Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 3630869, 9 pages
https://doi.org/10.1155/2017/3630869

Research Article

Synergy of Genetic Algorithm with Extensive Neighborhood Search for the Permutation Flowshop Scheduling Problem

Rong-Chang Chen,[1] Jeanne Chen,[2] Tung-Shou Chen,[2] Chien-Che Huang,[2] and Li-Chiu Chen[2]

[1]*Department of Distribution Management, National Taichung University of Science and Technology, Taichung 404, Taiwan*
[2]*Department of Computer Science and Information Engineering, National Taichung University of Science and Technology, Taichung 404, Taiwan*

Correspondence should be addressed to Chien-Che Huang; charles0928@gmail.com

Received 19 August 2016; Revised 24 December 2016; Accepted 17 January 2017; Published 13 February 2017

Academic Editor: Jean J. Loiseau

The permutation flowshop scheduling problem (PFSP) is an important issue in the manufacturing industry. The objective of this study is to minimize the total completion time of scheduling for minimum makespan. Although the hybrid genetic algorithms are popular for resolving PFSP, their local search methods were compromised by the local optimum which has poorer solutions. This study proposed a new hybrid genetic algorithm for PFSP which makes use of the extensive neighborhood search method. For evaluating the performance, results of this study were compared against other state-of-the-art hybrid genetic algorithms. The comparisons showed that the proposed algorithm outperformed the other algorithms. A significant 50% test instances achieved the known optimal solutions. The proposed algorithm is simple and easy to implement. It can be extended easily to apply to similar combinatorial optimization problems.

1. Introduction

The permutation flowshop scheduling problem (PFSP) [1–7] is the combination of n jobs and m machines of production scheduling problems. Each job must be processed on m machines. All jobs are processed in the same sequence on each machine, but with different processing times. Each machine is identical and can only process one job at a time, and the processing cannot be interrupted or preempted. Specifically, let p_{ij} denote the processing time of job i on Machine j, and let P denote a permutation of the n jobs. Then the jobs are processed consecutively on Machine 1 in the order given by P, with no delays between the executions of consecutive jobs. The jobs are processed in the same order on Machine 2, but each job must wait until it is finished processing on Machine 1 so that it can be started on Machine 2. Similarly, a job must wait to be processed on Machine j until it has finished processing on Machine $j-1$.

Subject to the foregoing stipulations, find a permutation P that minimizes the maximum makespan, that is, the time to compete the processing of the last job in P on the last Machine m. By the preceding definition, PTSP has $n!$ possible schedules. With an increase in the number of jobs and machines, the complexity of PTSP grows exponentially. Garey et al. [8] have proven that PFSP belonged to an NP-complete problem for $m \geqq 3$. Thus, in recent years, approximation methods have been popularly used to solve PFSP. One such method is the metaheuristic algorithms [9]. In metaheuristic algorithms, studies are focused on the hybrid genetic algorithms which combine genetic algorithms and local search methods. The performance of hybrid genetic algorithms, or memetic algorithms (MAs) [10], is affected by local search methods. However, the local search methods are easily trapped in the local optima which increase the difficulty to find global optima.

This study proposes a new hybrid genetic algorithm (GA_ENS algorithm) which combines genetic algorithm and extensive neighborhood search. The diversity of the local search is thus significantly increased. When the GA_ENS algorithm explores the neighborhood of the current solution,

the neighborhood will be dynamically changed to prevent redundant searches in the same region. This increases the probability of finding the global optimum.

In this study, we compared the GA_ENS algorithm with two other algorithms which belonged to state-of-the-art hybrid genetic algorithms for PFSP. One is the HGA_RMA algorithm proposed by Ruiz et al. [11] and the other is the NEGAVNS algorithm proposed by Zobolas et al. [12]. The two algorithms used different local search methods.

The remainder contents of this paper are organized as follows: Section 2 is a brief overview of the related algorithms for PFSP. Section 3 is the detailed description of this study proposing the new hybrid genetic algorithm. Section 4 is the experimental results and analyses. Finally, conclusions are presented in Section 5.

2. Related Work

In 1954, Johnson [13] proposed Johnson's rule to solve two-machine flowshop scheduling problem. As a result, much research had been focused on resolving the different flowshop scheduling problems. Currently, the m-machine ($m > 2$) PFSP is the main trend of research.

Reza Hejazi and Saghafian [14] presented a complete review of flowshop scheduling problems with makespan objectives from 1954 to 2005. They reviewed various production scheduling problems of flowshops, including PFSP.

PFSP belongs to an NP-complete problem [8]. Therefore, approximation methods are applied to solve PFSP. However, the approximation methods cannot provide optimal solutions. They can only provide acceptable solutions within a reasonable time range. Approximate methods can generally be classified as heuristic algorithms and metaheuristic algorithms [15].

