2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Minimizing makespan for a no-wait flowshop using Tabu Mechanism Improved Iterated Greedy Algorithm

Jianya Ding, Shiji Song, Rui Zhang, Cheng Wu

***Abstract*— This paper proposes a tabu mechanism improved iterated greedy (TMIIG) algorithm to solve the no-wait flowshop scheduling problem with makespan criterion. The motivation of seeking for further improvement in the iterated greedy (IG) algorithm framework is based on the observation that the construction phase of the original IG algorithm may lead to repeated search when applying the insertion neighborhood search. To overcome the drawback, we modified the IG algorithm by a tabu-based reconstruction strategy to enhance its exploitation ability. A powerful neighborhood search method which involves insert, swap, and double-insert moves is then applied to obtain better soluions from the reconstructed solution in the previous step. Numerical computations verified the advantages of utilizing the new reconstruction scheme. In addition, comparisons with other high-performing algorithms demonstrated the effectiveness and robustness of the proposed algorithm.**

I. Introduction

The no-wait flowshop scheduling problem (NWFSP) is an important scheduling problem that arises in many manufacturing industries. Examples include chemical processing [1], plastic molding [2], pharmaceutical processing [3] and steel rolling [4] industries. In a no-wait flowshop, n jobs are to be processed successively through m machines in a predetermined processing order. Typically, there should be no waiting period in between consecutive machines when a job is being processed. In this situation, the start of a job may get delayed to satisfy the no-wait constraint. A comprehensive survey on the research and application of no-wait flowshop scheduling problems is given in Ref. [2].

The problem has been addressed with respect to various performance measures, including makespan [5], total flowtime [6], maximum lateness [7], total tardiness [8] and number of tardy jobs [9]. In this study, we consider the makespan criterion, which is defined as the completion time of the last job leaving the system.

The no-wait flowshop scheduling problem with makespan criterion is known to be NP-hard [10]. Due to the NP-hard nature, exact algorithms that guarantee global optimality require unacceptably long computational time even for moderate-sized problems. On the other hand, heuristics can provide excellent results that are optimal or near-optimal for large-scale problems. Therefore, considerable research attention has been attracted to the design of heuristics in recent years.

Heuristic algorithms can be broadly classified into two groups: constructive methods and improvement methods. Constructive methods generally exploit the problem-specific information and present a single pass algorithm to build a complete schedule under some predetermined priority rule. Bertolissi [11], Rajendran [1], Bonney and Gundry [12], King and Spachis [13], and Laha and Chakraborty [14] introduced different constructive heuristics for the no-wait flowshop problem to minimize makespan. In addition, the famous NEH algorithm which was first proposed by Nawaz, Enscore, and Ham [15] to solve the permutation flowshop problem also suits the problem well. The improvement methods consist of a wide variety of meta-heuristics developed in the past three decades. Meta-heuristics are general algorithm frameworks and have been successfully implemented for many complex optimization problems. Well-known meta-heuristics include genetic algorithm (GA) [16], simulated annealing (SA) [17], tabu search (TS) [18] and variable neighborhood search (VNS) algorithm [19]. In recent years, many more computationally effective meta-heuristic algorithms are proposed, such as estimation of distribution (EDA) algorithm [20], ant colony optimization (ACO) [21], discrete particle swarm optimization (DPSO) [5], differential evolution algorithm [22] and memetic algorithm [23].

Although the newly developed meta-heuristic algorithms provide excellent results for scheduling problems, they are fairly sophisticated and their performance largely depends on complicated parameter tuning schemes. This has created difficulty of obtaining the same computational effectiveness when re-implementing the reported algorithms. As Pan and Ruiz [24] pointed out, simple and easily adaptable algorithms are highly desirable. The iterated greedy (IG) [25] algorithm is an example of such algorithms. It can provide comparable or even better computational results to the other optimization algorithms, while the search mechanism is simple and general. The IG algorithm iteratively applies a neighborhood search scheme to obtain a local optimal solution and a greedy reconstruction scheme to escape from the obtained local minima. This process is repeated until the termination condition is satisfied.

