An Improved Multiobjective Evolutionary Algorithm for Solving the No-Wait Flow Shop Scheduling Problem

Tsung-Su Yeh and Tsung-Che Chiang
Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan
anikiboy@gmail.com, tcchiang@ieee.org

Abstract - **The no-wait flow shop extends the classical flow shop by considering a practical constraint (in steel, plastic, and several industries) that operations of each job should be processed continuously on machines. In this paper, we propose to use a multiobjective evolutionary algorithm based on decomposition (MOEA/D) for no-wait flow shop scheduling with minimization of makespan and maximum tardiness as two objectives. First, we propose a crossover operator that inherits gene blocks with smaller machine idle time from parent solutions. Second, we investigate the effects of initial population by using different job ordering rules. Third, we generate ninety problem instances and conduct experiments on these instances. Experimental results confirm that our idle-time-based crossover and multi-rule initialization lead to good solution quality. We make all data of problem instances and sets of solutions publicly accessible to promote future research on this topic.**

Keywords – **no-wait, flow shop, multiobjective, evolutionary algorithms, idle time**

<< This paper is included in the Proceedings of the International Conference on Industrial Engineering and Engineering Management (IEEM 2018), held in Singapore during Dec. 16-19, 2018. >>

I. INTRODUCTION

Since no-wait scheduling problems were proposed in 1970s [1], the no-wait flow shop scheduling problem (NWFSP) has been one of the problems that researchers endeavored to solve. The main feature in the no-wait shop is that operations of each job should be processed without interruptions on machines. Thus, the first operation of a job might be scheduled to start late to avoid waiting time of the following operations. Previous studies usually considered only a single objective, such as minimization of makespan. In this paper, we aim to minimize makespan and maximum tardiness simultaneously in a Pareto manner. A solution X *dominates* a solution Y when X is not worse than Y in all objectives and X is not equal to Y. When no solution can dominate a solution X, it is called *Pareto optimal*. The *Pareto set* collects all Pareto optimal solutions, and the *Pareto front* refers to the projection of the Pareto set into the objective space. Our goal of solving the multiobjective NWFSP (MONWFSP) is to find the Pareto set.

In the NWFSP, there are n jobs and m machines. The processing time of job j on machine k is denoted by $p(j, k)$. Each job has to be processed by machine 1, machine 2, …, and finally by machine m. Given a permutation $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ of n jobs, the completion time $C(j, k)$ of job j on machine k is defined by formula (1)-(3).

$$C(\pi_1, 1) = p(\pi_1, 1), \tag{1}$$
$$C(\pi_j, 1) = C(\pi_{j-1}, 1) + e(\pi_{j-1}, \pi_j) + p(\pi_j, 1),\ j = 2, \ldots, n, \tag{2}$$
$$C(\pi_j, k) = C(\pi_j, k-1) + p(\pi_j, k),\ j = 1, \ldots, n, k = 2, \ldots, m, \tag{3}$$

where $e(\pi_{j-1}, \pi_j)$ denotes the required delay time between the first operations of jobs π_{j-1} and π_j to keep continuously processing operations of job π_j. Based on the completion time and due dates of jobs, makespan and maximum tardiness are defined as (4) and (5), where $d(\pi_j)$ denotes the due date of job π_j.

$$f_1(\pi) = C_{max}(\pi) = C(\pi_n, m), \text{ and} \tag{4}$$
$$f_2(\pi) = T_{max}(\pi) = \mathbf{max}_{j=1}^{n}\{\max\{0, C(\pi_j, m) - d(\pi_j)\}\}. \tag{5}$$

Below is an example of a NWFSP with 4 jobs and 4 machines. Let the permutation π be $\{1, 2, 3, 4\}$. The minimal idle time $e(1, 2)$ is 0, $e(2, 3)$ is 1, and $e(3, 4)$ is 2. Fig. 1 illustrates the derived schedule. The makespan $C_{max}(\pi)$ is 11, and the maximum tardiness $T_{max}(\pi)$ is 3.

$$(p(j,k))_{4x4} = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 2 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (d(j))_{4x1} = \begin{bmatrix} 4 \\ 7 \\ 7 \\ 10 \end{bmatrix}$$

