Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

A Discrete Differential Evolution Algorithm for the No-Wait Flowshop Scheduling Problem with Total Flowtime Criterion

M. Fatih Tasgetiren, Quan-Ke Pan, P. N. Suganthan, and Yun-Chia Liang

Abstract— **In this paper, a discrete differential evolution (DDE) algorithm is presented to solve the no-wait flowshop scheduling problem with the total flowtime criterion. The DDE algorithm is hybridized with the variable neighborhood descent (VND) algorithm to solve the well-known benchmark suites in the literature. The DDE algorithm is applied to the 110 benchmark instances of Taillard [1] by treating them as the no-wait flowshop problem instances with the total flowtime criterion. The solution quality is evaluated with optimal solutions, lower bounds and best known solutions provided by Fink & Voß [2]. The computational results show that the DDE algorithm generated better results than those in Fink & Voß [2].**

I. Introduction

Among all types of scheduling problems, no-wait flowshop has important applications in different industries including chemical processing by Rajendran [3], food processing by Hall & Sriskandarayah [4], concrete ware production by Grabowski & Pempera [5], and pharmaceutical processing by Raaymakers & Hoogeveen [6]. In a no-wait flowshop, each of *n* jobs consists of *m* operations owning a predetermined processing order through machines. Each job is to be processed without preemption and interruption on or between *m* machines. That is, once a job is started on the first machine, it has to be continuously processed through machines without interruption. In addition, each machine can handle no more than one job at a time and each job has to visit each machine exactly once. Therefore, when needed, the start of a job on the first machine must be delayed in order to meet the no-wait requirement. Given that the release time of all jobs is zero and set-up time on each machine is included in the processing time, the no-wait flowshop problem is to schedule jobs that minimize the makespan or total flowtime over all jobs. For the computational complexity of the no-wait flowshop scheduling problem, Garey and Johnson [7] proved that it is NP-Hard. Therefore, only small-sized instances of the no-wait flowshop problem can be solved optimally with reasonable computational time using exact algorithms. When the problem size increases, the computational time of exact methods grows exponentially. On the other hand, heuristic algorithms have generally acceptable time and memory requirements to reach a near-optimal or optimal solution. In past decades, most research focused on developing heuristic algorithms. These solution techniques can be broadly classified into two groups referred to as constructive method and improvement method. In the first group, heuristics were introduced for the makespan criterion by Bonney & Gundry [8], King & Spacins [9], Gangadharan & Rajendran [10], and Rajendran [3]. As for the flowtime objective, Rajendran and Chaudhuri [11] proposed a simple construction heuristic with two priority rules, and Fink & Voß [2] used several simple construction heuristics such as nearest neighbor (NN), cheapest insertion (Chins), and pilot method (Pilot). The second group has grown quickly with the development of computer technology. The literature related to the makespan criterion consists of simulated annealing (SA) by Aldowaisan & Allahverdi [12], genetic algorithm (GA) by Aldowaisan & Allahverdi [12], hybrid GA and SA (GASA) by Schuster & Framinan [13], variable neighbourhood search (VNS) by Schuster & Framinan [13], descending search (DS) by Grabowski & Pempera [14], and tabu search (TS) by Grabowski & Pempera [14]. When considering the total flowtime criterion, the literature is limited. However, Chen et al. [15] used a GA, Fink & Voß [2] presented SA and TS algorithms and Pan et al. [16] proposed a discrete particle swarm optimization (DPSO) algorithm to test a set of benchmark problems proposed by Taillard [1]. In addition, a comprehensive survey of the no-wait flowshop scheduling problem can be found in Hall & Sriskandarayah [4].

Differential evolution (DE) is one of the latest evolutionary optimization methods proposed by Storn and Price [17]. Like other evolutionary-type algorithms, DE is a population-based, stochastic global optimizer. In a DE algorithm, candidate solutions are represented as chromosomes based on floating-point numbers. In the mutation process of a DE algorithm, the weighted difference between two randomly selected population members is added to a third member to generate a mutated solution followed by a crossover operation to combine the mutated solution with the target solution so as to generate a trial solution. Then a selection operator is applied to compare the fitness values of both competing solutions, namely, target and trial solutions to determine the winner to survive for the next generation.

M. Fatih Tasgetiren is with the Department of Operations Management and Busısness Statistics, Sultan Qaboos University, P.O.Box 20, Al Khod 123, Muscat, Sultanate of Oman: mfatih@squ.edu.om

Quan-Ke Pan is with the College of Computer Science, Liaocheng, University, Liaocheng, Shandong Province, 252059, P. R. China; qkpan@lctu.edu.cn

P. N. Suganthan is with the School of Electrical and Electronic Engineering Nanyang Technological University, Singapore 639798; *epnsugan@ntu.edu.sg* .

Yun Chia-Liang is with the Department of Industrial Engineering and Management, Yuan Ze University, 135 Yuan-Tung Road, Chungli, Taoyuan, 320 Taiwan, R.O.C. ycliang@saturn.yzu.edu.tw

Since DE was first introduced to solve the Chebychev polynomial fitting problem by Storn and Price [17, 18], it has been successfully applied to solve a variety of applications by Corne et al. [19], Lampinen [20], Babu and Onwubolu [21], and Price et al. [22].