The simplicity and effectiveness of IG algorithm motivate us to exploit further improvement of the algorithm's structure. In particular, a more suitable perturbation method and a more powerful neighborhood search approach are to be considered in the algorithm's framework. In the origi-

This research is partially supported by the National Natural Science Foundation of China under Grants 61273233 and 61104176, the Research Foundation for the Doctoral Program of Higher Education under Grants 20120002110035 and 20130002130010.

Ding, Song and Wu are with the Department of Automation, Tsinghua University, Beijing 100084, China (emails: ding-jy12@mails.tsinghua.edu.cn, shijis@tsinghua.edu.cn, wuc@tsinghua.edu.cn). Zhang is with the School of Economics and Management, Nanchang University, Nanchang 330031, China (email: zhangrui@ncu.edu.cn)

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

nal perturbation scheme, some jobs are randomly removed from a complete sequence and then reinserted greedily into the obtained partial sequence one by one to build another complete sequence. However, this insertion reconstruction strategy utilizes a mechanism similar to the commonly used neighborhood search method based on insert moves. This may lead to repeated search and poor exploitation ability in the search progress. To overcome this drawback, we propose a tabu mechanism based construction method in the perturbation scheme to avoid repeated search. In addition, a more powerful neighborhood structure based on insert, swap and double-insert moves is adopted to further enhance the exploration ability.

The rest of the paper is organized as follows. In section 2, the NWFSP problem with makespan performance measure is formulated. Section 3 proposes the tabu mechanism improved iterated greedy algorithm for the considered problem. The computational effectiveness of the proposed algorithm is verified through numerical comparison in section 4. Finally, section 5 summarizes the paper and highlight some future research directions.

II. Problem Formulation

The no-wait flowshop scheduling problem is described as follows. There are n jobs to be processed sequentially through m machines in the same order. Every job $j\,(j = 1, 2, \ldots, n)$ requires a predetermined processing time $p_{i,j}$ on every machine $i\,(i = 1, 2, \ldots, m)$. To satisfy the no-wait constraint, all the jobs must be processed without any waiting time between consecutive machines. This implies that the start of a job must be delayed on the first machine when necessary. In this work, the target of scheduling is to minimize the makespan denoted as $C_{\max}$ (or $C_{m,n}$) which equals the finish time of the last job on the last machine in the shop. It is noticeable that some other assumptions on the permutation flowshop scheduling problems described in Ref. [26] apply to this problem as well.

The no-wait characteristic of the problem ensures that the completion time difference between two adjacent jobs is determined by the processing times of the two jobs, regardless of the other jobs in the permutation. Thus, a *completion time distance* can be defined between each pair of adjacent jobs. The completion time distance from job i to job j is calculated in advance as follows [27].

$$D_{i,j} = \max_{k=1,\ldots,m} \left\{ \sum_{h=k}^{m} (p_{h,j} - p_{h,i}) + p_{k,i} \right\}. \qquad (1)$$

It is clear that the makespan of a feasible permutation schedule $\pi = \{\pi(1), \ldots, \pi(n)\}$ is given by

$$\begin{aligned} C_{\max}(\pi) &= C_{m,n}(\pi) \\ &= \sum_{j=2}^{n} D_{[j-1],[j]} + \sum_{k=1}^{m} p_{k,\pi(1)}, \end{aligned}$$

where $D_{[j-1],[j]}$ represents the completion time distance between the $(j-1)$-th and j-th job in schedule π, i.e. $D_{\pi(j-1),\pi(j)}$.

To simplify the expression of the makespan, a dummy job $\pi(0)$, which has zero processing time on all machines is introduced at the beginning of permutation π. Thus, the schedule π is redefined as $\pi' = \{\pi(0), \pi(1), \ldots, \pi(n)\}$. With the above developments, we obtain the objective function, i.e. makespan of a permutation π as follows:

$$C_{\max}(\pi) = C_{\max}(\pi') = \sum_{i=1}^{n} D_{[j-1],[j]}. \qquad (2)$$

Let Π denote the set of all $n!$ possible permutation schedules in the solution space for the NWFSP. The problem is then to find a permutation schedule $\pi^* \in \Pi$ such that:

$$C_{\max}(\pi^*) = \min_{\pi \in \Pi} C_{\max}(\pi).$$

III. Tabu mechanism improved IG algorithm

This section presents a tabu mechanism improved iterated greedy (TMIIG) algorithm for solving the no-wait flowshop scheduling problem with makespan criterion. The proposed algorithm consists of the following three phases: an NEH algorithm to generate an initial schedule, a tabu-based reconstruction technique to escape from local minima, and a neighborhood search method to improve the current solution. The reconstruction phase and neighborhood search phase continue until a pre-defined termination condition is met. In this section, we first describe each phase of the proposed algorithm and the stopping criterion used. Then, we present the steps of the proposed TMIIG algorithm.