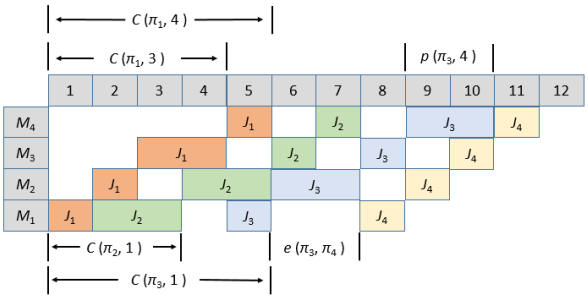

Fig. 1: A 4-job, 4-machine schedule in an NWFSP

Our approach follows the framework of the multiobjective evolutionary algorithm based on decomposition (MOEA/D) [2], which is a well-known algorithm in the field of evolutionary multiobjective optimization (EMO). We design the problem-specific operators – initialization, crossover, and mutation, carefully. We investigate different job ordering rules to generate the initial population; we incorporate machine idle time (a useful but rarely considered factor) in the crossover operator; we utilize the mutation operator to maintain population diversity.

In the rest of this paper, we review related studies in Section 2. Section 3 details the proposed algorithm. Experiments and results are given in Section IV. Section V concludes the paper and lists future work.

II. LITERATURE REVIEW

A. *Metaheuristic Algorithms*

The NWFSP is known to be an NP-hard problem when there are more than two machines [3]. It means that enumeration methods could not handle large-size problems effectively. In the literature, heuristics and metaheuristics showed their potential in solving the NWFSP. Metaheuristics including particle swarm optimization (PSO) [4][5], tabu search (TS) [6], simulated annealing (SA) [7], variable neighborhood search (VNS) [8], genetic algorithm (GA) [8], and differential evolution (DE) [9] have been applied to the problem.

Most of the studies focused on the single objective NWFSP. One of the earliest studies on MONWFSP was proposed by Allahverdi and Aldowaisan in 2002 [10]. Later, Allahverdi and Aydilek [11] proposed an algorithm to minimize total completion time under an upper bound of makespan. Pan *et al*. [9] presented a discrete DE algorithm. Pan *et al*. [4] presented a multi-objective PSO for minimization of makespan and maximum tardiness.

In the field of evolutionary computation, EMO is an important and popular research area. Multiobjective evolutionary algorithms aim to approximate the Pareto front. NSGA-II [12], MOPSO [4] and MODE [9] are dominance-based algorithms. They use the Pareto dominance relation as the primary criteria to select individuals. MOEA/D [2], on the other hand, belongs to decomposition-based algorithms. It evaluates individuals by scalarizing functions with uniform weight vectors of objectives. NSGAII and MOEA/D are both representative EMO algorithms.

B. *Initialization Procedures*

The evolutionary algorithm starts from an initial population and then repeats the variation and selection steps to evolve the solutions (to make them better). In practice, the quality of the initial solutions has significant effects on the results of the evolutionary process. The NEH heuristic, proposed by Nawaz *et al*. [13], is a famous heuristic to generate a schedule with the goal of minimizing makespan in the flow shop. The jobs are sorted in the non-increasing order of total processing time (also known as the longest-processing-time, LPT, rule). Then, the jobs are taken one by one in this order, and each job is inserted into the best position in the partial schedule. Specifically speaking, when the j^{th} job is to be inserted, we test j possible positions in the sequence of the previous $(j-1)$ jobs and then insert it into the position that results in the smallest makespan. Based on this sort-and-insert-greedily procedure, Ding *et al*. [6] proposed a modified NEH (MNEH), in which the jobs were sorted in the non-increasing order of the standard deviation of operation processing times over m machines. They found that the MNEH outperforms the original NEH by 0.35%.

