A new hybrid ant colony algorithm for scheduling of no-wait flowshop

Oper Res Int J
DOI 10.1007/s12351-016-0253-x

ORIGINAL PAPER

A new hybrid ant colony algorithm for scheduling of no-wait flowshop

Vahid Riahi[1] · **Morteza Kazemi**[2]

Received: 8 May 2015 / Revised: 5 April 2016 / Accepted: 16 June 2016

Abstract In this paper, the no-wait flow shop scheduling problem under makespan and flowtime criteria is addressed. The no-wait flowshop is a variant of the well-known flowshop scheduling problem where all processes follow the previous one without any interruption for operations of a job. Owing to the problem is known to be NP-hard for more than two machines, a hybrid meta-heuristic algorithm based on ant colony optimization (ACO) and simulated annealing (SA) algorithm is improved. First, at each step, due to the characteristic of ACO algorithm that include solution construction and pheromone trail updating, some different areas of search space are checked and best solution is selected. Then, to enhance the quality and diversity of the solution and finding best neighbor of this solution, a novel SA is presented. Moreover, a new principle is applied for global pheromone update based on the probability function like SA algorithm. The proposed approach solution is compared with several the state-of-the-art algorithms in the literature. The reported results show that the proposed algorithms are effective and the new approach for local search in ACO algorithm is efficient for solving the no-wait flow shop problem. Then, we employed another hybrid ACO algorithm based on hybridization of ACO with variable neighborhood search (VNS) and compare the results given by two proposed algorithms. These results show that our new hybrid provides better results than ACO-VNS algorithm.

✉ Vahid Riahi
Vahid.riahi@griffithuni.edu.au

Morteza Kazemi
kaazemi@sutech.ac.ir

[1] Institute for Integrated and Intelligent Systems, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia

[2] Department of Computer and Information Technology Engineering, School of Industrial Engineering, Shiraz University of Technology, Shiraz, Iran

Published online: 04 July 2016

Keywords No-wait flow shop scheduling · Local search · Maximum completion time · Variable neighborhood search · Simulated annealing

1 Introduction

The general assumption of majority of flow-shop applications is that the sequencing of the jobs relies on buffers, otherwise they are considered as intermediate storage between machines. The no-wait flow-shop scheduling problem (NWFSP) with makespan criterion is considered in this paper. In NWFSP, each job has to be processed from the first machine to the last without any interruption and the job sequence is unique on all machines. 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, the start of a job on the first machine may 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 flow-shop problem is to schedule jobs that minimize the makespan over all jobs. The no-wait flow shop scheduling problem has important applications in modern industry. It has an extensive background in industrial applications, including steel production, mining, logistics, chemical industry and food processing. For example, in steel factories, to avoid cooling and defects in steel, the liquid steel undergoes a chain of operations such as molding into ingots, unmolding, and soaking. Similarly, in the food processing, the canning operation must be operated after cooking operations immediately to ensure freshness (Hall and Sriskandarajah 1996). A comprehensive survey on the research and application of no-wait flow-shop scheduling problem can be found in Hall and Sriskandarayah's review paper (Hall and Sriskandarajah 1996). In addition, Bagchi et al. (2006) showed that it can be transformed into the asymmetric traveling salesman problem (ATSP) and presented some no-wait and blocking scheduling models.

In the operation research literature, many elegant mathematical models and methods have been developed to deal with the real-world problems. Some exact approaches have been proposed to unravel the problems optimally that have the limitation of solving only small-sized problems.

On the other hand, heuristics which are based on polynomial time algorithms are the most suitable methods for solving large scheduling problems. In general, heuristics attain good solutions in a reasonable time interval. Additionally, in the recent years meta-heuristics with techniques such as bee colony algorithm (Khorramizadeh and Riahi 2015), genetic algorithm (GA) (Guo et al. 2005), memetic algorithm (MA) (Frutos and Tohmé 2013), particle swarm optimization (PSO) (Marinakis et al. 2009), Electromagnetism-like Mechanism (SEM) (Bonyadi and Li 2010), and ant colony optimization (ACO) (Riahi and Kazemi 2015) have been developed to generate competitive results for many combinatorial optimization problems.

In the recent years, several ant colony algorithms have developed for various kinds of problems including the scheduling problem. Kashan and Karimi (2008) proposed two ACO algorithms with two different visibility functions for total

weighted tardiness single machine environment with formation of processing batches. To minimize total completion time in a no-wait two-machine flow-shop, Shyu et al. (2004) designed some specific features including a heuristic for initializing the initial pheromone, a hybrid state transition rule and a hybrid local search. Li et al. (2011) dealt with the minimization of the sum of completion time in a sequence-dependent setups permutational flowshop. The authors used a time-limited dynamic programming algorithm to perform a post-optimization strategy. Mirabi (2011) proposed an ant colony optimization technique for the sequence-dependent flowshop scheduling problem. The proposed algorithm contained a new pheromone initialization procedure and a local search method. Rui et al. (2014) presented an ant colony algorithm to solve the integrated job shop scheduling problem with tool flow in flexible manufacturing system. Huang et al. (Huang et al. 2015) proposed a no-wait FSMP problem with time window constraint which a new ant colony optimization (ACO), known as ant colony optimization with flexible update (ACOFU), is presented to solve the problem. The results show that ACOFU is superior to ACO when applied to small-scale problems. Recently, due to these insights, ACO is selected to solve this problem based on its ability and performance in solving scheduling problems. This is our incentive to employ ACO for solving the presented research problem.

On the other side, however SA usually is considered as one of the best algorithms that applied as a local search to improve the quality of solutions (Jolai et al. 2013; Wang et al. 2011; Xiao et al. 2015; Elmi et al. 2011; Mirsanei et al. 2011), a wide range of research on the literature review indicates that research with hybridization of ACO and SA is very rare. Behnamian et al. (2009) considered a scheduling problem with parallel machine and setup times. The authors combined ACO algorithm with the VNS and SA to balance exploration and exploitation.

In this paper, we develop an effective hybrid ant colony algorithm where the new approach to simulated annealing as a local search algorithm is presented as an improvement phase. To show its performance, we use another method based on one popular method, VNS, as a local search in proposed ACO. In addition, based on the main characteristics of the considered problem, we propose a new approach for global update that takes advantage of increasing diversity by a probability criterion. The computational comparison demonstrated the effectiveness of the presented hybrid ACO algorithm and shows that our algorithm is able to provide competitive results in reasonable CPU times and our new simulated annealing method has a better performance than VNS algorithm for a local search.

This paper is structured as follows: the no-wait flow shop problem is described Sect. 2. Section 3 introduced new ant colony optimization algorithm, whereas the experimentation is given in Sect. 4. The paper is concluded in Sect. 5.

