Omega ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics ☆

Shih-Wei Lin [a,b], Kuo-Ching Ying [c,*]

[a] Department of Information Management, Chang Gung University, Taoyuan, Taiwan
[b] Department of Medical Research and Development, Linkou Chang Gung Memorial Hospital, Taoyuan, Taiwan
[c] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 14 July 2015
Accepted 5 December 2015

Keywords:
Scheduling
No-wait flowshop
Makespan
Matheuristics

ABSTRACT

The no-wait flowshop scheduling problem (NWFSP) with makespan minimization is a well-known strongly *NP*-hard problem with applications in various industries. This study formulates this problem as an asymmetric traveling salesman problem, and proposes two matheuristics to solve it. The performance of each of the proposed matheuristics is compared with those of the best existing algorithms on 21 benchmark instances of Reeves and 120 benchmark instances of Taillard. Computational results show that the presented matheuristics outperform all existing algorithms. In particular, all tested instances of the problem, including a subset of 500-job and 20-machine test instances, are solved to optimality in an acceptable computational time. Moreover, the proposed matheuristics can solve very hard and large NWFSPs to optimality, including the benchmark instances of Vallada et al. and a set of 2000-job and 20-machine problems. Accordingly, this study provides a feasible means of solving the *NP*-hard NWFSP completely and effectively.

1. Introduction

The flowshop scheduling problem (FSP) has been one of the most intensively discussed classes of problems in operations research literature over the past five decades [1–5]. Of particular practical interest is variants of FSPs, called no-wait FSPs (NWFSPs), that are widely applied in various industries, such as the chemicals, plastics, metals, electronics, pharmaceuticals, and food-processing industries [6,7]. For technological reasons, in these industries, no in-process waiting is allowed between any two consecutive machines, such that once the processing of a job begins, subsequent processing must be continuously carried out on all machines with no interruption until completion. This paper focuses on the NWFSP with the objective of minimizing the makespan, which can be written as $F_m|nwt|C_{\max}$ using the standard 3-tuple notation of Graham et al. [8], where F_m is a flowshop with m machines, nwt denotes the no-wait restriction and $C_{\max}$ indicates that the objective is to minimize the makespan. This problem is a member of the set of strongly *NP*-hard problems for three or more machines [9].

In view of the significance of the $F_m|nwt|C_{\max}$ problem in both theory and engineering applications, effective and efficient algorithms for solving it are required. Gilmore and Gomory solved the two-machine case of the $F_m|nwt|C_{\max}$ problem using an $O(n \log n)$ time algorithm with a sub-tour patching technique [10]. Reddi and Ramamoorthy [11] and Wismer [12] were the first to address the $F_m|nwt|C_{\max}$ problem with three or more machines. Many researchers have since attempted to develop effective and efficient algorithms for solving this problem. An early comprehensive survey of the $F_m|nwt|C_{\max}$ problem can be found in Hall and Sriskandarajah [13].

With respect to exact methods, Selen and Hott [14] presented a mixed integer goal programming model for solving the multi-objective NWFSP. Van der Veen and Van Dal [15] have proven that some special cases of NWFSPs are solvable using polynomial time solution algorithms if the processing times on all but two machines are fixed. To the best of our knowledge, no exact method has yet been proposed for solving the $F_m|nwt|C_{\max}$ problem. Given the *NP*-nature of this problem, all previous studies of this topic have focused on developing heuristic algorithms in order to find good (although not necessarily optimal) solutions to this intractable problem in a relatively short time.

The heuristic algorithms that are available for solving the $F_m|nwt|C_{\max}$ problem can be classified into two main categories: constructive heuristics and meta-heuristics. Table 1 summarizes the various constructive heuristics and meta-heuristics in

☆"This manuscript was processed by Associate Editor Sterna".

* Corresponding author. Tel.: +886 2 2771 2171; fax: +886 2 2731 7168.
E-mail address: kcying@ntut.edu.tw (K.-C. Ying).

http://dx.doi.org/10.1016/j.omega.2015.12.002
0305-0483/

Table 1
Constructive heuristics and meta-heuristics for the $F_m|nwt|C_{\max}$ problem.

Year	Author(s)	Acronym	Type[a]	Performance be superior to
1976	Bonney and Gundry [16]	S/M	C	Palmer's method, Gupta's algorithm
1980	King and Spachis [17]	LBJD(sc), LBJD(sc)*), LBJD(mc), MLSS(mc), MCL(mc)	C	MCL(mc) is better than other four compared heuristics
1993	Gangadharan and Rajendran [18]	GAN-RAJ	C	S/M, SC, MC
1994	Rajendran [19]	RAJ	C	S/M, SC, MC
1995	Gonzalez et al. [22]	GA	M	Palmer's method, Gupta's algorithm, RAJ
2003	Aldowaisan and Allahverdi [23]	SA, SA-1, SA-2, GEN, GEN-1, GEN-2	M	GAN-RAJ, RAJ
2005	Grabowski and Pempera [26]	DS, DS+M, TS, TS+M, TS+MP	M	RAJ, VNS, GASA
2006	Schuster [27]	FTS	M	GASA
2007	Liu et al. [28]	HPSO	M	VNS, GASA
2008	Pan et al. [29]	DPSO	M	HPSO, RAJ, VNS, GASA
2008	Pan et al. [30]	HDPSO	M	HPSO, DPSO
2008	Pan et al. [31]	IIG	M	RAJ, TS, TS+M, TS+MP, DPSO
2008	Li et al. [20]	CH	C	GAN-RAJ, RAJ
2009	Laha and Chakraborty [21]	LC	C	GAN-RAJ, RAJ and two compared heuristics
2009	Qian et al. [32]	HDE	M	HPSO
2010	Tseng and Lin [33]	HGA	M	RAJ, VNS, GASA, TS, HPSO
2011	Jarboui et al. [34]	GA-VNS	M	SA, TS, VNS, DPSO
2012	Samarghandi and ElMekkawy [35]	TS-PSO	M	VNS, GASA, DS, DS+M, TS, TS+M, TS+MP
2013	Davendra et al. [36]	DSOMA	M	DPSO
2015	Ding et al. [37]	TMIIG	M	DPSO, IIG, HDE, HGA, GA-VNS, TS-PSO

[a] C: constructive heuristic, M: meta-heuristic.

chronological order by publication. Several noteworthy constructive heuristics have been proposed for solving the $F_m|nwt|C_{\max}$ problem. Bonney and Gundry [16] and King and Spachis [17] pioneered constructive heuristic algorithms to solve the $F_m|nwt|C_{\max}$ problem. In 1976, Bonney and Gundry [16] developed a slope matching (S/M) method which used geometrical relationships between the cumulative process times. In the same year, King and Spachis [17] proposed two single-chain heuristics (LBJD(sc) and LBJD(sc)*) and three multiple-chain heuristics (LBJD(mc), MLSS (mc) and MCL(mc)), to solve the $F_m|nwt|C_{\max}$ problem. Their computational results revealed that the overall performance of the MCL(mc) heuristic to be excellent. Gangadharan and Rajendran [18] and Rajendran [19] presented additional heuristics, named GAN-RAJ and RAJ, for solving the same problem; their heuristics were shown to be superior to the S/M [16], SC and MC heuristics [17]. Li et al. [20] introduced a composite heuristic (CH), based on an objective increment method, which outperformed GAN-RAJ [18] and RAJ [19] and had the lowest CPU time of all the algorithms to which it is compared. Laha and Chakraborty [21] presented a constructive heuristic, called the LC heuristic, for solving the $F_m|nwt|C_{\max}$ problem, based on the principle of job insertion. The empirical results demonstrated that the solutions found using the LC heuristic were significantly better than those using the GAN-RAJ [18], RAJ [19] and two other compared heuristics. To the best of the authors' knowledge, the LC heuristic is currently the state-of-the-art constructive heuristic for solving the $F_m|nwt|C_{\max}$ problem.

Some remarkable meta-heuristics have been developed for solving the $F_m|nwt|C_{\max}$ problem. Gonzalez et al. [22] developed a hybrid genetic algorithm (GA) for solving the $F_m|nwt|C_{\max}$ problem; it produced comparable or better solutions to benchmark problems than known heuristic algorithms. Aldowaisan and Allahverdi [23] proposed six meta-heuristics (SA, SA-1, SA-2, GEN, GEN-1, GEN-2) based on simulated annealing (SA) and GA to solve the problem. Their computational results showed the best two of the six algorithms to be SA-2 and GEN-2, which outperformed GAN-RAJ [18] and RAJ [19], but required significantly more processing time.

In the same year that Aldowaisan and Allahverdi [23] proposed their six meta-heuristics, Schuster and Framinan [24] provided two algorithms, including a variable neighborhood search (VNS) algorithm and a hybrid algorithm that used both SA and GA (GASA) for solving the no-wait jobshop scheduling problem. The authors showed that the VNS and GASA algorithms were superior to the RAJ [19], even though they were not specifically designed for solving the $F_m|nwt|C_{\max}$ problem. Framinan and Schuster [25] improved upon the results of Schuster and Framinan [24] in the jobshop case, using a meta-heuristic called complete local search with memory (CLM).