A. Initial Solution

The NEH algorithm is an effective constructive heuristic algorithm to minimize makespan in a permutation flowshop. Starting from an initial sort of jobs in non-increasing order of their total processing times, it constructs a complete solution by inserting these jobs one by one into a partial schedule to generate an approximate solution for the problem. Since the NWFSP problem in this work also aims to obtain a job permutation such that the makespan is minimized, the NEH algorithm can suit this problem very well. The procedures of the NEH algorithm is described as follows:

Algorithm 1 NEH algorithm

1: Generate an initial sequence in non-increasing order of their total processing times. Denote this sequence as $\pi_0 = (\pi_0(1), \pi_0(2), \ldots, \pi_0(n))$.
2: Select the first two jobs in π_0 and evaluate the two possible partial sequences of the two jobs. The sequence with the smaller makespan will become the current partial sequence $\pi_2 = (\pi_2(1), \pi_2(2))$.
3: **for** $k = 3$ to n **do**
4: Find the best position of inserting $\pi_0(k)$ into the current partial sequence π_{k-1}. The solution obtained in this step is denoted as the new current partial sequence π_k.
5: **end for**

Output: sequence π_n and $C_{\max}(\pi_n)$

Since the completion time matrix $(D_{i,j})_{n\times n}$ can be calculated in advance, the main computational burden of the NEH algorithm lies in the inserting process of Step 3. By considering the acceleration method for job insertion in Ref. [28], the NEH algorithm can be completed in $O(n^2)$ computational effort.

B. Tabu-based reconstruction

The original Iterated Greedy (IG) algorithm generates a sequence of solutions by iteratively applying greedy constructive heuristics including destruction and construction phases. In these two phases, some jobs are firstly removed from a complete sequence and then reinserted into the obtained partial sequence using the NEH-Insertion method. However, this reconstruction mechanism may not have a good performance in jumping out of local minima since it utilizes a similar mechanism as in the neighborhood search phase which may lead to repeated search. Faced with this issue, a new reconstruction technique which forbids some insert positions is proposed as follows.

Once the initial solution is obtained, we create a tabu list TL_j for each job j. When inserted into the partial sequence, job j is not allowed to be placed at the positions immediately after any job in the tabu list TL_j. The tabu lists are updated in each iteration.

To start the reconstruction, d jobs ($d \in \{1,\ldots,n-1\}$) are randomly selected and removed from the current complete candidate solution π which contains n jobs. For each removed job (say $\pi(j)$), the job that immediately precedes it ($\pi(j-1)$) is added to its tabu list ($TL_{\pi(j)}$). The removed d jobs form a partial sequence π_R in the same order that they were selected. The partial solution with $n-d$ remaining jobs also forms another partial sequence denoted as π_D in its original order of jobs.

Next, the jobs in π_R are re-inserted into π_D one after another. These jobs will be inserted at the positions such that the makespan of partial sequence is minimized while the tabu constraint is not violated. The procedure of the tabu based reconstruction is given in Algorithm 2.

The main computational burden in the reconstruction lies in line 9 where each of the possible inserting positions is evaluated. By considering a similar acceleration method as in the implementation of NEH algorithm, the reconstruction phase in each iteration can be completed in $O(dn)$ time.

C. Neighborhood Search

The schedule obtained in the reconstruction phase may not be a good enough solution since there may be better sequences in its neighborhood. Therefore, a local search method is adopted to further improve the current sequence. In particular, the variable neighborhood search (VNS) method is applied in the presented algorithm to find more promising solutions in the solution space by changing the neighborhood structures in the search progress.

The selection on neighborhood structures can greatly influence the performance of the VNS method [19]. Since neighborhood is usually defined based on moves of jobs,

Algorithm 2 Tabu Based Reconstruction

Input: Tabu list $TL_j\,(j=1,\ldots,n)$, sequence π, number of removed jobs d.