The applications of DE on combinatorial optimization problems are still considered limited, but the advantages of DE include a simple structure, immediately accessible for practical applications, ease of implementation, speed to acquire solutions, and robustness as demonstrated in the literature. However, the major obstacle of successfully applying a DE algorithm to solve combinatorial problems in the literature is due to its continuous nature. To remedy this drawback, this research proposes a novel discrete differential evolution (DDE) algorithm to solve the no-wait flowshop scheduling problem with the objective of minimizing total flowtime.

The paper is organized as follows. Section II introduces the no-wait flowshop scheduling problem. Section III gives the details of the proposed DDE and VND algorithms. The computational results over benchmark problems are discussed in Section IV. Finally, Section V summarizes the concluding remarks.

II. NO-WAIT FLOWSHOP SCHEDULING PROBLEM

The no-wait flowshop scheduling problem can be described as follows: Given the processing times $p(j,k)$ for job *j* and machine *k*, each of *n* jobs $(j = 1,2,...,n)$ will be sequenced through *m* machines $(k = 1,2,...,m)$. Each job *j* has a sequence of *m* operations $(o_{j1}, o_{j2},..,o_{jm})$. To satisfy the no-wait restriction, the completion time of the operation o_{jk} must be equal to the earliest time to start of the operation $o_{j,k+1}$ for $k = 1,2,..,m-1$. In other words, there must be no waiting time between the processing of any consecutive operation of each of *n* jobs. The problem is then to find a schedule such that the processing order of jobs is the same on each machine and the maximum completion time or total flowtime is minimized.

Suppose that the job permutation $\pi = \{\pi_1, \pi_2,..., \pi_n\}$ represents the schedule of jobs to be processed. Let $d(\pi_{j-1}, \pi_j)$ be the minimum delay on the first machine between the start of jobs π_j and π_{j-1} restricted by the no-wait constraint when the job π_j is directly processed after the job π_{j-1}. To compute the minimum delay, we propose an alternative approach. As shown in Fig. 1, the $2/m/P/C_{\max}$ problem is equavalent to the $2/m/no-wait\ \ P/C_{\max}$ problem. Let $M_{j-1,j}(\pi_i, m)$ denotes the makespan of two independent jobs in the permutation flowshop, $\forall(\pi_{j-1}, \pi_j) \in \Pi$ in the $2/m/P/C_{\max}$ problem. Since the makespan for the permutation flowshop and no-wait flowshop problems is the same as shown in Fig. 1, the minimum delay can be computed by substracting the total processing time of job π_j from the the makespan $M_{j-1,j}(\pi_j, m)$ as follows:

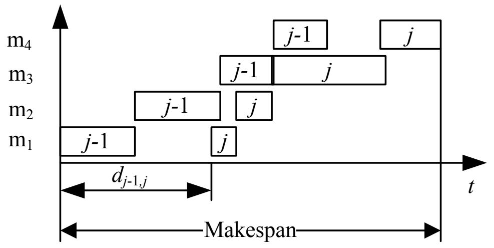
$$d(\pi_{j-1}, \pi_j) = M_{j-1,j}(\pi_j, m) - \sum_{k=1}^{m} p_{jk} \tag{1}$$

for $j = 2,..,n$. Then, the total flow time of *n* jobs can be given by

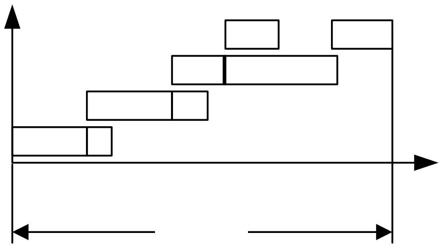
$$TF(\pi) = \sum_{j=2}^{n} (n+1-j) d(\pi_{j-1}, \pi_j) + \sum_{j=1}^{n} \sum_{k=1}^{m} p(j,k) \tag{2}$$

Therefore, the no-wait flowshop scheduling problem with the total flowtime criterion is to find a permutation π^* in the set of all permutations Π such that

$$TF(\pi^*) \le TF(\pi) \qquad \forall \pi \in \Pi. \tag{3}$$



a. No-wait flow shop with only two jobs



b. Permutation flow shop with only two jobs

Fig.1. Computation of Delay

III. DISCRETE DIFFERENTIAL EVOLUTION ALGORITHM

Currently, there exist several mutation variations in DE. The *DE/rand/1/bin* schemes of Storn and Price [17] is presented below. The DE algorithm starts with initializing the initial population with the size of *NP*. Each individual has an *n*-dimentional vector with parameter values determined randomly and uniformly between predefined search range.

To generate a mutated individual, the DE mutates vectors from the target population by adding the weighted difference between two randomly selected target population members to a third member as follows:

$$v_{ij}^t = x_{aj}^{t-1} + F\left(x_{bj}^{t-1} - x_{cj}^{t-1}\right) \tag{4}$$

where a, b, and c are three randomly chosen individuals from the population such that $\left(a \neq b \neq c \in (1,..,NP)\right)$. $F > 0$ is a mutation scale factor which affects the differential variation between two individuals.

