New Greedy Randomized Adaptive Search Procedure based on Differential Evolution algorithm for solving no-wait flowshop scheduling problem

Hanen Akrout, Bassem Jarboui, Abdelwaheb Rebaï
Faculté des Sciences Economiques et de Gestion
MODILS
Sfax, Tunisia
hanen.akrout@laposte.net

Patrick Siarry
Université de Paris-Est Créteil
LiSSi
Paris, France
siarry@u-pec.fr

***Abstract*—Scheduling in a production workshop consists of assigning a number of different tasks to the existing machines with respect to the constraints of the problem in order to optimise the objective function. In this paper, we focus on the No-Wait Flow Shop Scheduling Problem (NWFSSP) with the makespan objective. We present a new hybrid algorithm NEW-GRASP-DE that combines Greedy Randomised Adaptive Search Procedure (GRASP) with Differential Evolution Algorithm (DEA). Furthermore, we used an Iterative Local Search (ILS) as an improvement phase in the GRASP method. Our algorithm is tested on instances that have been proposed in the literature. We compared our results with those of different authors. The experimental results show the effectiveness of our algorithm.**

Keywords—no-wait flowshop; greedy randomized adaptive search procedure; differential evolution algorithm; makespan

I. INTRODUCTION

The most scheduling problems are generally difficult to resolve. Indeed, the tasks should be scheduled on fixed periods and under certain conditions. These conditions may be resource constraints and precedence constraints between tasks. Recently, scheduling problems have received special attention especially in the field of production.

In the following, we will consider the No-Wait Flow Shop Scheduling Problem (NWFSSP) with makespan (C_{max}) objective. This kind of problem was studied in many different fields in the literature such as: still production [1], chemical industry [2], production of concrete wares [3], food processing [4] and pharmaceutical industry [5]. First, the NWFSSP was tackled by [6]. Reference [1] showed how to reduce the problem to Asymmetric Travelling Salesman Problem (ATSP). Later, [7] proved that the NWFSSP with makespan criterion is NP-complete. In this context, [4] studied several deterministic flowshop, jobshop, and openshop problems. They also described enumerative algorithms and some heuristics and they reviewed the stochastic no-wait and blocking scheduling problems.

For the production steel, a series of operations must be executed without interruption before the metal has cooled. In fact, in the steel industry, we must have to pay attention to the degradation of metal and to avoid defects in the material composition.

By the three parameter notation, $\alpha / \beta / \gamma$, the considered problem can be notated as $Fm/nwt/C_{max}$, according to [8].

The no-wait constraint means that there is no intermediate storage or work-in-process inventories between two machines or stages. Therefore, all jobs are executed by all machines in the same order without having to wait between two successive machines. But, sometimes it is necessary to delay the start of execution of the job on a particular machine, so that the end of the operation coincides with the start of the operation on the following machine. The difference between the completion time of the last operation and the beginning of the first operation is equal to the sum of their processing time on all machines.

In this paper, we will present a new hybrid algorithm which combines the Greedy Randomised Adaptive Search Procedure (GRASP) [9] and the Differential Evolution Algorithm (DEA) [10] for solving the NWFSSP. In fact, we will try to improve the effectiveness of the GRASP method through the introduction of an auto adjustment procedure of the GRASP parameter by using the DEA. Furthermore, we use the Iterative Local Search (ILS) [11] as an improvement phase in the GRASP method. Our algorithm is tested on benchmark instances from the literature. Compared with the previous results available in the literature, our proposed approach is able to outperform the best competing approaches and to improve the best known solutions of the tested instances.

The rest of this paper is structured as follows. The literature review is presented in section II. Section III describes the problem. Sections IV, V and VI define respectively the GRASP method, the Differential Evolution Algorithm and the Iterated Local Search. In section VII, we present the proposed approach NEW-GRASP-DE. The experimental results are presented in section VIII. We conclude the paper with section IX.

978-1-4799-0313-9/13/$31.00 ©2013 IEEE

II. LITERATURE REVIEW

To obtain an optimal solution for the NWFSSP, we must use enumeration procedures that require a great amount of time. In addition, it is difficult to propose the most adequate heuristic since it varies from one problem to another. For these reasons, some exact methods and constructive heuristics were discussed in the literature. Moreover, the instances relative to this type of problem are very large, so the exact methods are not suitable for solving it. In this context, [12] proposed a new heuristic and dominance relations for the two-machine no-wait separate setup flowshop problem. The total flowtime criterion was considered. For the purpose of evaluating the efficiency of this heuristic, a lower bound is generated and it's used along with the dominance relations in a branch-and-bound algorithm.

Furthermore, some constructive heuristics are developed. For example, [13] presented two simple heuristic algorithms for the NWFSSP. First, they present the methods in order to generate the initial seed sequences. Second, they obtained better solutions by an improvement technique. Reference [2] proposed a simple heuristic based on heuristic preference relations and job insertion for solving the NWFSSP. The considered criterion is the makespan. Reference [14] developed a new constructive heuristic on the basis of the principle of job insertion. This heuristic draws inspiration from the Nawaz–Enscore–Ham heuristic (NEH) proposed by [15]. The considered criterion for all these authors is the makespan.

The majority of the methods proposed in the literature are metaheuristics. In fact they are able to solve any kind of problem since they can be adapted to any problem. Moreover, they can obtain good quality solutions with reasonable computation time.

Some authors used single solution based metaheuristics. For example, [16] developed three heuristics that are based on adapting Simulated Annealing (SA) of [17]. They developed an improved heuristic to SA by applying NEH heuristic [15] and their proposed insertion techniques to the obtained sequence from the SA algorithm. Furthermore, they improved this algorithm by applying the pairwise procedure to the sequence obtained from it. Further, they proposed improvement procedures to these heuristics. Reference [18] presented a local search algorithm $VNS_{(SF)}$ that is based on a decomposition of the problem into a sequencing and a timetabling problem. Reference [19] developed and compared five variants of descending search and tabu search algorithms: descending search algorithm $DS_{(GP)}$, descending search algorithm with multimoves $DS+M_{(GP)}$ (the multimove consists of several moves that are executed at the same time in a single iteration of the algorithm), tabu search algorithm $TS_{(GP)}$, tabu search algorithm with multimoves $TS+M_{(GP)}$ and tabu search algorithm which the multimove is created and executed just in the specific situations $TS+MP_{(GP)}$. The multimoves are proposed for guiding the search process to more promising search space regions and accelerating the convergence to good solutions. Furthermore, a dynamic tabu list is proposed to avoid trapped at a local optimum. Reference [20] proposed an improved iterated greedy algorithm. First, they developed a speed-up method for the insert neighbourhood solution in order to improve the efficiency. Second, they presented a modified Nawaz-Enscore-Ham heuristic based on the NEH heuristic [15] to obtain an initial solution with a certain quality. Third, to perform exploitation, they applied a simple local search on the basis of the speed-up method into their algorithm. Reference [21] described a Variable Neighbourhood Search which is based on two structures of neighbourhoods (swap-local-search and insert-local-search). The swap-local-search makes all possible swaps of pair of job's positions within all parts of solutions. The insert-local-search performs all possible insert moves of jobs within all parts of the generated solution.