1: Set $\pi_R = \varnothing$ and $\pi_D = \pi$.
2: **for** $k = 1$ to d **do**
3: Select a job (without repetition) at a random position j in π. Add $\pi(j-1)$ to the end of tabu list $TL_{\pi(j)}$. Insert job $\pi(j)$ at the end of sequence π_R and remove it from sequence π_D.
4: **if** $Length(TL_{\pi(j)}) > maxLength$ **then**
5: Delete the first element in $TL_{\pi(j)}$
6: **end if**
7: **end for**
8: **for** $k = 1$ to d **do**
9: Evaluate each of the possible inserting positions for job $\pi_R(k)$ that is not prohibited by the tabu list $TL_{\pi_R(k)}$. Insert job $\pi_R(k)$ into π_D at the position with the smallest makespan among the evaluated solutions.
10: **end for**

Output: sequence π_D.

the search progress can benefit much from suitably selected moves. Among various types of moves considered in the literature, insert and pair-wise swap moves are most commonly used for the no-wait flowshop scheduling problems. The neighborhood based on insert moves is defined by enumerating all possible pairs of positions $j,k \in \{1,\ldots,n\}$ in sequence π ($j \neq k$), where job $\pi(j)$ is removed and then reinserted at position k. The neighborhood based on swap moves is defined similarly which considers exchanging the positions of two jobs in the sequence. The acceleration technique for the evaluation of both neighborhood solutions is described in Ref [5].

In this work, we adopted the VNS moves which include not only the insert and swap moves, but also the rarely tested double-insert moves [14], [29]. The double-insert moves consider removing two consecutive jobs from position j ($j = 1,\ldots,n-1$) and re-inserting them together into position k ($k = 1,\ldots,n-1$, and $k > j+1$ or $k < j$) in the same order. Let π denote the original sequence and π' denote the sequence after the double-insert moves. It is easily verified that the makespan increment is given as follows:

$$\begin{aligned}\Delta C_{\max} &= C_{\max}(\pi') - C_{\max}(\pi)\\ &= D_{[k-1][j]} + D_{[j+1][k]} - D_{[k-1][k]}\\ &\quad + D_{[j-1][j+2]} - D_{[j-1][j]} - D_{[j+1][j+2]}\end{aligned} \tag{3}$$

Based on the above discussion, the procedure of the neighborhood search phase is described as in Algorithm 3. By considering the acceleration described in Ref. [5] and equation (3), the computational complexity in this phase can be reduced to $O(n^2)$.

D. Acceptance Criterion and Stopping Rules

After a new sequence π' is generated by the local search procedure, it is to be decided whether to accept it as the

Algorithm 3 Neighborhood Search

Input: sequence π.

Set the swap neighborhood structure as N_1, the insert neighborhood structure as N_2 and the double-insert as N_3.
while $l \leqslant 3$ **do**
  Find the best neighbor solution π' of π in $N_l(\pi)$.
  if $C_{\max}(\pi') < C_{\max}(\pi)$ **then**
    Set $\pi = \pi'$ and $l = 1$.
  else
    Set $l = l + 1$.
  end if
end while

Output: sequence π.

incumbent solution for the next iteration. Inspired by the simulated annealing (SA) method, a SA-like acceptance criterion is adopted in this paper. In this acceptance criterion, π' is accepted with a probability of

$$p = \min\left\{\exp\left(\frac{C_{\max}(\pi) - C_{\max}(\pi')}{T}\right), 1\right\},$$

where π is the solution of current iteration and T is a constant temperature. Following the suggestions of Osman and Potts [30], the temperature is set as follows

$$T = \frac{\sum_{j=1}^{n}\sum_{k=1}^{m} p_{kj}}{10 \cdot m \cdot n} \cdot T_0,$$

where T_0 is a parameter to be adjusted.

The maximum number of iterations and/or the maximum computational time are the stopping rules adopted in this paper. The proposed hybrid TMIIG algorithm stops when a certain stopping condition is met, whichever first.

E. Algorithm framework

Based on the above developments, the procedures of the proposed TMIIG algorithm are described in Algorithm 4.