Following the mutation phase, the crossover operator is applied to obtain the trial individual such that:

$$u_{ij}^{t} = \begin{cases} v_{ij}^{t}, if \quad r_{ij}^{t} \leq CR \quad or \quad j = D_j \\ x_{ij}^{t-1}, \qquad Otherwise \end{cases} \tag{5}$$

where D_j refers to a randomly chosen dimension (*j=1,..,n*), which is used to ensure that at least one parameter of each trial individual u_{ij}^{t} differs from its counterpart in the previous generation u_{ij}^{t-1}. CR is a user-defined crossover constant in the range [0, 1], and r_{ij}^{t} is a uniform random number between 0 and 1. In other words, the trial individual is made up with some parameters of mutant individual, or at least one of the parameters randomly selected, and some other parameters of the target individual.

To decide whether or not the trial individual u_{ij}^{t} should be a member of the target population for the next generation, it is compared to its counterpart target individual x_{ij}^{t-1} at the previous generation. The selection is based on the survival of the fitter one among the trial population and target population such that:

$$x_{ij}^{t} = \begin{cases} u_{ij}^{t}, if \quad f\left(u_{ij}^{t}\right) \leq f\left(x_{ij}^{t-1}\right) \\ x_{ij}^{t-1}, otherwise \end{cases} \tag{6}$$

The pseudo code of the DE algorithm is given in Fig. 2.

```
Initialize parameters
Initialize target population
Evaluate target population
Do {
    Obtain mutant population
    Obtain trial population
    Evaluate trial population
    Make selection
    Apply local search (optional)
While (Not Termination)
```

Fig. 2. Standard DE Algorithm.

It is obvious that standard DE equations cannot be used to generate a discrete job permutation since positions are real-valued. Instead we propose a DDE algorithm whose solutions are based on discrete job permutations. In the DDE algorithm, the target population is constructed based on the discrete job permutation as represented by $X_i = [X_1, X_2,,, X_{NP}]$. For the mutant population the following equations can be used:

$$V_i^t = m_1 \oplus F_p\left(X_i^{t-1}\right) \tag{7}$$

$$V_i^t = m_1 \oplus F_p\left(X_a^{t-1}\right) \tag{8}$$

$$V_i^t = m_1 \oplus F_p\left(G^{t-1}\right) \tag{9}$$

where X_a^{t-1} is randomly chosen individual from the target population; G^{t-1} is the global best solution; m_1 is the mutation probability; and F_p is the mutation operator with the mutation strength of p. In other words, a uniform random number *r* is generated between [0, 1]. If *r* is less than m_1 then the mutation operator with the mutation strength is applied to generate the mutant individual. In other words, mutation operator is applied p times. In this paper, the $V_i^t = m_1 \oplus F_p\left(G^{t-1}\right)$ version of mutation operators is employed in order to provide the target population with information about the global best. In the mutation equation, *k* represents the mutation strength, which is the key to the success of the algorithm. The higher the value of *p* is, the lower the possibility that the algorithm would escape from the local minima. On the other hand, the lower the value of *p* is, the higher the possibility that the algorithm would have excessive randomness. So care must be taken in the choice of the value of the mutation strength.

Following the mutation phase, the trial individual is obtained such that:

$$U_i^t = c_1 \oplus CR\left(X_i^{t-1}, V_i^t\right) \tag{10}$$

where *CR* is the crossover operator, and c_1 is the crossover probability. In other words, the *ith* individual is recombined with its corresponding mutant individual to generate the trial individual.

Finally, the selection is based on the survival of the fitter one among the trial and target solutions such that:

$$X_i^t = \begin{cases} U_i^t & if\ f\left(U_i^t\right) \leq f\left(X_i^{t-1}\right) \\ X_i^{t-1} & otherwise \end{cases} \quad 1 \leq i \leq NP \tag{11}$$

The two-cut PTL crossover proposed by Pan et al. [16] is used in the DDE algorithm.The PTL crossover is able to produce a pair of distinct permutations even from two identical parents. An illustration of the two-cut PTL crossover is shown in Table I.

TABLE I
PTL CROSSOVER OPERATOR

Two-Cut PTL Crossover						Two-Cut PTL Crossover					
P1	5	**1**	**4**	2	3	P1	5	**1**	**4**	2	3
P2	*3*	*5*	**4**	*2*	**1**	P2	*5*	**1**	**4**	*2*	*3*
O1	*3*	*5*	*2*	**1**	**4**	O1	*5*	*2*	*3*	**1**	**4**
O2	**1**	**4**	*3*	*5*	*2*	O2	**1**	**4**	*5*	*2*	*3*

In the PTL crossover, a block of jobs from the first parent is determined by two cut points randomly. This block is either moved to the right or left corner of the permutation. Then the offspring permutation is filled out with the remaining jobs from the second parent. This procedure will always produce two distinctive offspring even from the same two parents as shown in Table I. In this paper, one of these two unique offspring is chosen randomly with equal probability.

To demonstrate how the individuals are updated in the DDE algorithm, an example is illustrated in Table II2. Assume that the mutation and crossover probabilities are 1.0, two-cut PTL crossover operator and insert mutation operator are employed. Given the individual and the global best solution, the individual is first mutated. For example, in Table 2, the job 3 is inserted after the job 4 in the global best solution G, thus generating the mutant individual V_i. Then the individual V_i is recombined with its corresponding individual in the target population to generate the trial individual U_i. Finally, the target individual X_i is compared to the trial individual U_i to determine which one would survive for the next generation based on the survival of the fitter one.