Other authors adopted the population based metaheuristics. Reference [16] presented three heuristics that are based on Genetic Algorithm (GA). They adapted the GA proposed by [22] to the NWFSSP. After that, they developed an improved heuristic to GA by applying the NEH heuristic [15] and their insertion techniques to the generated sequence from GA. Finally, they improved this second heuristic by applying the pairwise procedure to the sequence generated from it. They also proposed improvement procedures to these heuristics. Reference [23] developed a Discrete Particle Swarm Optimization algorithm $DPSO_{(PTL)}$. These particles are presented as discrete job permutations and a new position update method is described based on the discrete domain. Reference [24] presented a discrete self-organising migrating version based on the algorithm proposed by [25]. This algorithm was inspired from competitive–cooperative behaviour of intelligent creatures. Reference [21] introduced a genetic algorithm based on a new crossover operator that takes advantage of common sub-sequences provided by the best solutions. This new crossover operator is established on the blocks of jobs in both parents in any position.

In addition, several hybrid methods are considered in the literature. Reference [18] presented a hybrid approach $GASA_{(SF)}$ that combines GA with SA algorithm. This approach is based on the method proposed by [26]. Reference [27] developed a hybrid Genetic Algorithm. In this method, the representation is a variant of one used on GA treatments of the TSP. The genetic operators contain edge recombination and three additional operators conceived on analogy to known heuristic methods. Reference [28] described a hybrid particle swarm optimization. In fact, in order to balance the exploration and the exploitation, they combine the population-based evolutionary searching ability of particle swarm optimization with some local search heuristics. They developed a novel encoding scheme based on random key representation. They proposed two local search methods which the first is based on the NEH heuristic [15] and the second is based on SA with an adaptive meta-Lamarckian learning strategy [29]. To improve the solution quality, [23] presented a $DPSOVND_{(PTL)}$ algorithm which they hybridized their DPSO algorithm with the variable neighbourhood descent algorithm. They developed speed-up methods for both the swap and insert neighbourhood structures. Reference [30] described a hybrid discrete particle swarm optimization. In the beginning, they calculated the makespan of a job permutation through a simple approach. After that, they presented a speed-up method for evaluating the similar insert neighbourhood solution. Finally, they combined a discrete particle swarm optimization algorithm based on the permutation representation and a local search algorithm based

on the insert neighbourhood so as to enhance the searching ability and to balance the exploration and the exploitation. Reference [31] proposed a hybrid differential evolution. Initially, they described a largest-order-value rule for transforming the individuals in the DEA from real vectors to job permutations. Then, they combine the DEA with local search algorithm in the promising region. In the end, they developed a speed-up evaluation method and a fast Insert-based neighbourhood examining method in order to reduce the computing complexity of solution evaluation. Reference [32] presented a hybrid genetic algorithm $HGA_{(TL)}$ where they combined the genetic algorithm with a novel local search scheme. To improve the capability of intensification in their algorithm, they used an orthogonal-array-based crossover operator. Their local search scheme hybridizes the insertion search method and the insertion search with cut-and-repair. For the search process, the first method is used for searching a small neighbourhood and the second method is used for searching a large neighbourhood. Reference [21] also described a hybrid genetic algorithm $GA\text{-}VNS_{(JES)}$ where the Variable Neighbourhood Search was used as an improvement procedure after the mutation in order to avoid the algorithm from trapping into local optima. We note that the makespan criterion was studied by all these authors. Furthermore, the total flow time criterion was considered in the literature. We can cite [33], [34], [12], [35], [30], [36], [21] and [37]. Moreover, [38] treated the maximum lateness criterion. Finally, the total completion time was optimized by [39] and [40].

III. NO-WAIT FLOW SHOP SCHEDULING PROBLEM

The NWFSSP can be described as follows: a set of n jobs are available for processing on m machines. Each machine can processes one job at a time and each job must be performed only on one machine at a time. All jobs must be completed without interruption. The jobs should not wait to be scheduled between two successive machines. The processing order is the same for all jobs. The processing time of each job is known and fixed for each machine. The objective is to find an ordering of jobs which minimizes $C_{\max}$ for all machines.

Let PT_{ij} denotes the processing time of job i ($i = 1, 2, \ldots, n$) on machine j ($j = 1, 2, \ldots, m$), $PT_{w(i)j}$ denotes the processing time of job which is in the i^{th} position in the sequence $w = \{w_{(1)}, w_{(2)}, \ldots, w_{(n)}\}$ on machine j.

The NWFSSP can be transformed into an Asymmetric Travelling Salesman Problem (ATSP). In this context, let D_{ir} denotes the minimum delay on the first machine between the starts of job i and job r (if the job r is scheduled immediately after job i in the sequence).

$$D_{ir} = PT_{i1} + \max\left\{\max_{2\le j\le m}\left[\sum_{t=2}^{j} PT_{it} - \sum_{t=1}^{j-1} PT_{rt}\right], 0\right\} \quad (1)$$

Fig. 1 shows an illustrative example of the transformation of NWFSSP into ATSP.

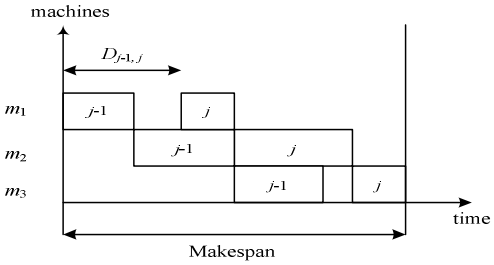

Figure 1. Transformation of NWFSSP into ATSP.

Let $CT_{w(i)}$ the completion time of the job in the i^{th} position in the sequence.

$$CT_{w(1)} = \sum_{j=1}^{m} PT_{w(1)j} \quad (2)$$

$$CT_{w(i)} = \sum_{t=2}^{i} D_{w(t-1)w(t)} + \sum_{j=1}^{m} PT_{w(i)j} \quad (3)$$

where $i \in \{2, 3, \ldots, n\}$.

The makespan is given by the following equation:

$$C_{\max} = CT_{w(n)} \quad (4)$$

Fig. 2 presents the makespan of NWFSSP with three jobs and three machines.

