Efficient Metaheuristics for Multi-stage No-Wait Flexible Flowshop Scheduling Problem

Mageed A. Ghaleb
Department of Industrial Engineering
King Saud University
Riyadh, Riyadh 11421, KSA
mghaleb87@gmail.com

Ibrahim M. Alharkan
Department of Industrial Engineering
King Saud University
Riyadh, Riyadh 11421, KSA
imalhark@ksu.edu.sa

Abstract

The issue of sequencing and scheduling has received considerable critical attention in many fields since the early 1950s. In this research, we studied an important production scheduling system, which generally called flexible (or hybrid) flowshop scheduling problem (FFSP or HFSP). We studied this production system with an important restriction called the no-wait. The objective was to optimize the maximum completion time (Makespan) in the no-wait flexible flowshop scheduling problem (NWFFSP) with multi-stage. The considered problem (NWFFSP) is found to be least explored to the field of research. Moreover, the problem is found not to be only NP-hard, but also has a well-earned reputation of being one of the most computationally difficult combinatorial optimization problems considered to date. Therefore, the application of efficient metaheuristics procedures to the problem, in order to find an approximate solution close to the optimum with considerably less computational time, is the primary focus of this research. We consider the use of simulated annealing (SA), tabu search (TS), and Geometric particle swarm optimization (GPSO) to solve the NWFFSP. An extensive computational study is carried out, and the results showed the superiority of the TS compared to other algorithms.

Keywords
Flexible flow shop; No-wait; Metaheuristics; Makespan.

1. Introduction
Production sequencing and scheduling is one of the most important activities in production planning and control. (Morton and Pentico 1993) discussed how important the sequencing and scheduling role is, stating that "it pervades all economic activity". The FFSP, one of the most well-known problems in the area of production scheduling, naturally originates from manufacturing environments and widely exists in a variety of real-world industries. Ever since the FFSP was identified in 1970's (Arthanari and Ramamurthy 1971), it has attracted considerable attention during the past decades (Wang 2005, Ruiz and Vázquez Rodríguez 2010). The flexible flow shop is also known in the literature as flexible flow line (FFL), hybrid flow shop (HF), or multi-processor flow shop (Quadt and Kuhn 2007, Jungwattanakit *et al.* 2008, Ribas *et al.* 2010, Ruiz and Vázquez Rodríguez 2010, Pinedo 2012). The FFSP is a generalization of two particular scheduling problems, namely the classic flow shop scheduling problem (FSP) and the parallel machine scheduling (PMS) problem (Ribas *et al.* 2010, Ruiz and Vázquez Rodríguez 2010, Pinedo 2012). However, it was pointed out in the surveys by (Sriskandarajah and Ladet 1986) and (Hall and Sriskandarajah 1996) that the no-wait FFSP has received less attention from both theoretical and practical aspects than classic FFSSP. The no-wait requirement, which is our interest, is a phenomenon that may occur in the flow shops and flexible flow shops machine environments. Jobs are not allowed to wait between two successive machines, jobs must be processed from the start to finish, remove of any interruption on machines or between them. In some industries, due to the temperature or other characteristics of the material it is required that each operation follow the previous one immediately. Such situations appear in the chemical processing (Rajendran 1994), food processing (Hall and Sriskandarajah 1996),

concrete production (Grabowski and Pempera 2000), pharmaceutical processing (Raaymakers and Hoogeveen 2000) and production of steel, plastics, and aluminum products (Aldowaisan and Allahverdi 2004). An example of such environment is a steel rolling mill in which a slab of steel is not allowed to wait, as it would cool off during a wait (Pinedo 2012). If any waiting happened the slab of steel will be either a rework or a scrap product. Thus, such requirement took a highly importance in the flow shop and flexible flow shop machine environments.

According to the previous research, it has been found that most of the attempts to solve the NWFFSP have used heuristic algorithms or specific heuristics and metaheuristics. For the best of our knowledge no research work presented any exact methods to solve the no-wait FFSP even in its simple picture, the two-stage no-wait FFSP. Several heuristic algorithms have been applied to the problem of no-wait in FFSPs. Some of them applied only to the problem with two stages. For example, (Liu *et al.* 2003) introduced a two-stage no-wait FFSP with single machine in the first stage. A least deviation algorithm, which is a greedy heuristic, was designed to solve the problem. (Xie *et al.* 2004) developed a new heuristic algorithm, called minimum deviation algorithm (MDA), to solve the two-stage no-wait FFSP with minimizing the makespan. (Wang *et al.* 2005) developed a heuristic called MLPT to minimize the makespan for the two-stage no-wait FFSP. The considered problem also assumed no idle time between two consecutive processed jobs on machines of the second stage. Moreover, the developed heuristic was with a tight error bound. (Chang *et al.* 2004) proposed two heuristic algorithms to minimize the makespan for a two-stage no-wait FFSP. The considered problem was with a single machine in one of the two stages and with setup and removal times. (Carpov *et al.* 2012) proposed an adaptive randomized list scheduling heuristic to solve the two-stage no-wait FFSP with precedence constraints. The classical two-stage FFSP was also considered. The three stages problem has been considered by (Vila and Pascual 2009). The author studied the three-stage no-wait FFSP with batch dependent setup costs, which comes from an ice cream manufacturing company. The problem has been formulated as a mixed integer programming, and two heuristics were developed to solve the problem. For the best of our knowledge no research work presented any heuristic algorithm to solve the problem with more than three stages.