TABLE II
DDE OPERATORS

X_i	3	1	4	2	5	
G	1	3	2	4	5	
Insert Mutation						
G	1	**3**	2	**4**	5	
V_i	1	2	**4**	**3**	5	
Two-Cut PTL Crossover						F
X_i	3	**1**	**4**	2	5	45
V_i	*1*	2	*4*	3	5	
U_i	2	3	5	**1**	**4**	40
Selection						
$f(U_i) = 40 < f(X_i) = 45 \quad X_i = U_i$,So						
X_i	2	3	5	**1**	**4**	

A. Initial Population

The initial population is constructed by two popular heuristics, namely the nearest neighbor (NN), and the NEH insertion heuristic of Nawaz et al. [23]. The NN heuristic appends at each step an unscheduled job with a minimal inevitable delay to the last job of the partial sequence unscheduled yet. On the other hand, the NEH heuristic has two phases. In phase I, jobs are ordered in descending sums of their processing times. In phase II, a job sequence is established by evaluating the partial schedules based on the initial order of the first phase. Suppose a current sequence is already determined for the first *k* jobs, *k*+1 partial sequences are constructed by inserting job *k*+1 in *k*+1 possible slots of the current sequence. Among the *k*+1 sequence, the one generating the minimum total flowtime is kept as the current sequence for the next iteration. Then job *k*+2 from phase I is considered and so on until all jobs have been sequenced.

The target population is constructed as follows: Starting from the first job of an identity permutation (1,2,...,*n*), the NN heuristic is applied to build a complete NN permutation; then the first phase of the NEH heuristic is ignored and the second phase of the NEH heuristic is applied to the NN permutation to generate a final permutation of the individuals to be included in the initial target population. We repeat the heuristics for all possible jobs in the identity permutation as the first job to construct the initial target population. By doing so, the diversity of the population is achieved. For this reason, the target population size is taken to be equal to the number of dimensions/jobs (*n*). The pseudo code used to generate the initial population is given in Figure 3.

```
$initpop()\{$
$i = 1$
 $do\{$
        $s_0 = \{1,2,..,n\}$
        $J_{(1)} = i$;
          $s_1 = s_0 - \{J_{(1)}\}$
$s_2 = J_{(1)} + NN(s_1)$
$s_3 = s_2 - \{J_{(1)} + J_{(2)}\}$
$X_i = \{J_{(1)} + J_{(2)}\} + NEH(s_3)$
$i = i + 1$
 $\}While(i \le NP)\ \}$
```

Fig. 3. Initial Population.

B. VND Local Search for DDE Algorithm

VNS is a recent meta-heuristic proposed by Mladenovic & Hansen [24] systematically exploiting the idea of neighborhood change, both in descent to local minima and in escape from the valleys containing them. Let N_k , $k = 1,2,..,k_{\max}$ be a set of neighborhood structures and $N_k(s)$ denote a set of solutions in the *kth* neighborhood of s . Then VNS systematically exploits the following observations:

1. A local minimum with respect to one neighborhood structure is not necessary so for another;
2. A global minimum is a local minimum with respect to all possible neighborhood structures;
3. For many problems, local minima with respect to one or several neighborhoods are relatively close to each other.

VND, a deterministic variant of VNS, is based on the observation (1) above. That is, a local optimum within the neighborhood $N_1(s)$ is not necessary one within the neighborhood $N_2(s)$. Thus, it may be advantageous to combine descent heuristics [24].

It should be noted that Tasgetiren et al. [25] recently developed a PSO algorithm for the unrestricted permutation flowshop sequencing problem with makespan and total flowtime criteria. In their PSO algorithm, the VNS algorithm was embedded in the PSO algorithm to improve the solution quality where detailed results have been reported on both the benchmark suites of Taillard [1] and Watson et al. [26], consisting of more than 14,000 problem instances in total. According to the results, the VNS version of the PSO algorithm was proven to be efficient and was able to improve the best known solutions of Taillard [1] and Watson et al. [26] for total flowtime and makespan criteria, respectively. Since the no-wait flowshop problem is a special case of the classical flowshop scheduling problem, the deterministic variant, VND, of the VNS algorithm was also embedded as a local search in the DDE algorithm in order to obtain competitive or better results than those recently reported in the literature.

It is obvious that the performance of the local search algorithms depends on the choice of neighborhood structure. For the no-wait flowshop scheduling problem, the following neighborhood structure was considered:

- Swap two jobs between η^{th} and κ^{th} dimensions, $\eta \neq \kappa$ (*Swap*)
- Remove the job at the η^{th} dimension and insert it in the κ^{th} dimension $\eta \neq \kappa$ (*Insert*)

The sequence of VND neighborhood structures is chosen as *Swap+Insert*. It should be noted that the VND local search for the no-wait flowshop scheduling problem was only applied to the global best solution, G^t at each iteration *t*. The pseudo code of the VND local search is given in Fig. 4.

```
$VND\{$
$s_0 = G^t$
$Choose\, N_K, \quad k = 1,..,k_{\max}$
$s = perturbation(s_0)$
$k \leftarrow 1$
$do\{$
$s_1 \leftarrow N_k(s)$
$if \quad f(s_1) \leq f(s)\{$
    $s \leftarrow s_1$
    $k \leftarrow 1$ $\}$
$else$
    $k \leftarrow k + 1$
 $\}while(k \leq k_{\max})$
    $if \quad f(s) \leq f(G^t) \quad then \quad G^t = s$
    $else \quad if \quad random < \exp(-(f(s) - f(G^t))/T) \quad then$
        $G^t = s$ $\}$
```

Fig. 4. VND Local Search.