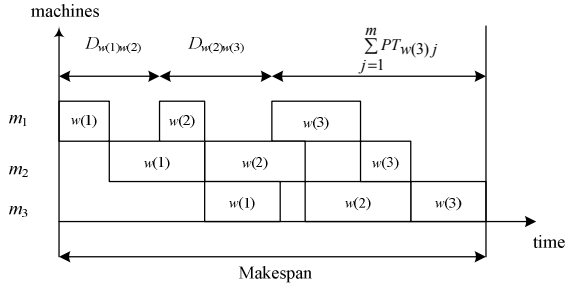

Figure 2. Makespan of NWFSSP ($n = 3$ and $m = 3$).

IV. GREEDY RANDOMISED ADAPTIVE SEARCH PROCEDURE

The GRASP method was introduced by [9]. It's a multi-start iterative metaheuristic which is used to be applied for the combinatorial optimization problems. GRASP is considered as a metaheuristic that keeps the good features of greedy algorithms and random construction procedure. The basic version of GRASP consists of two phases: construction and local search.

First, the constructive phase consists of building a feasible solution to the problem. Second, the local search algorithm is used to improve the solution already found. These two phases are repeated until a stopping criterion is reached.

During the constructive phase, the solution is built one element at a time. So, in each iteration, we add an item to the current partial solution. In order to determine which element will be added to this solution, we choose randomly one element from the Restricted Candidate List (RCL_α) obtained through a greedy function (where α denotes the threshold parameter of GRASP method). The RCL_α contains only the best elements and it's dynamically updated after each iteration of the construction. This constructive phase continues until a complete solution is obtained.

In fact, the solution generated is not guaranteed to be local optimum. To improve this solution, the local search phase is used. This phase replaces successively the current solution by the best solution found in its neighbourhood. Local search algorithm stops when there is no better solution found.

V. Differential Evolution Algorithm

The DEA is an evolutionary algorithm which has been introduced by [10]. Generally, it's used for solving continuous optimization problems. The DEA is a population based method which consists in four phases: initialization, mutation, crossover and selection. It is characterized by using the differential mutation. In fact, this method is based on a vector difference paradigm. This means that the difference between randomly indexed candidate solutions is considered to construct a new candidate solution. The nomenclature scheme used in this paper is: "DE/rand/1/bin". For the DEA, three control parameters are considered: the scaling factor F, the crossover rate CR and the population size P.

In the initialization phase, we start with the random initialization of a population of individuals in the search space with P individuals and we determine the best individual with the best objective value. We consider the i^{th} individual in the D-dimensional search space at generation (iteration) g be $S_i(g) = [s_{i,1}, s_{i,2}, ..., s_{i,D}]$ ($i = 1, 2, ..., P$), where $S_i(g)$ is a real vector.

In the mutation phase, we generate a mutant vector by adding the weighted difference between a defined number of individuals randomly selected from the previous population to another individual. The following equation shows how to combine three different, randomly chosen vectors to create a mutant vector:

$$M_i(g+1) = S_{r1}(g) + F * (S_{r2}(g) - S_{r3}(g)) \quad (5)$$

The indexes $r1$, $r2$, $r3 \in \{1, 2, ..., P\}$ are different and chosen randomly. They are also different from the current index i. The scaling factor F is a positive real number that controls the rate at which the population evolves. It's in the range between 0 and 2.

In the crossover phase, the Differential Evolution crosses each vector target individual $S_i(g)$ with a mutant vector in order to obtain the trial vector. The crossover operator is applied so that increasing the diversity of the population. Following the mutation phase, the trial individual is generated such that:

$$T_{i,j}(g+1) = \begin{cases} M_{i,j}(g+1) & \text{if } (r_j \le CR \text{ or } j = j_{rand}) \\ S_{i,j}(g) & \text{otherwise} \end{cases} \quad (6)$$

The crossover rate is in the range between 0 and 1. It's considered as a user-defined value that controls the fraction of parameter values that are copied from the mutant. r_j denotes the j^{th} evaluation of a random number uniformly distributed in the range of [0, 1] where ($j = 1, 2, \ldots, N$). j_{rand} denotes the index randomly chosen from the set (1, 2, …, D).

For the selection phase, we compare the objective function values of target and trial individuals. So, for a minimization problem, if the trial individual better minimizes the objective function, then it replaces the target vector in the next generation. Otherwise, the target keeps its place in the population for at least one more generation. The corresponding equation can be presented as follows: (f denotes the objective function)

$$S_i(g+1) = \begin{cases} T_i(g+1) & \text{if } f(T_i(g+1)) \le f(S_i(g)) \\ S_i(g) & \text{otherwise} \end{cases} \quad (7)$$

After generation a new population, the process of mutation, crossover and selection is repeated until the stopped criterion is satisfied.

VI. Iterated Local Search

The ILS metaheuristic is introduced by [11]. It's a stochastic local search method that iteratively applies local search to perturbations of the locally optimal feasible solution. In fact, the ILS method consists in combining local search method and perturbation of solutions. The main idea behind ILS algorithm is to alternate between the local search and the perturbation.

To describe this method, four basic steps must be defined: an initialization strategy which generates an initial solution from the search space, the perturbation which modifies the current solution in a different way so as to obtain the intermediate solution, the local search which leads the intermediate solution to the local optimum, and the acceptance criterion which determines whether the local optimum is applied for the next perturbation.

The ILS algorithm starts from a local optimal feasible solution generated by the local search. For escaping from this local optimum, a random perturbation is applied to the current solution and a new local optimum is generated and the local search is applied again to it. In order to decide if this new local optimal solution should be accepted as new current solution, it must be satisfies the acceptance criterion. In the contrary case, the new current solution does not change. The best solution is updated and the above process is repeated until a termination criterion is fulfilled.

VII. NEW APPROACH FOR SOLVING THE NWFSSP

In this work, we will try to improve the performance of GRASP method through the introduction of an auto adjustment of the threshold α in the constructive phase based on the DEA.

A. Constructive Phase

In each iteration of the constructive phase, the GRASP method randomly selects one element of the solution from the Restricted Candidate List RCL_α until a new solution is generated. The key to the success of this method depends on the choice of the threshold α that restricts the set of candidate elements. The most important failure of this method is presented in this characteristic which requires a good adjustment of the parameter α considering that this parameter will be constant for all instances and during the different iterations of the constructive phase. In reality, the good choice of the parameter α varies from one instance to another instance. Sometimes, it will be interesting to vary this parameter during the constructive phase of the same solution.

For this reason, we propose an auto adjustment of this parameter which can be adapted from one instance to another instance and to take into account possible changes of parameter α during the constructive phase of the solution. This goal was achieved through the hybridization of GRASP method with DEA. This algorithm will adjust a parameter vector $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, where α_i denotes the selection threshold into the *RCL* after the choice realized in the iteration (i-1). In our case, the parameter α_i will depend on the choice of the job during the iteration (i-1).