The design core of the heuristic algorithms is based on experiences. In other words, the design of the algorithms relies heavily on understanding the problems and experiences. Heuristic algorithms can solve problems quickly and obtain reasonable solutions within reasonable time. However, they cannot guarantee the same quality solution in different data. The greatest drawback is that they cannot guarantee global and consistent optimal solutions.

The heuristic algorithms can mainly be divided into two categories: constructive methods and improvement methods [15]. The constructive methods construct solutions from scratch according to some special rules and can provide solutions rather quickly. However, their solution quality is not guaranteed, examples of which include Palmer's heuristic method [16], CDS heuristic [17], rapid access (RA) [18], and NEH heuristic [19]. NEH heuristic is the best algorithm in the constructive methods [15, 20]. NEH is a quick local search for minimum makespan. For the n jobs ($n!$), for the first-level current permutation, the kth job which minimizes the partial makespan is inserted. The sequence is calculated with $n(n+1)/2-1$. Taillard's insertion moves jobs by new neighborhoods $N(s)$ where s is the best solution. Current research has been focused on improving the NEH mechanisms which also involved the hybrid designs [21, 22]. The improvement methods are also called local search methods. They can provide good solutions. However, they are time-consuming. The basic design idea of the improvement methods is to improve the initial solution by some specific rules and expect to obtain the better solutions, examples of which are RACS and RAES [18].

In the last 20 years, much work has been done on the relatively new type approximate metaheuristic algorithms which are also known as modern heuristics. The algorithms inject the probability concept into the process of solving problems. Compared with the heuristic algorithms, metaheuristic algorithms require more time for getting the solutions. However, they resulted in higher quality solutions than heuristic algorithms [15].

In order to enhance the quality of solutions in metaheuristic algorithms, algorithms must be designed with the balance between diversification and intensification. Diversification belongs to the capability of global search and maintaining the population diversity; it can explore all areas of the search space. Intensification belongs to the capability of local search; it can exploit the neighborhood of the current solution and find a local optimum.

Metaheuristic algorithms can be classified as trajectory methods and population-based methods according to the number of solutions used in the search process [9]. Trajectory methods generate a feasible solution in each iteration and attempt to find the best solution along the trajectory of the search space, such as simulated annealing (SA) [23–28], Tabu search (TS) [29–34], iterated local search (ILS) [9, 15], and variable neighborhood search (VNS) [9, 35]. Population-based methods generate a set of feasible solutions to perform parallel search in each generation and get the best solution after iterative evolution, such as genetic algorithms (GAs) [36–40], ant colony optimization (ACO), and particle swarm optimization (PSO).

3. A New Hybrid Genetic Algorithm

Traditional genetic algorithms utilize the population to execute multiple points search using the genetic operators. As a result, they have the capability of global search and population diversity. In order to enhance the efficiency of local search for obtaining better quality solutions, this study proposes a new hybrid genetic algorithm: the GA_ENS algorithm (Figure 1). The difference between the GA_ENS algorithm and traditional genetic algorithms is that the GA_ENS algorithm added an operator of extensive neighborhood search, which can enhance the force of intensification; we named this operator "extensive neighborhood search operator" (ENS operator). In the following, we will describe all operators of the GA_ENS algorithm.

3.1. Encoding. In accordance with the definition of PFSP and the objective of this study, the permutation encoding was used to design chromosomes in the GA_ENS algorithm. Every chromosome consisted of genes, which the number of jobs. The order of genes of a chromosome was the processing sequence of all jobs; in other words, jobs was processed

Table 1: An instance of PFSP ($n = 3, m = 3$).

	Job 1	Job 2	Job 3
Machine 1	2	5	4
Machine 2	4	3	6
Machine 3	3	2	2

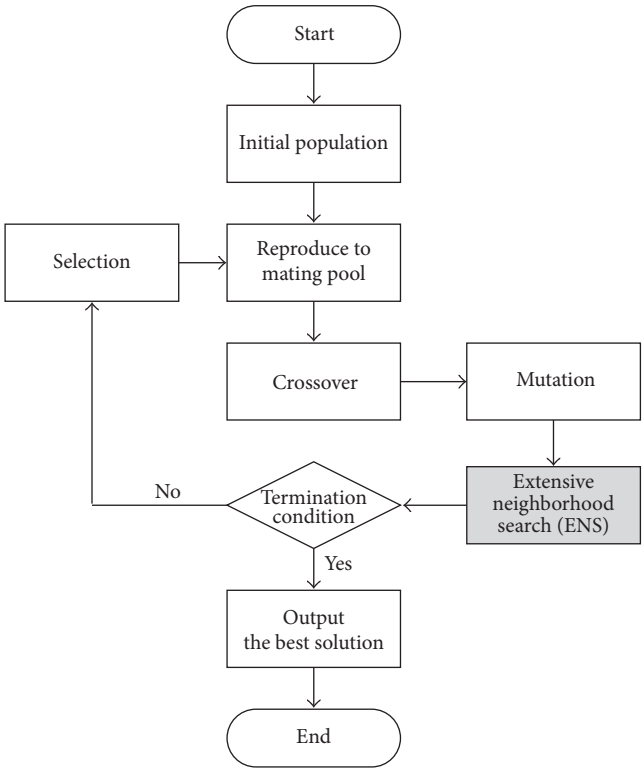

Figure 1: The GA_ENS algorithm flowchart.

according to the order of chromosome encoding until all jobs had been processed.