IV. Computational results

This section describes the computational results for the evaluation of TMIIG in makespan minimization for NWFSP. For this purpose, The well-known Taillard's benchmark problem dataset [31] was adopted for our computational experiments where the no-wait constraints are assumed to be active. This benchmark consists of 120 problem instances with 12 different sizes, ranging from 20 jobs and 5 machines to 500 jobs and 20 machines. For each size, 10 instances are provided. All proposed algorithms and procedures were coded in Visual C++ and computational experiments were executed on a PC with Intel Core (TM) CPU running at 3.20 GHz.

Pilot experiments were conducted under a set of potential parameter values to find the best combinations of parameter settings. The parameter settings are as follows: $d = 10$, $maxLength = 1$, and $Tem = 0.4 \times \frac{\sum_{j=1}^{n}\sum_{k=1}^{m} p_{k,j}}{10nm}$. All the tested algorithms are allowed to run for $t_{\max} = (mn/2) \times 20$ ms before terminated.

Algorithm 4 TMIIG algorithm

(Initial solution): Generate an initial sequence using the NEH algorithm (described in Algorithm 1). Let π be the solution obtained in this step.
Initialize all the tabu lists as empty set: $TL_j := \varnothing, j = 1, \ldots, n$.
while the termination condition is not met **do**
  (Tabu based reconstruction): With respect to the tabu lists, apply the tabu-based reconstruction method (described in Algorithm 2) to sequence π to construct another complete sequence π'. Update the tabu lists.
  (Neighborhood search): Use the neighborhood search method (described in Algorithm 3) that hybridizes the insert, swap, and double-insert moves to improve sequence π' in the previous step. The improved solution is denoted as π''.
  (Acceptance): Set $\pi = \pi''$ with probability

$$p = \begin{cases} 1, & \text{if } \Delta < 0, \\ e^{-\Delta/Tem}, & \text{if } \Delta \geq 0, \end{cases}$$

  where $\Delta = C_{\max}(\pi'') - C_{\max}(\pi)$ and Tem is a predetermined temperature parameter.
end while

Output: The best solution found.

A. Effectiveness of the tabu based reconstruction

To test the effectiveness of the proposed tabu based reconstruction procedure, computational experiments are designed as follows. For one instance, the proposed TMIIG algorithm is evaluated twice. In the first trial, TMIIG is executed as described in the previous sections without any modification. In the second trial, however, TMIIG is executed by selecting the tabu length to be zero and thus the effect of tabu mechanism disappears. By comparing the experimental results in the two trials, the contribution of the tabu mechanism is clearly identified. Note that five independent runs are conducted for each problem instance in both trials. For notational simplicity, the algorithm evaluated in the first and second trial is referred to as TMIIG1 and TMIIG2, respectively.

To test the difference between the two algorithms, a series of paired sample t-test at 95% significance level was conducted. Let μ_1 and μ_2 denote the average makespan of the TMIIG1 and TMIIG2 algorithm, respectively. The null hypothesis is $H_0 : \mu_1 = \mu_2$ and the alternative hypothesis is $H_1 : \mu_1 < \mu_2$.

It is observed in Table I that the null hypothesis is rejected and the alternative hypothesis is accepted for the middle and large scale instances ($n \geqslant 100$). It means that the average makespan obtained by TMIIG1 algorithm is smaller than that of the TMIIG2 algorithm at the significance level of 0.95. This result revealed the effectiveness of involving tabu mechanism in the reconstruction phase.

TABLE II

COMPARISON OF RESULTS ON TAILLARD'S BECHMARK WITH MAKESPAN CRITERION OVER 5 RUNS

$n \times M$	Upper bound	IIGA		$DPSO_{VND}$		TMIIG	
		ARPD	SD	ARPD	SD	ARPD	SD
20×5	1480.3	0.00	0.00	0.00	0.00	0.00	0.00
20×10	1983	0.01	0.01	0.00	0.00	0.00	0.00
20×20	2971.9	0.02	0.01	0.02	0.02	0.00	0.00
50×5	3272.7	0.30	0.20	0.34	0.18	0.19	0.14
50×10	4276.1	0.27	0.14	0.22	0.16	0.13	0.11
50×20	5898.6	0.22	0.14	0.28	0.16	0.08	0.10
100×5	6236.3	0.51	0.18	0.27	0.28	0.25	0.16
100×10	8029.9	0.42	0.19	0.48	0.25	0.17	0.14
100×20	10688.9	0.62	0.19	0.52	0.30	0.21	0.16
200×10	15326.1	0.56	0.20	0.23	0.23	0.18	0.14
200×20	19979	0.75	0.20	0.43	0.21	0.15	0.13
500×20	47226	0.87	0.14	0.58	0.19	0.22	0.18
Average	NA	0.38	0.13	0.28	0.16	0.14	0.12