Subsequently, more meta-heuristics have been developed for solving the $F_m|nwt|C_{\max}$ problem. Grabowski and Pempera [26] developed and compared two variants of descending search (DS, DS+M) and three Tabu search (TS)-based algorithms (TS, TS+M, TS+MP) which were more effective in finding high quality solutions than all other previous methods, including RAJ [19], VNS [24] and GASA [24]. Schuster [27] implemented a fast Tabu search (FTS) algorithm for solving the no-wait jobshop scheduling problem and the $F_m|nwt|C_{\max}$ problem, and find that it compared extremely well to the GASA algorithm [24]. Liu et al. [28] presented an effective hybrid particle swarm optimization (HPSO) for solving the problem. Their comparisons of HPSO with other algorithms, such as the VNS [24] and GASA [24] algorithms, demonstrated the effectiveness of the HPSO algorithm. Pan et al. conducted a series of studies and proposed a discrete particle swarm optimization (DPSO) algorithm [29], a hybrid discrete particle swarm optimization (HDPSO) algorithm [30] and an improved iterated greedy (IIG) algorithm [31]. Their computational results showed that these meta-heuristic algorithms to be superior to several of the best heuristics reported in the literature, in terms of quality of the search, robustness and efficiency. Qian et al. [32] proposed an effective hybrid differential evolution (HDE) algorithm for solving the same problem; the simulation results demonstrated that it was superior to the HPSO algorithm [28]. Tseng and Lin [33] proposed a hybrid genetic algorithm (HGA), which hybridized the genetic algorithm and a novel local search scheme. Their computational results, based on two well-known benchmarks, showed that the proposed HGA yielded better results than those obtained using the RAJ [19], VNS [24], GASA [24], TS [26] and HPSO [28] algorithms. Jarboui et al. [34] propose a hybrid genetic algorithm (GA-VNS) that applied VNS as an improvement procedure in the final step of the genetic algorithm. Their computational results show that GA-VNS provided competitive results and better upper bounds (UBs), while the VNS algorithm [24] was better than the GA-VNS for large test instances. Samarghandi and ElMekkawy [35] developed a

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

hybrid TS and PSO algorithm (TS-PSO). Their computational results showed that it outperformed the VNS [24], GASA [24], DS [26], DS+M [26], TS [26], TS+M [26] and TS+MP [26] algorithms and improved upon some of the best-known solutions to the considered test problems. Recently, Davendra et al. [36] introduced a novel discrete self-organizing migrating algorithm (DSOMA) for solving the $F_m|nwt|C_{\max}$ problem. Although it is not the best algorithm for solving this class of problem, the results obtained using DSOMA are comparable with those of the best-performing heuristics in the literature. More recently, Ding et al. [37] proposed a Tabu-mechanism-improved iterated greedy (TMIIG) algorithm for solving the problem. Their empirical results confirmed that the TMIIG algorithm was more effective than all of the other well-performing heuristic algorithms. To the best of our knowledge, TMIIG is by far the best meta-heuristic algorithm for solving the $F_m|nwt|C_{\max}$ problem.

In spite of the fact that the $F_m|nwt|C_{\max}$ problem has been extensively studied over the last four decades, none of the available algorithms can generate optimal solutions to large $F_m|nwt|C_{\max}$ problems in a reasonable amount of time. Therefore, this work presents two matheuristics to solve this problem optimally. The proposed matheuristics consist of three phases. In the first phase, an initial seed sequence is quickly generated using a modified Nawaz–Enscore–Ham (MNEH) constructive heuristic based on the well-known Nawaz–Enscore–Ham (NEH) heuristic [38,39]. In the second phase, after the $F_m|nwt|C_{\max}$ problem is reformulated as a special case of the asymmetric traveling salesman problem (ATSP), an improved solution that serves as an upper bound (UB) of the optimal solution is obtained by using the Lin–Kernighan–Helsgaun (LKH) heuristic [40] to enhance the search efficiency. In the third phase, an optimal solution is obtained by solving the binary integer programming (BIP) mathematical model that corresponds to the transformed ATSP.

The remainder of this paper is organized as follows. Section 2 formally defines the $F_m|nwt|C_{\max}$ problem, which is then formulated as a BIP mathematical model of its equivalent ATSP. Section 3 describes in detail the procedures for implementing the proposed matheuristics. Section 4 discusses the computational results obtained by using four benchmark problem sets, and then compares the performance of the proposed matheuristics with that of the best-so-far algorithms. Section 5 provides some final remarks and suggests directions for future research.

2. Problem formulation

This section defines the $F_m|nwt|C_{\max}$ problem. The connection of this problem to a special case of the ATSP is exploited and a corresponding BIP mathematical model is described.

2.1. Problem definition

The following notation is used to define the $F_m|nwt|C_{\max}$ problem and to simplify the exposition of the mathematical models.

2.1.1. Parameters

i, i'	machine index
j, j'	job/city index
k	position index
n	number of jobs to be scheduled
m	number of machines in the no-wait flowshop
$p_{j,i}$	processing time of job j on machine i
$o_{j,i}$	operation of job j on machine i

2.1.2. Decision variables

π	a feasible solution/schedule
π_k	job in position k of a feasible schedule π
d_{π_{k-1},π_k}	minimum delay on the first machine between the initiations of two consecutive jobs in positions $k-1$ and k, rendered necessary by the no-wait restriction
C_j	completion time of job j
$C_{k,i}$	completion time of job in position k on machine i
$C_{\max}$	makespan
$X_{j,k}$	a binary variable, $X_{j,k} = \begin{cases} 1, \text{ if job } j \text{ occupies position } k \\ 0, \text{otherwise} \end{cases}$
$C_{\max}(\pi)$	makespan of a given schedule π
φ	a Hamiltonian tour of the ATSP
$d(\varphi)$	length of a given Hamiltonian tour φ
$c_{j,j'}$	difference between the completion times of two consecutive jobs/cities j and j' in a given Hamiltonian tour φ
$x_{jj'}$	a binary variable, $x_{jj'} = \begin{cases} 1, \text{if job } j' \text{ is directly visited followed by job } j \\ 0, \text{otherwise} \end{cases}$
Q	a nonempty proper subset of n jobs
X	a feasible solution to the ATSP
S	a feasible solution set for the ATSP, where $S = \{x_{jj'} \mid \sum_{j \in Q} \sum_{j' \notin Q} x_{jj'} \geq 1$, for every nonempty proper subset Q of the n jobs$\}$

Based on the above notation, the $F_m|nwt|C_{\max}$ problem that is considered herein can be defined formally as follows. Consider a set $\vartheta = \{J_1, \ldots, J_n\}$ of n pre-determined jobs to be processed on m machines, such that the sequences of jobs on all machines are identical. All jobs are simultaneously available for processing at the beginning of the planning horizon, while all machines are continuously available. Each job j $(j = 1, 2, \ldots, n)$ comprises of a strictly ordered sequence of m operations $(o_{j,1}, o_{j,2}, \ldots, o_{j,m})$, where $o_{j,i}$ is the operation of job j that is pre-assigned to machine i $(i = 1, 2, \ldots, m)$, and should be processed during processing time $p_{j,i}$ without interruption. At any moment, every job is being processed at most by one machine, and every machine can execute no more than one job at a time. No job queue is allowed to form at any machine, except the first one. Once the processing of a job is initiated on the first machine, that job must be processed continuously on all machines until completion. In order to ensure that the jobs are processed without interruption on all machines, the jobs must be processed in the same order on all machines (assuming non-zero processing times). Therefore, the $F_m|nwt|C_{\max}$ problem is generally assumed to be a permutation FSP. To satisfy the no-wait constraint, the beginning processing time of the job on the first machine may be delayed as required to ensure that it need not wait for processing on subsequent machines. The goal of this study is to find a feasible schedule $\pi = (\pi_1, \ldots, \pi_n)$ for the n jobs that minimizes the maximum completion time (or makespan), where π_k $(k = 1, \ldots, n)$ denotes the job that is assigned to position k. The no-wait constraint follows the fact that the makespan of π, $C_{\max}(\pi)$ can be obtained using the following equation [12,41]:

$$C_{\max}(\pi) = \sum_{k=2}^{n} d_{\pi_{k-1},\pi_k} + \sum_{i=1}^{m} p_{\pi_n, i}$$

where $d_{\pi_{k-1},\pi_k} = \max_{1 \leq i \leq m} \left\{ \sum_{i'=1}^{i} p_{\pi_{k-1},i'} - \sum_{i'=2}^{i} p_{\pi_k,i'-1} \right\}$ for $k = 2, \ldots, n$.

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

The mixed integer programming (MIP) model of the addressed $F_m|nwt|C_{\max}$ problem can be formulated as follows:

$$\text{Minimize } C_{\max} \tag{1}$$

subject to

$$C_{k,i} = C_{k,i-1} + \sum_{j=1}^{n} X_{j,k} \cdot p_{j,i}, \forall k, i > 1, \tag{2}$$

$$C_{k,i} \geq \sum_{j=1}^{n} X_{j,k} \cdot p_{j,i}, \forall k, i = 1, \tag{3}$$

$$C_{k,i} \geq C_{k-1,i} + \sum_{j=1}^{n} X_{j,k} \cdot p_{j,i}, \forall k > 1, i, \tag{4}$$

$$C_{\max} \geq C_{k,m}, \forall k, \tag{5}$$

$$\sum_{k=1}^{n} X_{j,k} = 1, \forall j, \tag{6}$$

$$C_{k,i} \geq 0, \forall k, i, \tag{7}$$

$$X_{j,k} \in \{0,1\}, \forall j, k. \tag{8}$$

The objective function (1) is to minimize the maximal completion time (makespan). Constraint set (2) specifies the relationship between the completion times of each job on two sequential machines. Constraint set (3) ensures that the completion time of each job on the first machine is more than or equal to its processing time on that machine. Constraint set (4) presents the relationship between the completion times of two consecutive jobs on each machine. Constraint set (5) sets the maximal completion time. Constraint set (6) ensures that every job is assigned to a single job position. Constraint set (7) defines that the completion time of each job is nonnegative. Finally, Constraint set (8) defines the binary variables.