Pan *et al*. [5] proposed an initialization method based on the nearest neighbor (NN) heuristic [14] and the NEH heuristic. The NN heuristic was originally used for the traveling salesman problem, and it constructs the route by repeatedly appending the customer with the shortest distance to the last customer to the partial route. Pan *et al*. followed this concept to construct the job permutation by appending the next job with the minimal required idle time on the first machine to the partial schedule. Then, the greedy insertion procedure of NEH constructs a schedule based on this permutation. Given n jobs, each job was taken as the first job to start the NN heuristic. In other words, NN generated n permutations for NEH to

generate n schedules accordingly. The NN heuristic was also used by Lin and Ying [15] in their initialization method. They repeatedly appended the next job that leads to the minimal makespan to the end of job permutation. Then, the Kernighan-Helsgaun (LKH) [16] algorithm was used to improve the permutation. Pan *et al*. [4] proposed a new multi-objective heuristic, named the PWQ heuristic. We will briefly describe it in Section III-C.

C. Crossover Operators

Crossover is the major operator to generate new solutions in the evolutionary algorithm. Many crossover operators have been proposed to solve permutation problems [17]. Jarboui *et al*. [8] proposed a new crossover operator called "Block Order Crossover (BOC)". They thought that identical consecutive permutation blocks of parents are better to be preserved, regardless of the block positions. Except BOC, most of the crossover operators exchange job sub-sequences randomly. In our crossover, we will consider the machine idle time required by the job sub-sequences and keep the sub-sequence with the shortest idle time.

III. THE PROPOSED ALGORITHM

A. Overview

To solve the MONWFSP, we propose an algorithm based on MOEA/D [2]. Using MOEA/D, a multiobjective problem is decomposed into many single-objective sub-problems. Each sub-problem i is associated with a weight vector λ^i and an individual X_i in the population. The individual X_i is responsible for solving the sub-problem i by optimizing the scalarizing function (e.g. weighted sum) controlled by the weight vector λ^i . A set of evenly distributed weight vectors is generated, and solutions to these sub-problems are collected to be the approximation of the Pareto set of the original multiobjective problem. The second key concept of MOEA/D is the neighborhood-based selection. The top $|B(i)|$ sub-problems with the smallest Euclidean distance $d(\lambda^i, \lambda^j)$ are included in the neighborhood $B(i)$ of a sub-problem i. Mating selection is carried out only in the neighborhood. We use an external population (EP) to store the non-dominated solutions found during the evolutionary process. Table I gives the pseudo code of the proposed algorithm.

TABLE I: PSEUDO CODE OF THE PROPOSED ALGORITHM

Initialization
1.1 set EP = ∅
1.2 generate weight vectors and set neighborhood
1.3 generate the initial population: mixed PWQ heuristic

Update
2.1 mating selection
2.2 reproduction: ITX crossover operator
2.3 mutation: swap or insert
2.4 replacement
2.5 duplicate elimination
2.6 update EP
Repeat 2.1~ 2.6 until the stopping criterion is satisfied

B. Encoding

The first task to solve a problem by metaheuristics is to represent solutions of the problem as chromosomes. We adopt the popular encoding scheme to represent a schedule as a permutation of jobs. The permutation denotes the processing order of the jobs on machines.

C. Initialization

We apply four rules together with the earliest-due-date (EDD) rule in the framework of the PWQ heuristic [4] to generate the initial population. The four rules include:

Rule 1 (LPT): It sorts jobs in the non-increasing order of total processing time [13].
Rule 2 (RND-NEH): It takes a random job permutation to run the NEH heuristic to generate a job sequence [8].
Rule 3 (STD): It sorts jobs in the non-increasing order of the standard deviation of processing times over machines [6].
Rule 4 (NN): It arranges jobs by the NN heuristic considering machine idle time [5][15].

The above four rules aim to minimize makespan, and the EDD rule aims to minimize maximum tardiness. We generate the initial job permutations for the NEH heuristic based on the weighted sum $s(j) = \lambda_1^i \cdot pos_{rulei}(j) + \lambda_2^i \cdot pos_{EDD}(j)$ of the position values of job j in the result of rule i (i = 1, 2, 3, and 4) and in the result of EDD rule, where $\lambda^i = (\lambda_1^i, \lambda_2^i)$ are the weight vectors defined in MOEA/D.

