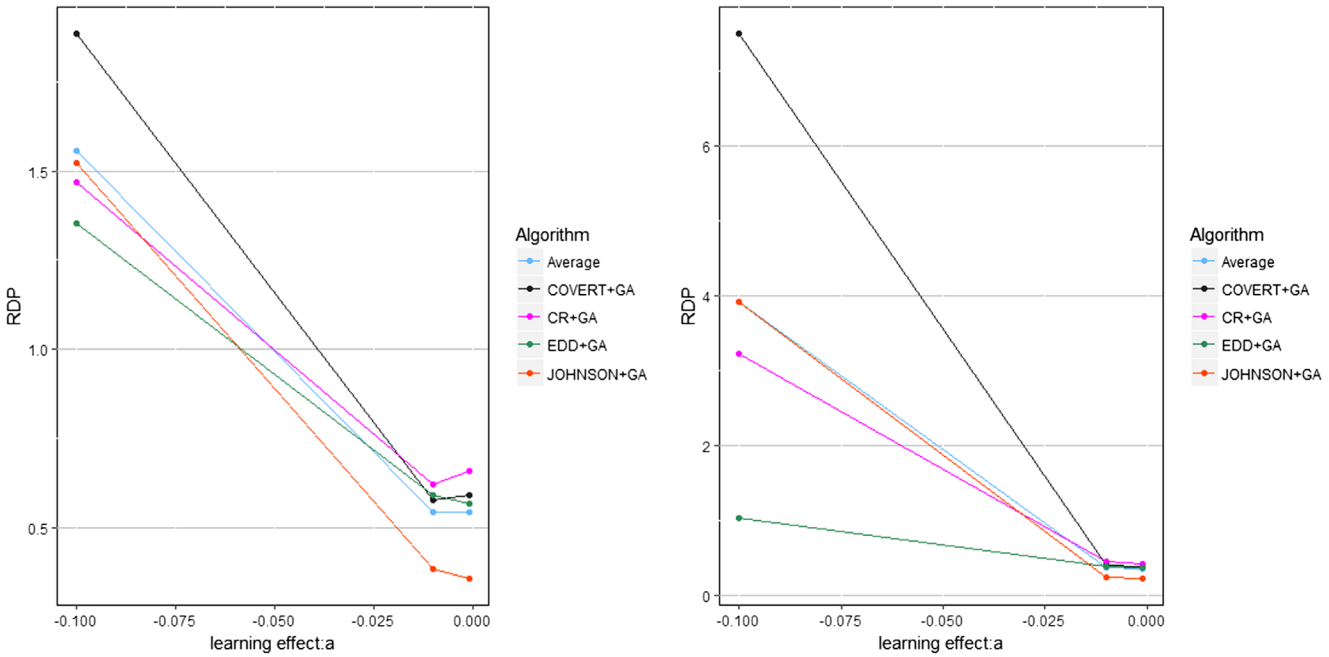



**Figure 9.** The performance of four algorithms when $a$ changes (n = 40, left, and n = 80).

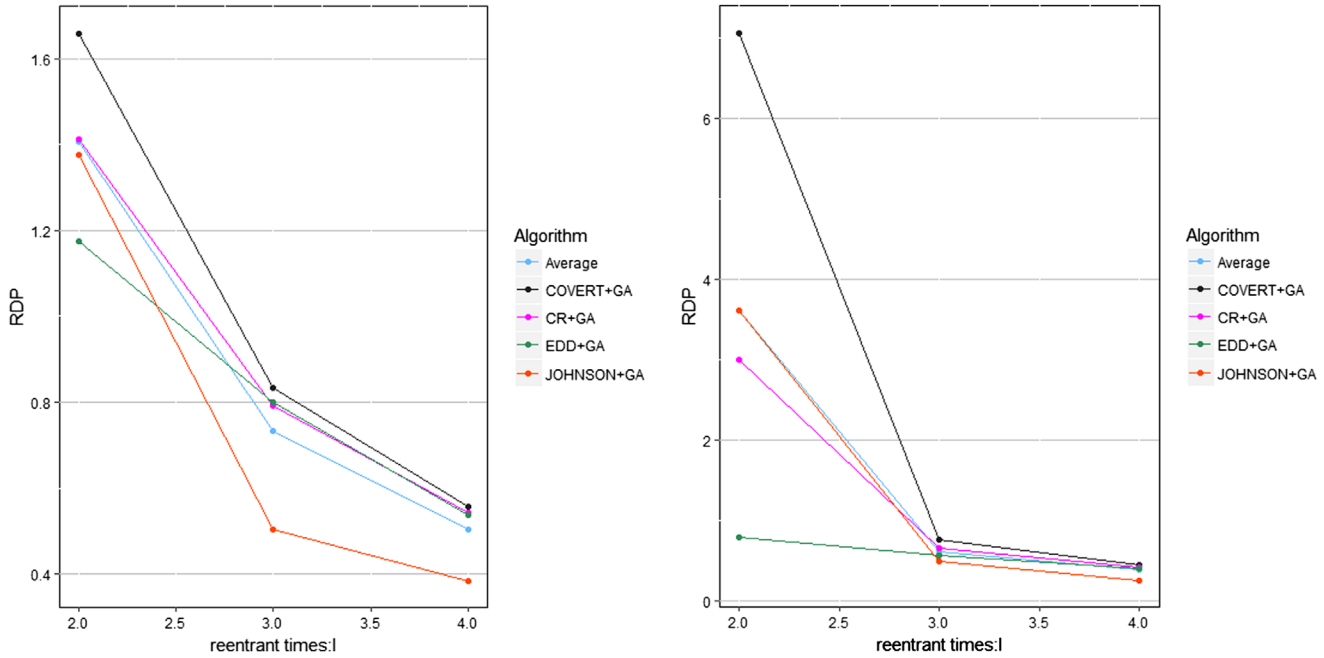



**Figure 10.** The performance of four algorithms when *l* changes (n = 40, left, and n = 80).

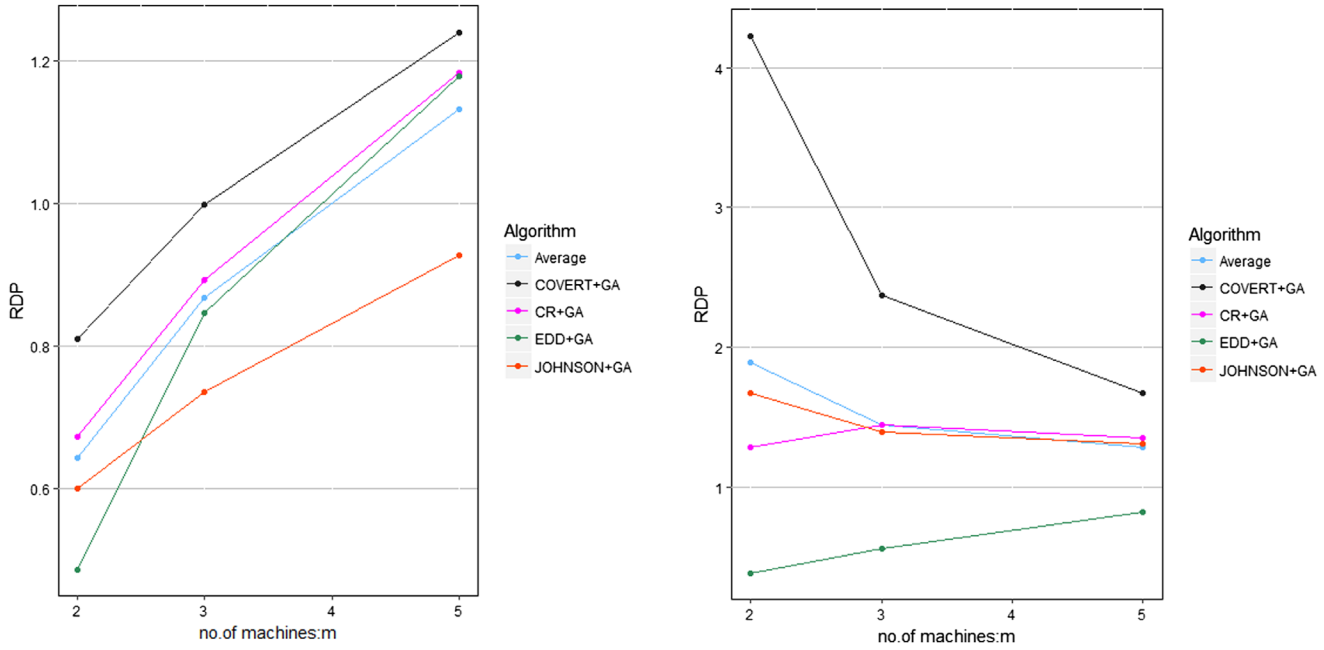



**Figure 11.** The performance of four algorithms when *m* changes (n = 40, left, and n = 80).

Over all, it was easily seen that the HOHNSON+GA performed the best and the algorithm COVERT+GA generated the worst solutions among the four algorithms. The differences of mean were 0.1128 for job size 8 and 0.1513 for job size 10.

Furthermore, to compare the statistical significant differences in solution quality of the four GA algorithms, we conducted an ANOVA for the small size of jobs ($n$ = 8 and $n$ = 10), the $p$-value is 0.0001. There exist significant differences between AEP means of 4 GAs under the level of significance 0.05. Accordingly, Fisher's least significant difference (LSD) test was employed to detect the differences between four AEP means. The result was shown in Table 4 that JOHNSON+GA performed the best among the four algorithms, for small size of jobs, was confirmed by the statistical test.