2.2. Converting the problem into an ATSP, and formulating it as a BIP model

The $F_m|nwt|C_{\max}$ problem can be transformed into a special case of the $(n+1)$-city ATSP [12]. Let G denote the complete, arc-weighted, directed graph with vertex set $\{J_0, J_1, \ldots, J_n\}$ and the traffic cost $c_{j,j'}$ $(\forall j, j' \in \{0, 1, \ldots n\})$ on the arc from job/city J_j to $J_{j'}$ defined as follows, where J_0 is a dummy job with zero processing time on all machines:

$$c_{j,j'} = \begin{cases} \sum_{i=1}^{m} p_{j',i}, \text{if } j = 0 \\ 0, \quad \text{if } j' = 0 \\ C_{j'} - C_j = \max_{1 \leq i \leq m} \left\{ 0, \sum_{i'=i}^{m} (p_{j',i'} - p_{j,i'}) + p_{j,i'} \right\}, \text{otherwise} \end{cases} \tag{9}$$

As is well known [12,15], a feasible schedule $\pi = (\pi_1, \ldots, \pi_n)$ of the $F_m|nwt|C_{\max}$ problem can be represented as a directed Hamiltonian cycle $\varphi = (0, \pi_1, \ldots, \pi_n, 0)$ in the digraph G, which begins from the dummy city 0, goes to city π_1 and finally returns from city π_n to the dummy city 0. The $(n+1)$-city ATSP is solved by finding a minimal length $d(\varphi)$ of the directed Hamiltonian cycle $\varphi = (0, \pi_1, \ldots, \pi_n, 0)$ of all the cities; this process is equivalent to finding an optimal sequence $\pi = (\pi_1, \ldots, \pi_n)$ with a minimal makespan $C_{\max}(\pi)$ as the solution to the $F_m|nwt|C_{\max}$ problem, and $d(\varphi) = C_{\max}(\pi)$.

The above $(n+1)$-city ATSP can be formulated as the following BIP mathematical model, which is used in the third phase of the proposed matheuristics:

$$\text{Minimize } z = \sum_{j=0}^{n} \sum_{j'=0}^{n} c_{j,j'} x_{jj'} \tag{10}$$

subject to

$$\sum_{\substack{j=0 \\ j \neq j'}}^{n} x_{jj'} = 1, j' = 0, 1, \ldots, n, \tag{11}$$

$$\sum_{\substack{j'=0 \\ j' \neq j}}^{n} x_{jj'} = 1, j = 0, 1, \ldots, n, \tag{12}$$

$$X = \{x_{jj'}\} \in S, \tag{13}$$

$$x_{jj'} \in \{0,1\}, \forall j, j' = 0, 1, \ldots, n, \text{ and } j \neq j'. \tag{14}$$

The objective function (10) is to find a minimal length $d(\varphi)$ of the directed Hamiltonian cycle $\varphi = (0, \pi_1, \ldots, \pi_n, 0)$ of all the cities. Constraint set (11) ensures that only one city can be directly visited before city $j'(j' = 0, 1, \ldots, n)$. Constraint set (12) ensures that only one city can be directly visited after city $j(j = 0, 1, \ldots, n)$. Constraint (13) prevents sub-tours. Constraint set (14) defines the decision variable $x_{jj'}$ $(\forall j, j' = 0, 1, \ldots, n; j \neq j')$ as binary.

3. Proposed matheuristics

The proposed matheuristics consist of three phases. In the first phase, two versions of the MNEH constructive heuristics are used to obtain the initial seed sequences for solving the $F_m|nwt|C_{\max}$ problem. Then, in the second phase, the $F_m|nwt|C_{\max}$ problem is converted into an ATSP, and the LHK algorithm is then applied to improve upon the initial solution and obtain a near-optimal solution. Finally, in the third phase, the near-optimal solution obtained by the LHK algorithm is set as the UB of the corresponding BIP mathematical model of the transformed ATSP; the BIP model is solved using the Gurobi optimizer, a state-of-the-art mathematical programming solver, in order to obtain the optimal solution. Since two constructive heuristics are used to generate the initial seed sequence in the first phase, two versions of the matheuristic exist, which are Matheuristic$_1$ and Matheuristic$_2$. The three phases are described in detail below.

Phase I. : Using MNEH constructive heuristics to obtain initial seed sequences

The procedures for implementing the two versions of the MNEH constructive heuristics, which can rapidly generate an initial seed sequence for solving the $F_m|nwt|C_{\max}$ problem, are described as follows.

Step 1: Generate a job list $\pi = (\pi_1, \ldots, \pi_n)$ with respect to an indicator value. In this work, two versions of constructive heuristics, MNEH$_1$ and MNEH$_2$, are used to generate π. MNEH$_1$ serves to sort n jobs in a non-increasing order of their total processing times to yield a job list π. MNEH$_2$ uses the nearest neighbor (NN) algorithm [42] to yield a job list π. The procedure for executing the NN algorithm is as follows.

Step 1.1: Start from a dummy job as the current vertex.

Step 1.2: Find the shortest edge that connects the current vertex to an unvisited vertex V.

Step 1.3: Set the current vertex to V and mark V as visited.

Step 1.4: If all of the vertices in the domain are visited, then terminate; otherwise, go back to *Step* 1.2.

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

Step 2: Select the first two jobs from the job list $\pi=(\pi_1,\ldots,\pi_n)$; determine the best (minimum) makespan of the two sequences by placing π_1 in the first place and π_2 in the second place, and then by reversing that order. Do not change the relative positions of the two jobs with respect to each other in the remaining steps of the $MNEH_1$/$MNEH_2$ constructive heuristic.
Step 3: Pick the job that is in the next position in π; find the best sequence by placing this job in all possible positions in the partial sequence as developed thus far.
Step 4: Repeat Step 3 until all jobs are placed in the initial seed sequence.

Phase II. : Using the LKH heuristic to improve the initial seed sequence.

After the initial seed sequence is obtained in Phase I, a dummy job/city (denoted as zero) is added to the beginning of the initial seed sequence. In the second phase, the $F_m|nwt|C_{\max}$ problem is transformed into a special case of the ATSP, and is solved using the LKH heuristic [40], which is a well-known and highly effective algorithm. Eq. (9) gives the traffic cost $c_{j,j'}$ on the arc of the transformed ATSP from job/city J_j to $J_{j'}$ $(\forall j,j' \in \{0,1,\ldots n\})$. Fig. 1 presents a sketch of the LKH heuristic. First, the specification of the problem to be solved is read. In order to create a candidate set of tours, the algorithm generates a set of initial tours in some random fashion, and then repeatedly performs the local optimization on each initial tour to reduce its length until a tour is obtained for which no exchange can yield an improvement. After the initial length of the best tour is set to a large floating-point number (denoted as DBL_MAX), the algorithm repeatedly improves the candidate solutions for a specified number of runs. Owing to space limitations, this work does not provide details of the LKH heuristic. The reader is referred to the paper by Helsgaun [40] for a detailed discussion of the use of the LKH heuristic for solving the traveling salesman problem. Execution of the LKH heuristic yields an improved feasible solution, which serves as a UB on the optimum.

Phase III. : Solving the corresponding BIP model to obtain an optimal solution.

Fig. 2 presents an outline of Phase III. Firstly, the UB obtained in Phase II is input into the BIP mathematical model. Then, as shown in Fig. 2, a relaxed ATSP model is used repeatedly to obtain feasible solutions of the BIP mathematical model that corresponds to the transformed ATSP. Constraint (13), which prevents sub-tours of the transformed ATSP, significantly affects the performance of the proposed algorithm in solving the MIP model. The MIP model is therefore relaxed by removing constraint (13), such that the relaxed model is much more easily solved at the expense of generating sub-tours. When the optimal solution with sub-tours is obtained for the relaxed TSP model, these sub-tours are added to the model as new constraints; the relaxed model is then solved again. Finally, an optimal solution can be obtained using the Gurobi optimizer (version 6.0).

LKH heuristic

```
void main ()  {
    ReadProblemData ();
    CreateCandidateSet ();
    BestCost= DBL_MAX;
    For  (Run= 1; Run <= Runs; Run++)  {
        double Cost= FindTour ();
        if (Cost < BestCost)  {
            RecordBestTour ();
            BestCost= Cost;
        }
    }
}
```

Fig. 1. A sketch of the LKH heuristic [33].

Phase III of the proposed matheuristics

```
Begin
    Input upper bound obtained in the Phase II;
    Solve the relaxed model (BIP model without Constraint (13));
    While the sub-tour exist
    Begin
        Add the sub-tour as new constraint;
        Solve relaxed model;
    End
End
```

Fig. 2. An outline of Phase III of the proposed matheuristics.

4. Computational results

This section elucidates the computational experiments that are performed to evaluate the performance of the proposed matheuristics in solving the $F_m|nwt|C_{\max}$ problem. The following subsections compare the computational results obtained by applying the proposed matheuristics to the test problems with those obtained using other state-of-the-art algorithms.

4.1. Test problems

Four sets of benchmark problem instances are used to verify the effectiveness and efficiency of the proposed matheuristics, the first being the set of benchmark problems from the OR-Library as provided by Reeves [43]. This problem set comprises of 21 test instances in seven combinations, ranging from 20 jobs with five machines to 75 jobs with 20 machines. For each size of problem, three instances are provided.