TABLE I

PAIRED SAMPLE T-TEST FOR TMIIG1 ALGORITHM AND TMIIG2 ALGORITHM ON THE TAILLARD INSTANCES

Instance	H_0	H_1	p-value	H_0	H_1
20×5	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.1717	Accept	Reject
20×10	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.1717	Accept	Reject
20×20	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.1717	Accept	Reject
50×5	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0461	Reject	Accept
50×10	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0033	Reject	Accept
50×20	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0071	Reject	Accept
100×5	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0914	Accept	Reject
100×10	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0012	Reject	Accept
100×20	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0000	Reject	Accept
200×10	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0003	Reject	Accept
200×20	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0000	Reject	Accept
500×20	$\mu_1 = \mu_2$	$\mu_1 < \mu_2$	0.0000	Reject	Accept

B. Comparison with Existing Algorithms

To test the effectiveness and efficiency of the proposed TMIIG algorithm in searching better quality schedules, we compared its computational results with the high-performance algorithms: IIGA [32] and $DPSO_{VND}$ [5] to solve the problem. For this purpose, the IIGA and $DPSO_{VND}$ algorithms were re-implemented on the same PC using the same coding language. Each of the 120 problem instances in the Taillard's dataset was solved using each of the three algorithms. To compare the results obtained from these experiments, the following statistics were collected.

1) *Average relative percentage deviation* (ARPD): the criterion to measure the average relative quality of solutions

$$\text{ARPD} = \frac{1}{R}\sum_{r=1}^{R}\frac{C_r - C_r^*}{C_r^*} \times 100,$$

2) *Standard deviation* (SD): the criterion to measure the degree of solutions' closeness to the mean solutions (and thus a measure of algorithm robustness)

$$\text{SD} = \sqrt{\frac{1}{R}\sum_{r=1}^{R}\left[\frac{C_r - C_r^*}{C_r^*} \times 100 - \text{ARPD}\right]^2},$$

where C_r is the solution generated by a specific algorithm A, $(\text{A} \in \{\text{IIGA}, \text{DPSO}_{\text{VND}}, \text{TMIIG}\}$ in the r-th $(r = 1, 2, \cdots, R)$ experiment for a given size problem, and C_r^* is the best solution found so far. Obviously, the less the value of ARPD (or SD) is, the better the algorithm's performance is.

Table II summarizes the computational results for the IIGA, $DPSO_{VND}$, and TMIIG algorithms. As revealed in the table, the total average of ARPD obtained by the proposed TMIIG algorithm is 0.14, superior to the corresponding value 0.38 and 0.28 obtained by the IIGA and $DPSO_{VND}$ algorithms, respectively. Therefore, the proposed TMIIG algorithm outperforms the IIGA and $DPSO_{VND}$ in solution quality. Similarly, we can conclude from the table that the proposed algorithm outperforms the other two tested algorithms in terms of solution robustness. Since all the three algorithms were executed in the same computational environment, the results indicate that the proposed TMIIG algorithm is superior to all the algorithms compared for solving the NWFSP problem with makespan criterion.

V. CONCLUSION

In this work, a tabu mechanism improved iterated greedy algorithm is developed for the no-wait flowshop scheduling problem to minimize makespan. Unlike many newly reported evolutionary algorithms, the proposed TMIIG algorithm is simple and can be easily replicated. Despite its simplicity, the developed algorithm provides promising computational results. Therefore, the TMIIG algorithm is a suitable metaheuristic for solving the NWFSP problem.

The good performance of TMIIG algorithm largely depends on the tabu-based reconstruction part in the algorithm framework. This reconstruction technique improves the exploitation ability of the algorithm and leads to better performance when compared with the high-performance IIGA and $DPSO_{VND}$ algorithms. Further study related to this work will focus on introducing more powerful neighborhood search approaches to strengthen the exploration ability.