Identical solutions might be generated, and we remove the duplicates by applying the mutation operator by k times where k is a random number between one and five. After the whole population is generated, we associate each sub-problem (and its weight vector) with the individual that has the best performance (i.e. that leads to the minimal weighted Tchebycheff value, detailed in Section III-F and eq. (6)). Finally, unassociated weight vectors are assigned to unassociated individuals randomly.

D. Mating Selection

Given N sub-problems (and also N individuals in the population), we do crossover on two parents to generate an offspring for each sub-problem i. One parent is the individual X_i of sub-problem i, and the other parent is a random individual in the neighborhood $B(i)$.

E. Crossover

Traditional crossover operators usually exchange random gene blocks between parents. In this paper, we incorporate the domain knowledge in the operator. Our operator, the idle-time-based crossover (ITX), considers the machine idle time of the inherited gene block (a job sub-sequence and also a partial schedule). The following procedure explains how we generate one offspring O_1. By exchanging the roles of parents P_1 and P_2, we can generate the offspring O_2.

Step 1. We select and evaluate two random sub-sequences with the same size from the parent P_1. The size is a random number between 1 and $n/2$. Taking Fig. 2 as an example and assuming the size is 4, we select one sub-sequence [2, 3, 4, 5] and the other sub-sequence [4, 5, 6, 7]. Then, we derive two (partial) schedules by these two sub-sequences and calculate the total idle times of the first machine in these two schedules.

Step 2. The sub-sequence that leads to a smaller total idle time (assume [4, 5, 6, 7] here) will be inherited by the offspring O_1. To increase diversity, this sub-sequence is put in a random position in O_1.

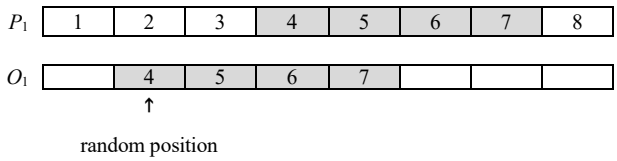

Fig. 2. The ITX crossover – inherit the gene block with a smaller machine idle time

Step 3. The missing gene values are inherited from the other parent O_2, as illustrated in Fig. 3.

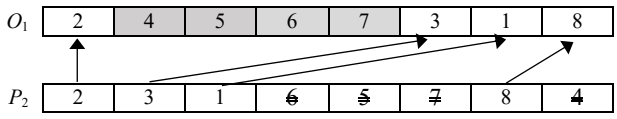

Fig. 3. The ITX crossover – inherit the missing gene values

F. Replacement

We use weighted Tchebycheff as the fitness function for sub-problems. The aggregated objective value $F_i(X)$ of an individual X with respect to the sub-problem i is defined by (6), where λ_1^i denotes the weight of the first objective (i.e. makespan in this paper), $f_j()$ denote the j^{th} objective function, and f_j^{min} and f_j^{max} denote the minimum and maximum values of the j^{th} objective over all individuals in the population, respectively. Normalization is carried out because the scale of the objective functions may differ a lot.

$$F_i(X) = \max\{\lambda_1^i \times \frac{f_1(X) - f_1^{min}}{f_1^{max} - f_1^{min}}, (1-\lambda_1^i) \times \frac{f_2(X) - f_2^{min}}{f_2^{max} - f_2^{min}}\} \quad (6)$$

After an offspring X_{new} is generated by the ITX operator, we calculate the aggregated objective values $F_i(X_{new})$ of X_{new} over all sub-problems i. We identify the sub-problem i^* that the offspring X_{new} performs the best by the minimal aggregated objective value, i.e. $i^* = arg\ min\{F_i(X_{new})\}$ for i in [1, N], where N is the population size. Then, the offspring X_{new} replaces the original individual X_i of sub-problem i if X_{new} has a smaller aggregated objective value than X_i.

G. Mutation and Duplicate Elimination

The role of mutation in an evolutionary algorithm is to maintain or increase the population diversity. Each offspring undergoes mutation in probability p_m. One of the insertion and swap operators is selected in equal probability and is applied by k times where k is a random number between one and five. When we found individuals with the same values of both objectives, the above procedure is applied, too.