In each time when a new vector $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ was generated by DEA, we construct a new solution by a GRASP method through selecting at each iteration a job from the list RCL_{α_i} which denotes the set of candidate jobs satisfying $pr_{w(i)j} \geq \alpha_i \max_{j\in\Omega} \{pr_{w(i)j}\}$, where Ω is the set of jobs not yet scheduled and $pr_{w(i)j}$ denotes the selection probability of the job j after the job $w(i)$.

$$pr_{w(i)j} = \frac{1/d_{w(i)j}}{\sum_{k\in\Omega} 1/d_{w(i)k}} \quad \forall j \in \Omega \tag{8}$$

This probability is inspired from the nearest neighbour used in the Travelling Salesman Problem. The RCL_{α_i} depends on parameter α_i which is associated to each job i. The selection of elements of a candidate list depends on the last job scheduled in the sequence. The parameters α_i are improved by conserving only the best parameter, that contribute to generate good solution, in the population of DEA. The selection criterion of DEA is used directly after the evaluation of the solution generated by the constructive phase of GRASP method.

B. Improvement Phase

When we use a simple local search, we cannot escape from local optima. Therefore, the use of the ILS algorithm will be more interesting at this level. In our paper, we propose the ILS algorithm as an improvement phase into the GRASP method. So, perturbations based on very large scale neighbourhood allow us to escape local optima and to change the solution. The local search relative to the ILS method consists in applying sequentially four neighbourhood structures in the following order: N_1 (insertion of one job in a new position), N_2 (swap of two jobs), N_3 (insertion of subsequence of jobs in a new position), N_4 (swap of two subsequences of jobs). The local search procedure stops when it converges to a local optimum of four neighbourhood structures. In other words, the procedure terminates when we cannot improve the current solution. So as to perturb the solution, we apply a swap and insertion moves selected at random. The number of moves is a random number which belongs to the range [1, $n_{\max}$], where $n_{\max}$ denotes the maximum number of moves.

The four neighbourhood structures used in our algorithm are described in Fig. 3: the neighbourhood structure N_1 consists of selecting a job and inserting it into a new position. The neighbourhood structure N_2 consists of selecting two jobs and swapping their positions. The neighbourhood structure N_3 consists of selecting a subsequence jobs and inserting it into a new position. The neighbourhood structure N_4 consists of selecting two subsequence jobs and exchanging their positions.

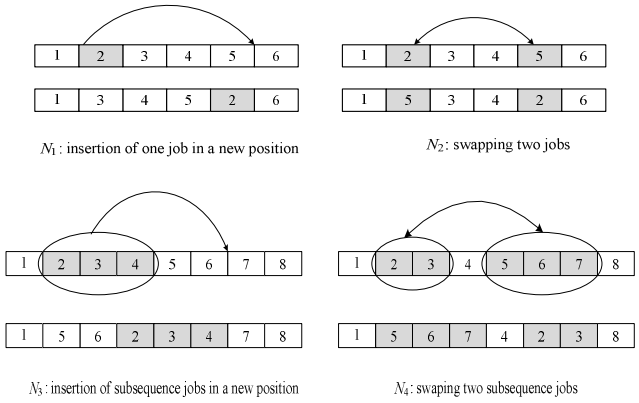

Figure 3. The proposed neighbourhood structures.

VIII. COMPUTATIONAL RESULTS

To show the performance of our proposed approach NEW-GRASP-DE for the NWFSSP, our experimentation is carried out on the benchmarks of Reeves [41], Heller [42] and Taillard [43]. We compared our results with the ones obtained by the other authors in the literature while considering the makespan criterion.

The NEW-GRASP-DE algorithm is coded in C++. The experiments are executed in Windows XP on desktop PC with Intel Pentium IV, running at 3.2 GHz processor and 512 MB memory. The proposed approach is tested on 133 benchmarks where 21 instances are taken from [41], 2 instances are taken from [42] and 110 instances are taken from [43].

So as to calculate the average, we solved each instance 10 times. Based on the best results, we fixed the parameters of the algorithm after a set of experiments. The parameters are fixed as follows: P = 50, F = 1.5, CR = 0.85, $n_{\max}$ = 10. We set a

maximal computational time equal to $n^2/2$ ms as a stopping criterion.

Moreover, when we consider all the results presented in the literature, we remark that the authors used different softwares and hardwares. In addition, some authors did not mention all their results. For this reason, we are sometimes unable to compare our results with other results.

A. *Experiment results for Reeves's instances and Heller's instances*

Considering the Reeves's instances [41], n = {20, 30, 50, 75} denotes the number of jobs and m = {5, 10, 15, 20} denotes the number of machines. But considering the Heller's instances [42], n = {20, 100} denotes the number of jobs and m = {10} denotes the number of machines.

For the instances of Reeves and Heller, we compared in Tables I and II our NEW-GRASP-DE algorithm to those previously obtained by other authors such as the (RAJ) algorithm proposed by [2], the $(VNS)_{(SF)}$ and the $(GASA)_{(SF)}$ algorithms developed by [18], the $(DS)_{(GP)}$, the $(DS + M)_{(GP)}$, the $(TS)_{(GP)}$, the $(TS + M)_{(GP)}$ and the $(TS + MP)_{(GP)}$ algorithms used by [19], the $(HGA)_{(TL)}$ algorithm presented by [32], the $(DPSO)_{(PTL)}$ and the $(DPSOVND)_{(PTL)}$ algorithms developed by [23], the $(GA\text{-}VNS)_{(JES)}$ algorithm used by [21].

Tables I and II showed the results in term of deviation. We presented in Tables III and IV the computational times for all algorithms cited above. However, the results of the $(HGA)_{(TL)}$ algorithm for Heller's instances are not available.

According to the averages shown in Tables I and II, we note that we obtained the best results for the instances of Reeves and Heller. For the minimum: 0.03 (NEW-GRASP-DE) < 0.05 $(GA\text{-}VNS)_{(JES)}$ < 0.09 $(DPSOVND)_{(PTL)}$ < 1.71 $(DPSO)_{(PTL)}$. For the average: 0.14 (NEW-GRASP-DE) < 0.22 $(DPSOVND)_{(PTL)}$ < 0.24 $(GA\text{-}VNS)_{(JES)}$ < 2.24 $(DPSO)_{(PTL)}$. For the maximum: 0.30 (NEW-GRASP-DE) < 0.44 $(GA\text{-}VNS)_{(JES)}$ < 0.44 $(DPSOVND)_{(PTL)}$ < 2.94 $(DPSO)_{(PTL)}$.

For other algorithms, the authors noted that their minimum values: 0.40 $(TS + M)_{(GP)}$ < 0.42 $(TS + MP)_{(GP)}$ < 0.49 $(TS)_{(GP)}$ < 1.44 $(VNS)_{(SF)}$ < 2.62 $(DS + M)_{(GP)}$ < 2.64 $(DS)_{(GP)}$ < 6.31 $(GASA)_{(SF)}$.