We give an instance of PFSP in Table 1: the number of jobs is 3 ($n = 3$) and the number of machines is also 3 ($m = 3$). Each number in Table 1 denotes the processing time of each job on each machine. To assume that a chromosome representation is $\boxed{1\ 2\ 3}$, the processing sequence of this chromosome is "Job 1 → Job 2 → Job 3" according to the encoding of the GA_ENS algorithm.

The objective of this study is to find the minimum makespan ($C_{\max}$); the makespan denotes the total completion time of all jobs that have been processed. Algorithm 1 is the pseudocode for makespan calculation: n is the total number of jobs; m is the total number of machines; span $[j]$ denotes the accumulative processing time on the jth machine (default value = 0); p_{ij} denotes the processing time of the job of the ith position in the chromosome on Machine j. The computational result of the above chromosome makespan is 19 (see Table 2 and Figure 2).

3.2. Reproduction. The initial population of the test instance in the GA_ENS algorithm must be generated randomly before

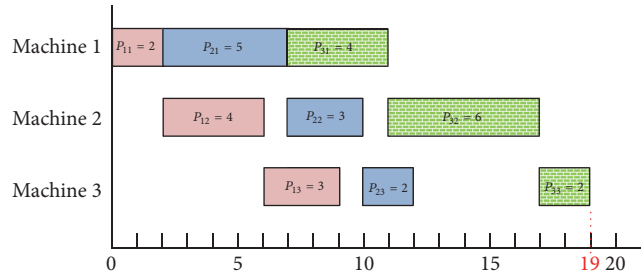

Figure 2: Gantt chart of "Job 1 → Job 2 → Job 3" from Table 1.

```
for i = 1 to n
   for j = 1 to m
      if (j = 1) then
         span[j] = span[j] + p_ij
      else
         span[j] = max(span[j − 1], span[j]) + p_ij
      end if
   end for
end for
C_max = span[m]
```

Algorithm 1: Pseudocode for makespan calculation.

evolution. The population consisted of multiple chromosomes according to population size, which is a parameter. A chromosome is usually called individual, and an individual is a solution of PFSP, namely, a point in the search space.

The GA_ENS algorithm used multiple solutions to execute multiple points search in the search space and is expected to find the high quality solutions. First, the GA_ENS algorithm picked chromosomes and reproduced them to the mating pool. Second, it proceeded to evolve these chromosomes through the crossover and mutation operators that simulated biological evolution. Next, it utilized the ENS operator to improve quality of solutions. Finally, the GA_ENS algorithm utilized the selection operator to select better quality chromosomes into the next evolutionary generation. Before evolution of each generation, the total number of chromosomes called population size was kept the same and obtained the best solution which was an optimal schedule through iterative evolution until a termination condition was reached.

In order to maintain the population diversity, the mechanism of pick at random was adopted in the reproduction operator of this study. In other words, multiple pairs of chromosomes (called parents) were selected and reproduced to the mating pool for breeding new chromosomes (called offspring) in the next stage.

The total number of reproductions in this stage was the number of population sizes multiplied by crossover rate. After the above stage, the successive evolution stages continued.

3.3. Crossover. The main duty of the crossover operator was to generate the offspring from parents of the mating pool by mating. The purpose of this stage was that the offspring

Table 2: Calculate the makespan of "Job 1 → Job 2 → Job 3" from Table 1.

	i = Job 1	i = Job 2	i = Job 3
j = 1 (Machine 1)	span[1] = 0 + 2 = 2	span[1] = 2 + 5 = 7	span[1] = 7 + 4 = 11
j = 2 (Machine 2)	span[2] = 2 + 4 = 6	span[2] = 7 + 3 = 10	span[2] = 11 + 6 = 17
j = 3 (Machine 3)	span[3] = 6 + 3 = 9	span[3] = 10 + 2 = 12	span[3] = 17 + 2 = 19

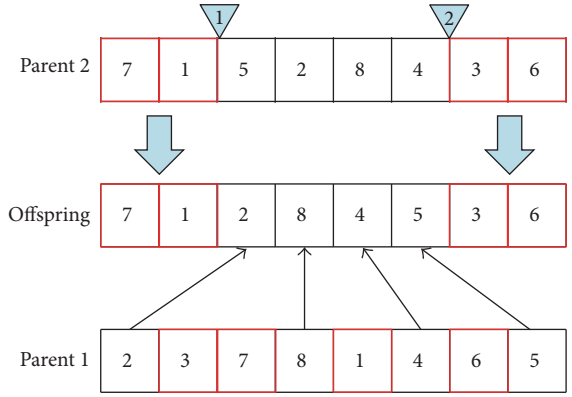

Figure 3: Two-point crossover.