IV. EXPERIMENTS AND RESULTS

A. Data Set

To test the performance of our proposed algorithm, we conducted experiments using Taillard's data set [18], which was popularly used in the literature. The data set consists of nine categories with different problem sizes ($n \times m$), where n ranges from 20 to 100 and m ranges from 5 to 20. Each category contains ten instances. Since the data set does not include job due dates, we generated due dates by the following way: first, generate a random job permutation and derive the corresponding schedule; second, set the due date of job i by $d(i) = C(i, m) + rand(-n, n)$, where $C(i, m)$ denotes the completion time of job i.

B. Performance Indicator：IGD

The inverted generational distance (IGD) [19] is a widely used performance indicator in the field of EMO. Given an approximation set of solutions A obtained by an algorithm and the reference set of solutions R, the IGD is calculated by (7):

$$IGD(A) = \frac{\sum_{Y \in R} \min_{X \in A} d(Y,X)}{|R|}, \quad (7)$$

where $d(Y, X)$ is the Euclidean distance between the objective vectors of solutions Y and X. Before calculating the IGD, the solution vectors are normalized so that the range of each objective is in [0, 1]. In the experiments, the reference front R is the set of the non-dominated solutions among all solutions obtained by all algorithm variants in Tables III-V. Each problem instance was solved by each algorithm variant by ten times. The average (AVG) and the standard deviation (STD) of IGD values over instances in each problem category are calculated and listed in Tables III-V. Since the IGD values of solutions to different problem instances might differ a lot, we follow the procedure in [19] to normalize the values.

C. Experimental Setting

Our algorithm has only a few parameters. We set their values by some pilot runs and previous experience. Table II summarizes the values.

TABLE II PARAMETER SETTING

Parameter	Value	Meaning
N	100	Population size
$\|B(i)\|$	20	Neighborhood size
p_m	0.6	Mutation rate
T	$10mn$	Computation time (10^{-3} s)

D. *Experiment I: Effect of the Initialization Methods*

In the first experiment, we want to examine the effect of the initial population (Section III-B) on the solution quality. We used the classical PMX operator here. In Table III we mark the best result (the minimal average IGD) among the four rules in bold. We can see that no single rule always performs the best. The last column shows the results of using four rules simultaneously. We mark the average IGD values in bold and italics if they are smaller than the best among the four sole rules. We can see that the use of multiple rules can lead to close or even better results than the use of sole rule, especially in solving larger-scale problem instances.

E. *Experiment II: Effect of the Crossover Operators*

The second experiment aims to examine whether the proposed ITX operator is useful. We compared the ITX operator with the PMX[17], and BOC [8] operators. Table IV summarizes the results. It shows that the MOEA/D+ITX variant outperforms the other two variants with PMX and BOC operators in all categories of problem instances.

F. *Experiment III: Effect of the Whole MOEA*

The last experiment examines the performance of the proposed MOEA. Since there is no public reference solutions for comparison, we took NSGA-II plus PMX as the benchmark algorithm. IGD results are summarized in Table V. Except the smallest-size problem category, the proposed MOEA/D-ITX leads to better performance than NSGA-II-PMX in eight out of nine problem categories. Performance difference is even larger for larger-scale problem categories. Fig. 4 shows the approximated front in a single run of solving problem instance TA90. The solutions of MOEA/D-ITX dominate those of NSGA-II-PMX.