The second benchmark problem set is composed of the 120 benchmark instances of Taillard [44], for which the processing time $p_{j,i}$ $(j=1,\ldots,n; i=1,\ldots,m)$ is an integer generated from the uniform distribution [1, 99]. The number of jobs $n=\{20, 50, 100, 200, 500\}$ and the number of machines $m=\{5, 10, 20\}$ yield 12 combinations, each with 10 test instances. Therefore, 120 test instances of the second problem set are used.

The third set of benchmark problem instances is the 480 new hard benchmark instances of the permutation FSP as proposed by Vallada et al. [45]. This benchmark problem set consists of 240 large instances and 240 small instances, with up to 800 jobs on 60 machines. Vallada et al. [45] generated thousands of instances and selected the hardest ones. Small instances include 24 combinations of $n=\{10, 20, 30, 40, 50, 60\}$ jobs with $m=\{5, 10, 15, 20\}$ machines. Large instances refer to 24 combinations of $n=\{100, 200, 300, 400, 500, 600, 700, 800\}$ jobs with $m=\{20, 40, 60\}$ machines. Ten instances of each combination yield a total of 480 instances.

In order to verify the capability of the proposed matheuristics to yield the optimal solution in solving the $F_m|nwt|C_{\max}$ problem, a new benchmark problem set is generated herein using the same experiment design as Taillard [44], which consists of 30 very large-scale test instances. The fourth benchmark problem set comprises of three combinations of $n=\{1000, 1500, 2000\}$ jobs with $m=20$ machines. Ten test instances are generated for each size of problem, denoted as ta121 to ta130, ta131 to ta140, and ta141 to ta150 for $n=\{1000, 15000, 2000\}$, respectively.

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

4.2. Results and discussion

The proposed matheuristics are implemented in C++, compiled with the maximum execution speed option and run on a PC with an Intel® Core(™) 2 Quad Q9400 processor that runs at 2.66 GHz with 20 GB of RAM.

In order to demonstrate the effectiveness of the $MNEH_1$/$MNEH_2$ heuristics in the first phase of the matheuristic, computational experiments are conducted on the four benchmark problem sets. The heuristics are compared with the LC heuristic [21], a state-of-the-art constructive heuristic for solving the $F_m|nwt|C_{max}$ problem. Table 2 lists the average computational times (CPU times in seconds) required by the LC heuristic and the $MNEH_1$/$MNEH_2$ heuristics when applied to each benchmark problem set; the numbers of instances for which the former yield better, equally good, or worse solutions than the latter are shown. As seen in Table 2, the $MNEH_1$ and the $MNEH_2$ heuristics both outperform the LC heuristic, especially for large problems. The three constructive heuristics are run for approximately the same CPU time. Specifically, the proposed $MNEH_1$ heuristic finds better solutions than the LC heuristic in 10 out of 21, 62 out of 120, 328 out of 480, and 30 out of 30 test instances when applied to the first, second, third and fourth benchmark problem sets, respectively, whereas the proposed $MNEH_2$ heuristic finds better solutions than the LC heuristic in 11 out of 21, 61 out of 120, 340 out of 480, and 30 out of 30 test instances when applied to the first, second, third and fourth benchmark problem sets, respectively.

In order to verify the effectiveness and efficiency of the proposed matheuristics, the results are compared with those found using some state-of-the-art algorithms (DPSO [29], IIGA [31], HDE [32], HGA [33], GA-VNS [34], TS-PSO [35] and TMIIG [37]). Reeves [43] previously tested all seven of these algorithms on the 21 instances considered herein. Our study compares the proposed matheuristics with those seven algorithms when applied to the same benchmark problem set. Table 3 lists the computational results obtained by using $Matheuristic_1$, $Matheuristic_2$, the matheuristic without the UB, DPSO, IIGA, HDE, HGA, TS-PSO, HDE and TMIIG algorithms for solving the 21 Reeves's test instances. Column 1 in Table 3 (Name) provides the name of each instance. Column 2 ($n \times m$) specifies the combination of the number of jobs and the number of machines. Column 3 (Opt.) presents the optimal solutions obtained using $Matheuristic_1$ and $Matheuristic_2$. Columns 4–6 present the required execution times T(s) (CPU times in seconds) of $Matheuristic_1$, $Matheuristic_2$ and the matheuristic without the UB. The other columns specify the relative error rates (RER) and the execution times of the UB and the seven compared algorithms. The RER is given by: $RER = (C^h - Opt.)/Opt. \times 100\%$, where C^h denotes the makespan obtained by algorithm h; Opt. is the optimal solution obtained by the proposed matheuristics. As shown in Table 3, both $Matheuristic_1$ and $Matheuristic_2$ obtain

Table 2
Results of statistical tests (the $MNEH_1$/$MNEH_2$ heuristic vs. the LC heuristic).

Benchmark problem set	$MNEH_1$ vs. LC			$MNEH_2$ vs. LC			$MNEH_1$	$MNEH_2$	LC
	Better	Tie	Loss[a]	Better	Tie	Loss	Time (s)[a]	Time (s)	Time (s)
First	10	0	11	11	0	10	< 0.001	< 0.001	< 0.001
Second	62	2	54	61	0	59	0.001	0.001	0.001
Third	328	11	141	340	6	134	0.004	0.004	0.006
Forth	30	0	0	30	0	0	0.046	0.052	0.088

[a] CPU time in second.

Table 3
Computational Results for Reeves's benchmark test instances.

Name	$n \times m$	Opt.	$Matheuristic_1$	$Matheuristic_2$	Matheuristic w/o UB	UB	IIGA		DPSO		GA-VNS		HGA		TS-PSO		HDE		TMIIG	
			T (s)	T (s)	T (s)	RER	RER	T (s)	RER	T (s)	RER	T (s)	RER	T (s)	RER	T (s)	RER	T (s)	RER	T (s)
Rec1	20 × 5	1526	0.02	0.02	0.07	0.00	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.34	0.00	3.10	0.00	0.20
Rec3	20 × 5	1361	0.04	0.04	0.04	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.31	0.00	3.10	0.00	0.20
Rec5	20 × 5	1511	0.06	0.05	0.05	0.00	0.01	0.02	0.02	0.02	0.01	0.00	0.00	0.01	0.42	0.28	0.09	3.10	0.00	0.20
Rec7	20 × 10	2042	0.21	0.27	0.28	0.00	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.01	0.20	0.38	0.00	3.90	0.00	0.20
Rec9	20 × 10	2042	0.05	0.05	0.04	0.00	0.00	0.02	0.00	0.00	0.02	0.02	0.00	0.01	0.00	0.40	0.00	3.80	0.00	0.20
Rec11	20 × 10	1881	0.02	0.02	0.02	0.00	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.01	0.21	0.39	0.15	3.90	0.00	0.20
Rec13	20 × 15	2545	0.09	0.09	0.16	0.00	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.01	0.11	0.42	0.00	5.10	0.00	0.20
Rec15	20 × 15	2529	0.08	0.08	0.10	0.00	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.01	0.17	0.40	0.00	5.20	0.00	0.20
Rec17	20 × 15	2587	0.08	0.08	0.13	0.00	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.01	0.04	0.44	0.00	5.20	0.00	0.20
Rec19	30 × 10	2850	0.07	0.08	0.06	0.00	0.00	0.04	0.19	0.02	0.11	0.00	0.00	0.03	0.72	0.51	0.06	10.40	0.00	0.45
Rec21	30 × 10	2821	0.13	0.13	0.16	0.00	0.18	0.04	0.11	0.04	0.09	0.02	0.28	0.03	0.17	0.49	0.18	10.20	0.07	0.45
Rec23	30 × 10	2700	0.07	0.07	0.07	0.00	0.00	0.04	0.05	0.05	0.07	0.05	0.00	0.03	0.07	0.53	0.12	10.40	0.00	0.45
Rec25	30 × 15	3593	0.15	0.14	0.15	0.00	0.01	0.04	0.05	0.03	0.00	0.02	0.00	0.03	0.52	0.55	0.03	15.00	0.00	0.45
Rec27	30 × 15	3431	0.10	0.10	0.13	0.00	0.01	0.05	0.25	0.06	0.08	0.07	0.00	0.03	0.18	0.58	0.18	14.80	0.01	0.45
Rec29	30 × 15	3291	0.06	0.07	0.07	0.00	0.01	0.06	0.09	0.02	0.13	0.06	0.00	0.03	0.29	0.61	0.20	14.90	0.03	0.45
Rec31	50 × 10	**4307**[a]	0.24	0.23	0.25	0.00	0.37	0.11	0.42	0.33	0.51	0.34	0.63	0.27	1.10	1.29	0.64	3.80	0.22	1.25
Rec33	50 × 10	**4424**	0.12	0.11	0.09	0.00	0.76	0.10	0.62	0.34	0.71	0.36	0.77	0.25	1.07	1.91	0.96	3.70	0.39	1.25
Rec35	50 × 10	4397	0.18	0.19	0.19	0.00	0.15	0.10	0.29	0.34	0.40	0.30	0.62	0.23	0.63	1.85	0.74	3.70	0.19	1.25
Rec37	75 × 20	**8008**	0.54	0.56	0.52	0.00	0.77	0.18	0.85	1.16	0.90	0.35	1.41	1.45	1.29	3.42	0.89	14.70	0.41	2.81
Rec39	75 × 20	**8419**	0.67	0.67	0.89	0.00	0.73	0.18	0.82	0.99	0.76	0.98	1.02	1.28	1.43	3.51	0.82	14.60	0.57	2.81
Rec41	75 × 20	**8437**	0.43	0.44	0.54	0.00	0.81	0.18	0.71	0.88	0.68	1.12	0.81	1.07	1.58	3.50	0.79	14.70	0.50	2.81
Average			0.16	0.17	0.19	0.00	0.18	0.06	0.21	0.21	0.21	0.18	0.26	0.23	0.49	1.05	0.28	7.89	0.11	0.79

[a] The bold values indicate the optimal solutions that have not been found before.