2 The no-wait flowshop scheduling problem

In this Section, we first define the NWFSP with a makespan criterion. Then, we review the relevant literature.

2.1 Problem formulation

The NWFSP can be defined as: a set of jobs is available for processing on machines. Each job has a sequence of operations. Each machine can process 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 and the jobs should not wait to be scheduled between two successive machines. The problem is finding a schedule in a way that the processing order of jobs is the same on each machine and the maximum completion time is minimized.

The following notations are used to formulate the no-wait flow-shop problems:

n Number of jobs,
m Number of machines,
pij Processing time for the job i on the machine j
$p_{[i]j}$ Processing time for the job in the position i on the machine j
d_{ik} Minimum delay between the start of job i and job $k$on the first machine due to the no-wait restriction,
$C_{[i]}$ The completion time of the job located on position i

The minimum delay time d_{ik} and completion time $C_{[i]}$ can be calculated using the following equations:

$$d_{ik} = p_{i1} + \max\left\{\max_{2\le j\le m}\left(\sum_{l=2}^{j} p_{il} - \sum_{l=1}^{j-1} p_{kl}\right), 0\right\} \tag{1}$$

$$C_{[1]} = \sum_{j=1}^{m} p_{[1]j} \tag{2}$$

$$C_{[i]} = \sum_{l=2}^{i} d_{[l-1][l]} + \sum_{j=1}^{m} p_{[i]ji=2,3,..,n} \tag{3}$$

So, the maximum completion time (makespan) can be obtained as follows:

$$C_{\max} = \max_{1\le i\le n}\left\{C_{[i]}\right\}. \tag{4}$$

2.2 Literature review

According to Garay and Johnson (1979), the no-wait flow-shop scheduling problem is NP-hard. Therefore, heuristic procedures due to their polynomial time complexity are preferred as the most suitable optimization methods to solve these scheduling problems. Important heuristics for the no-wait flowshop to minimize the makespan are slope index (Bonney and Gundry 1976), Minimum covering level (MCL) (King and Spachis 1980), Raj (Gangadharan and Rajendran 1994), Framinan-Nagano heuristic (FN) (Framinan et al. 2010), Fast composite heuristic (FCH) (Li and Wu

2008), Li-Wang-Wu heuristic (LWW) (Li et al. 2008) and Laha-Chakraborty heuristic (LCH) (Laha and Chakraborty 2009).

In addition, it has been examined by some researchers applying different meta-heuristics. Aldowaisan and Allahverdi (2003) designed methods based on simulated annealing and genetic algorithm. Schuster and Framinan (2003) implemented a variable neighborhood search algorithm (VNS) and a hybrid algorithm GASA to solve the no-wait job shop scheduling problems. Grabowski and Pempera (2005) presented five heuristic algorithms for NWFSP, where three Tabu Search (TS) based algorithms (TS, TS + M, TS + MP) were developed by using the TS with multi-moves and the other were based on descending search (DS) and descending search with multi-moves (DS + M). Revealed by experimental results, these tabu search methods outperform the VNS and the GASA. Liu et al. (2007) proposed an effective algorithm based on hybrid particle swarm optimization (HPSO) and Nawaz-Enscore-Ham (NEH) heuristic (Nawaz et al. 1983) to minimize makespan. Pan et al. (2008) presented a DPSO algorithm for the NWFSP. They also hybridized the variable neighborhood descent (VND) algorithm and discrete particle swarm optimization (DPSO) to further enhance its searching ability. The results show that the presented algorithm was effectiveness and is better than other algorithms both heuristic and meta-heuristic in the literature.

Pan et al. (2008) applied an improved iterated greedy algorithm (IIGA) to solve NWFSP. In that paper, they introduced a speed-up method for insert neighborhood and a new heuristic algorithm based on NEH heuristic algorithm. In 2009, Qian et al. (2009) proposed a hybrid differential evolution (HDE) approach with a new local search to solve NWFSP. Tseng and Lin (2010) proposed a hybrid GA (HGA), which hybridizes a GA and a novel local search scheme which combines two local search methods: the Insertion Search (IS) and a novel local search method called the Insertion Search with Cut-and-Repair (ISCR). Samarghandi and ElMekkawy (2012) proposed a hybrid TS and PSO algorithm and utilized a new coding and decoding technique to improve its efficiency. Ding et al. (2015) proposes a Tabu-mechanism improved iterated greedy (TMIIG) algorithm with a Tabu reconstruction strategy which improves the exploitation ability of the algorithm and leads to better performance.

3 Description of the proposed ACO algorithm

In this section, we described the proposed ACO algorithm for the no-wait flowshop scheduling problem. The ACO algorithm is utilized to generate initial scheduling. Then the ACO solutions are improved by using a new local search method. The explanations of these algorithms are as follows:

The ACO meta-heuristics algorithm, was introduced by Colorni et al. (1991) for the first time, and has been successfully applied to a large number of combinatorial optimization problems. This algorithm is inspired by the behavior of real ants in finding the shortest path from a food source to the nest.

In ACO algorithm a feasible solution is usually shown as a path in a graph. In NWFSP, the graph of the problem is shown with a complete graph where the

vertices correspond to the activities. Each sequence of scheduling the jobs can be shown as a directed path which includes all the vertices in the graph of the problem. Therefore, finding a feasible solution lead to find such directed path.

According to previous studies, ACO can work effectively to find good solutions in scheduling problems if it benefits from effective local search and constructive pheromone updating rule. This inspired us to develop a new ACO to solve no-wait flowshop problem. Flowshop scheduling problem can be regarded as a hard optimization problem. As a result, simple ACO with traditional pheromone updating trail and completely random search methods to find following values may not perform efficiently in an appropriate time. Consequently, the ACO developed in this paper has been benefited of a new approach to Global update of pheromone trail and applies effective local search based on a new simulated annealing algorithm.

The general structure of the proposed ACO algorithm is then represented as follows (the details are presented in the remainder of this section):

Step 1. Set parameter; generate an initial solution and initialize the pheromone trails

Step 2. While the termination condition is not met, do the following:

Step 2.1 For each ant in the colony, do:

a. Construct a solution by repeatedly applying the transition rule

b. Apply the pheromone local update rule;

Step 2.2. Find the best solution generated in step 2.1 and improve the solution quality by the local search;

Step 2.3. Modify the pheromone trails according to the global updating rule;

Step 2.4. Update the minimum and maximum trail bounds, and limit the pheromone trails.

Step 3. Return the best solution found.

3.1 Pheromone initialization

τ_{ij} is considered as the intensity of assigning ith job in jth position. This desirability is changed during the running of the ACO algorithm. So, there is a pheromone value for every job i for every feasible position j that can be updated at each iteration of ACO algorithm.