## 5. Four Heuristic-based Genetic Algorithms on the Large Number of Jobs

For the large-sized problem, the number of jobs $n$ is 40 and 80, we measured the performance of a heuristic algorithm in terms of the relative deviation percentage (RDP), defined as:

$$\text{RDP} = \frac{V - V_{min}}{V_{min}} \times 100\%,$$

Where $V$ is the objective value obtained by a heuristic and $V_{min}$ is the best solution among the four algorithms, COVERT+GA, CR+GA, EDD+GA and JOHNSON+GA. The experimental results were summarized in Table 4.

As Table 5 indicated that the overall means were (in an increasing order) 0.7548, 0.8372, 0.9159, and 1.0165 for JOHNSON+GA, EDD+GA, CR+GA, and COVERT+GA, respectively, for job sizes 40. Like the small jobs case, the JOHNSON+GA performed the best among the four algorithms. But for job size 80, the EDD+GA performed the best. The overall means were (in an increasing order) 0.5914, 1.3615, 1.4588, and 2.7590 for EDD+GA, CR+GA, JOHNSON+GA, and COVERT+GA, respectively. For all test cases (job sizes 8, 10, 40, and 80), the COVERT+GA performed overall the worst among the four algorithms.

Table 5 presents the performance trend for the algorithms in association with the parameters tardiness factor (τ), due date range ($R$), learning effect ($a$), the number of machines ($m$), and the number of times of reentrant ($l$). For clarifications, 5 figures were given. Figure 5 revealed; as tardiness factor (τ) increases, the performance of all four algorithms decrease for job size 40, on the contrary, as tardiness factor (τ) increases, the performance of all algorithms increase, except the algorithm EDD+GA, for job size 80.

Figures 8, 9 and 10 told us as the due date range ($R$), learning effect ($a$), and the number of times of reentrant ($l$) decrease the performance of all four algorithms generally decrease for both job sizes 40 and 80.

Regarding the number of machines ($m$), Figure 11 indicates that as the number of machines increase the performance of all four algorithms increase for job size 40 and the performance of all algorithms decrease, except the algorithm EDD+GA which increase, for job size 80.

In summary, for the large job sizes, it seemed the algorithm EDD+GA performed better. However, we conducted another ANOVA for the large size of jobs ($n$ = 40 and $n$ = 80), the $p$-value is 0.0934. The differences between RPD means of 4 GAs are not significant under the level of significance 0.05, although the means are 1.8877, 1.1387, 1.1068 and 0.7143 for COVERT+GA, CR+GA, JOHNSON+GA and EDD+GA, respectively.