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

optimal solutions for all test instances in less computational time than all the other algorithms, except for IIGA. The matheuristic without the UB algorithm can also find optimal solutions in all test instances, but requires more computational time than that of Matheuristic$_1$ and Matheuristic$_2$. For Reeves's benchmark test instances, applying the UB reduces the total average times required for Matheuristic$_1$ and Matheuristic$_2$ by 15.79% and 21.05%, respectively. The RERs of the UB in all test instances are zero, proving that the UB is very tight in the 21 instances of Reeves [43]. Notably, none of the seven state-of-the-art algorithms yields the optimal solutions in test instances Rec31, Rec33, Rec37, Rec39 and Rec41. The optimal solutions that have not been found before are presented in bold.

To the best of the authors' knowledge, only the GA-VNS and TMIIG algorithms have been tested on the Taillard benchmark problem set. Jarboui et al. [34] used only the GA-VNS to solve 110 out of 120 test instances, and did not execute it for the ten 500-job with 20-machine instances. The GA-VNS was coded in C++ programming language and ran in Windows XP on a desktop PC with an Intel Pentium IV, 3.2 GHz processor with 512 MB of memory. Jarboui et al. [34] implemented 20 replications for each of the 110 test instances, which output the best solutions as new UBs. The TMIIG was coded using C++, with the computational experiment executed on a personal computer (PC) with an Intel Core (TM) CPU running at 3.20 GHz in a Windows 7 Operating System environment. Ding et al. [37] utilized the TMIIG algorithm to solve all 120 Taillard problem instances, providing 43 new best solutions. The termination condition of the maximum running time was set as $T_{max}=n^2/2$ ms for both GA-VNS and TMIIG.

Tables 4 and 5 present the computational results of Matheuristic$_1$, Matheuristic$_2$, the matheuristic without the UB, GA-VNS and TMIIG for the Taillard benchmark problem set. Table 4 lists the relative error rates (RER) and the average computational times, respectively, when Matheuristic$_1$, Matheuristic$_2$, the matheuristic without the UB, GA-VNS and TMIIG are applied to problems of various sizes. In Table 4, Column 1 ($n \times m$) presents the combination of the number of jobs and the number of machines; Column 2 presents the average optimal solutions over 10 instances of each combination; Columns 3–5 list the average computational times T(s) (CPU times in seconds) required to execute Matheuristic$_1$, Matheuristic$_2$ and the matheuristic without the UB, respectively; Column 6 presents the average RERs of the UB; Columns 7 and 8 present the average RERs obtained and the computational times required using GA-VNS; and Columns 9 and 10 present the average RERs obtained and the computational times required using TMIIG. Table 4 reveals that both Matheuristic$_1$ and Matheuristic$_2$ yield optimal solutions in less computational time than the two other algorithms. The matheuristic without the UB also yields optimal solutions in all test instances, but requires more computational time than that of Matheuristic$_1$ or Matheuristic$_2$. The total average times saved by adding the UB are 46.12% and 46.47% for Matheuristic$_1$ and Matheuristic$_2$, respectively. The maximum RER among all test instances is 0.037%, thus verifying that the UB is very strict. Notably, neither GA-VNS nor TMIIG yields optimal solutions when the number of jobs is equal to, or larger than, 100. Table 5 lists the optimal solutions obtained using the proposed matheuristics in the 120 Taillard benchmark test instances; the bold values indicate the optimal solutions that have not been found before.

Matheuristics can typically be used in (at least) two ways. First, they can be used as heuristics, if the matheuristic is terminated after some time (possibly after some iterations), to yield an approximate solution. Second, they can also be used as exact

Table 5
Optimal solutions for 120 Taillard's benchmark test instances.

Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.
Ta01	1486	Ta31	3160	Ta61	**6361**[a]	Ta91	**15225**
Ta02	1528	Ta32	3432	Ta62	**6212**	Ta92	**14990**
Ta03	1460	Ta33	3210	Ta63	**6104**	Ta93	**15257**
Ta04	1588	Ta34	3338	Ta64	**5999**	Ta94	**15103**
Ta05	1449	Ta35	3356	Ta65	**6179**	Ta95	**15088**
Ta06	1481	Ta36	3346	Ta66	**6056**	Ta96	**14976**
Ta07	1483	Ta37	3231	Ta67	**6221**	Ta97	**15277**
Ta08	1482	Ta38	3235	Ta68	**6109**	Ta98	**15133**
Ta09	1469	Ta39	3070	Ta69	**6355**	Ta99	**14985**
Ta10	1377	Ta40	3317	Ta70	**6365**	Ta100	**15213**
Ta11	2044	Ta41	4274	Ta71	**8055**	Ta101	**19531**
Ta12	2166	Ta42	4177	Ta72	**7853**	Ta102	**19942**
Ta13	1940	Ta43	4099	Ta73	**8016**	Ta103	**19759**
Ta14	1811	Ta44	4399	Ta74	**8328**	Ta104	**19759**
Ta15	1933	Ta45	4322	Ta75	**7936**	Ta105	**19697**
Ta16	1892	Ta46	4289	Ta76	**7773**	Ta106	**19826**
Ta17	1963	Ta47	4420	Ta77	**7846**	Ta107	**19946**
Ta18	2057	Ta48	4318	Ta78	**7880**	Ta108	**19872**
Ta19	1973	Ta49	4155	Ta79	**8131**	Ta109	**19784**
Ta20	2051	Ta50	4283	Ta80	**8092**	Ta110	**19768**
Ta21	2973	Ta51	6129	Ta81	**10675**	Ta111	**46121**
Ta22	2852	Ta52	5725	Ta82	**10562**	Ta112	**46627**
Ta23	3013	Ta53	5862	Ta83	**10587**	Ta113	**46013**
Ta24	3001	Ta54	5788	Ta84	**10588**	Ta114	**46396**
Ta25	3003	Ta55	5886	Ta85	**10506**	Ta115	**46251**
Ta26	2998	Ta56	5863	Ta86	**10623**	Ta116	**46490**
Ta27	3052	Ta57	5962	Ta87	**10793**	Ta117	**46043**
Ta28	2839	Ta58	5926	Ta88	**10801**	Ta118	**46368**
Ta29	3009	Ta59	5876	Ta89	**10703**	Ta119	**46240**
Ta30	2979	Ta60	5957	Ta90	**10747**	Ta120	**46292**

[a] The bold values indicate the optimal solutions that have not been found before.

Table 4
Average execution times and RERs for Taillard's benchmark test instances.

$n \times m$	Opt. (average)	Matheuristic$_1$	Matheuristic$_2$	Matheuristic w/o UB	UB	GA-VNS		TMIIG	
		T(s)	T(s)	T(s)	RER	RER	T (s)	RER	T (s)
20 × 5	1480.3	0.05	0.05	0.05	0.007	0.00	0.20	0.00	0.20
20 × 10	1983.0	0.11	0.11	0.17	0.003	0.00	0.20	0.00	0.20
20 × 20	2971.9	0.29	0.34	0.35	0.009	0.00	0.20	0.00	0.20
50 × 5	3269.5	0.16	0.16	0.25	0.007	0.02	1.25	0.02	1.25
50 × 10	4273.6	0.28	0.28	0.35	0.000	0.00	1.25	0.00	1.25
50 × 20	5897.4	1.25	1.23	1.30	0.000	0.00	1.25	0.00	1.25
100 × 5	6196.1	0.41	0.40	0.71	0.000	0.44	5.00	0.35	5.00
100 × 10	7991.0	1.31	1.28	1.64	0.000	0.33	5.00	0.30	5.00
100 × 20	10658.5	1.75	1.73	2.29	0.000	0.32	5.00	0.32	5.00
200 × 10	15124.7	3.17	3.16	4.49	0.037	0.91	20.00	0.74	20.00
200 × 20	19788.4	5.85	5.79	6.76	0.012	0.86	20.00	0.78	20.00
500 × 20	46284.1	54.58	54.23	101.30	0.000	NA	NA	1.13	125.00

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

methods if the matheuristic must be run until an optimal solution is found. With respect to the first usage, the computational results in Tables 3–5 show the proposed Matheuristic$_1$ and Matheuristic$_2$ to be more effective than all compared heuristics except for the IIGA, with not only approximate solutions but also optimal solutions being found in all test instances in less computational time than that required by the compared heuristics. With respect to the second usage, the proposed MIP model is solved herein using Gurobi (Version 6.0), a state-of-the-art mathematical programming solver. The CPU times required to yield the optimal solutions in 16 very small instances with $m=3$ and $n=\{8, \ldots, 23\}$ using Matheuristic$_1$, Matheuristic$_2$ and the MIP model are measured. Fig. 3 presents the computational results, which reveal that the CPU time required to obtain the optimal solution using the MIP model increases rapidly with the size of the test instance; in instances with $n=23$, the CPU time required to obtain the optimal solution using the MIP model exceeds 24 h. In contrast, the CPU times required by Matheuristic$_1$ and Matheuristic$_2$ are only 0.02 and 0.02 s, respectively.