In this paper, like PACO proposed by Rajendran and Ziegler (2004), an initial differential setting of the trail intensities is considered. In this paper we consider an upper bound $\tau_{\max}$ and a lower bound $\tau_{\min}$ for the trail pheromone that calculated by the following formulas:

$$\tau_{max} = ((1-\rho)Z_{best})^{-1} \tag{5}$$

$$\tau_{min} = \beta\tau_{max} \tag{6}$$

where Z_{best} denotes the best makespan obtained so far and $\rho(0 \le \rho \le 1)$ is the parameter of the algorithm and for initially, it equals the makespan of Rajendran's heuristic (1993), so, the initial pheromone are chosen as $\tau_{ij} = \tau_{\max} = ((1-\rho)Z_{best})^{-1}$ for all i and j.

The Raj heuristic is a variant of the NEH heuristic (1983) that is regarded as the best heuristic for the flow shop. It has three phases which are explained as follows:

Step 1 Calculate the value of T(i) defined by Eq. (7) for each job i. Then, obtain a sequence $\pi = (\pi_1, \pi_2, \ldots \pi_i, \ldots \pi_n)$ by sorting jobs according to their increasing T(i)

$$T(i) = \sum_{j=1}^{m} \left[(m - j + 1) t_{ij} \right] \tag{7}$$

Step 2 The first two jobs of π are taken, and the two possible subsequences of these two jobs are evaluated, and then select the better one as the current sequence s. Set $k = 2$.

Step 3 Set $k = k + 1$. Take the kth job of π and insert it into p ($p \in [1, k - 1]$) possible positions of the current sequence s; we obtain p subsequences. Select the subsequence with the minimum makespan as the current sequence s for the next generation. Repeat this step until all jobs are sequenced, and the final sequence is found.

3.2 Make feasible solution

A set of artificial ants is initially created. Each ant starts with an empty sequence and then successively appends an unscheduled job to the partial sequence until a feasible solution is constructed (or all jobs are scheduled). In choosing the next job j to be appended after job i, the ant applies the following state transition rule:

$$j = \begin{cases} \arg\max\left(\tau_{ij}\right), & \text{if } q \le q_0 \\ S, & \text{otherwise} \end{cases} \tag{8}$$

q is a random number uniformly distributed in [0,1], and q_0 is a parameter between 0 and 1. If $q \le q_0$, the unscheduled job j with maximum value is placed after job i; otherwise, a job is chosen according to S. the random variable S is selected according to the probability distribution given in Eq. 9.

$$P_{ij}^{k} = \frac{\tau_{ij}}{\sum_{j \in N_j^k} \tau_{ij}}, \quad i \in N_j^k \tag{9}$$

where N_j^k is the set of unscheduled jobs.

3.3 Local update of pheromone trail

To avoid premature convergence, a local trail update is performed. The local update reduces the pheromone amount of a new component so that the following ants have a less probability to choose this job in the same place. This is achieved by the following local updating rule:

$$\tau_{ij}^{new} = (1-\rho)\tau_{ij}^{old} + \frac{\rho}{C_{\max}^{k}} \tag{10}$$

where $\rho(0 \leq \rho \leq 1)$ is the parameter of the algorithm. And C_{max}^{k} is the makespan of the complete sequence of ant k.

3.4 Local search

Once a complete sequence of jobs has been generated by all ants, the performance quality of the solution is improved by means of a local search procedure. In this paper, two algorithms are proposed based on the simulated annealing algorithm and variable neighborhood search method are used for the local search.

3.4.1 A variable neighborhood search for the local search

VNS (Mladenović and Hansen 1997) is a stochastic algorithm in which, first, a set of neighborhood structures N_k $(k = 1, ..., n)$ are defined. Then, each iteration of the algorithm is composed of three steps: shaking, local search, and change.

In this paper, a modified variable neighborhood search (VNS) that is incorporated into Liu and Liu (2013) algorithm as a hybrid strategy is used. In Liu and Liu (2013), they used two neighborhood structures called swap and insert local search respectively. Insert move that operates on sequence of jobs π and removes the job $\pi(x)$ from its original position x, and next insert it in a position y in π, whereas Swap move in which the jobs $\pi(x)$ and $\pi(y)$, $x \neq y$, are interchanged in some positions x and y in π. In presented VNS, each job is considered for possible insertion in all other positions after considering all possible swaps of pairs of job's positions. At used VNS, the best individual of ants is selected as the seed permutation. In fact, searching neighborhood for the best local optimum is repeated with first neighborhood structure until no improvement appears: the seed permutation is updated if the new solution is better than the seed permutation and search is continued; otherwise the neighborhood is changed, and search is continue with second neighborhood structure until no possible improvement.

3.4.2 The simulated annealing (SA) as a local search

SA is one of the most popular meta-heuristics for addressing combinatorial optimization problems over the past two decades.

SA method differs from the most local search heuristics because it uses two or more neighborhoods, instead of one, in its structure. In particular, they are based on the principle of systematic change of neighborhood during the search. A standard SA algorithm begins with generating an initial random solution. Then, iteratively produce a neighbor solution. The new solution will be accepted as the new current solution if it has a lower or equal cost; otherwise its acceptance is based on a probability function. The process is repeated until the termination condition is satisfied (Low et al. 2004).

In presented SA, in order to increase the quality of the solution, some neighborhood generation structures are used and to increase the searching speed of the proposed SA, some additional termination criteria are presented.

To start the procedure, SA generates an initial solution as the incumbent solution and after that SA proceeds in several iterations. After setting parameter, at each iteration, a random neighbor is generated. Moves that improve the objective function are always accepted. Otherwise, the neighbor is selected with a given probability that depends on the current temperature and the amount of degradation ΔE of the objective function. ΔE represents the difference in the objective value between the current solution and the generated neighboring solution. As the algorithm progresses, the probability that such moves are accepted decreases.

In our algorithm, contrast to the most other studies in literature (Ying et al. 2012; Dai et al. 2013; Rabiee et al. 2014) that just use temperature, T, as a stopping criterion, some additional termination criteria are presented. In this paper, the proposed SA is similar to the SA developed by Low (2005). The main difference between the proposed SA and those developed by Low (2005) is that in proposed algorithm, to increase the quality of solution and the speed of algorithm, different termination criteria have been considered. This difference allows the algorithm to find the final solution from fairly diversified solutions and relies more on solutions with good qualities. In addition, we've used a Moving List (*ML*) that is empty first ($ML \leftarrow \varnothing$) and after insertion and swap operators, find the eight operators that result in the smallest amount of makespan. Put these eight operators in ML.

For more clarification algorithms, steps are explained in Algorithm 1.