According to the Tables III and IV, we range the average minimum time: 0.00 $(DPSO)_{(PTL)}$ < 0.00 $(DS + M)_{(GP)}$ < 0.02 $(DS)_{(GP)}$ < 0.07 $(DPSOVND)_{(PTL)}$ < 0.13 $(GA\text{-}VNS)_{(JES)}$ < 0.23 (NEW-GRASP-DE) < 0.85 $(TS)_{(GP)}$ < 0.87 $(TS + M)_{(GP)}$ < 0.87 $(TS + MP)_{(GP)}$ < 23.78 $(VNS)_{(SF)}$ < 200.13 $(GASA)_{(SF)}$.

In terms of results, our algorithm is better than those of [30] and [21], but in terms of computation time, they overwhelm us with a few fractions of a second (it's very negligible). For these algorithms: $(DPSO)_{(PTL)}$, $(DS + M)_{(GP)}$, $(DS)_{(GP)}$, their results are the best in terms of computation time (with a few fractions of a second), but in terms of results we are the best.

TABLE I. RESULTS OF DIFFERENT ALGORITHMS CITED IN THE LITERATURE

	best	RAJ	$VNS_{(SF)}$	$GASA_{(SF)}$	$DS_{(GP)}$	$DS+M_{(GP)}$	$TS_{(GP)}$	$TS+M_{(GP)}$	$TS+MP_{(GP)}$	$HGA_{(TL)}$
Inst										
Rec01	1526	4,19	1,31	0,07	0,33	0,46	0,00	0,07	0,07	0,00
Rec03	1361	7,05	2,42	2,28	3,38	3,38	0,00	0,00	0,00	0,00
Rec05	1511	8,34	0,73	0,86	2,25	2,25	0,33	0,07	0,00	0,00
Rec07	2042	3,77	1,37	0,20	2,64	2,64	0,00	0,00	0,00	0,00
Rec09	2042	4,85	2,35	0,15	1,08	1,08	0,00	0,05	0,05	0,00
Rec11	1881	3,46	1,86	0,00	1,97	1,97	0,00	0,00	0,00	0,00
Rec13	2545	6,44	0,31	0,43	2,79	2,79	0,00	0,00	0,00	0,00
Rec15	2529	6,41	0,12	0,00	1,27	1,27	0,12	0,00	0,12	0,00
Rec17	2587	5,91	0,46	0,12	0,08	0,08	0,00	0,00	0,00	0,00
Rec19	2850	10,77	2,39	4,74	2,25	2,53	0,00	0,53	0,39	0,00
Rec21	2821	6,88	2,38	4,50	2,94	2,94	0,07	0,14	0,28	0,28
Rec23	2700	12,22	0,15	4,70	4,04	4,04	0,15	0,19	0,00	0,00
Rec25	3593	6,74	0,92	3,87	3,45	3,45	0,36	0,00	0,11	0,00
Rec27	3431	6,53	0,32	3,76	2,65	2,65	0,52	0,03	0,32	0,00
Rec29	3291	8,87	1,00	4,53	1,00	1,00	0,36	0,00	0,36	0,00
Rec31	4307	7,52	2,46	10,45	3,48	3,46	1,18	0,93	0,84	0,62
Rec33	4424	7,82	2,06	12,97	5,70	5,65	1,88	1,02	0,95	0,76
Rec35	4397	7,30	1,39	11,23	2,00	2,00	0,77	0,68	0,96	0,61
Rec37	8017	12,00	0,80	18,60	3,27	3,13	1,46	1,37	1,52	1,29
Rec39	8420	8,76	2,98	18,34	3,36	3,19	1,15	1,16	1,25	1,00
Rec41	8437	10,75	2,55	18,26	4,02	4,02	1,03	1,26	0,98	0,80
Hel01	704	10,80	2,13	24,57	5,11	4,69	1,85	1,70	1,56	-
Hel02	179	5,59	0,56	0,56	1,68	1,68	0,00	0,00	0,00	-
AVG			1,44	6,31	2,64	2,62	0,49	0,40	0,42	

TABLE II. RESULTS OF DIFFERENT ALGORITHMS CITED IN THE LITERATURE

		$DPSO_{(PTL)}$			$DPSOVND_{(PTL)}$			$GA\text{-}VNS_{(JES)}$			NEW-GRASP-DE		
Inst	**best**	**min**	**avg**	**max**	**min**	**avg**	**max**	**Min**	**avg**	**max**	**min**	**avg**	**max**
Rec01	1526	0,46	0,61	1,18	0,00	0,00	0,00	0,00	0,00	0,07	0,00	0,00	0,00
Rec03	1361	1,92	2,09	2,43	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Rec05	1511	3,84	1,65	2,25	0,00	0,02	0,20	0,00	0,01	0,07	0,00	0,00	0,00
Rec07	2042	0,05	0,45	0,98	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Rec09	2042	0,54	1,58	3,87	0,00	0,00	0,00	0,00	0,02	0,05	0,00	0,00	0,00
Rec11	1881	0,00	0,84	1,86	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Rec13	2545	0,43	0,83	1,92	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Rec15	2529	0,00	0,51	1,03	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Rec17	2587	0,08	0,33	0,58	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Rec19	2850	1,69	2,14	2,66	0,00	0,19	0,38	0,00	0,11	0,38	0,00	0,00	0,00
Rec21	2821	1,66	2,20	2,59	0,00	0,11	0,42	0,00	0,09	0,35	0,00	0,00	0,00
Rec23	2700	2,22	2,37	3,11	0,00	0,05	0,92	0,00	0,07	0,63	0,00	0,16	0,37
Rec25	3593	1,11	1,83	2,44	0,00	0,05	0,11	0,00	0,00	0,00	0,00	0,00	0,08
Rec27	3431	1,75	2,77	3,00	0,00	0,25	0,73	0,00	0,08	0,32	0,00	0,01	0,02
Rec29	3291	1,61	1,99	2,37	0,00	0,09	0,64	0,00	0,13	0,36	0,00	0,14	0,72
Rec31	4307	2,25	2,91	3,62	0,11	0,43	0,67	0,18	0,51	0,76	0,00	0,22	0,55
Rec33	4424	3,26	4,46	5,61	0,18	0,62	0,97	0,09	0,71	1,49	0,11	0,36	0,74
Rec35	4397	2,48	3,26	3,89	0,09	0,29	0,84	0,00	0,40	0,72	0,00	0,13	0,34
Rec37	8017	2,86	3,62	4,32	0,17	0,73	1,10	0,15	0,78	1,24	0,01	0,31	0,78
Rec39	8420	2,71	3,27	3,78	0,54	0,80	1,18	0,10	0,75	1,19	0,33	0,73	1,27
Rec41	8437	3,42	3,99	4,58	0,47	0,71	0,97	0,15	0,68	1,01	0,00	0,50	0,90
Hel01	704	3,98	4,60	4,97	0,42	0,64	0,99	0,57	1,12	1,56	0,28	0,59	0,99
Hel02	179	1,12	3,21	4,47	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
AVG		1,71	2,24	2,94	0,09	0,22	0,44	0,05	0,24	0,44	0,03	0,14	0,30