In the VND algorithm in Fig. 4, the global best solution G^t at each iteration *t* is assigned to the incumbent solution s_0. After choosing the neighborhood structure N_k (*Swap+Insert*), the incumbent solution was perturbed to avoid getting trapped at a local minimum. The perturbation strength was 5 insertions to the global best solution. Then the VND algorithm was applied to the perturbed solution s. Furthermore, $N_k(s)$ is concerned with applying the neighborhood structure N_k to the perturbed solution *s*. For the swap neighborhood, the size of the local search is $n(n-1)/2$ whereas it is $(n-1)^2$ for the insert neighborhood. It should be noted that the complete search of both neighborhood structure in a VND algorithm consumes significant amount of CPU time. In order to accelerate the search process in the VND algorithm, the speed-up methods proposed in Pan et al. [16] are used to compute the total flowtime. The speed-up methods proposed in Pan et al. [16] are significant contributions to the no-wait flowshop literature and they have enhanced the performance of the VND local search significantly.

In addition, a constant temperature is used in the simulated annealing type of acceptance criterion in the VND algorithm as suggested by Osman and Potts [27]:

$$T = \frac{\sum_{j=1}^{n} \sum_{k=1}^{m} p_{jk}}{n * m} * h \qquad (12)$$

where $h = 2.9$. In this way, the global best solution is diversified by giving chances to some inferior solutions during the search to escape from the local minima.

For simplicity, the DDE algorithm with the VND local search is denoted as DDE_{VND} throughout the paper from now on.

IV. EXPERIMENTAL RESULTS

The DDE and DDE_{VND} algorithms for the no-wait flowshop scheduling problem were coded in Visual C++ and run on an Intel P IV 3.0 GHz PC with 512MB memory. Regarding the parameters of the DDE and DDE_{VND} algorithms, m_1 and c_1 are taken as 0.8. Insert mutation operator (*F*) with the mutation strength of *p*=3 is used. As a crossover operator, the two-cut PTL is employed. Finally, the population size was the number of dimensions/jobs due to the nature of the construction of the initial population. The DDE and DDE_{VND} algorithms were applied to the 110 benchmark instances of Taillard [1] by treating them as no-wait flowshop problem instances with the total flowtime criterion. The solution quality was evaluated with the optimal solutions, lower bounds, and best known solutions provided by Fink & Voß [2]. The maximum number of generations was carefully set to 1000 for the DDE and DDE_{VND} algorithms with the total flowtime criterion.

R=5 runs were conducted for each problem instance consistent with the Fink & Voß [2] and each run was compared to the optimal solution, or the lower bound, or the best known solution reported by Fink & Voß [2]. The average relative percent deviation denoted by Δ_{avg} and the average CPU time to reach the best objective function value in each run averaged over *R* runs denoted by t_{avg} were given as statistics for performance measures. The average relative percent deviation was computed as follows:

$$\Delta_{avg} = \sum_{i=1}^{R} \left(\frac{(TF_i - TF_{ref}) * 100}{TF_{ref}} \right) / R \qquad (13)$$

where TF_i was the total flowtime generated by the DDE or DDE_{VND} algorithm in each run whereas TF_{ref} was the value of optimal, lower bound or best known total flowtime reported by Fink & Voß [2], and *R* was the number of runs. The CPU time requirements were compared using the conversion factor since different machines with different speeds are used.

Fink & Voß [2] have presented some construction methods and metaheuristics for the no-wait flowshop scheduling problem with the total flowtime criterion. Construction methods presented were the nearest neighbor (NN), the cheapest insertion (Chins), and the pilot (Pilot) heuristics whereas metaheuristics investigated were steepest descent (SD), iterated steepest descent (ISD), simulated annealing (SA), and tabu search (TS) algorithms. See Fink & Voß [2] for the details of construction methods and metaheuristics. The applications of above heuristics for the no-wait flowshop scheduling problem were based on *HotFrame*, a heuristic optimization tool developed by Fink & Voß [28]. All heuristics have been applied to the benchmark suite of Taillard [1], originally generated for the unrestricted permutation flowshop sequencing problem. The benchmark suite is available in OR library and contains problem instances with 20 (ta001-ta030), 50 (ta031-ta060), 100 (ta061-ta090), and 200 (ta091-ta110) jobs. Taillard's instances are originally given in the number of machines. However, Fink & Voß [2] presented computational results as the average for problem instances of the same number of jobs. Run times were also averaged in seconds. All computations were conducted using a Pentium II with a 266 MHz clock. In order to evaluate the performance of heuristics, the 3-index formulation of Picard & Queyranne [29] was used to compute optimal results for the problem instances with 20 jobs and to compute lower bounds provided by linear programming (LP) relaxation for problem instances with 50 and 100 jobs. Unfortunately, Fink & Voß [2] were not able to solve the LP relaxation of the problem instances with 200 jobs. For this reason, for n=20, Fink & Voß [2] compared to optimal results, for n=50 and n=100 to lower bounds, and for n=200 to the best results obtained during all experiments. The best objective function value obtained for each problem instance was also provided by Fink & Voß [2]. In addition, Fink & Voß [2] provided a detailed analysis of construction methods, different neighborhood structures embedded in SD, ISD, SA and TS algorithms. According to the results in Fink & Voß [2] , SA and reactive tabu search (RTS) algorithms generated better results with a 1000 second CPU time in connection with shift (insert) neighborhood on the basis of initial solutions provided by Chins and Pilot-10 heuristics. It should be noted that the DPSO and DPSO$_{VND}$ algorithms in Pan et al. [16] were applied to the same set of benchmark suite and compared to the results in Fink & Voß [2].