Algorithm 1. Simulated Annealing algorithm

Step 1	*Initialization*
Step 1.1	*Set the initial temperature, parameters, T, final temperature, T_f, and cooling rate, β,*
Step 1.2	*Obtain an initial solution, S, and set $S^* = S$.*
Step 1.3	*Set counter=0, temp=0.*
Step 2	*While not yet frozen, $T > T_f$,do the following:*
Step 2.1	*While (temp<temp$_f$) or (counter < counter$_f$) do the following :*
Step 2.1.1	*Neighborhood search. Select a neighbor S^{new} of S.*
Step 2.1.2	*Compute $\Delta E = Obj(S^{new}) - Obj(S)$ and update ML.*
Step 2.1.3	*If $\Delta E \leq 0$, set $S = S^{new}$;*
Step 2.1.3.1	*Compute $\Delta E = Obj(S^{new}) - Obj(S^*)$;*
Step 2.1.3.2	*If $\Delta E \leq 0$, set $S^* = S^{new}$ and temp=temp+1.*
Step 2.1.4	*If $\Delta E > 0$, generate random number, L, from the interval, (0, 1); If $L < e^{-\Delta E/T}$, set $S^{new} = S$ and counter = counter +1.*
Step 2.1.5	*Set $T = T^* \beta$ and counter=0, temp=0.*
Step 3	*Return the best solution found for (S^*).*

Based on the generic SA algorithm, the details of the proposed SA algorithm for the no-wait flow shop scheduling problem are given below.

3.4.2.1 Initial solution and neighborhood The best randomly solution generated by ants is used as an initial solution. Given a sequence S, a new sequence can be obtained for S using one of the swap or insertion operations described in the proposed VNS algorithm.

3.4.2.2 Move acceptance In order to escape from local minima, the probabilistic acceptance of a non-improving neighbor is used. The probability of accepting a non-improving neighbor is proportional to the temperature T and inversely proportional to the change of the objective function ΔE. So that, the probability of replacing one solution with its neighbor, when $\Delta E > 0$, is $e^{-\Delta E/T}$. In fact, the non-improving solution will be accepted if L, a random number between zero and one, was lower than $e^{-\Delta E/T}$.

3.4.2.3 Cooling schedule The cooling schedule temperature T_k has a great impact on the success of the SA optimization algorithm. It decrease after each iteration according to the formula $T_k = \beta * T_{k-1}$, where $0 < \beta < 1$.

3.4.2.4 Termination conditions In this paper, stopping criteria may be used are the following:

- Reaching a final temperature T_k (if T_k is lower than a pre-specified final temperature).
- Achieving a predetermined number of iterations with improvement in the best found solution.
- Reaching a fixed number of iterations that solutions are accepted with probability acceptance.

4 Global update of pheromone trail

The global updating rule is applied after all ants generate their solutions and the local search applied. Unlike most other studies that apply global update just on the best solution so far Ahmadizar (2012), Xu et al. (2012), in this paper, based on the idea of SA algorithm, global update is applied for all of ML's solutions with a probabilistic acceptance. The pheromone trails are modified according to the following global updating rule to make the search more directed:

$$\tau_{ij}^{new} = \begin{cases} \rho\tau_{ij}^{old} + \Delta\tau_{ij} & \textit{for all solutions in moving List} \\ \rho\tau_{ij}^{old} & \textit{Otherwise} \end{cases} \tag{11}$$

where $\Delta\tau_{ij}$ is controlled by probability criterion.

That is, if

$$C_{max}^{ML_i} = C_{max}^{best} or\ exp(-\Delta C/R) > rand(0,1) \tag{12}$$

$$\Delta\tau_{ij} = \frac{1}{C_{max}^{ML_i}} \tag{13}$$

In above equation, R is a constant and $\Delta C = C_{max}^{ML_i} - C_{max}^{best}$ where $C_{max}^{ML_i}$ is the makespan of the $ML_i (i = 1, \ldots, 8)$ solution and C_{max}^{best} is the best makespan obtained so far. If criterion is not satisfied $\Delta\tau_{ij} = 0$.

With proposed procedure, the immature homology can be effectively avoided and the diversity of solutions in later iterations increased as well.

After modifying the pheromone trail intensities according to the above global updating rule, upper bound and lower bound are updated Then, any pheromone trail out of this range (τ_{max} and τ_{min}) should be adjusted, so, in case of $\tau_{ij}^{new} > \tau_{max}$ or $\tau_{ij}^{new} < \tau_{min}$, the τ_{ij}^{new} is set to τ_{max} or τ_{min} respectively.

5 Computational result comparisons

The proposed modified ant colony optimization algorithm is implemented in Delphi 7.0. Tests are run on a dual core 3.6 GH, CPU with 2 GB memory.

All of the parameters in this study were determined experimentally. The parameter settings for the presented SS are listed in Table 1.

We conducted experiments on 29 benchmark problems which consists of eight problems (car1, car2… car8) provided by Carlier (1978) and 21 problems provided by Reeves (1995). As the optimal solution is unknown for the larger instances, the performance of the proposed algorithm is evaluated with a distance from the Rajendran heuristic algorithm (1994). t denotes the average CPU time (in second) of 20 runs (R = 20).

The performance in this paper, to report the statistics based on the percentage of relative deviations from Rajendran, they were denoted as RAJ. To be more specific percentage relative difference (PRD) was computed as follows:

$$PRD = \frac{(M_i - M_{ref})}{M_{ref}} \times 100 \tag{14}$$

where M_i and M_{ref}, were the makespan generated by algorithms and the reference makespan generated by RAJ.

The proposed algorithms are compared with 7 other heuristics which can be seen in Table 2. TS, TS + M and TS + MP are three local search algorithms based on

Table 1 Parameter settings for the ACO–SA algorithm

Number of ants = 10	*Iteration number* = 50	$q_0 = (n-4)\rho/_n$	$Temp_f = 6$
$\rho = 0.8$	$\beta = 0.2$	$T = 40$	$Counter_f = 22$
$Temp_f = 7$	$T_f = 0.1$	$R = n/2$	

several variants of descending search and tabu search algorithms employed by Garbowski and Pempera (2005), the VNS and GA-SA of Schuster and Framinan (2003), HPSO represents the results of Liu et al. (2007), HGA of Tseng and Lin (2010) and TS/PSO of the Samarghandi and ElMekkawy (2012). Due to the fact that the computational time consumed by these tested algorithms generally does not exceed one second on average, time consumption is not the main issue for this problem. Consequently, we conclude that the proposed ACO–SA algorithm can outperform these advanced algorithms for the NWFSP in minimizing the makespan.