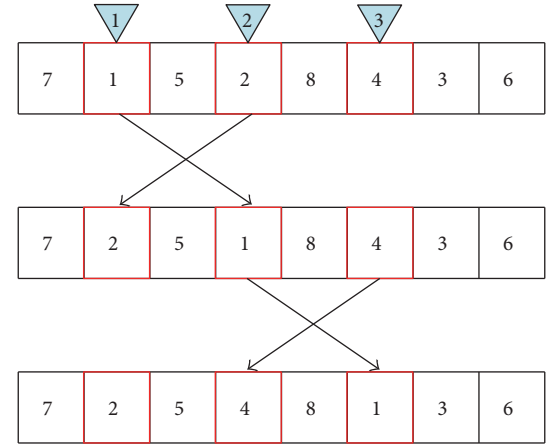

Figure 4: 3-position change mutation; **7 2 5**.

can obtain good genes from parents by simulating biological evolution and can obtain better quality solutions by the interchange of information. The difference between offspring and parents can guide the search to different regions and lead to the search regions expanded. This study decided to use the two-point crossover (Figure 3) because Murata et al. [41] proved that this two-point crossover was effective for PFSP.

3.4. Mutation. The mutation operator mutated genes of selected chromosomes self in order to enhance the population diversity and completely search the entire search space. If the mutation operator cleverly cooperates with the crossover operator, the premature convergence to the local optimum can be avoided and can obtain better quality solutions.

In addition, we need to set the mutation rate for deciding the total number of randomly selected chromosomes to undergo the mutation operator. For example, we assume the number of chromosomes after the crossover operator was performed is 15 and the mutation rate is 0.2; according to our mutation mechanism, there are 3 chromosomes ($15 \times 0.2 = 3$) that will be randomly selected to perform the mutation operator. The mutation mechanism used in this study was the 3-position change mutation (see Figure 4).

3.5. Extensive Neighborhood Search (ENS). The design concepts of the framework of the local search in this study were inspired by variable neighborhood search (VNS) [9, 35] and iterated greedy algorithm [6]. We integrated the above two algorithms and our designs into the proposed local search method. The searched neighborhood by this method will be dynamically changed with the perturbation of the current solution, so we named this method "extensive neighborhood search" (ENS).

```
function ENS(s)
  s_b = s
  for k = 1 to k_max
    s' = Shaking(s_b)
    N(s') = Insertion-Move(s')
    s'' = BestOf(N(s'))
    if C_max(s_b) > C_max(s'') then
      s_b = s''
    end if
  end for
  if C_max(s) > C_max(s_b) then
    s = s_b
    call ENS(s)
  end if
```

Algorithm 2: Pseudocode for ENS operator.

The neighborhood structure must be efficiently defined for the enhancement of searching efficiency. The Insertion-Move [20] was used to construct neighborhood structures in this study.

The probability of local search was set to 1; namely, all offspring which were produced by the crossover and mutation operators must undergo the ENS operator. The steps of the ENS operator are described as follows (Algorithm 2 and Figure 5 are the procedure of the ENS operator).

Step 1. This is the initial step. Let the best solution (s_b) = the current solution (s) and the number of perturbation (k) = 1.

Step 2. Execute the shaking of perturbation mechanism. The NEH heuristic [19] is used to reconstruct a new solution