TABLE III. COMPUTATION TIME OF DIFFERENT ALGORITHMS CITED IN THE LITERATURE

	$VNS^a_{(SF)}$	$GASA^a_{(SF)}$	$DS^b_{(GP)}$	$DS+M^b_{(GP)}$	$TS^b_{(GP)}$	$TS+M^b_{(GP)}$	$TS+MP^b_{(GP)}$	$HGA^c_{(TL)}$
Inst								
Rec01	0,00	6,00	0,00	0,00	0,20	0,20	0,20	0,009
Rec03	0,00	6,00	0,00	0,00	0,20	0,20	0,20	0,006
Rec05	0,00	7,00	0,00	0,00	0,20	0,20	0,20	0,008
Rec07	0,00	12,00	0,00	0,00	0,20	0,20	0,20	0,008
Rec09	0,00	11,00	0,00	0,00	0,20	0,20	0,20	0,008
Rec11	0,00	10,00	0,00	0,00	0,20	0,20	0,20	0,008
Rec13	0,00	17,00	0,00	0,00	0,30	0,30	0,30	0,009
Rec15	0,00	17,00	0,00	0,00	0,30	0,30	0,30	0,008
Rec17	0,00	16,00	0,00	0,00	0,30	0,30	0,30	0,008
Rec19	1,00	34,00	0,00	0,00	0,40	0,40	0,40	0,034
Rec21	1,00	35,00	0,00	0,00	0,40	0,40	0,40	0,030
Rec23	0,00	35,00	0,00	0,00	0,40	0,40	0,40	0,025
Rec25	1,00	55,00	0,00	0,00	0,50	0,50	0,50	0,031
Rec27	1,00	51,00	0,00	0,00	0,50	0,50	0,50	0,031
Rec29	1,00	54,00	0,00	0,00	0,50	0,50	0,50	0,031
Rec31	5,00	147,00	0,00	0,00	1,10	1,10	1,10	0,267
Rec33	7,00	145,00	0,00	0,00	1,10	1,10	1,10	0,252
Rec35	7,00	146,00	0,00	0,00	1,10	1,10	1,10	0,225
Rec37	122,00	907,00	0,10	0,00	2,50	2,60	2,60	1,447
Rec39	106,00	890,00	0,10	0,00	2,50	2,60	2,60	1,283
Rec41	110,00	904,00	0,10	0,00	2,50	2,60	2,60	1,073
Hel01	185,00	1088,00	0,10	0,00	3,80	3,90	3,90	-
Hel02	0,00	10,00	0,00	0,00	0,20	0,20	0,20	-
AVG	23,78	200,13	0,02	0,00	0,85	0,87	0,87	

a. $VNS_{(SF)}$ and $GASA_{(SF)}$ are run on Athlon 1400 MHz [18].

b. $DS_{(GP)}$, $(DS+M)_{(GP)}$, $TS_{(GP)}$, $(TS+M)_{(GP)}$ and $(TS+MP)_{(GP)}$ are run on Pentium 1000 MHz [19].

c. $HGA_{(TL)}$ is run on Intel Pentium III, 1266 MHz processor and 1GB memory [32].

TABLE IV. COMPUTATION TIME OF DIFFERENT ALGORITHMS CITED IN THE LITERATURE

	DPSO[d](PTL)			DPSOVND[d](PTL)			GA-VNS[e](JES)			NEW-GRASP-DE		
Inst	**min**	**Avg**	**max**	**min**	**avg**	**max**	**min**	**avg**	**max**	**min**	**avg**	**max**
Rec01	0,00	0,00	0,02	0,00	0,01	0,03	0,00	0,00	0,08	0,00	0,00	0,00
Rec03	0,00	0,00	0,02	0,00	0,00	0,02	0,00	0,00	0,01	0,00	0,00	0,00
Rec05	0,00	0,00	0,02	0,00	0,02	0,06	0,00	0,00	0,01	0,00	0,01	0,07
Rec07	0,00	0,00	0,00	0,00	0,00	0,02	0,00	0,01	0,06	0,00	0,03	0,11
Rec09	0,00	0,00	0,02	0,00	0,00	0,02	0,00	0,02	0,09	0,00	0,00	0,03
Rec11	0,00	0,00	0,00	0,00	0,01	0,03	0,00	0,01	0,09	0,00	0,00	0,01
Rec13	0,00	0,00	0,02	0,00	0,01	0,03	0,00	0,01	0,09	0,00	0,00	0,00
Rec15	0,00	0,00	0,00	0,00	0,00	0,02	0,00	0,01	0,09	0,00	0,00	0,03
Rec17	0,00	0,00	0,02	0,00	0,00	0,02	0,00	0,01	0,08	0,00	0,00	0,01
Rec19	0,00	0,01	0,02	0,00	0,02	0,08	0,00	0,00	0,01	0,01	0,06	0,23
Rec21	0,00	0,01	0,02	0,00	0,04	0,16	0,00	0,02	0,10	0,04	0,14	0,21
Rec23	0,00	0,00	0,02	0,01	0,05	0,17	0,00	0,05	0,21	0,01	0,01	0,03
Rec25	0,00	0,00	0,02	0,00	0,03	0,17	0,00	0,02	0,12	0,01	0,07	0,25
Rec27	0,00	0,00	0,02	0,00	0,06	0,16	0,01	0,07	0,21	0,01	0,12	0,21
Rec29	0,00	0,01	0,02	0,00	0,02	0,14	0,00	0,06	0,18	0,00	0,02	0,03
Rec31	0,00	0,02	0,03	0,06	0,33	0,56	0,10	0,34	0,62	0,14	0,43	0,78
Rec33	0,00	0,02	0,03	0,05	0,34	0,56	0,18	0,36	0,59	0,21	0,33	0,56
Rec35	0,00	0,02	0,05	0,16	0,34	0,58	0,09	0,30	0,60	0,12	0,33	0,75
Rec37	0,00	0,03	0,06	0,34	1,16	1,58	1,13	0,35	1,40	0,67	1,46	1,89
Rec39	0,00	0,04	0,11	0,17	0,99	1,63	0,36	0,98	1,43	0,96	1,54	2,03
Rec41	0,00	0,04	0,08	0,20	0,88	1,56	0,28	1,12	1,42	1,09	1,70	2,01
Hel01	0,00	0,05	0,22	0,67	1,86	2,66	0,82	1,68	2,50	2,01	3,23	4,01
Hel02	0,00	0,00	0,02	0,00	0,00	0,01	0,00	0,00	0,01	0,00	0,00	0,00
AVG	0,00	0,01	0,04	0,07	0,27	0,45	0,13	0,24	0,44	0,23	0,42	0,58

d. $DPSO_{(PTL)}$ and $DPSO_{VND(PTL)}$ are run on Intel Pentium IV, 3.0 GHz processor and 512 MB memory [23].

e. $GA\text{-}VNS_{(JES)}$ is run on Intel Pentium IV, 3.2 GHz processor and 512 MB memory [21].