The comparison is based on the instances of Carlier and reeves. Table 2 shows the experimental results for the nine algorithms in comparison. As shown in Tables 2, our ACO–SA algorithm is competitive with all both local search-based algorithms and population-based algorithms. In fact, Analysis of Table 2 reveals that the searching quality of proposed ACO–SA is superiorly relative to other algorithms with respect of quality of solutions. As can be observed in the Table 2, the total average of RPD by the proposed ACO–SA algorithm is -4.85, which is superior to the corresponding values of -3.81, -1.2, -4.52, -4.76, -4.74, -4.6, -4.82 and -4.67 obtained by the VNS, GASA, TS, TS + M, TS + MP, HPSO, HGA and TS/PSO, respectively. By comparing our algorithm with HGA that have a better performance than others, in 6 instances, it has a better results than ours. However, in these instances solutions were close. But in 6 instances, our algorithm has a superiority than HGA algorithm.

In addition, for proving the efficiency of presented SA and its importance in proposed algorithm, we used VNS as another method of local search. Table 3 summarizes the computational results for our algorithm without any local search, ACO-VNS and ACO–SA algorithms.

In addition, it is observed from Table 3 that the ACO–SA has better performance than ACO-VNS in both CPU times and the quality of solutions, besides it has a significant effect on our algorithm. It is clear that the new approach for simulated annealing as a local search and hybridization with ACO increases its quality of searching and decreases its CPU Times especially in large size benchmarks rather than VNS.

The comparison results of Tables 2 and 3 prove that our algorithm without local search (ACO) is competitive and better than some local search-based algorithms (GASA, DS, DS + M). In addition, we can see that the ACO-VNS is competitive as well compared to other algorithms in literature.

In the procedure, there is a parameter includes ρ, which may has great effect on the performance of the presented ACO–SA algorithm. Therefore, we further investigate the effect of ρ by experiments. We vary ρ from 0.1 to 0.9. The contrastive results are presented in Fig. 1. From Fig. 1, different percent of average PRD values was showed based on different values of ρ produce that revealed this fact that the parameter ρ affects the searching quality of the presented algorithm. Indeed, presented algorithm had its best performance when the parameter ρ is equal to 0.8 due to its minimum average PRD value.

On the other hand, we evaluate another experiment to test the performance of the proposed algorithm on the 110 problems presented by Taillard (1993). The results of algorithms after 10 runs are given in Table 4. Due to the efficiency of ACO-VNS

Table 2 Results of best algorithms

Instance	M * J	Opt	RAJ	VNS		GASA		TS		TS + M		TS + MP	
Name				PRD	t	PRD	t	PRD	t	PRD	t	PRD	t
Car01	5 * 11	8142		0.70	0	0.00	1	0.00	0.1	0.00	0.1	0.00	0.1
Car02	4 * 13	8242		0.20	0	0.00	1	0.00	0.1	0.00	0.1	0.00	0.1
Car03	5 * 12	8866		0.00	0	0.00	1	0.00	0.1	0.00	0.1	0.00	0.1
Car04	4 * 14	9195		1.60	0	0.00	2	0.00	0.1	0.00	0.1	0.00	0.1
Car05	4 * 10	9159		3.50	0	0.00	1	0.00	0.1	0.00	0.1	0.00	0.1
Car06	9 * 8	9690		0.00	0	0.00	1	0.00	0.1	0.00	0.1	0.00	0.1
Car07	7 * 7	7705		0.00	0	0.00	0	0.00	0.1	0.00	0.1	0.00	0.1
Car08	8 * 8	9372		0.00	0	0.00	1	0.00	0.1	0.00	0.1	0.00	0.1
REC01	5 * 20		1590	−2.77	0	−3.96	6	**−4.03**	0.2	−3.96	0.2	−3.96	0.2
REC03	5 * 20		1457	−4.32	0	−4.46	6	**−6.59**	0.2	**−6.59**	0.2	**−6.59**	0.2
REC05	5 * 20		1637	−7.03	0	−6.9	7	−7.39	0.2	−7.64	0.2	**−7.7**	0.2
REC07	10 * 20		2119	−2.31	0	−3.45	12	**−3.63**	0.2	**−3.63**	0.2	**−3.63**	0.2
REC09	10 * 20		2141	−2.38	0	−4.48	11	**−4.62**	0.2	−4.58	0.2	−4.58	0.2
REC11	10 * 20		1946	−1.54	0	**−3.34**	10	**−3.34**	0.2	**−3.34**	0.2	**−3.34**	0.2
REC13	15 * 20		2709	−5.76	0	−5.65	17	**−6.05**	0.3	**−6.05**	0.3	**−6.05**	0.3
REC15	15 * 20		2691	−5.91	0	**−6.02**	17	−5.91	0.3	**−6.02**	0.3	−5.91	0.3
REC17	15 * 20		2740	−5.15	0	−5.47	16	**−5.58**	0.3	**−5.58**	0.3	**−5.58**	0.3
REC19	10 * 30		3157	−7.57	1	−5.45	34	**−9.72**	0.4	−9.25	0.4	−9.38	0.4
REC21	10 * 30		3015	−4.21	1	−2.22	35	−6.31	0.4	−6.3	0.4	−6.17	0.4
REC23	10 * 30		3030	−10.76	0	−6.7	35	−10.76	0.4	−10.73	0.4	**−10.89**	0.4
REC25	15 * 30		3835	−5.45	1	−2.69	55	−5.97	0.5	**−6.31**	0.5	−6.21	0.5
REC27	15 * 30		3655	−5.83	1	−2.6	51	−5.64	0.5	−6.1	0.5	−5.83	0.5
REC29	15 * 30		3583	−7.23	1	−3.99	54	−7.64	0.5	**−8.28**	0.5	−7.94	0.5

Table 2 continued

Instance	M * J	Opt	RAJ	VNS		GASA		TS		TS + M		TS + MP	
Name				PRD	t	PRD	t	PRD	t	PRD	t	PRD	t
REC31	10 * 50		4631	−4.71	5	2.72	147	−5.9	1.1	−6.13	1.1	−6.22	1.1
REC33	10 * 50		4770	−5.35	7	4.78	145	−5.51	1.1	−6.31	1.1	−6.37	1.1
REC35	10 * 50		4718	−5.51	7	3.67	146	−6.08	1.1	−6.17	1.1	−5.91	1.1
REC37	20 * 75		8979	−10.00	122	5.89	907	−9.41	2.5	−9.49	2.6	−9.36	2.6
REC39	20 * 75		9158	−5.32	106	8.80	890	−4.97	2.5	−6.99	2.6	−6.91	2.6
REC41	20 * 75		9344	−7.41	110	6.79	904	−6.08	2.5	−8.57	2.6	−8.82	2.6
Mean				−3.81		−1.2		−4.52		−4.76		−4.74	