REFERENCES

[1] C. Rajendran, "A no-wait flowshop scheduling heuristic to minimize makespan," *Journal of the Operational Research Society*, pp. 472–478, 1994.

[2] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Operations research*, vol. 44, no. 3, pp. 510–525, 1996.

[3] W. Raaymakers and J. Hoogeveen, "Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing," *European Journal of Operational Research*, vol. 126, no. 1, pp. 131–151, 2000.

[4] T. Aldowaisan and A. Allahverdi, "Minimizing total tardiness in no-wait flowshops," *Foundations of Computing and Decision Sciences*, vol. 37, pp. 149–162, 2012.

[5] Q.-K. Pan, M. Fatih 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] S. J. Shyu, B. Lin, and P. Yin, "Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time," *Computers & industrial engineering*, vol. 47, no. 2, pp. 181–193, 2004.

[7] C. Wang, X. Li, and Q. Wang, "Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion," *European Journal of Operational Research*, vol. 206, no. 1, pp. 64–72, 2010.

[8] G. Liu, S. Song, and C. Wu, "Some heuristics for no-wait flowshops with total tardiness criterion," *Computers & operations research*, vol. 40, no. 2, pp. 521–525, 2013.

[9] T. A. Aldowaisan and A. Allahverdi, "No-wait flowshop scheduling problem to minimize the number of tardy jobs," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 1-4, pp. 311–323, 2012.

[10] H. Röck, "The three-machine no-wait flow shop is np-complete," *Journal of the ACM (JACM)*, vol. 31, no. 2, pp. 336–345, 1984.

[11] E. Bertolissi, "Heuristic algorithm for scheduling in the no-wait flow-shop," *Journal of Materials Processing Technology*, vol. 107, no. 1, pp. 459–465, 2000.

[12] M. Bonney and S. Gundry, "Solutions to the constrained flowshop sequencing problem," *Operational Research Quarterly*, pp. 869–883, 1976.

[13] J. King and A. Spachis, "Heuristics for flow-shop scheduling," *International Journal of Production Research*, vol. 18, no. 3, pp. 345–357, 1980.

[14] D. Laha and U. K. Chakraborty, "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1-2, pp. 97–109, 2009.

[15] M. Nawaz, E. Enscore, 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.

[16] T. Aldowaisan and A. Allahverdi, "New heuristics for no-wait flowshops to minimize makespan," *Computers & Operations Research*, vol. 30, no. 8, pp. 1219–1231, 2003.

[17] S. Kirkpatrick, M. Vecchi, *et al.*, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] J. Grabowski and J. Pempera, "Some local search algorithms for no-wait flow-shop problem with makespan criterion," *Computers & Operations Research*, vol. 32, no. 8, pp. 2197–2212, 2005.

[19] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[20] B. Jarboui, M. Eddaly, and P. Siarry, "An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems," *Computers & Operations Research*, vol. 36, no. 9, pp. 2638–2646, 2009.

[21] F. Ahmadizar, "A new ant colony algorithm for makespan minimization in permutation flow shops," *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 355–361, 2012.

[22] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.

[23] B. Liu, L. Wang, and Y.-H. Jin, "An effective pso-based memetic algorithm for flow shop scheduling," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 18–27, 2007.

[24] Q.-K. Pan and R. Ruiz, "Local search methods for the flowshop scheduling problem with flowtime minimization," *European Journal of Operational Research*, vol. 222, no. 1, pp. 31–43, 2012.

[25] 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.

[26] J. N. Gupta and E. F. Stafford Jr, "Flowshop scheduling research after five decades," *European Journal of Operational Research*, vol. 169, no. 3, pp. 699–711, 2006.

[27] D. Wismer, "Solution of the flowshop-scheduling problem with no intermediate queues," *Operations Research*, vol. 20, no. 3, pp. 689–697, 1972.

[28] X. Li, Q. Wang, and C. Wu, "Heuristic for no-wait flow shops with makespan minimization," *International Journal of Production Research*, vol. 46, no. 9, pp. 2519–2530, 2008.

[29] K. Gao, Q. Pan, P. Suganthan, and J. Li, "Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization," *The International Journal of Advanced Manufacturing Technology*, pp. 1–10, 2013.

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

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

[32] Q.-K. Pan, L. Wang, and B.-H. Zhao, "An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7-8, pp. 778–786, 2008.