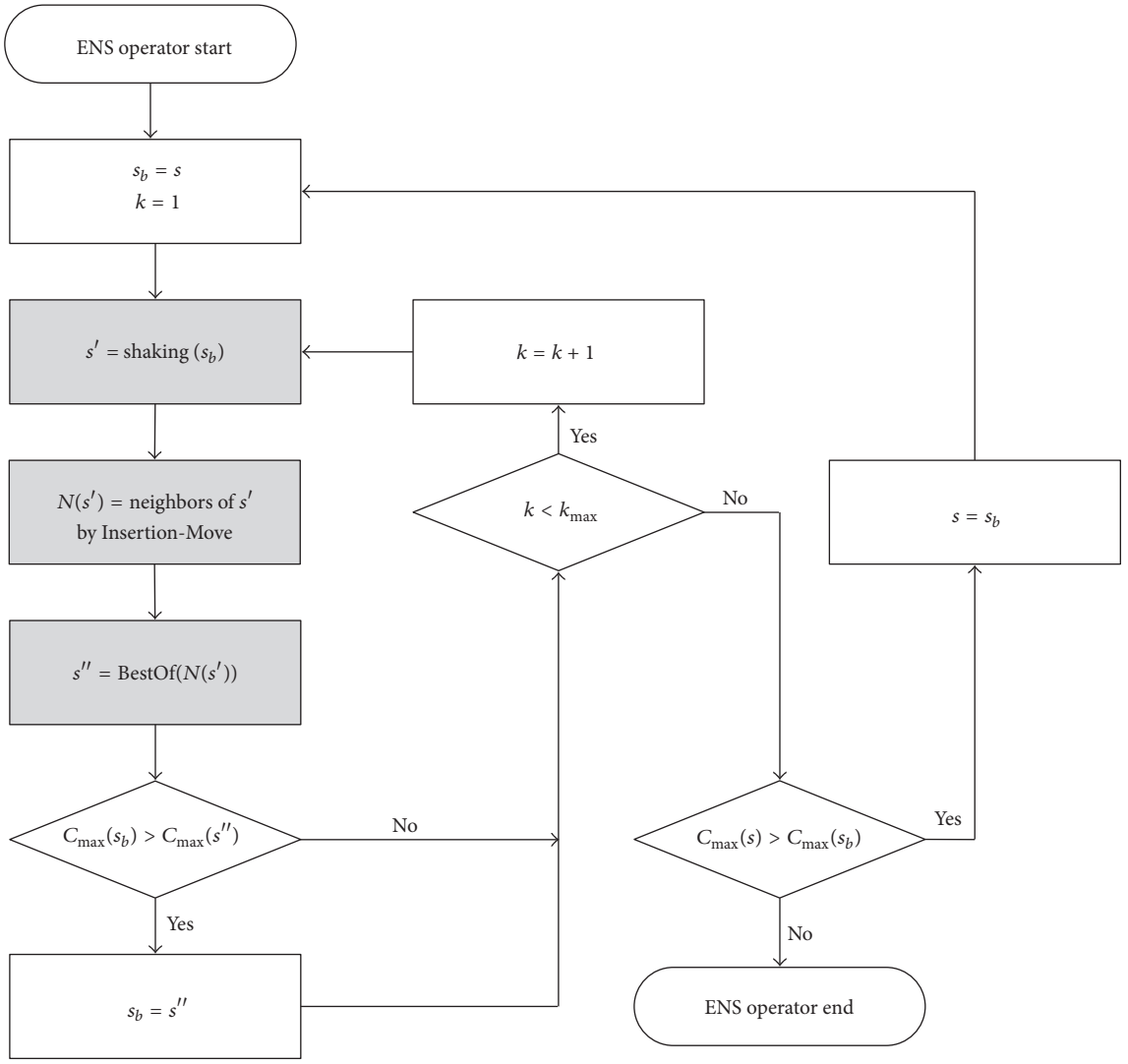

Figure 5: ENS operator flowchart.

(s_b is converted to s') through the destruction and construction phases [6].

Step 3. Construct the neighborhood $N(s')$ of s' by the Insertion-Move [20].

Step 4. s'' is the best neighbor of the neighborhood $N(s')$: $s'' = \text{BestOf}(N(s'))$.

Step 5. If $C_{\max}(s_b) > C_{\max}(s'')$, then $s_b = s''$; namely, the best solution needs to be updated when better solution is found.

Step 6. Let $k = k + 1$ and go to Step 2 when $k < k_{\max}$ ($k_{\max}$ is the maximum number of perturbations); on the contrary, go to Step 7.

Step 7. If $C_{\max}(s) > C_{\max}(s_b)$, then $s = s_b$ and go to Step 1; on the contrary, end the ENS operator.

3.6. Selection. After the crossover, mutation, and ENS operators were performed in each generation, a new population was selected into the next generation before a termination condition had been reached, called the selection operator. This selection operator belonged to the evolutional selection, and its duty was to select chromosomes for the next generation evolution.

The selection operator can guide the search to the correct direction and lead to the fact that the quality of solutions can be continuously improved. In order to maintain the population diversity and promote the evolution, the selection mechanism of this study was the tournament selection.

In the tournament selection, comparing each chromosome of the offspring with a randomly selected chromosome from the parents, the chromosome with lower makespan (better quality) was selected into the next generation. The above process was repeated until every chromosome of the offspring had performed the tournament selection.

For maintaining the population diversity and avoiding losing particular chromosomes which can guide the search to the correct direction, the acceptance criterion [24, 25] was applied to inferior offspring to decide whether to accept or reject them into the next generation.