Instance	M * J	HPSO		HGA		TS/PSO		ACO–SA	
Name		PRD	t	PRD	t	PRD	t	PRD	0.00
Car01	5 * 11	0.0	0.4	0.0	0.002	0	0.00	0	0.00
Car02	4 * 13	0.0	0.7	0.0	0.002	0	0.00	0	0.00
Car03	5 * 12	0.0	0.9	0.0	0.002	0	0.00	0	0.00
Car04	4 * 14	0.0	1.4	0.0	0.000	0	0.00	0	0.00
Car05	4 * 10	0.0	0.6	0.0	0.002	0	0.00	0	0.00
Car06	9 * 8	0.0	0.3	0.0	0.000	0	0.00	0	0.00
Car07	7 * 7	0.0	0.2	0.0	0.000	0	0.00	0	0.00
Car08	8 * 8	0.0	0.3	0.0	0.000	0	0.00	0	0.00
REC01	5 * 20	−3.77	3.9	**−4.03**	0.009	−4.025	0.029	**−4.03**	0.029
REC03	5 * 20	**−6.59**	4.8	**−6.59**	0.006	**−6.59**	0.022	**−6.59**	0.022
REC05	5 * 20	−7.39	4.1	**−7.70**	0.008	−7.31	0.019	**−7.70**	0.019
REC07	10 * 20	**−3.63**	6.6	**−3.63**	0.008	−3.44	0.022	**−3.63**	0.022
REC09	10 * 20	−4.58	6.7	**−4.62**	0.008	**−4.62**	0.025	**−4.62**	0.025

Table 2 continued

Instance	M * J	HPSO		HGA		TS/PSO		ACO–SA	
Name		PRD	t	PRD	t	PRD	t	PRD	0.00
REC11	10 * 20	**−3.34**	7.0	**−3.34**	0.008	−3.14	0.023	**−3.34**	0.023
REC13	15 * 20	**−6.05**	11.0	**−6.05**	0.009	−5.95	0.026	**−6.05**	0.026
REC15	15 * 20	**−6.02**	8.6	**−6.02**	0.008	−5.86	0.036	**−6.02**	0.036
REC17	15 * 20	**−5.58**	8.6	**−5.58**	0.008	−5.55	0.022	**−5.58**	0.022
REC19	10 * 30	−9.15	23.0	**−9.72**	0.034	−9.07	0.048	−9.67	0.048
REC21	10 * 30	−5.70	24.0	−6.17	0.03	−6.27	0.059	**−6.36**	0.059
REC23	10 * 30	−10.8	24.0	**−10.89**	0.025	−10.83	0.050	−10.82	0.050
REC25	15 * 30	−5.71	32.0	**−6.31**	0.031	−5.823	0.093	−6.23	0.093
REC27	15 * 30	**−6.13**	39.0	**−6.13**	0.031	−5.960	0.115	−6.04	0.115
REC29	15 * 30	−7.81	31.0	−8.15	0.031	−7.883	0.096	−8.01	0.096
REC31	10 * 50	−5.92	122.0	−6.41	0.267	−5.970	0.39	**−6.69**	0.39
REC33	10 * 50	−5.51	116.0	−6.54	0.252	−6.260	0.48	**−6.88**	0.48
REC35	10 * 50	−6.02	105.0	−6.23	0.225	−6.217	0.46	**−6.45**	0.46
REC37	20 * 75	−8.89	635.0	−9.56	1.447	−9.667	1.581	**−9.89**	1.581
REC39	20 * 75	−6.79	897.0	−7.13	1.283	−6.756	1.595	**−7.23**	1.595
REC41	20 * 75	−7.94	883.0	**−8.98**	1.073	−8.215	1.409	−8.8	1.409
Mean		−4.6		−4.82		−4.669		**−4.85**	

Bold values are the better solutions reached by presented algorithm than other algorithms

Table 3 Results of proposed algorithms

Instance	M * J	ACO			ACO-VNS			ACO–SA		
		PRD			PRD			PRD		
Name		Min	Avg	t	Min	Avg	t	Min	Avg	t
Car01	5 * 11	0.00	0.28	0.00	0.00	0.00	0.00	0	0	0.00
Car02	4 * 13	0.00	0.27	0.00	0.00	0.00	0.00	0	0	0.00
Car03	5 * 12	0.00	0.13	0.00	0.00	0.00	0.00	0	0	0.00
Car04	4 * 14	0.00	0.86	0.00	0.00	0.00	0.00	0	0	0.00
Car05	4 * 10	0.00	0.00	0.00	0.00	0.00	0.00	0	0	0.00
Car06	9 * 8	0.00	0.00	0.00	0.00	0.00	0.00	0	0	0.00
Car07	7 * 7	0.00	0.00	0.00	0.00	0.00	0.00	0	0	0.00
Car08	8 * 8	0.00	0.00	0.00	0.00	0.00	0.00	0	0	0.00
REC01	5 * 20	−3.52	−3.28	0.01	−4.03	−4.03	0.023	−4.03	−4.03	0.029
REC03	5 * 20	−5.4	−5.1	0.00	−6.59	−6.52	0.02	−6.59	−6.59	0.022
REC05	5 * 20	−6.96	−6.33	0.01	−7.68	−7.70	0.019	−7.7	−7.7	0.019
REC07	10 * 20	−3.44	−3.09	0.00	−3.63	−3.63	0.015	−3.63	−3.63	0.022
REC09	10 * 20	−3.6	−3.10	0.00	−4.62	−4.6	0.026	−4.62	−4.62	0.025
REC11	10 * 20	−2.98	−2.46	0.01	−3.34	−3.34	0.021	−3.34	−3.34	0.023
REC13	15 * 20	−5.8	−5.5	0.01	−6.05	−6.05	0.025	−6.05	−6.05	0.026
REC15	15 * 20	−5.9	−5.64	0.00	−6.02	−6.02	0.029	−6.02	−6.02	0.036
REC17	15 * 20	−5.47	−5.27	0.00	−5.58	−5.58	0.024	−5.58	−5.58	0.022
REC19	10 * 30	−8.11	−7.57	0.05	−9.72	−9.67	0.042	−9.72	−9.67	0.048
REC21	10 * 30	−5.30	−4.7	0.04	−6.43	−6.28	0.051	−6.43	−6.36	0.059
REC23	10 * 30	−8.81	−8.58	0.04	−10.89	−10.79	0.053	−10.89	−10.82	0.050
REC25	15 * 30	−5.45	−4.95	0.03	−6.31	−6.23	0.097	−6.31	−6.23	0.093
REC27	15 * 30	−4.57	−3.58	0.05	−6.12	−6.04	0.108	−6.13	−6.04	0.115
REC29	15 * 30	−6.67	−6.21	0.06	−8.15	−8.08	0.096	−8.15	−8.01	0.096
REC31	10 * 50	−4.71	−4.27	0.011	−6.32	−6.21	0.46	−6.76	−6.69	0.39
REC33	10 * 50	−4.57	−3.48	0.011	−6.78	−6.49	0.42	−7.25	−6.88	0.48
REC35	10 * 50	−4.3	−3.7	0.012	−6.42	−6.26	0.35	−6.76	−6.45	0.46
REC37	20 * 75	−7.87	−7.21	0.026	−9.88	−9.48	1.651	−10.32	−9.89	1.581
REC39	20 * 75	−5.54	−4.97	0.026	−7.31	−7.23	1.565	−7.56	−7.23	1.595
REC41	20 * 75	−6.36	−5.85	0.026	−8.80	−8.66	1.309	−8.97	−8.8	1.409
Mean			−3.56			−4.79			−4.85	

based on its results in Table 3, we compare our algorithm with this algorithm in this experiment. Results show that proposed algorithm, ACO–SA, is effective and has reliable performance.