B. Experiment results for Taillard's instances

We can find the instances of Taillard [43] on the website "OR-Library." We considered 11 different kinds of problems. Each type of problem possesses 10 problem instances. The number of jobs *n* is equal to 20, 50, 100 and 200 and the number of machines *m* is equal to 5, 10 and 20. In this paper, we propose new upper bounds for the makespan criterion. The results of Table V show us that NEW-GRASP-DE algorithm improved the best known solutions for 48 of the 110 tested instances. Our best results are presented in bold type. This means that we have improved the results by 43.64% compared to the results of [21]. Table VI presents our results in term of deviation from the results of [21]. We conclude that our algorithm is much more efficient than $(GA\text{-}VNS)_{(JES)}$ algorithm of [21].

The deviation of each algorithm is calculated by the following formula:

$$dev = \frac{(M - UB)}{UB} \times 100 \qquad (9)$$

M denotes the Makespan obtained by each algorithm for each given problem instance and *UB* denotes the corresponding upper bound. According to the experimental results presented in Table VI, we can see that our algorithm is more efficient than the algorithm relative to [21].

TABLE V. UPPER BOUND OF MAKESPAN FOR TAILLARD'S INSTANCES

Instance	**Up.B**	**Instance**	**Up.B**	**Instance**	**Up.B**	**Instance**	**Up.B**
tai001	1486	tai031	**3160**	tai061	**6370**	tai091	15319
tai002	1528	tai032	3432	tai062	**6224**	tai092	**15078**
tai003	1460	tai033	**3210**	tai063	**6115**	tai093	**15350**
tai004	1588	tai034	**3338**	tai064	**6008**	tai094	**15155**
tai005	1449	tai035	3356	tai065	**6186**	tai095	**15181**
tai006	1481	tai036	**3346**	tai066	**6057**	tai096	**15066**
tai007	1483	tai037	3231	tai067	**6234**	tai097	**15378**
tai008	1482	tai038	3235	tai068	**6123**	tai098	**15210**
tai009	1469	tai039	**3070**	tai069	**6366**	tai099	**15073**
tai010	1377	tai040	3317	tai070	**6370**	tai100	**15282**
tai011	2044	tai041	4274	tai071	**8077**	tai101	**19667**
tai012	2166	tai042	4177	tai072	**7880**	tai102	**20070**
tai013	1940	tai043	4099	tai073	8028	tai103	**19892**
tai014	1811	tai044	4399	tai074	**8338**	tai104	**19908**
tai015	1933	tai045	4322	tai075	**7950**	tai105	**19790**
tai016	1892	tai046	4289	tai076	**7792**	tai106	**19910**
tai017	1963	tai047	4420	tai077	**7864**	tai107	**20077**
tai018	2057	tai048	4318	tai078	**7903**	tai108	**19989**
tai019	1973	tai049	4155	tai079	**8147**	tai109	**19898**
tai020	2051	tai050	4283	tai080	**8108**	tai110	**19892**
tai021	2973	tai051	6129	tai081	**10697**		
tai022	2852	tai052	5725	tai082	10594		
tai023	3013	tai053	5862	tai083	10611		
tai024	3001	tai054	5788	tai084	10607		
tai025	3003	tai055	5886	tai085	10539		
tai026	2998	tai056	5863	tai086	**10667**		
tai027	3052	tai057	5962	tai087	**10831**		
tai028	2839	tai058	5926	tai088	**10829**		
tai029	3009	tai059	5876	tai089	**10718**		
tai030	2979	tai060	5958	tai090	10798		

TABLE VI. COMPARISON OF NEW-GRASP-DE WITH $GA\text{-}VNS_{(JES)}$ FOR TAILLARD'S INSTANCES

	$GA\text{-}VNS_{(JES)}$			**NEW-GRASP-DE**		
n×m	**min**	**avg**	**max**	**min**	**avg**	**max**
20×05	0,00	0,03	0,10	0,00	0,03	0,15
20×10	0,00	0,06	0,15	0,00	0,02	0,11
20×20	0,00	0,03	0,07	0,00	0,00	0,00
50×05	0,17	0,55	0,94	-0,01	0,19	0,51
50×10	0,09	0,47	0,87	0,01	0,23	0,61
50×20	0,07	0,47	1,04	0,04	0,31	0,67
100×05	0,42	0,84	1,27	-0,29	-0,13	0,05
100×10	0,43	0,85	1,32	-0,10	0,14	0,51
100×20	0,42	0,91	1,44	0,03	0,38	0,83
200×10	1,16	1,68	2,24	-0,33	-0,06	0,22
200×20	1,08	1,52	1,98	-0,24	0,08	0,41
AVG	0,35	0,67	1,04	-0,08	0,11	0,37

IX. CONCLUSION

In this paper, we developed a NEW-GRASP-DE algorithm so that resolving the NWFSSP. The objective is to minimize the makespan. Our proposed approach is based on the hybridization of GRASP method with DEA. Furthermore, we used the ILS algorithm as an improvement phase in the GRASP method. The proposed algorithm has been tested on 133 benchmarks that have been proposed in the literature.

We can see that our algorithm is very competitive and it outperformed other algorithms according to the experimental results above cited. Besides, when we consider the instances of Taillard [43], we show that our approach is able to improve 48 instances out of 110 instances. In addition, our computational time is much less than other works described in the literature.

Finally, as future avenues of research, we first propose to hybridize the GRASP method with other algorithms such as the Particle Swarm Optimization or the Estimation of Distribution Algorithm. Second, we can use different algorithms for the improvement phase of GRASP method such as Tabu Search Algorithm or Simulated Annealing Algorithm.

REFERENCES

[1] D. A. Wismer, "Solution of the flowshop scheduling problem with no intermediate queues," Oper. Res., vol. 20, pp. 689–697, 1972.

[2] C. Rajendran, "A no-wait flowshop scheduling heuristic to minimize makespan," J. Oper. Res. Soc., vol. 45, pp. 472–478, 1994.

[3] J. Grabowski and J. Pempera, "Sequencing of jobs in some production system," Eur. J. Oper. Res., vol. 125, pp. 535–550, 2000.