## 6. Conclusions and Suggestions

In this study we addressed the re-entrant permutation flowshop scheduling problem with learning consideration to minimize the total tardiness. Given the intractability of the problem, we built four heuristic search algorithms in addition to a NEH-type locally improvement method to approximately solve the problem. Moreover, in order to find good quality near-optimal solutions, we applied a genetic algorithm (GA) by embedding the solutions obtained by the heuristics as initial solutions to solve the problem. The computational results showed that the GA algorithm accompanied with Johnson's rule performed best in terms of solution quality for the small-sized problem ($n$ = 8, 10) and job size 40. While the GA algorithm accompanied with EDD performed best for the large-sized problems ($n$ = 80). Future research may consider including both the due dates and jobs ordered by different agents or using non-linear optimization methods to solve the problem.

## Acknowledgements

We are grateful to the Editor and three anonymous referees for their many helpful comments on an earlier version of our paper. This paper was supported in part by the Ministry of Science Technology (MOST) of Taiwan under grant numbers MOST105-2221-E-035-053-MY3 and NSC 102-2221-E-035-070-MY3.

## Disclosure statement

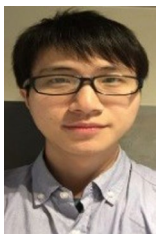
No potential conflict of interest was reported by the authors.

## Notes on contributors

***Win-Chin Lin*** is an associate professor in the Department of Statistics, Feng Chia University, Taiwan. He received a doctoral degree from the Department of Statistics, Iowa State University, USA, in 1998. His teaching and research interests include applied statistics, experimental designs, and scheduling.

***Chin-Chia Wu*** is a professor in the Department of Statistics, Feng Chia University, Taiwan. He received a doctoral degree from the Graduate Institute of Management, School of Management, National Taiwan University of Science and Technology, Taiwan in 1997. His teaching and research interests include applied statistics and operations research.

***Kejian Yu*** is a professor in the School of Management, Zhejiang Shuren University, China. He received a Master's degree from the Graduate Institute of Management, School of Management, Daebul University, Korea in 2005. His teaching and research interests include applied economics and operations research.



***Yong-Han Zhuang*** is a soft engineering operator in ASE GROUP. He received a bachelor's degree and B.S. degree from Department of Statistics and Graduate Institute of Statistics and Actuarial Science, Feng Chia University, Taiwan, in 2011 and 2013. His research interests include applied statistics and operations research.

***Shang-Chia Liu*** is an associate professor in the Department of Business Administration at Fu Jen Catholic University, Taiwan. He has obtained an MBA and MS degree in Information. He received the Ph.D. degree in Business Administration from Fu Jen Catholic University. His teaching and research interests include production scheduling, and database marketing.

## References

Alfieri, A. (2009). Workload simulation and optimization in multi-criteria hybrid flowshop scheduling: A case study. *International Journal of Production Research, 47*, 5129–5145.

Bellman, R., & Ernest, R. (1982). *Mathematical aspects of scheduling and applications*. Oxford: Pergamon Press.

Bengu, G. (1994). A simulation-based scheduler for flexible flowlines. *International Journal of Production Research, 32*, 321–344.

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research, 115*, 173–178.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effect. *European Journal of Operational Research, 188*, 315–329.

Bispo, C.F., & Tayur, S. (2001). Managing simple re-entrant flow lines: Theoretical foundation and experimental results. *IIE Transaction, 33*, 609–623.

Boudhar, M., & Meziani, N. (2010). Two-stage hybrid flow shop with recirculation. *International Transactions in Operational Research, 17*, 239–255.

Chen, J.-S. (2006). A branch-and-bound procedure for the re-entrant permutation flow-shop scheduling problem. *International Journal of Advanced Manufacturing Technology, 29*, 1186–1193.

Chen, C.-L., Vempati, V.S., & Aljaber, N. (1995). An application of genetic algorithms for flow shop problems. *European Journal of Operational Research, 80*, 389–396.