TABLE III
COMPARISON OF RESULTS (Δ_{avg})

	SA		TS(Chins)		TS(Pilot 10)		DPSO	
Jobs	Avg	CPU	Avg	CPU	Avg	CPU	Avg	CPU
20	0.00	7.0	0.00	1000	0.00	1000	1.18	0.00
50	0.98	44.0	0.88	1000	0.74	1000	4.33	0.03
100	2.64	196.4	2.20	1000	1.88	1000	6.61	0.14
200	1.00	982.1	1.19	1000	0.08	1000	4.11	0.74

TABLE III. CONTINUED

	DPSO$_{VND}$		DDE		DDE$_{VND}$	
Jobs	Avg	CPU	Avg	CPU	Avg	CPU
20	0.00	0.00	0.91	0.01	0.00	0.00
50	0.57	0.36	4.28	0.03	0.47	0.50
100	1.15	3.61	6.54	0.13	1.04	4.11
200	-1.38	27.78	3.94	0.75	-1.49	28.89

The DDE and DDE$_{VND}$ algorithms were applied to the benchmark suite of Taillard [1] to be compared to the computational results provided by Fink & Voß [2]. However, lower bounds were not reported in the paper by Fink & Voß [2]. In order to have a fair comparison, the lower bounds are obtained through personal communication. Then the DDE and DDE$_{VND}$ algorithms were compared to the optimal solutions for n=20, to the lower bounds for n=50 and n=100, and to the best known solutions for n=200 as in Fink & Voß [2]. The maximum number of generations was fixed at 1000 generations.

Table III summarizes the computational results for the DDE and DDE$_{VND}$ algorithms to be compared to the DPSO and DPSO$_{VND}$ algorithms in Pan et al. [16] and SA, TS (Chins) and TS (Pilot 10) in Fink & Voß [2]. From Table III, it is obvious that the DPSO and DDE algorithm were not able to compete with SA, TS (Chins), and TS (Pilot-10) algorithms owing to the fact that the results of the DPSO and DDE algorithms were based on a short run of 1000 generations taking 0.74 and 0.75 seconds for 200-job problems, respectively. However, the inclusion of the VND algorithm in the DPSO and DDE algorithms has significantly improved the solution quality. In order to compare the DDE$_{VND}$ algorithm to those in Fink & Voß [2] and in Pan et al. [16], we conducted the paired t-tests based on the best known solutions presented by Fink & Voß [2]. The paired t-test results are given in Table IV, V and VI, respectively. In addition, the new best known solutions generated by DDE$_{VND}$ are given in Table VII on which the paired t-test comparisons are made.

As seen in Table IV, the p-value is zero so the null hypothesis was rejected on the behalf of the DPSO$_{VND}$ algorithm. It indicates that the difference in the total flowtime generated between two algorithms was meaningful at the significance level of 0.95. For this reason, it can be concluded that the DPSO$_{VND}$ algorithm was superior to those presented in Fink & Voß [2]. In addition, the DPSO$_{VND}$ algorithm was able to find all the optimal solutions for n=20, and to improve the best known solutions for n=50, n=100 and n=200. The new best known solutions collected from 5 runs were given in Pan et al. [16]. Totally, 74 out of 80 best known solutions provided by Fink & Voß [2] were ultimately improved by the DPSO$_{VND}$ algorithm.

Table V shows the paired t-test results for the DDE$_{VND}$ versus Fink & Voß [2]. the p-value is zero so the null hypothesis was rejected on the behalf of the DDE$_{VND}$ algorithm. It implies that the difference in the total flowtime of solutions generated by both algorithms was meaningful at the significance level of 0.95. For this reason, it can be concluded that the DDE$_{VND}$ algorithm was superior to those presented in Fink & Voß [2]. In addition, the DDE$_{VND}$ algorithm was able to find all the optimal solutions for n=20, and to improve the best known solutions for n=50, n=100 and *n*=200. The new best known solutions collected from 5 runs were given in Table VII. In total, 77 out of 80 best known solutions provided by Fink & Voß [2] were ultimately improved by the DDE_{VND} algorithm.

When comparing the DDE_{VND} to the $DPSO_{VND}$ algorithm, the *p* value is 0.011 which is less than 0.05 indicating that the difference in total flowtime generated by both algorithms was meaningful at the significance level of 0.95. However, it was not meaningful at the significance level of 0.99. The permutations obtained for the optimal and new best known solutions are available upon request.

In terms of the CPU time requirement, even though we employed a machine approximately 11.28 times (3000/266=11.28) faster than the one used by Fink & Voß [2], the DDE_{VND} algorithm was much faster than those in Fink & Voß [2] because 28.89*11.26=325.3015 seconds were much smaller than 1000 seconds for *n*=200 in Fink & Voß [2]. It was due to the fact that the speed-up methods presented in Pan et al. [16] have led the DDE_{VND} algorithm to consume less CPU times.