In order to prove the efficiency of our algorithm, we evaluated it with respect to the total flowtime criterion. Fink and Voß (2003) presented some construction methods and meta-heuristics for the no-wait flowshop scheduling problem with the total flowtime criterion. The benchmark experimentation results have been given in Table 5. The results are presented as each the job instance sizes. The average, minimum and

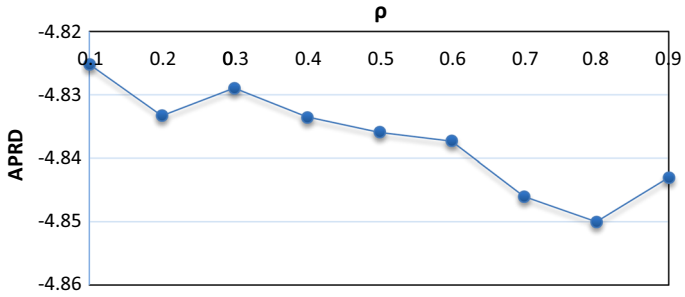

Fig. 1 Effect of the parameter ρ

Table 4 Results of proposed algorithms on Taillard's benchmark problems

N * M	ACO_{VNS}				ACO–SA			
	PRD		CPU		PRD		CPU	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
20 * 5	1479.6	1489.55	0.21	0.24	1477.9	1481.67	0.22	0.28
20 * 10	1985	1990.21	0.43	0.48	1981.9	1989.34	0.41	0.44
20 * 20	2976.7	3007.77	0.51	0.59	2972.9	2986.54	0.51	0.56
50 * 5	3285.7	3335.48	1.41	1.69	3280.8	3297.36	1.44	1.79
50 * 10	4288.8	4659.54	2.47	2.56	4279.8	4311.42	2.61	2.96
50 * 20	5920.4	6096.45	2.89	3.06	5903	6001.77	2.99	3.18
100 * 5	6263.4	6505.21	31.89	32.15	6248.6	6375.7	25.49	30.20
100 * 10	8165.3	8225.27	49.49	56.58	8039.5	8133.28	45.49	57.23
100 * 20	10,755.3	10,913.59	52.16	60.81	10,722.3	10,823.33	53.23	61.26
200 * 10	15,588.7	15,663.36	173.69	185.38	15,428.6	15,532.26	169.08	189.17
200 * 20	20,219.5	20,386.24	187.32	198.62	20,111.5	20,221.85	181.14	200.82
Mean	7357.13	7479.33	45.68	49.29	7313.35	7377.68	43.87	49.81

maximum were given. ACO–SA has 10 independent replications for each instance and outcomes have been given in Table 6. According to the results, ACO–SA has better performance than F&V algorithm, due to an average PRD of −0.5405.

6 Conclusion and future work

In the current research, a no-wait flow-shop scheduling problem (NWFSP) for minimizing the makespan was discussed. This problem is known to be strongly NP-hard. In this paper, ant colony algorithm with the simulated annealing as a local search method are proposed to solve this problem. Unlike most of other reported population-based algorithms in the literature, the proposed ACO–SA algorithm is simple and easy to implement and replicate. The evaluation of the proposed methods was carried out against the 8 best performing methods from the literature.

Table 5 Summarized results of F&V

$J * M$	*F&V*		
	Min	Max	Average
20 * 5	15,317	17,970	160,895
20 * 10	21,939	26,342	23,605.6
20 * 20	37,000	39,663	38,467.8
50 * 5	75,842	83,901	802,732
50 * 10	105,433	116,560	112,974.2
50 * 20	160,213	172,845	165,414.2
100 * 5	278,811	308,052	2,969,583
100 * 10	391,359	422,301	405,038.4
100 * 20	556,311	578,119	565,978.1
200 * 10	1,489,457	1,541,419	15,166,852
200 * 20	2,012,785	2,057,409	2,040,911.9

Table 6 Computation comparison of ACO–SA with F&V

$J \times M$	*ACO–SA*			
	MaxPRD	MinPRD	APRD	T(s)
20 * 5	0	0	0	0.26
20 * 10	0	0	0	0.51
20 * 20	0	0	0	0.59
50 * 5	0.01,187	−0.63427	−0.273255	1.73
50 * 10	0.00343	−0.268361	−0.082688	2.83
50 * 20	0.08232	−0.16538	−0.02806	3.21
100 * 5	−0.61619	−1.55893	−1.09585	27.27
100 * 10	−0.52152	−1.08169	−0.75413	52.32
100 * 20	−0.33321	−0.83356	−0.51207	66.31
200 * 10	−1.45381	−2.75527	−2.0195	189.56
200 * 20	−0.75124	−1.83107	−1.27409	215.62
Average	−0.3253	−0.8298	−0.5405	50.92

Statistical analyses and extensive experimental demonstrate the efficiency of the proposed ACO–SA algorithm owing to an average PRD of −4.85. In order to prove the efficiency of our algorithms, we use another method based on VNS for local search phase. The computational results are indicated that hybridization ACO with SA is viable and reliable method and really competitive and provides promising computational results. As the future work, other neighborhood's structures can be applied in SA algorithm. In addition, the proposed algorithm can be used to solve the NWFSP with set-up times, maintenance of machines and transportation times. Furthermore, combining ACO with other local search-based algorithms such as Iterated local search or tabu search algorithm is possible research method.