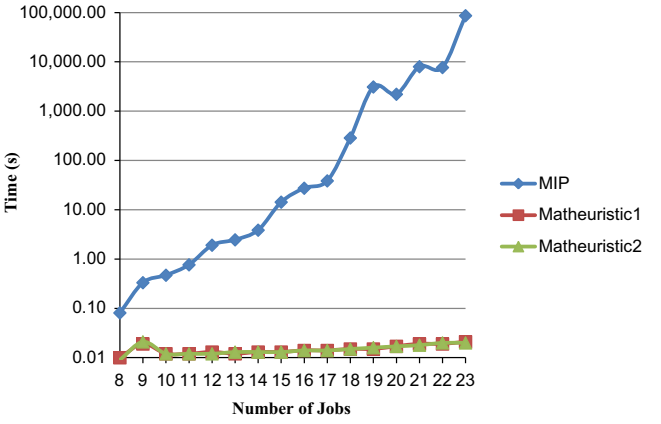

Fig. 3. The CPU times need to get the optimal solution by using the Matheuristic$_1$, Matheuristic$_2$ and MIP model.

Tables 6 and 7 list the optimal solutions obtained using the proposed matheuristics when applied to small and large test instances in the third benchmark problem set, respectively. Notably, all 480 benchmark instances can be optimally solved using the proposed matheuristics, and the maximum required computational time is 1441 s. These results confirm that the proposed matheuristics are highly effective and efficient in solving the $F_m|nwt|C_{\max}$ problem.

Table 8 shows the computational results obtained using the UB, Matheuristic$_1$ and Matheuristic$_2$ for solving 30 newly generated very large test instances in the fourth benchmark problem set. In Table 8, Columns 1–4 show the name, number of jobs, number of machines and optimal solution in each instance, respectively; Column 5 provides the average RER obtained using the UB; Columns 6 and 7 present the solutions obtained using MNEH$_1$ with the computational times required; Columns 8 and 9 present the

Table 6
Optimal solutions for 240 small instances of Vallada et al. benchmark problem set.

Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.
10_5_1	760	20_5_1	1414	30_5_1	2072	40_5_1	2842	50_5_1	3577	60_5_1	3906
10_5_2	759	20_5_2	1481	30_5_2	1960	40_5_2	2875	50_5_2	3303	60_5_2	3779
10_5_3	823	20_5_3	1588	30_5_3	2029	40_5_3	2592	50_5_3	3289	60_5_3	3858
10_5_4	776	20_5_4	1355	30_5_4	2111	40_5_4	2637	50_5_4	3391	60_5_4	3899
10_5_5	798	20_5_5	1520	30_5_5	1967	40_5_5	2738	50_5_5	3405	60_5_5	3941
10_5_6	849	20_5_6	1333	30_5_6	2127	40_5_6	2598	50_5_6	3302	60_5_6	3758
10_5_7	843	20_5_7	1388	30_5_7	2036	40_5_7	2649	50_5_7	3088	60_5_7	4001
10_5_8	768	20_5_8	1340	30_5_8	2051	40_5_8	2829	50_5_8	3238	60_5_8	4138
10_5_9	841	20_5_9	1499	30_5_9	2046	40_5_9	2753	50_5_9	3117	60_5_9	3784
10_5_10	719	20_5_10	1546	30_5_10	1546	40_5_10	2797	50_5_10	3372	60_5_10	3980
10_10_1	1253	20_10_1	2017	30_10_1	2653	40_10_1	3550	50_10_1	4121	60_10_1	5067
10_10_2	1278	20_10_2	1998	30_10_2	2861	40_10_2	3416	50_10_2	4261	60_10_2	5185
10_10_3	1171	20_10_3	2036	30_10_3	2796	40_10_3	3408	50_10_3	4227	60_10_3	4953
10_10_4	1181	20_10_4	1932	30_10_4	2762	40_10_4	3622	50_10_4	4320	60_10_4	5006
10_10_5	1294	20_10_5	2032	30_10_5	2773	40_10_5	3488	50_10_5	4356	60_10_5	5140
10_10_6	1198	20_10_6	2059	30_10_6	2808	40_10_6	3565	50_10_6	4205	60_10_6	5146
10_10_7	1256	20_10_7	2051	30_10_7	2683	40_10_7	3496	50_10_7	4096	60_10_7	5130
10_10_8	1220	20_10_8	2018	30_10_8	2532	40_10_8	3427	50_10_8	4322	60_10_8	4976
10_10_9	1243	20_10_9	1979	30_10_9	2693	40_10_9	3501	50_10_9	4289	60_10_9	5001
10_10_10	1317	20_10_10	1963	30_10_10	2647	40_10_10	3447	50_10_10	4268	60_10_10	5040
10_15_1	1516	20_15_1	2663	30_15_1	3347	40_15_1	4370	50_15_1	4972	60_15_1	5972
10_15_2	1596	20_15_2	2523	30_15_2	3243	40_15_2	4214	50_15_2	5079	60_15_2	5965
10_15_3	1611	20_15_3	2392	30_15_3	3301	40_15_3	4251	50_15_3	5136	60_15_3	6070
10_15_4	1649	20_15_4	2392	30_15_4	3406	40_15_4	4249	50_15_4	5248	60_15_4	5974
10_15_5	1602	20_15_5	2502	30_15_5	3463	40_15_5	4353	50_15_5	5092	60_15_5	6004
10_15_6	1529	20_15_6	2634	30_15_6	3478	40_15_6	4120	50_15_6	5194	60_15_6	6149
10_15_7	1702	20_15_7	2580	30_15_7	3416	40_15_7	4299	50_15_7	5297	60_15_7	6059
10_15_8	1720	20_15_8	2521	30_15_8	3444	40_15_8	4279	50_15_8	5174	60_15_8	5974
10_15_9	1683	20_15_9	2511	30_15_9	3314	40_15_9	4116	50_15_9	5096	60_15_9	5760
10_15_10	1687	20_15_10	2519	30_15_10	3390	40_15_10	4301	50_15_10	5173	60_15_10	6092
10_20_1	1913	20_20_1	3082	30_20_1	3894	40_20_1	4935	50_20_1	5854	60_20_1	6925
10_20_2	1973	20_20_2	2872	30_20_2	4017	40_20_2	4854	50_20_2	5825	60_20_2	6928
10_20_3	1989	20_20_3	2935	30_20_3	4022	40_20_3	5103	50_20_3	5952	60_20_3	7151
10_20_4	1971	20_20_4	2828	30_20_4	3786	40_20_4	4837	50_20_4	5960	60_20_4	7077
10_20_5	1979	20_20_5	3078	30_20_5	3781	40_20_5	4712	50_20_5	5893	60_20_5	6699
10_20_6	2152	20_20_6	3172	30_20_6	3971	40_20_6	4936	50_20_6	6042	60_20_6	6781
10_20_7	1893	20_20_7	2999	30_20_7	3999	40_20_7	5092	50_20_7	5984	60_20_7	6909
10_20_8	1933	20_20_8	2837	30_20_8	4016	40_20_8	4999	50_20_8	5906	60_20_8	6871
10_20_9	1941	20_20_9	3094	30_20_9	4019	40_20_9	5041	50_20_9	5977	60_20_9	6833
10_20_10	1876	20_20_10	2884	30_20_10	4113	40_20_10	4726	50_20_10	5926	60_20_10	6724

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

Table 7
Optimal solutions for 240 large instances of Vallada et al. benchmark problem set.

Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.	Instance	Opt.
100_20_1	10441	200_40_1	26652	300_60_1	45767	500_20_1	46305	600_40_1	72374	700_60_1	99706
100_20_2	10617	200_40_2	26434	300_60_2	45455	500_20_2	46646	600_40_2	72497	700_60_2	99288
100_20_3	10693	200_40_3	26320	300_60_3	45622	500_20_3	46489	600_40_3	72353	700_60_3	98604
100_20_4	10622	200_40_4	26576	300_60_4	46023	500_20_4	46187	600_40_4	72648	700_60_4	99206
100_20_5	10762	200_40_5	27038	300_60_5	45763	500_20_5	46517	600_40_5	72471	700_60_5	99327
100_20_6	10544	200_40_6	26586	300_60_6	45936	500_20_6	46171	600_40_6	72535	700_60_6	99394
100_20_7	10875	200_40_7	26555	300_60_7	46563	500_20_7	46503	600_40_7	72533	700_60_7	98785
100_20_8	10640	200_40_8	26844	300_60_8	45932	500_20_8	46377	600_40_8	72426	700_60_8	99317
100_20_9	10549	200_40_9	26487	300_60_9	46112	500_20_9	46323	600_40_9	73289	700_60_9	99617
100_20_10	10495	200_40_10	26723	300_60_10	46245	500_20_10	45754	600_40_10	72324	700_60_10	100481
100_40_1	14968	200_60_1	32175	400_20_1	37222	500_40_1	60765	600_60_1	86234	800_20_1	72360
100_40_2	14761	200_60_2	32140	400_20_2	37693	500_40_2	61655	600_60_2	86026	800_20_2	72008
100_40_3	14599	200_60_3	32091	400_20_3	37482	500_40_3	61557	600_60_3	86187	800_20_3	72097
100_40_4	14651	200_60_4	31886	400_20_4	37329	500_40_4	61180	600_60_4	86477	800_20_4	71910
100_40_5	14737	200_60_5	32242	400_20_5	37520	500_40_5	61746	600_60_5	86109	800_20_5	72427
100_40_6	14470	200_60_6	31902	400_20_6	37433	500_40_6	61060	600_60_6	86122	800_20_6	72344
100_40_7	14894	200_60_7	31793	400_20_7	37748	500_40_7	60982	600_60_7	85911	800_20_7	71870
100_40_8	14807	200_60_8	31745	400_20_8	37657	500_40_8	61772	600_60_8	85978	800_20_8	71986
100_40_9	14778	200_60_9	32162	400_20_9	37452	500_40_9	61725	600_60_9	87162	800_20_9	71761
100_40_10	14490	200_60_10	32134	400_20_10	37735	500_40_10	61274	600_60_10	86200	800_20_10	71859
100_60_1	17851	300_20_1	28476	400_40_1	49529	500_60_1	73039	700_20_1	63478	800_40_1	94679
100_60_2	17887	300_20_2	28583	400_40_2	49565	500_60_2	72660	700_20_2	63252	800_40_2	94360
100_60_3	17786	300_20_3	28623	400_40_3	49555	500_60_3	73038	700_20_3	63354	800_40_3	94358
100_60_4	18030	300_20_4	28742	400_40_4	50155	500_60_4	73211	700_20_4	63390	800_40_4	94936
100_60_5	18123	300_20_5	28749	400_40_5	49884	500_60_5	72498	700_20_5	63484	800_40_5	95372
100_60_6	18167	300_20_6	28811	400_40_6	49759	500_60_6	73448	700_20_6	63589	800_40_6	94806
100_60_7	17984	300_20_7	28574	400_40_7	49989	500_60_7	72735	700_20_7	63751	800_40_7	94295
100_60_8	18191	300_20_8	28734	400_40_8	49747	500_60_8	73479	700_20_8	63685	800_40_8	94883
100_60_9	17810	300_20_9	28591	400_40_9	49875	500_60_9	72443	700_20_9	63459	800_40_9	95475
100_60_10	17831	300_20_10	29154	400_40_10	49789	500_60_10	72458	700_20_10	63166	800_40_10	94725
200_20_1	19731	300_40_1	38247	400_60_1	59650	600_20_1	55209	700_40_1	83864	800_60_1	112634
200_20_2	19768	300_40_2	38450	400_60_2	59530	600_20_2	54776	700_40_2	83773	800_60_2	112306
200_20_3	19895	300_40_3	38028	400_60_3	59583	600_20_3	55247	700_40_3	83657	800_60_3	111782
200_20_4	19624	300_40_4	38270	400_60_4	60001	600_20_4	54825	700_40_4	84147	800_60_4	112154
200_20_5	19500	300_40_5	38511	400_60_5	58865	600_20_5	54911	700_40_5	83641	800_60_5	112351
200_20_6	19878	300_40_6	38477	400_60_6	59605	600_20_6	55181	700_40_6	83650	800_60_6	112377
200_20_7	19619	300_40_7	38274	400_60_7	59235	600_20_7	54747	700_40_7	83580	800_60_7	112640
200_20_8	19850	300_40_8	38196	400_60_8	59245	600_20_8	54868	700_40_8	84074	800_60_8	112589
200_20_9	19551	300_40_9	38026	400_60_9	59784	600_20_9	55177	700_40_9	84266	800_60_9	112950
200_20_10	19798	300_40_10	38250	400_60_10	59537	600_20_10	54530	700_40_10	83550	800_60_10	111427

solutions obtained using $MNEH_1$+LKH with the computational times required; Columns 10 and 11 present the numbers of iterations (Iter.) of $Matheuristic_1$ with the computational times required; and Columns 12 to 17 present equivalent information concerning $MNEH_2$, $MNEH_2$+LKH and $Matheuristic_2$. Table 8 reveals that $MNEH_2$ outperforms $MNEH_1$. However, $MNEH_1$+LKH and $MNEH_2$+LKH have similar performance, as do $Matheuristic_1$ and $Matheuristic_2$. LKH significantly improves the quality of the solutions obtained using $MNEH_1$ and $MNEH_2$. However, LKH cannot provide the optimal solution to the test instances in the fourth benchmark problem set. The maximum RER among all test instances is 0.142%, verifying that the UB is very strict. Notably, optimal solutions can be found using either $Matheuristic_1$ or $Matheuristic_2$, even though the number of jobs is 2000, with the longest computation time at approximately 9300 s (155 min).

We analyzed the percentage of the searching nodes that could be eliminated when the UB is applied to Phase III of each of the proposed matheuristics; in doing so, the total number of nodes searched by the BIP model in Phase III is compared with those searched without the UB. The analytical results in Table 9 reveal that a very large percentage of the nodes searched is eliminated by applying the UB obtained in Phase II. With respect to Phase III of $Matheuristic_1$, the total average reduction rates (%) of nodes searched in the first, second, third and fourth benchmark problem sets are 18.66%, 46.74%, 22.58% and 21.31%, respectively. In Phase III of $Matheuristic_2$, the total average reduction rates (%) of nodes searched in the first, second, third and fourth benchmark problem sets are 18.66%, 46.74%, 22.58% and 21.31%, respectively. This improvement is significant, ensuring that the optimal solution of the $F_m|nwt|C_{\max}$ problem can be obtained more efficiently by using the proposed matheuristics than by using the BIP model.

5. Conclusions and recommendations for future studies

The $F_m|nwt|C_{\max}$ problem that is studied herein arises often in many industries. Finding an optimal solution to the $F_m|nwt|C_{\max}$ problem has long been a challenging task, in spite of the fact that the problem is so extensively studied. This study proposes two matheuristics to solve this problem effectively and efficiently. Experiments on four benchmark problem sets reveal that the proposed matheuristics yield optimal solutions for very large test instances in an acceptable computational time. This fact demonstrates the main contribution provided by the proposed matheuristics, especially since the problem is NP-hard in the strong sense. The optimality of the solution and the modest computational requirement make the proposed matheuristics highly valuable for use in practical manufacturing systems, thus bridging the gap between research and practice.

Some important directions for further research are as follows. First, the proposed matheuristics can be applied to minimize the makespan in the single machine scheduling problem with sequence-dependent setup times. Second, the proposed matheuristics can be easily modified to minimize the makespan in

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

Table 8
Optimal solutions for the test instances of the fourth benchmark problem set.

Inst.	n	m	Opt.	UB	$MNEH_1$		$MNEH_1+LKH$		$Matheuristic_1$		$MNEH_2$		$MNEH_2+LKH$		$Matheuristic_2$	
				RER	Sol.	T (s)	Sol.	T (s)	Iter.	T (s)	Sol.	T (s)	Sol.	T (s)	Iter.	T (s)
Ta121	1000	20	88,855	0.001	95,932	0.01	88,856	31.59	12	498.80	88,810	0.016	88,856	31.27	12	498.52
Ta122	1000	20	89,165	0.002	96,240	0.02	89,167	30.79	12	628.74	89,119	0.018	89,167	28.74	12	625.19
Ta123	1000	20	88,986	0.000	96,604	0.02	88,986	31.25	4	156.29	88,961	0.018	88,986	28.36	4	153.32
Ta124	1000	20	88,494	0.001	95,561	0.02	88,495	33.59	26	1092.94	88,458	0.018	88,495	30.94	26	1089.60
Ta125	1000	20	89,210	0.001	96,335	0.02	89,211	32.59	20	975.33	89,123	0.018	89,211	30.32	20	972.56
Ta126	1000	20	88,789	0.001	95,764	0.02	88,790	31.62	10	350.65	88,730	0.018	88,790	29.62	10	348.51
Ta127	1000	20	89,052	0.002	96,205	0.02	89,054	28.44	12	614.42	89,010	0.016	89,054	28.52	12	615.47
Ta128	1000	20	89,120	0.000	96,454	0.02	89,120	31.72	3	125.38	89,099	0.016	89,120	31.25	3	124.87
Ta129	1000	20	89,226	0.001	95,979	0.02	89,227	31.16	11	448.36	89,175	0.016	89,227	30.46	11	447.40
Ta130	1000	20	88,807	0.002	95,673	0.02	88,809	31.07	7	266.10	88,772	0.018	88,809	30.41	7	264.92
Ta131	1500	20	131,191	0.008	141,941	0.04	131,202	87.74	15	1848.84	129,811	0.078	131,202	86.45	15	1841.73
Ta132	1500	20	131,085	0.007	142,227	0.04	131,094	86.48	3	349.30	129,934	0.078	131,094	88.38	3	351.16
Ta133	1500	20	130,561	0.018	141,266	0.04	130,585	91.55	7	797.50	128,821	0.094	130,585	93.29	7	776.34
Ta134	1500	20	130,505	0.016	141,046	0.04	130,526	87.83	7	720.29	128,780	0.078	130,526	89.54	7	722.19
Ta135	1500	20	131,037	0.011	141,618	0.04	131,051	87.05	16	1979.60	129,794	0.093	131,051	86.90	16	1980.63
Ta136	1500	20	131,300	0.015	141,890	0.04	131,320	92.21	10	1121.68	130,141	0.078	131,320	92.71	10	1122.52
Ta137	1500	20	130,483	0.005	141,489	0.04	130,489	83.00	13	1342.32	129,445	0.094	130,489	82.83	13	1341.39
Ta138	1500	20	131,124	0.007	141,989	0.04	131,133	78.29	12	1190.34	130,247	0.094	131,133	80.77	12	1192.78
Ta139	1500	20	130,734	0.009	141,567	0.04	130,746	76.63	4	445.12	129,925	0.094	130,746	80.39	4	449.01
Ta130	1500	20	130,719	0.005	141,630	0.04	130,726	79.36	13	1376.18	129,862	0.078	130,726	79.59	13	1377.51
Ta141	2000	20	172,205	0.131	187,500	0.07	172,431	270.55	28	5577.38	160,641	0.219	172,431	285.82	28	5591.84
Ta142	2000	20	172,652	0.119	187,104	0.07	172,857	281.45	33	9225.61	161,133	0.218	172,857	272.56	33	9202.63
Ta143	2000	20	171,937	0.001	186,780	0.07	171,939	236.08	15	2905.49	171,900	0.219	171,939	230.64	15	2901.51
Ta144	2000	20	170,972	0.003	185,784	0.07	170,977	248.25	11	2644.79	170,934	0.234	170,977	236.31	11	2632.91
Ta145	2000	20	172,295	0.001	186,778	0.07	172,297	243.78	30	7857.97	172,262	0.219	172,297	237.12	30	7856.53
Ta146	2000	20	172,086	0.002	186,947	0.07	172,090	239.27	12	2854.90	172,042	0.234	172,090	242.19	12	2856.63
Ta147	2000	20	171,993	0.002	186,879	0.07	171,996	236.64	26	4943.32	171,953	0.218	171,996	240.45	26	4943.22
Ta148	2000	20	172,375	0.000	186,322	0.08	172,375	232.91	10	2530.51	172,328	0.218	172,375	238.91	10	2535.92
Ta149	2000	20	172,745	0.003	187,364	0.08	172,750	228.91	21	6046.95	172,693	0.218	172,750	244.07	21	6048.95
Ta150	2000	20	171,858	0.142	186,736	0.07	172,102	252.01	9	2152.50	158,884	0.218	172,102	278.82	9	2177.20