TABLE IV

PAIRED-T TEST FOR H_O:$DPSO_{VND}$=F&V VS H_1:DDE_{VND} # F&V

	N	Mean	StDev	SE Mean
$DPSO_{VND}$	110	472275	637377	60771
Fink & Voß	110	478400	647517	61738
Difference	110	-6125	10815	1031

95% CI for mean difference: (-8168, -4081)

T-Test of mean difference = 0 (vs not = 0): T-Value = -5.94 P-Value = 0.000

TABLE V

PAIRED-T TEST FOR H_O:DDE_{VND}=F&V VS H_1:DDE_{VND} # F&V

	N	Mean	StDev	SE Mean
DDE_{VND}	110	471852	636818	60718
Fink & Voß	110	478400	647517	60718
Difference	110	-6548	11518	1098

95% CI for mean difference: (-8724, -4371)

T-Test of mean difference = 0 (vs not = 0): T-Value = -5.96 P-Value = 0.000

TABLE VI

PAIRED-T TEST FOR H_O:DDE_{VND}=DPSOVND VS H_1:DDE_{VND} # $DPSO_{VND}$

	N	Mean	StDev	SE Mean
DDE_{VND}	110	471852	636818	60718
$DPSO_{VND}$	110	472275	637377	60771
Difference	110	-423	1716	164

95% CI for mean difference: (-747, -99)

T-Test of mean difference = 0 (vs not = 0): T-Value = -2.58 P-Value = 0.011

V. CONCLUSIONS

DE is one of the recent evolutionary optimization methods. It has been widely used in a wide range of applications. To the best of our knowledge, this is the first reported application of DDE and DDE_{VND} algorithms to the no-wait flowshop scheduling problem with the total flowtime criterion. Unlike the standard DE, the DDE algorithm employs a permutation representation and work on the discrete domain. The DDE algorithm is also hybridized with the VND local search to solve well-known benchmark suites in the literature. The DDE and DDE_{VND} algorithms were applied to the 110 benchmark instances of Taillard [1] by treating them as the no-wait flowshop problem instances with the total flowtime criterion.

The computational results show that the proposed DDE and DDE_{VND} algorithms outperformed the metaheuristic algorithms presented by Fink & Voß [2] for the total flowtime criterion. In addition, it generated slightly better results than the DPSO algorithm. Besides finding all the optimal solutions for *n*=20 problems, 77 out of 80 best known solutions for the total flowtime criterion reported by Fink & Voß [2] were ultimately improved by the proposed DDE_{VND} algorithm. It should be noted that the inclusion of the VND local search and the speed-up methods in the DDE algorithm has enhanced the solution quality significantly.

As the future work, the proposed DDE algorithm will be applied to a variety of combinatorial optimization problems in the literature.

Acknowledgement: Dr P. N. Suganthan acknowledges the financial support offered by the A*Star (Agency for Science, Technology and Research) under the grant # 052 101 0020.

REFERENCES

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

[2] Fink A, Voß S. Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research* 2003;151:400-14.

[3] Rajendran C. A no-wait flowshop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society* 1994;45:472-8.

[4] Hall NG, Sriskandarayah C. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research* 1996; 44:510-25.

[5] Grabowski J., Pempera J. Sequencing of jobs in some production system. *European Journal of Operational Research* 2000;125:535-50.

[6] Raaymakers W, Hoogeveen J. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing, *European Journal of Operational Research* 2000; 126:131-51.

[7] Garey MR, Johnson DS. *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco, 1979.

[8] Bonney MC, Gundry SW. Solutions to the constrained flowshop sequencing problem. *Operational Research Quarterly* 1976; 24:869-83.

[9] King JR, Spachis AS. Heuristics for flowshop scheduling. *International Journal of Production Research* 1980; 18:343-57.

[10] Gangadharan R, Rajedran C. Heuristic algorithms for scheduling in no-wait flowshop. *International Journal of Production Economics* 1993;32:285-90.

[11] Rajendran C, Chaudhuri D. Heuristic algorithms for continuous flow-shop problem. *Naval Research Logistics* 1990, 37, 695-705.

[12] Aldowaisan T, Allahverdi A. New heuristics for no-wait flowshops to minimize makespan. *Computers and Operations Research* 2003;30:1219-31.

[13] Schuster C J, Framinan J M. Approximative procedure for no-wait job shop scheduling. *Operations Research Letters* 2003;31:308-18.

[14] Grabowski J, Pempera J. Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers and Operations Research* 2005; 32:2197-212.

[15] Chen C-L, Neppalli R V, Aljaber N. Genetic algorithms applied to the continuous flowshop problem. *Computers and Industrial Engineering*, 1996, 30, 919-929.

[16] Pan Q-K, Tasgetiren M. F, Liang Y-C, 2005, "A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem with Makespan and Total Flowtime Criteria", Accepted to "Bio-inspired metaheuristics for combinatorial optimization problems". Special issue of *Computers & Operations Research*, 2005 .

[17] Storn, R. and Price, K. (1995) "Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," *Technical Report TR-95-012*, ICSI, 1995.

[18] Storn, R. and Price, K. (1997) "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space," *Journal of Global Optimization*, vol. 11, pp. 341-359.

[19] Corne, D., Dorigo, M., and Glover, F. (eds.) (1999) "*Part Two: Differential Evolution," New Ideas in Optimization*, McGraw-Hill, pp. 77-158.