[4] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," Oper. Res., vol. 44, pp. 510–525, 1996.

[5] W. H. M. Raaymakers and J. A. Hoogeveen, "Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing," Eur. J. Oper. Res., vol. 126, pp. 131–151, 2000.

[6] S. Reddi and C. Ramamoorthy, "On the flowshop sequencing problem with no-wait in process," Oper. Res. Q., vol. 23, pp. 323–331, 1972.

[7] H. Röck, "The three-machine no-wait flow shop is NP-complete," J. Assoc. Comput. Mach., vol. 31, pp. 336–345, 1984.

[8] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," Ann. Discrete Math., vol. 5, pp. 287–326, 1979.

[9] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," J. Glob. Optim., vol. 6, pp. 109–133, 1995.

[10] R. Storn and K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," J. Glob. Optim., vol. 11, pp. 341–359, 1997.

[11] T. Stützle, "Applying iterated local search to the permutation flow shop problem," Technical Report AIDA–98–04, FG Intellektik, TU Darmstadt, 1998.

[12] T. Aldowaisan, "A new heuristic and dominance relations for no-wait flowshops with setups," Comput. Oper. Res., vol. 28, pp. 563–584, 2001.

[13] R. Gangadharan and C. Rajendran, "Heuristic algorithms for scheduling in the no-wait flowshop," Int. J. Prod. Econ., vol. 32, pp. 285–290, 1993.

[14] D. Laha and U. K. Chakraborty, "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling," Int. J. Adv. Manuf. Tech., vol. 41, pp. 97–109, 2009.

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

[16] T. Aldowaisan and A. Allahverdi, "New heuristics for no-wait flowshops to minimize makespan," Comput. Oper. Res., vol. 30, pp. 1219–1231, 2003.

[17] K. Chakravarthy and C. Rajendran, "A heuristic for scheduling in a fowshop with the bicriteria of makespan and maximum tardiness minimization," Prod. Plan. Control, vol. 10, pp. 707–714, 1999.

[18] C. J. Schuster and J. M. Framinan, "Approximative procedure for no-wait job shop scheduling," Oper. Res. Lett., vol. 31, pp. 308–318, 2003.

[19] J. Grabowski and J. Pempera, "Some local search algorithms for no-wait flow-shop problem with makespan criterion," Comput. Oper. Res., vol. 32, pp. 2197–2212, 2005.

[20] 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," Int. J. Adv. Manuf. Tech., vol. 38, pp. 778–786, 2008.

[21] B. Jarboui, M. Eddaly, and P. Siarry, "A hybrid genetic algorithm for solving no-wait flowshop scheduling problems," Int. J. Adv. Manuf. Tech., vol. 54, pp. 1129–1143, 2011.

[22] C. L. Chen, R. V. Neppalli, and N. Aljaber, "Genetic algorithms applied to the continuous flow shop problem," Comput. Ind. Eng., vol. 30, pp. 919–929, 1996.

[23] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," Comput. Oper. Res., vol. 35, pp. 2807–2839, 2008.

[24] D. Davendra, I. Zelinka, M. Bialic-Davendra, R. Senkerik, and R. Jasek, "Discrete Self-Organising Migrating Algorithm for flow-shop scheduling with no-wait makespan," Math. Comput. Model., vol. 57, pp. 100–110, 2013.

[25] I. Zelinka and J. Lampinen, "Soma – Self-Organizing Migrating Algorithm," in Proceedings of the 6th International Conference on Soft Computing (Mendel), Brno, Czech Republic, pp. 177–187, 2000.

[26] L. Wang and D. Zheng, "An effective hybrid optimization strategy for job-shop scheduling problems," Comput. Oper. Res., vol. 28, pp. 585–596, 2001.

[27] B. Gonazelez, M. Torres, and J. A. Moreno, "A hybrid genetic algorithm approach for the ''no-wait'' flowshop scheduling problem," in Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, London, UK, pp. 59–64, 1995.

[28] B. Liu, L. Wang, and Y. H. Jin, "An effective hybrid particle swarm optimization for no-wait flow shop scheduling," Int. J. Adv. Manuf. Tech., vol. 31, pp. 1001–1011, 2007.

[29] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," IEEE Trans. Evolut. Comput., vol. 8, pp. 99–110, 2004.

[30] Q. K. Pan, L. Wang, M. F. Tasgetiren, and B. H. Zhao, "A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion," Int. J. Adv. Manuf. Tech., vol. 38, pp. 337–347, 2008.

[31] B. Qian, L. Wang, R. Hu, D. X. Huang, and X. Wang, "A DE-based approach to no-wait flow-shop scheduling," Comput. Ind. Eng., vol. 57, pp.787–805, 2009.

[32] L. Y. Tseng and Y. T. Lin, "A hybrid genetic algorithm for no-wait flowshop scheduling problem," Int. J. Prod. Econ., vol. 128, pp. 144–152, 2010.

[33] C. Rajendran and D. Chaudhuri, "Heuristic algorithms for continuous flow-shop problem," Nav. Res. Log., vol. 37, pp. 695–705, 1990.

[34] E. Bertolissi, "Heuristic algorithm for scheduling in the no-wait flow-shop," J. Mater. Process. Tech., vol. 107, pp. 459–465, 2000.

[35] A. Fink and S. Voβ, "Solving the continuous flowshop scheduling problem by metaheuristics," Eur. J. Oper. Res., vol. 151, pp. 400–414, 2003.

[36] K. Z. Gao, Q. K. Pan, and J. Q. Li, "Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion," Int. J. Adv. Manuf. Tech., vol. 56, pp. 683–692, 2011.

[37] J. Czogalla and A. Fink, "Fitness landscape analysis for the no-wait flow-shop scheduling problem," J. Heuristics, vol. 18, pp. 25–51, 2012.

[38] C. Wang, X. Li, and Q. Wang, "Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion," Eur. J. Oper. Res., vol. 206, pp. 64–72, 2010.

[39] T. Aldowaisan and A. Allahverdi, "New heuristics for m-machine no-wait flowshop to minimize total completion time," Omega, vol. 32, pp. 345–352, 2004.

[40] S. J. Shyu, B. M. T. Lin, and P. Y. Yin, "Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time," Comput. Ind. Eng., vol. 47, pp. 181–193, 2004.

[41] C. R. Reeves, "A genetic algorithm for flowshop sequencing," Comput. Oper. Res., vol. 22, pp. 5–13, 1995.

[42] J. Heller, "Some numerical experiments for an M×J flow shop and its decision-theoretical aspects," Oper. Res., vol. 8, pp. 178–184, 1960.

[43] E. Taillard, "Benchmarks for basic scheduling problems," Eur. J. Oper. Res., vol. 64, pp. 278–285, 1993.