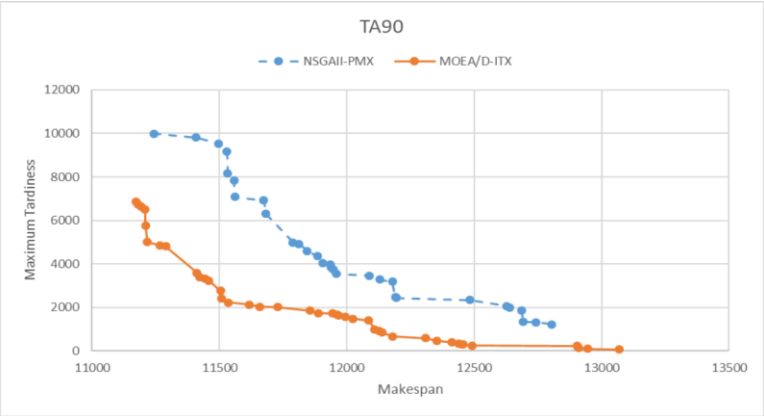

Fig. 4. Approximated front of MOEA/D-ITX and NSGAII-PMX by solving TA90

V. CONCLUSION

NWFSP is a challenging optimization problem and models the real condition in several industries. In this paper, we proposed an evolutionary algorithm to minimize makespan and maximum tardiness in a Pareto manner. Our algorithm is based on MOEA/D. We proposed a crossover operator by utilizing the machine idle time to identify good gene blocks. Through comprehensive experiments on a data set of ninety problem instances, we showed that the proposed crossover improves the solution quality. Besides, we also showed the potential benefit of multi-rule initialization and the use of MOEA/D framework. All data of the problem instances and reference solutions are available on the authors' website.

There are some limitations in this research and will be addressed in our future work. First, we will re-implement some state-of-the-art algorithms as benchmarks to verify the performance of our algorithm. Second, we will keep improving our algorithm by incorporating local search and speedy evaluation/decoding procedure. Third, we will do analysis to know more about the effects of parameter values.

REFERENCES

[1] A. Allahverdi, "An survey of scheduling problems with no-wait in process," *European Journal of Operational Research*, vol. 255, pp.665-686, 2016.

[2] Q. Zhang, and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol.11, no.6, pp. 712-731, 2007.

[3] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[4] Q. K. Pan, L. Wang, and B. Qian, "A novel multi-objective particle swarm optimization algorithm for no-wait flow shop scheduling problems," in *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 222, no. 4, pp. 519-539, 2008.

[5] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang. "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Computers & Operations Research*, vol. 35, no. 9, pp. 2807-2839, 2008.

[6] J. Y. Ding, S. Song, J. N. Gupta, R. Zhang, R. Chiong, and C. Wu, "An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem," *Applied Soft Computing*, vol. 30, pp. 604-613, 2015.

[7] J. Y. Ding, S. Song, R. Zhang, and C. Wu, "A novel block-shifting simulated annealing algorithm for the no-wait flowshop scheduling problem," in *Proceedings of the 2015 IEEE Congress on. Evolutionary Computation (CEC)*, pp. 2768-2774, 2015.

[8] B. Jarboui, M. Eddaly, and P. Siarry, "A hybrid genetic algorithm for solving no-wait flowshop scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 54, pp. 1129–1143, 2011.

[9] Q. K. Pan, L. Wang, and B. Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Computers & Operations Research*, vol. 36, no. 8, pp. 2498-2511, 2009.

[10] A. Allahverdi, and T. Aldowaisan, "No-wait flowshops with bicriteria of makespan and total completion time," *Journal of the Operational Research Society*, vol. 53, no. 9, pp. 1004-1015, 2002.

[11] A. Allahverdi and H. Aydilek, "Algorithms for no-wait flowshops with total completion time subject to makespan," *International Journal of Advanced Manufacturing Technology*, vol. 68, no. 9-12, pp. 2237-2251, 2013.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Myerarivan, "A fast and elitist multi-objective genetic algorithms: NSGA-II", *IEEE Transactions on Evolutionary Computation*, vol. 6, no.2, pp. 182-197, 2002.

[13] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, pp. 91-95, 1983.

[14] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6, no.3, pp. 563-581, 1977.

[15] S. W. Lin and K. C. Ying, "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics," *Omega*, vol. 64, pp. 115-125, 2016.

[16] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*. vol. 126, no. 1, pp. 106-130, 2000.

[17] T. Kellegöz, B. Toklu, and J. Wilson, "Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem," *Applied Mathematics and Computation*, vol. 199, no. 2, pp. 590-598, 2008.

[18] E. Taillard, "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, vol. 64, pp. 278-285, 1993.

[19] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review" *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117-132 Apr. 2003.

[20] Y. T. Chang, and T. C. Chiang, " Multiobjective permutation flow shop scheduling using MOEA/D with local search," in *Proceedings of the 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 262-269, 2016.