The acceptance criterion (see (1)) proposed by Osman and Potts [27] for PFSP was used in this study; the inferior offspring could be accepted into the next generation when (2) [6] is met.

$$\text{Temperature} = T \times \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij}}{n \times m \times 10}. \quad (1)$$

$$\text{AR}_{\text{random}} \leq \exp\left\{-\frac{(\Delta C_{\max})}{\text{Temperature}}\right\}. \quad (2)$$

In (1), the parameter T is 0.4 suggested by Ruiz and Stützle [6]; n is the total number of jobs; m is the total number of machines; P_{ij} denotes the processing time of the job i on Machine j ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$).

In (2), $\text{AR}_{\text{random}}$ is a random number drawn uniformly from $[0, 1]$; $\Delta C_{\max}$ denotes $C_{\max}$(offspring) − $C_{\max}$(parent); temperature is the value of (1).

Based on the above description, there are two situations in the tournament selection:

(1) An offspring chromosome is better than a parent chromosome: the offspring chromosome replaces the parent chromosome into the next generation.

(2) An offspring chromosome is worse than a parent chromosome: when (2) is met, the offspring chromosome will replace the parent chromosome into the next generation.

4. Experimental Results and Analyses

The GA_ENS algorithm was implemented in Microsoft Visual C++, running on an Intel Pentium 2.5 GHz PC with 1 GB of main memory.

Test datasets of all the experiments in this study were taken from Taillard's benchmark [42, 43]. It includes 12 sets of different PFSP problems, and each set is composed of 10 different instances. So Taillard's benchmark has 120 different PFSP instances altogether.

All the experiment results in this study need to be normalized for fair comparison, converting the makespan values obtained by the GA_ENS algorithm to the relative percentage deviation (RPD) values with the following equation:

$$\text{RPD} = \frac{\text{GA\_ENS}_{\text{sol}} - \text{Known}_{\text{best}}}{\text{Known}_{\text{best}}} \times 100. \quad (3)$$

In (3), $\text{GA\_ENS}_{\text{sol}}$ denotes the makespan value obtained by the GA_ENS algorithm; $\text{Known}_{\text{best}}$ denotes the known optimal makespan value of Taillard's benchmarks [43]. According to the above equation, the quality of the solution is better when this solution has a lower RPD value.

An experiment of termination condition was firstly executed: the test datasets were 120 different PFSP instances of Taillard's benchmark, and the total number of generations was 2000. To observe this experimental result (see Figure 6), the improvement range of RPD is very little or nothing after 500 generations; therefore, we decided that the termination condition of each follow-up experiment with the GA_ENS algorithm was 500 generations.

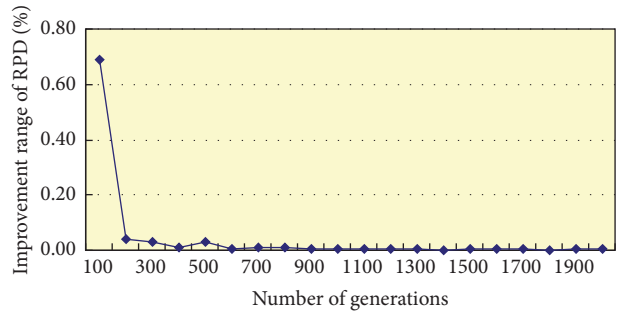

Figure 6: Experiment of termination condition.

We proceeded to execute an experiment of parameters which included population size, crossover rate, mutation rate, and the maximum number of perturbations in the ENS operator. Several values of these parameters were set as follows based on related experimental experiences:

(1) Population size: 10, 20, and 30.

(2) Crossover rate: 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0.

(3) Mutation rate: 0.1, 0.2, 0.3, and 0.4.

(4) The maximum number of perturbations: 5, 10, and 15.

The 216 combinations ($3 \times 6 \times 4 \times 3 = 216$) of the aforementioned parameters were tested with 9 instances of Taillard's benchmark which consisted of $n = \{20, 50, 100\}$ and $m = \{5, 10, 15, 20\}$. The termination condition of this experiment was 500 generations and it was repetitively executed 5 times. According to this experimental result, we can obtain the best parameter combination which has the lowest value of the average RPD:

$$\text{Average RPD} = \frac{\sum_{R=1}^{5} \left(\left(\text{GA}_{\text{ENS}_{\text{solR}}} - \text{Known}_{\text{best}}\right)/\text{Known}_{\text{best}} \times 100\right)}{5}. \quad (4)$$

In (4), $\text{GA}_{\text{ENS}_{\text{solR}}}$ denotes the makespan value obtained by the GA_ENS algorithm at a time experiment; $\text{Known}_{\text{best}}$ denotes the known optimal makespan value of Taillard's benchmarks [43]; R is the number of execution times and the total number is 5 in this study.

Four best parameter values were obtained from this experiment, shown as follows:

(1) Population size: 30.

(2) Crossover rate: 0.8.

(3) Mutation rate: 0.2.

(4) The maximum number of perturbations: 15.