Chen, J.-S., Pan, J.C.H., & Wu, C.K. (2007). Minimizing makespan in re-entrant flow-shops using hybrid tabu search. *International Journal of Advanced Manufacturing Technology, 34*, 353–361.

Cheng, M. (2013) Flowshop scheduling problems with a position-dependent exponential learning effect. *Mathematical Problems in Engineering, 2013*: Article ID 753123, 5 pages, https://doi.org/10.1155/2013/753123.

Cheng, T.C.E., & Wang, G. (2000). Single machine scheduling with learning effect considerations. *Annals of Operations Research, 98*, 273–290.

Cheng, T.C.E., Wu, C.-C., Chen, J.-C., Wu, W.-H., & Cheng, S.-R. (2013). Two-machine flowshop scheduling with a truncated learning function to minimize the makespan. *International Journal of Production Economics, 141*, 79–86.

Choi, S.-W. & Kim, Y.-D. (2009). Minimizing total tardiness on a two-machine re-entrant flowshop. *European Journal of Operational Research, 199*, 375–384.

Chu, F., Chu, C., & Desprez, C. (2010). Series production in a basic re-entrant shop to minimize makespan or total flow time. *Computers & Industrial Engineering, 58*, 257–268.

Chung, Y.-H. & Tong, L.-I. (2012). Bi-criteria minimization for the permutation flowshop scheduling problem with machine-based learning effects. *Computers & Industrial Engineering, 63*, 302–312.

Etiler, O. & Toklu, B. (2001). Comparison of genetic crossover operators using in scheduling problems. *Journal Inst. Technology, Gazi University, Turkey, 14*, 21–32.

Etiler, O., Toklu, B., Atak, M., & Wilson, J. (2004). A generic algorithm for flow shop scheduling problems. *Journal of Operations Research Society, 55*, 830–835.

Falkenauer, E., Bouffoix, S. (1991). A genetic algorithm for job shop. Proceedings of the 1991 IEEE International Conference on Robotics and Automation.

Higgins, P., Le Roy, P., & Tierney, L. (1996). *Manufacturing planning and control-beyond MRP II*. London: Chapman & Hall.

Janiak, A., Krysiak, T., & Trela, R. (2011). Scheduling problems with learning and ageing effects: A survey. *Decision Making in Manufacturing, 5*, 19–36.

Koulamas, C. (1994). The total tardiness problem: Review and extensions. *Operation Research, 42*, 1025–1041.

Kubiak, W., Lou, S.X.C., & Wang, Y. (1996). Mean flow time minimization in re-entrant job-shops with a hub. *Operations Research, 44*, 764–776.

Kuo, W.-H., Hsu, C.-J., & Yang, D.-L. (2012). Worst-case and numerical analysis of heuristic algorithms for flowshop scheduling problems with a time-dependent learning effect. *Information Sciences, 184*, 282–297.

Lin, D. & Lee, C.K.M. (2011). A review of the research methodology for the re-entrant scheduling problem. *International Journal of Production Research, 49*, 2221–2242.

Liu, C.-H. (2010). A genetic algorithm based approach for scheduling of jobs containing multiple orders in a three-machine flowshop. *International Journal of Production Research, 48*, 4379–4396.

Nadler, D. & Smith, W.D. (1963). Manufacturing progress functions for types of processes. *International Journal of Production Research, 2*, 115–135.

Nawaz, M., Enscore, E.E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine, *n*-job sequencing problem. *Omega, 11*, 91–95.

Pan, J.C.H., & Chen, J.-S. (2003). Minimizing makespan in re-entrant permutation flow-shops. *Journal of the Operational Research Society, 54*, 642–653.

Rau, H., Cho, K.-H. (2009). Genetic algorithm modeling for the inspection allocation in re-entrant production systems. *Expert Systems with Application, 36* (Compendex): 11287–11295.

Sule, D.R. (1997). *Industrial scheduling*. Boston, MA: PWS Publishing Company.