TABLE III. IGD VALUES OF INITIALIZATION METHODS OF THE MOEA/D

	Rule 1: LPT [13]		Rule 2: RND-NEH [4]		Rule 3: STD [6]		Rule4: NN [5][15]		Multi-Rule	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
20×5	**1.0915**	0.0548	1.1136	0.0882	1.1049	0.0528	1.1043	0.0651	1.0986	0.0634
20×10	1.0796	0.0283	1.0837	0.0298	1.0841	0.0236	**1.0751**	0.0265	1.0823	0.0274
20×20	1.0659	0.0163	1.0636	0.0150	1.0659	0.0165	**1.0603**	0.0184	1.0618	0.0203
50×5	**1.1005**	0.0433	1.1077	0.0690	1.1329	0.0730	1.1166	0.0735	1.1077	0.0656
50×10	1.0864	0.0278	**1.0859**	0.0200	1.1167	0.0443	1.0901	0.0198	***1.0843***	0.0170
50×20	**1.0963**	0.0280	1.0984	0.0370	1.0981	0.0387	1.1040	0.0363	1.0978	0.0317
100×5	**1.0852**	0.0263	1.1111	0.0647	1.1458	0.0600	1.0979	0.0457	1.1001	0.0416
100×10	1.0968	0.0258	**1.0954**	0.0309	1.1215	0.0460	1.1116	0.0299	***1.0862***	0.0283
100×20	**1.1221**	0.0292	1.1270	0.0471	1.1351	0.0471	1.1198	0.0363	***1.1154***	0.0426
AVG	**1.0916**	0.0311	1.0985	0.0446	1.1117	0.0447	1.0977	0.0391	1.0927	0.0375

TABLE IV IGD VALUES OF CROSSOVER OPERATORS OF THE MOEA/D

	BOC [4]		PMX [17]		ITX	
	AVG	STD	AVG	STD	AVG	STD
20×5	1.0899	0.0484	1.0986	0.0634	**1.0388**	0.0203
20×10	1.0798	0.0295	1.0823	0.0274	**1.0219**	0.0135
20×20	1.0650	0.0185	1.0618	0.0203	**1.0080**	0.0049
50×5	1.1331	0.0674	1.1077	0.0656	**1.0206**	0.0077
50×10	1.1113	0.0280	1.0843	0.0170	**1.0163**	0.0070
50×20	1.1223	0.0361	1.0978	0.0317	**1.0177**	0.0074
100×5	1.1172	0.0584	1.1001	0.0416	**1.0135**	0.0097
100×10	1.1297	0.0381	1.0862	0.0283	**1.0100**	0.0049
100×20	1.1494	0.0548	1.1154	0.0426	**1.0129**	0.0066
AVG	1.1109	0.0421	1.0927	0.0375	**1.0177**	0.0091

TABLE V. IGD VALUES OF THE MOEAD-ITX AND NSGAII-PMX ALGORITHMS

	NSGAII-PMX				MOEA/D-ITX			
	MIN	MAX	AVG	STD	MIN	MAX	AVG	STD
20×5	1.0000	1.0470	**1.0177**	0.0062	1.0199	1.0671	1.0388	0.0203
20×10	1.0049	1.0410	1.0227	0.0103	1.0071	1.0421	**1.0219**	0.0135
20×20	1.0077	1.0346	1.0199	0.0108	1.0009	1.0211	**1.0080**	0.0049
50×5	1.0402	1.1013	1.0717	0.0861	1.0008	1.0470	**1.0206**	0.0077
50×10	1.0624	1.0977	1.0808	0.0339	1.0000	1.0361	**1.0163**	0.0070
50×20	1.0927	1.1387	1.1163	0.0482	1.0000	1.0369	**1.0177**	0.0074
100×5	1.0578	1.0741	1.0670	0.0295	1.0000	1.0263	**1.0135**	0.0097
100×10	1.0722	1.0891	1.0818	0.0375	1.0000	1.0202	**1.0100**	0.0049
100×20	1.1098	1.1254	1.1189	0.0568	1.0000	1.0279	**1.0129**	0.0066
AVG	1.0498	1.0832	1.0663	0.0355	1.0032	1.0361	**1.0177**	0.0091