Table 3: The experimental results (average RPD) of GA_ENS (5 runs).

Instance		Run 1	Run 2	Run 3	Run 4	Run 5	Average
n	m						
20	5	0.00	0.00	0.00	0.00	0.00	0.00
20	10	0.00	0.00	0.00	0.00	0.00	0.00
20	20	0.00	0.00	0.00	0.00	0.00	0.00
50	5	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.34	0.30	0.40	0.37	0.37	0.35
50	20	0.64	0.50	0.49	0.39	0.58	0.52
100	5	0.00	0.00	0.00	0.00	0.00	0.00
100	10	0.03	0.05	0.03	0.09	0.05	0.05
100	20	0.91	0.82	0.82	0.89	0.82	0.85
200	10	0.04	0.03	0.03	0.04	0.04	0.04
200	20	1.00	0.95	0.99	1.02	0.96	0.99
500	20	0.38	0.41	0.41	0.44	0.41	0.41
Average		*0.279*	*0.255*	*0.265*	*0.270*	*0.269*	*0.268*

Table 4: The comparison of GA_ENS with other algorithms (average RPD).

Instance		HGA_RMA	$NEGA_{VNS}$	GA_ENS (without ENS)	GA_ENS
n	m				
20	5	0.04	0.00	1.40	0.00
20	10	0.02	0.01	3.45	0.00
20	20	0.05	0.02	2.73	0.00
50	5	0.00	0.00	1.46	0.00
50	10	0.72	0.82	6.43	0.35
50	20	0.99	1.08	8.04	0.52
100	5	0.01	0.00	0.92	0.00
100	10	0.16	0.14	4.29	0.05
100	20	1.30	1.40	9.27	0.85
200	10	0.14	0.16	3.28	0.04
200	20	1.26	1.25	9.12	0.99
500	20	0.69	0.71	6.87	0.41
Average		*0.448*	*0.466*	*4.773*	*0.268*

This completes setting the values of all operators and parameters in the GA_ENS algorithm. The next step is to execute the main experiment of the GA_ENS algorithm (see Table 3). The makespan values from the GA_ENS algorithm are converted to the RPD in (3) and the average RPD in (4) for comparison tests with the two selected algorithms [11, 12]. When a solution has a lower RPD value than other solutions, this solution is closer to the known optimal solution. This means that the quality of this solution is better than other solutions.

In Table 3, n is the total number of jobs; m is the total number of machines. Table 3 presents the average experimental results from 12 sets of different PFSP problems, and each set is composed of 10 different instances, so the instances total to 120 different PFSP instances from Taillard's benchmark.

Table 4 presents the comparison of GA_ENS with other algorithms: the test datasets include 12 sets of different PFSP problems from Taillard's benchmark; n is the total number of jobs; m is the total number of machines. The main compared algorithms were described in Section 1 of this paper, namely, the HGA_RMA algorithm [11] and the $NEGA_{VNS}$ algorithm [12].

The "GA_ENS (without ENS)" in Table 4 denotes that the GA_ENS algorithm without local search (i.e., local search probability = 0), so as to observe the performance of the ENS operator. The values of GA_ENS (without ENS) are the average experimental results after 5 runs.

In Table 3, the average RPD in 5 runs of the GA_ENS algorithm is 0.268 and the standard deviation is about 0.009. According to the above results, the quality and stability of solving PFSP of the GA_ENS algorithm both are good, so it is a stable and effective algorithm.

When $n = 20$ or $m = 5$ in Table 3, all the test instances can achieve the known optimal solutions. In addition, some instances of other sets of different problems can achieve the

known optimal solutions, so about 50% of the test instances can achieve the known optimal solutions.

In Table 4, firstly to observe the results between the GA_ENS algorithm and the GA_ENS (without ENS) algorithm, the former is far better than the latter (0.268 versus 4.773) so that the proposed ENS operator can enhance the quality of solutions. Next, compared with other algorithms (the HGA_RMA algorithm [11] and the NEGA_{VNS} algorithm [12]), the comparison results showed that the proposed GA_ENS algorithm outperformed compared algorithms and can effectively improve the probability of finding known optimal solutions.

5. Conclusions

This study proposes a GA_ENS algorithm for the PFSP, which combines two design strategies: the genetic algorithm (the kernel algorithm) and the extensive neighborhood search mechanism (namely, ENS operator). The genetic algorithm is responsible for the diversification force, and the ENS operator controls the intensification force. They are coordinately important and complementary to each other. For Taillard's benchmark, the experimental results show that about 50% of the test instances can achieve the known optimal solutions, small instances especially. And then, in the inferior instances, the difference in makespan between the inferior quality instances and the known optimal solutions can be reduced to within 1%. For the aforementioned reasons, the GA_ENS algorithm can effectively solve the PFSP.

Competing Interests

The authors declare that they have no competing interests.