Uzsoy, R., Lee, C.Y., & Martin-Vega, L.A. (1992). A review of production planning and scheduling models in the semiconductor industry—Part 1: System characteristics, performance evaluation and production planning. *IIE Transaction, 24*, 47–60.

Vargas-Villamil, F.D., & Rivera, D.E. (2001). A model predictive control approach for real-time optimization of re-entrant manufacturing lines. *Computers in Industry, 45*, 45–57.

Wang, J.B. (2010). Single-machine scheduling with a sum-of- actual-processing- time- based learning effect. *Journal of the Operational Research Society, 61*, 172–177.

Wang, J.-B., & Guo, Q. (2010). A due-date assignment problem with learning effect and deteriorating jobs. *Applied Mathematical Modelling, 34*, 309–313.

Wang, J.B. & Wang, M.Z. (2011). Worst-case analysis for flow shop scheduling problems with an exponential learning effect. *Journal of the Operational Research Society, 63*, 130–137.

Wang, J.B. & Xia, Z.Q. (2005). Flow-shop scheduling with a learning effect. *Journal of the Operational Research Society, 56*, 1325–1330.

Wang, M.Y., Sethi, S.P., & van de Velde, S.L. (1997). Minimizing makespan in a class of reentrant shops. *Operations Research, 45*, 702–712.

Wang, X.-Y., Zhou, Z., Zhang, X., Ji, P., & Wang, J.-B. (2013). Several flow shop scheduling problems with truncated position-based learning effect. *Computers & Operations Research, 40*, 2906–2929.

Wu, C.-C. & Lee, W.C. (2009). A note on the total completion time problem in a permutation flowshop with a learning effect. *European Journal of Operational Research, 192*, 343–347.

Wu, C.-C., Yin, Y.Q., Cheng, T.C.E., Cheng, Y., Liu, S.-Y., Lin, W.-C. (2016). Re-entrant flowshop scheduling with learning considerations to minimize the makespan. *Iranian Journal of Science and Technology, Transactions A: Science*, doi: 10.1007/s40995-017-0236-7

Xu, Y., Li, K., Hu, J., & Li, K. (2014). A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Information Sciences, 270*, 255–287.

Xu, J., Lin, W.-C., Wu, J., Cheng, S.-R., Wang, Z.-L., & Wu, C.-C. (2016). Heuristic based genetic algorithms for the re-entrant total completion time flowshop scheduling with learning consideration. *The International Journal of Computational Intelligence Systems, 9*, 1082–1100.

Yang, S.-J. & Yang, D.-L. (2012). Scheduling problems with past-sequence-dependent delivery times and learning effects. *Journal of the Operational Research Society, 63*, 1508–1515.

Yang, S.-J., Hsu, C.-J., & Yang, D.-L. (2010). Parallel-machine scheduling with setup and removal times under consideration of the learning effect. *Journal of the Chinese Institute of Industrial Engineers, 27*, 372–378.

Yelle, L.E. (1979). The learning curve: Historical review and comprehensive survey. *Decision Science, 10*, 302–328.

Yin, N. & Wang, X.-Y. (2011). Single machine scheduling with controllable processing times and learning effect. *International Journal of Advanced Manufacturing Technology, 54*, 743–748.

Yin, Y., Xu, D., Sun, K., & Li, H. (2009). Some scheduling problems with general position-dependent and time-dependent learning effects. *Information Sciences, 179*, 2416–2425.

Yin, Y., Xu, D., & Wang, J. (2010a). Some single-machine scheduling problems past-sequence-dependent setup times and a general learning effect. *The International Journal of Advanced Manufacturing Technology, 48*, 1123–1132.

Yin, Y., Xu, D., & Wang, J. (2010b). Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect. *Applied Mathematical Modelling, 34*, 3623–3630.

Yin, Y., Xu, D., & Wang, J. (2010). Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect. *Applied Mathematical Modelling, 34*, 3623–3630.

Zhang, X., Liu, S.C., Yin, Y., Wu, C.-C. (2016). Single-machine scheduling problems with a learning effect matrix. *Iranian Journal of Science and Technology, Transactions A: Science* doi:10.1007/s40995-016-0080-1.