References

Ahmadizar F (2012) A new ant colony algorithm for makespan minimization in permutation flow shops. Comput Ind Eng 63:355–361

Aldowaisan T, Allahverdi A (2003) New heuristics for no-wait flowshops to minimize makespan. Comput Oper Res 30:1219–1231

Bagchi TP, Gupta JND, Sriskandarajah C (2006) A review of TSP based approaches for flowshop scheduling. Eur J Oper Res 169:816–854

Behnamian J, Zandieh M, Fatemi Ghomi SMT (2009) Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. Expert Syst Appl 32:9637–9644

Bonney MC, Gundry SW (1976) Solutions to the constrained flowshop sequencing problem. Oper Res Q 27:869–883

Bonyadi MR, Li X (2010) A new discrete electromagnetism-based meta-heuristic for solving the multidimensional knapsack problem using genetic operators. Oper Res Int J 12:229–252

Carlier J (1978) Ordonnancements a contraints disjonctives. RAIRO Rech Oper 12:331–351

Colorni A, Dorigo M, Maniezzo (1991) Distributed optimization by ant colonies. In: Proceedings of the first european conference on artificial life, Paris

Dai M, Tang D, Giret A, Salido MA, Li WD (2013) Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robot Comput Inter Manuf 29:418–429

Ding JY, Song S, Gupta JND, Zhang R, Chiong R, Wu C (2015) An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. Appl Soft Comput 30:604–613

Elmi A, Solimanpur M, Topaloglu S, Elmi A (2011) A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. Comput Ind Eng 61:171–178

Fink A, Voβ S (2003) Solving the continuous flowshop scheduling problem by meta-heuristics. Eur J Oper Res 151:400–414

Framinan JM, Nagano MS, Moccellin JV (2010) An efficient heuristic for total flowtime minimisation in no-wait flowshops. Int J Adv Manuf Technol 46(9–12):1049–1057

Frutos M, Tohmé F (2013) A multi-objective memetic algorithm for the job-shop scheduling problem. Oper Res Int J 13:233–250

Gangadharan R, Rajendran C (1994) Heuristic algorithms for scheduling in the no-wait flowshop. Int J Prod Econ 32:285–290

Garay MR, Johnson DS (1979) Computers and Intractability: a guide to the theory of NP-completeness. Freedman, San Francisco

Grabowski J, Pempera J (2005) Some local search algorithms for no-wait flow-shop problem with makespan criterion. Comput Oper Res 32:2197–2212

Guo Y, Suhl L, Thiel MP (2005) Solving the airline crew recovery problem by a genetic algorithm with local improvement. Oper Res Int J 5:241–259

Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with locking and no-wait in process. Oper Res 44:510–525

Huang RH, Yang CL, Liu SC (2015) No-wait flexible flow shop scheduling with due windows. Math Probl Eng 2015:9

Jolai F, Asefi H, Rabiee M, Ramezanian P (2013) Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. Sci Iran 20:861–872

Kashan AH, Karimi B (2008) Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework. J Oper Res Soc 59:1269–1280

Khorramizadeh M, Riahi V (2015) A bee colony optimization approach for mixed blocking constraints flow shop scheduling problems. Math Probl Eng 2015:612604. doi:10.1155/2015/612604

King JR, Spachis AS (1980) Heuristics for flowshop scheduling. Int J Prod Res 18:343–357

Laha D, Chakraborty UK (2009) A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. Int J Adv Manuf Technol 41:97–109

Li X, Wu C (2008) Heuristic for no-wait flow shops with makespan minimization based on total idle-time increments. Sci China Ser F Inf Sci 51(7):896–909

Li X, Wang Q, Wu C (2008) Heuristic for no-wait flow shops with makespan minimization. Int J Prod Res 46(9):2519–2530

Li X, Baki MF, Aneja YP (2011) Flowshop scheduling to minimize the total completion time with a permanently present operator: models and ant colony optimization metaheuristic. Comput Oper Res 38:152–164
Liu Y-F, Liu S-Y (2013) A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. Appl Soft Comput 13:1459–1563
Liu B, Wang L, Jin YH (2007) An effective hybrid particle swarm optimization for no-wait flowshop scheduling. Int J Adv Manuf Technol 31:1001–1011
Low C (2005) Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Comput Oper Res 32:2013–2025
Low C, Yeh JL, Huang KI (2004) A robust simulated annealing heuristic for flow shop scheduling problems. Int J Adv Manuf Technol 23:762–767
Marinakis Y, Marinaki M, Matsatsinis N, Zopounidis C (2009) Evolution of the population of a genetic algorithm using particle swarm optimization: application to clustering analysis. Oper Res Int J 9:105–120
Mirabi M (2011) Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. Int J Adv Manuf Technol 55:317–326
Mirsanei HS, Zandieh M, Moayed MJ, Khabbazi MR (2011) A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. J Intell Manuf 22:965–978
Mladenovíc M, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100
Nawaz M, Enscore E, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11:11–95
Pan QK, Tasgetiren MF, Liang YC (2008a) A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Comput Oper Res 35:2807–2839
Pan QK, Wang L, Zhao BH (2008b) An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. Int J Adv Manuf Technol 38:778–786
Qian B, Wang L, Huang DX, Wang X (2009) A DE-based approach to no-wait flow-shop scheduling. Comput Ind Eng 57:787–805
Rabiee M, Sadeghi rad R, Mazinani M, Sadeghi rad R (2014) An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines. Int J Adv Manuf Technol 71:1229–1245
Rajendran C (1993) Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. Int J Prod Econ 29:65–73
Rajendran C (1994) A no-wait flowshop scheduling heuristic to minimize makespan. J Oper Res Soc 45:472–478
Rajendran C, Ziegler H (2004) Ant-colony algorithm for permutation flow-shop scheduling to minimize makespan/total flowtime of jobs. Eur J Oper Res 155:426–438
Reeves C (1995) A genetic algorithm for flow-shop sequencing. Comput Oper Res 22:5–13
Riahi V, Kazemi M (2015) A hybrid heuristic algorithm for the no-wait flowshop scheduling problem. Computer science and software engineering (CSSE), 2015 In: International symposium on 1–6
Rui Z, Shilong W, Zheqi Z (2014) An ant colony algorithm for job shop scheduling problem with tool flow. In: Proceedings of the IMechE, part B: Journal of engineering manufacture
Samarghandi H, ElMekkawy TY (2012) A meta-heuristic approach for solving the no-wait flow-shop problem. Int J Prod Res 50:7313–7326
Schuster CJ, Framinan JM (2003) Approximative procedures for no-wait job shop scheduling. Oper Res Lett 31:308–318
Shyu SJ, Lin BMT, Yin PY (2004) Application of ant colony optimization for no-wait flow-shop scheduling problem to minimize the total completion time. Comput Ind Eng 47:181–193
Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285
Tseng LY, Lin YT (2010) A hybrid genetic algorithm for no-wait flowshop scheduling problem. Int J Prod Econ 128:144–152
Wang HM, Chou FD, Wu FC (2011) A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan. Int J Adv Manuf Technol 53:761–776
Xiao J, Yang H, Zhang C, Zheng L, Gupta JND (2015) A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times. Comput Oper Res 63:72–82
Xu R, Chen H, Li X (2012) Makespan minimization on single batch-processing machine via ant colony optimization. Comput Oper Res 39:582–593
Ying KC, Lee ZJ, Lu CC, Lin SW (2012) Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups. Int J Adv Manuf Technol 58:671–682