References

[1] H. Abedinnia, C. H. Glock, and A. Brill, "New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem," *Computers and Operations Research*, vol. 74, pp. 165–174, 2016.

[2] M. Fatih Tasgetiren, Q.-K. Pan, P. N. Suganthan, and O. Buyukdagli, "A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem," *Computers and Operations Research*, vol. 40, no. 7, pp. 1729–1743, 2013.

[3] V. Fernandez-Viagas and J. M. Framinan, "On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 45, pp. 60–67, 2014.

[4] W. B. Liu, "A hybrid differential evolution algorithm based on dynamic variable neighborhood search for permutation flowshop scheduling problem," *Applied Mechanics and Materials*, vol. 835, pp. 847–857, 2016.

[5] B. Naderi and R. Ruiz, "A scatter search algorithm for the distributed permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 239, no. 2, pp. 323–334, 2014.

[6] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[7] D. R. Sule, *Industrial Scheduling*, PWS Publishing, Boston, Mass, USA, 1996.

[8] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[9] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[10] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[11] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.

[12] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *Computers & Operations Research*, vol. 36, no. 4, pp. 1249–1267, 2009.

[13] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[14] S. Reza Hejazi and S. Saghafian, "Flowshop-scheduling problems with makespan criterion: a review," *International Journal of Production Research*, vol. 43, no. 14, pp. 2895–2929, 2005.

[15] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, vol. 165, no. 2, pp. 479–494, 2005.

[16] D. S. Palmer, "Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum," *Journal of the Operational Research Society*, vol. 16, no. 1, pp. 101–107, 1965.

[17] H. G. Campbell, R. A. Dudek, and M. L. Smith, "A heuristic algorithm for the n job, m machine sequencing problem," *Management Science*, vol. 16, no. 10, pp. B630–B637, 1970.

[18] D. G. Dannenbring, "An evaluation of flow shop sequencing heuristics," *Management Science*, vol. 23, no. 11, pp. 1174–1182, 1977.

[19] M. Nawaz, E. E. Enscore Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[20] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 65–74, 1990.

[21] X. Dong, H. Huang, and P. Chen, "An improved NEH-based heuristic for the permutation flowshop problem," *Computers & Operations Research*, vol. 35, no. 12, pp. 3962–3968, 2008.

[22] P. J. Kalczynski and J. Kamburowski, "An improved NEH heuristic to minimize makespan in permutation flow shops," *Computers & Operations Research*, vol. 35, no. 9, pp. 3001–3008, 2008.

[23] H. Ishibuchi, S. Misaki, and H. Tanaka, "Modified simulated annealing algorithms for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 81, no. 2, pp. 388–398, 1995.

[24] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[25] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[26] F. A. Ogbu and D. K. Smith, "The application of the simulated annealing algorithm to the solution of the $n/m/C_{max}$ flowshop problem," *Computers and Operations Research*, vol. 17, no. 3, pp. 243–253, 1990.

[27] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[28] S. H. Zegordi, K. Itoh, and T. Enkawa, "Minimizing makespan for flow shop scheduling by combining simulated annealing with sequencing knowledge," *European Journal of Operational Research*, vol. 85, no. 3, pp. 515–531, 1995.

[29] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[30] F. Glover, "Tabu search—part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[31] F. Glover, "Tabu search—part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[32] J. Grabowski and M. Wodecki, "A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion," *Computers & Operations Research*, vol. 31, no. 11, pp. 1891–1909, 2004.

[33] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flow-shop problem," *European Journal of Operational Research*, vol. 91, no. 1, pp. 160–175, 1996.

[34] M. Widmer and A. Hertz, "A new heuristic method for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 41, no. 2, pp. 186–193, 1989.

[35] P. Hansen and N. Mladenović, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[36] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich , USA, 1975.

[37] R.-C. Chen and T.-T. Hu, "A decision-making mechanism considering carbon footprint and cost to fulfil orders for multi-site global companies," *International Journal of Shipping and Transport Logistics*, vol. 7, no. 3, pp. 295–318, 2015.

[38] R.-C. Chen and P.-H. Hung, "Multiobjective order assignment optimization in a global multiple-factory environment," *Mathematical Problems in Engineering*, vol. 2014, Article ID 673209, 14 pages, 2014.

[39] C.-L. Chen, V. S. Vempati, and N. Aljaber, "An application of genetic algorithms for flow shop problems," *European Journal of Operational Research*, vol. 80, no. 2, pp. 389–396, 1995.

[40] M. Gen, R. Cheng, and L. Lin, *Network Models and Optimization: Multi-Objective Genetic Algorithm Approach*, Springer, Berlin, Germany, 2008.

[41] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.

[42] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

[43] E. Taillard, "Summary of Best Known Lower and Upper Bounds of Taillard's Instances," http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html.