[20] Lampinen, J. (2001) "A Bibliography of Differential Evolution Algorithm," *Technical Report*, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing.

[21] Babu, B. V. and Onwubolu, G. C. (eds.) (2004) *New Optimization Techniques in Engineering*, Springer Verlag.

[22] Price, K., Storn, R., and Lampinen, J. (2006) *Differential Evolution – A Practical Approach to Global Optimization*, Springer-Verlag.

[23] Nawaz M, Enscore E. E Jr., Ham I, A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, *OMEGA* 11(1), (1983) 91-95.

[24] Mladenovic N, Hansen P, Variable neighborhood search, *Computers and Operations Research* 24 (1997) 1097-1100.

[25] Tasgetiren M. F., Yun-Chia Liang, Sevkli M., Gencyilmaz G, Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in Permutation Flowshop Sequencing Problem, *European Journal of Operational Research,* 2005, in press.

[26] Watson J P, Barbulescu L, Whitley L D, Howe A E, Contrasting structured and random permutation flowshop scheduling problems: search space topology and algorithm performance, *ORSA Journal of Computing* 14(2) (2002) 98-123.

[27] I. Osman, C. Potts, Simulated annealing for permutation flow shop scheduling, *OMEGA* 17(6) (1989) 551-557.

[28] Fink A, Voß S. *HotFrame: Aheuristic optimization framework*, in: Voß S., Woodruff D (Eds), Optimization Software Class Libraries. Kluwer, Boston, 2002, pp. 81-154.

[29] Picard J-C, Queyranne M. The time dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 1978, 26, 86-110.

TABLE VII
NEW BEST KNOWN SOLUTIONS GENERATED BY DDE$_{VND}$

Instance	F&V	DDE$_{VND}$	Instance	F&V	DDE$_{VND}$	Instance	F&V	DDE$_{VND}$	Instance	F&V	DDE$_{VND}$
Ta001	15674	15674	Ta031	76016	**75688**	Ta061	308052	**304801**	Ta091	1521201	**1494498**
Ta002	17250	17250	Ta032	83403	**82874**	Ta062	302386	**297712**	Ta092	1516009	**1476685**
Ta003	15821	15821	Ta033	78282	**78103**	Ta063	295239	**291175**	Ta093	1515535	**1492717**
Ta004	17970	17970	Ta034	82737	**82422**	Ta064	278811	**275783**	Ta094	1489457	**1462173**
Ta005	15317	15317	Ta035	83901	**83493**	Ta065	292757	**288700**	Ta095	1513281	**1476067**
Ta006	15501	15501	Ta036	80924	**80702**	Ta066	290819	**286901**	Ta096	1508331	**1473770**
Ta007	15693	15693	Ta037	78791	**78669**	Ta067	300068	**298047**	Ta097	1541419	**1503849**
Ta008	15955	15955	Ta038	79007	**78672**	Ta068	291859	**287677**	Ta098	1533397	**1492257**
Ta009	16385	16385	Ta039	75842	**75647**	Ta069	307650	**304237**	Ta099	1507422	**1477116**
Ta010	15329	15329	Ta040	83829	**83569**	Ta070	301942	**296929**	Ta100	1520800	**1488330**
Ta011	25205	25205	Ta041	114398	**114077**	Ta071	412700	**408763**	Ta101	2012785	**1983701**
Ta012	26342	26342	Ta042	112725	**112180**	Ta072	394562	**390187**	Ta102	2057409	**2027057**
Ta013	22910	22910	Ta043	105433	**105345**	Ta073	405878	**402478**	Ta103	2050169	**2020138**
Ta014	22243	22243	Ta044	113540	**113364**	Ta074	422301	**418390**	Ta104	2040946	**2015944**
Ta015	23150	23150	Ta045	115441	**115404**	Ta075	400175	**396496**	Ta105	2027138	**2010428**
Ta016	22011	22011	Ta046	112645	**112459**	Ta076	391359	**387754**	Ta106	2046542	**2017860**
Ta017	21939	21939	Ta047	116560	**116451**	Ta077	394179	**390626**	Ta107	2045906	**2014715**
Ta018	24158	24158	Ta048	115056	**114947**	Ta078	402025	**398165**	Ta108	2044218	**2024692**
Ta019	23501	23501	Ta049	110482	**110367**	Ta079	416833	**412351**	Ta109	2037040	**2003859**
Ta020	24597	24597	Ta050	113462	**113427**	Ta080	410372	**407960**	Ta110	2046966	**2020700**
Ta021	38597	38597	Ta051	172845	172931	Ta081	562150	**557792**			
Ta022	37571	37571	Ta052	161092	**160805**	Ta082	563923	**560516**			
Ta023	38312	38312	Ta053	160213	**160104**	Ta083	562404	**559681**			
Ta024	38802	38802	Ta054	161557	**161492**	Ta084	562918	**559309**			
Ta025	39012	39012	Ta055	167640	**167081**	Ta085	556311	**551593**			
Ta026	38562	38562	Ta056	161784	**161460**	Ta086	562253	**558368**			
Ta027	39663	39663	Ta057	167233	**167098**	Ta087	574102	**570010**			
Ta028	37000	37000	Ta058	168100	168113	Ta088	578119	**573686**			
Ta029	39228	39228	Ta059	165292	**165207**	Ta089	564803	**560367**			
Ta030	37931	37931	Ta060	168386	168386	Ta090	572798	**568533**			