Metaheuristics have also been applied to the problem of no-wait in FFSPs. Some of the designed metaheuristics applied only to the two-stage problem. For example, (Wang and Liu 2013) proposed a GA to minimize the makespan for a two-stage no-wait FFSP. (Shafaei *et al.* 2011b) proposed an intelligent system framework, with a number of heuristic algorithms applied in it, to minimize makespan for a no-wait two-stage FFSP. The proposed intelligent system was an adaptive neuro fuzzy inference system (ANFIS). (Rabiee *et al.* 2011) proposed two metaheuristic algorithms (SA and GA) to minimize the makespan for a two-stage no-wait FFSP. The results were compared to the MDA algorithm with the same data sets. (Jolai *et al.* 2013) developed three optimization methods based on simulated annealing to solve a bi-objective problem of two-stage no-wait FFSP. The considered objectives were the minimization of makespan and maximum tardiness. The developed optimization methods were classical weighted simulated annealing (CWSA), normalized weighted simulated annealing (NWSA), and fuzzy simulated annealing (FSA). (Shafaei *et al.* 2011a) developed six metaheuristic algorithms to minimize mean flow time for a two-stage no-wait FFSP. The developed metaheuristics were based on imperialist competitive algorithm (ICA), ant colony optimization (ACO), and particle swarm optimization (PSO). (Ghaleb *et al.* 2015) solved the two-stage NWFFSP, the objective was the makespan. Two metaheuristics were proposed, which are tabu Search (TS) and particle swarm optimization (PSO). The results of the study proposed the effectiveness and the efficiency of the TS. Metaheuristics have also been applied to the multi-stage problem. For instance, (Jolai *et al.* 2009) introduced a mixed integer linear programming model for a multi-stage no-wait FFSP with due window and job rejection. The objective was the maximization of the total profit gained from scheduled jobs. A genetic algorithm was proposed to solve the problem. (Jolai *et al.* 2012) proposed a three metaheuristic algorithms to minimize the makespan for a no-wait FFSP with sequence dependent setup times. The proposed metaheuristic algorithms were population based simulated annealing (PBSA), adapted imperialist competitive algorithm (AICA) and hybridization of AICA and PBSA. (Shilong *et al.* 2009) developed a mathematics model for the problem of single hoist-scheduling in no-wait FFSP with constant process times. A hybrid GA and SA was proposed to minimize the makespan for the studied problem.

The remaining content of this paper are organized as follows. In section 2, the multi-stage NWFFSP is introduced. In section 3, the SA, TS, GA and PSO are proposed after presenting the starting solution chosen and the constructive heuristic. In section 4, experimental study is presented with the chosen instances, parameter tuning, comparison process and the results. Finally, in section 5, we end the paper with some conclusions and future work propositions.

2. The multi-stage NWFFSP

In this study, we restrict our attention to deterministic flexile flow shop scheduling, where all the data that define a problem instance are known in advance. The no-wait requirement implies that the starting time of a job at the first stage has to be delayed to ensure that the job can go through the flow shop without having to wait for any stage. In another way, the completion time of a job on a given stage must be equal to the starting time of that job on the next (successive) stage. The no-wait flexible flow shop scheduling problem (NWFFSP) can be described as shown in Figure 1 as follows: Each of n jobs is to be sequentially processed on stages $1,\ldots,S$. Each stage is consist of one machine or a set of identical parallel machines $1,\ldots,m$. The processing time $p_{j,s}$ of job j on stage s is given. At any moment of time, each machine can processed at most one job and similarly, each operation of a job can only be processed on one machine. The no-wait requirement implies that the starting time of a job at the first machine has to be delayed to ensure that the job can go through the flow shop without having to wait for any machine. To satisfy the no-wait restrictions, the completion time of a job on a given stage must be equal to the starting time of the job on the next stage. The objective is to find a sequence that the maximum completion time, i.e., makespan (C_{max}), is minimized. To compute the makespan a developed constructive heuristic is used. Also the developed constructive heuristic has been used to ensure the feasibility of the generated solutions (Schedules). The problem can be described using Graham notation ($\alpha|\beta|\gamma$) as follows: $FF(S)|nwt|C_{max}$, $where\ "S"\ is\ the\ number\ of\ stages.$