Table 9
The total average reduction rates (%) of nodes searched when the UB is imposed to the proposed matheuristics.

Benchmark problem set	Phase III of $Matheuristic_1$	Phase III of $Matheuristic_2$
First	18.66	18.66
Second	46.74	46.74
Third	22.58	22.58
Forth	21.31	21.31

NWFSPs, with the additional consideration of sequence-dependent setup times. We hope to report on the same in the near future. Third, the generalization of the proposed matheuristics to no-wait jobshop scheduling problems with makespan as the objective function, for which few exact methods have been developed, is of practical interest. Finally, the application of the same matheuristic technique to NWFSPs with different objective functions is worth further research.

Acknowledgment

The first author of the work is grateful to the Ministry of Science and Technology, Republic of China (Taiwan) and the Linkou Chang Gung Memorial Hospital, Taiwan for financially supporting this research Grants MOST104-2410-H-182-011 and CARPD3B0012, respectively. The second author of the work is grateful to the Ministry of Science and Technology, Republic of China (Taiwan) for financially supporting this research Grant MOST104-2221-E-027-045.

References

[1] Pan QK, Wang L. Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. Omega 2012;40(2):218–229.
[2] Lin SW, Ying KC. Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. Omega 2013;41(2):383–389.
[3] Shabtay D, Arviv K, Stern H, Edan Y. A combined robot selection and scheduling problem for flow-shops with no-wait restrictions. Omega 2014;43(1):96–107.
[4] Pan QK, Ruiz R. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega 2014;44(1):41–50.
[5] Yenisey MM, Yagmahan B. Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega 2014;45(1):119–135.
[6] Aldowaisan T, Allahverdi A. New heuristics for m-machine no-wait flowshop to minimize total completion time. Omega 2004;32(5):345–352.
[7] Sapkal SU, Laha D. A heuristic for no-wait flow shop scheduling. International Journal of Advanced Manufacturing Technology 2013;68(5–8):1327–1338.
[8] Graham R, Lawler E, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.
[9] Röck H. The three-machine no-wait flow shop is NP-complete. Journal of the ACM 1984;31(2):336–345.
[10] Gilmore PC, Gomory RE. Sequencing a one-state variable machine: a solvable case of the travelling salesman 32problem. Operations Research 1964;12(5):655–679.
[11] Reddi SS, Ramamoorthy CV. On the flow-shop sequencing problem with no wait in process. Operational Research Quarterly 1972;23(3):323–331.
[12] Wismer DA. Solution of the flowshop sequencing problem with no intermediate queues. Operational Research 1972;20(3):689–697.
[13] Hall NG, Sriskandarajah C. A survey of machine scheduling problems with blocking and no-wait in process. Operational Research 1996;44(3):510–525.
[14] Selen WJ, Hott DD. A new formulation and solution of the flowshop scheduling problem with no in process waiting. Applied Mathematical Modelling 1986;10(4):246–248.
[15] Van der Veen JAA, Van Dal R. Solvable cases of the no-wait flow-shop scheduling problem. Journal of the Operational Research Society 1991;42(11):971–980.
[16] Bonney MC, Gundry SW. Solutions to the constrained flowshop sequencing problem. Operational Research Quarterly 1976;27(4):869–883.
[17] King JR, Spachis AS. Heuristics for flowshop scheduling. International Journal of Production Research 1980;18(3):345–357.

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002

[18] Gangadharan R, Rajendran C. Heuristic algorithms for scheduling in the no-wait fowshop. International Journal of Production Economics 1993;32(3):285–290.
[19] Rajendran C. A no-wait flowshop scheduling heuristic to minimize makespan. Journal of the Operational Research Society 1994;45(4):472–478.
[20] Li X, Wang Q, Wu C. Heuristic for no-wait flow shops with makespan minimization. International Journal of Production Research 2008;46(9):2519–2530.
[21] Laha D, Chakraborty UK. A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. International Journal of Advanced Manufacturing Technology 2009;41(1–2):97–109.
[22] Gonzalez B, Torres M, Moreno JA. A hybrid genetic algorithm approach for the "no-wait" flowshop scheduling problem. In: Proceedings of IEEE conference on genetic algorithms in engineering system: innovations and applications. Sheffield, United Kingdom; 1995. p. 59–64.
[23] Aldowaisan T, Allahverdi A. New heuristics for no-wait foowshops to minimize makespan. Computers & Operations Research 2003;30(8):1219–1231.
[24] Schuster CJ, Framinan JM. Approximative procedures for no-wait job shop scheduling. Operations Research Letters 2003;31(4):308–318.
[25] Framinan JM, Schuster CJ. An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach. Computers & Operations Research 2006;33(5):1200–1213.
[26] Grabowski J, Pempera J. Some local search algorithms for no-wait flow-shop problem with makespan criterion. Computers & Operations Research 2005;32(8):2197–2212.
[27] Schuster CJ. No-wait job shop scheduling: Tabu search and complexity of subproblems. Mathematical Methods of Operations Research 2006;63(3):473–491.
[28] Liu B, Wang L, Jin YH. An effective hybrid particle swarm optimization for no-wait flow shop scheduling. International Journal of Advanced Manufacturing Technology 2007;31(9–10):1001–1011.
[29] Pan QK, Tasgetiren MF, Liang YC. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers & Operations Research 2008;35(9):2807–2839.
[30] Pan QK, Liang YC, Tasgetiren MF, Zhao BH. A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. International Journal of Advanced Manufacturing Technology 2008;38(3–4):337–347.
[31] Pan QK, Wang L, Zhao BH. An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. International Journal of Advanced Manufacturing Technology 2008;38(7–8):778–786.
[32] Qian B, Wang L, Hu R, Huang DX, Wang X. A DE-based approach to no-wait flow-shop scheduling. Computers & Industrial Engineering 2009;57(3):787–805.
[33] Tseng LY, Lin YT. A hybrid genetic algorithm for no-wait flowshop scheduling problem. International Journal of Production Economics 2010;128(1):144–152.
[34] Jarboui B, Eddaly M, Siarry P. A hybrid genetic algorithm for solving no-wait flowshop scheduling problems. International Journal of Advanced Manufacturing Technology 2011;54(9–12):1129–1143.
[35] Samarghandi H, ElMekkawy TY. A meta-heuristic approach for solving the no-wait flow-shop problem. International Journal of Production Research 2012;50(24):7313–7326.
[36] Davendra D, Zelinka I, Bialic-Davendra M, Senkerik R, Jasek R. Discrete self-organising migrating algorithm for flow-shop scheduling with no-wait makespan. Mathematical and Computer Modelling 2013;57(1–2):100–110.
[37] Ding JY, Song S, Gupta JND, Zhang R, Chiong R, Wu C. An improved iterated greedy algorithm with a Tabu-basedreconstruction strategy for the no-wait flowshop scheduling problem. Applied Soft Computing 2015;30(5):604–613.
[38] Nawaz M, Enscore Jr E, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 1983;11(1):91–95.
[39] Kalczynski PJ, Kamburowski J. On the NEH heuristic for minimizing the makespan in permutation flow shops. Omega 2007;35(1):53–60.
[40] Helsgaun K. An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research 2000;126(1):106–130.
[41] Zhu X, Li X, Wang X. Objective increment based iterative greedy heuristic for no-wait flowshops with total flowtime minimization. Chinese Journal of Computers 2009;31(1):132–141.
[42] Rosenkrantz DJ, Stearns RE, Lewis PM. An analysis of several heuristics for the traveling salesman problem. SIAM Journal of Computing 1977;6(3):563–581.
[43] Reeves CR. Genetic algorithms and neighbourhood search. In: Terence C. Fogarty (Ed.), Evolutionary computing. Springer-Verlag Berlin, Heidelberg, 1994. p. 115–130.
[44] Taillard E. Benchmarks for basic scheduling problems. European Journal Operational Research 1993;64(2):278–285.
[45] Vallada E, Ruiz R, Framinan JM. New hard benchmark for flowshop scheduling problems minimising makespan. European Journal Operational Research 2015;240(3):666–677.

Please cite this article as: Lin S-W, Ying K-C. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. Omega (2015), http://dx.doi.org/10.1016/j.omega.2015.12.002