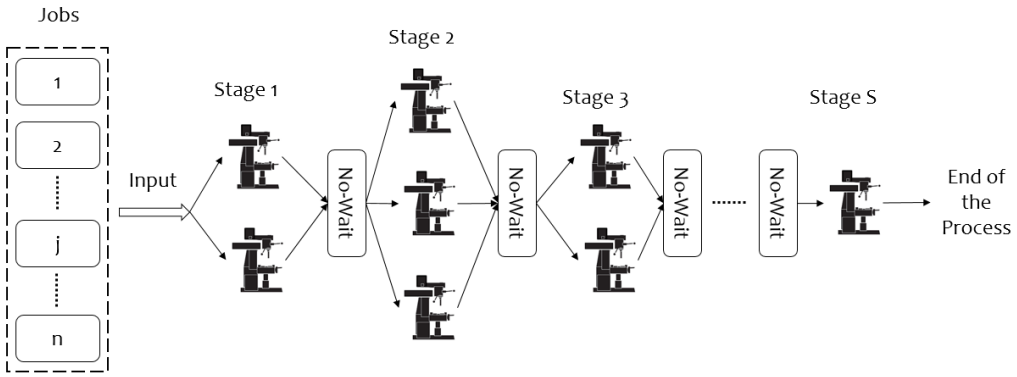

Figure 1: Flexible Flow Shop Scheduling environment with No-Wait restriction.

Production scheduling problems are generally considered NP-hard as combinatorial optimization problems (Pinedo 2012). Although, flexible flow shop scheduling problems (FFSP) are, in most cases, NP-hard. Back to 1979, (Michael and David 1979) showed that the FFSP with makespan objective is NP-complete, and along the years the NP-nature of most FFSPs has been shown (Ribas *et al.* 2010, Ruiz and Vázquez Rodríguez 2010). For instance, FFSP restricted to two processing stages, even in the case when one stage contains two machines and the other one a single machine, is NP-hard, after the results of (Gupta 1988). Similarly, the FFSP with preemptions is strongly NP-hard problems even with $m = 2$, according to (Hoogeveen *et al.* 1996). Moreover, the special case where there is a single machine per stage, known as the flow shop, and the case where there is a single stage with several machines, known as the parallel machines environment, are also NP-hard, (Michael and David 1979, Pinedo 2012). However, with some special properties and precedence relationships, the problem might be polynomially solvable (Djellab and Djellab 2002). Similarly, Rock (Röck 1984) proved that the no-wait FSP, with makespan and more than two machines, is strongly NP-hard. FFSP with no-wait, even with only two stages, is also NP-hard in the strong sense (Hall and Sriskandarajah 1996, Rabiee *et al.* 2011).

3. The Proposed Solution Methods

The proposed solution methods are based on SA, TS, GA and PSO approaches. They are categorized based on the algorithms applied for job sequencing and machine assignment. With reference to the algorithms presented in this paper, the job sequences are generated using the stochastic search process of SA, TS, GA and PSO. The jobs are then allocated to the machines using a constructive heuristic algorithm, which allocates the job with the first priority to the machine with the earliest available time. In the rest of this section, the starting solution and the constructive heuristic are explained. Then the structures of the proposed metaheuristics are presented. This is followed by a summary of the metaheuristics parameters that will be used in the tuning process in Section 4.

3.1 The Developed Constructive Heuristic

To construct a solution for such type of environment for the given constraint, which is the no-wait (nwt) constraint, and to the fulfillment of the no-wait restrictions, a constructive heuristic was developed to make sure that the completion time of a job on a given stage must be equal to the starting time of that job on the next (successive) stage. The most important points that should be considered in constructing a solution are:

- The jobs are assigned according to the availability of the machines in each stage.
- Since the following machine $m_i(s)$ busy, the job j will not be processed on the machine that precede the machine $m_i(s)$ until that; after processing that job on the precedence machine, the machine $m_i(s)$ should be available for processing that job.
- The time that the job j will take in the machine m_i in stage s for satisfying the no-wait constraint will be added to its processing time in that stage.

Notations used in the constructive heuristic:

- $S = no.Stages.$
- $M_s = no.machines\ in\ stage\ s, \quad where\ s = 1,2,\dots,S.$
- $m_i(s) = machine\ m_i\ at\ stage\ s, \quad where\ i = 1,2,\dots,M_s.$
- $AT_{sm_i} = Availability\ Time\ for\ machine\ m_i\ in\ stage\ s.$
- $AT_s = Availability\ Time\ of\ stage\ s.$
- $P_{sj} = Processing\ time\ of\ job\ j\ at\ stage\ s.$
- $CT_{sj} = Completion\ Time\ of\ job\ j\ at\ stage\ s.$
- $EST_{sj} = Expected\ Starting\ Time\ of\ job\ j\ at\ stage\ s = Actual\ CT_{(s-1)j}$.

Constructive heuristic steps: The steps of the constructive heuristic for solving such type of environment for the given constraint (the no-wait constraint) are as follows:

Step 1: Initialization

- $Select\ a\ starting\ sequence,\ set\ \ J^C = \{1,2,\dots,n\},\ \ and\ \ \ J = \{\emptyset\}.$
- $Set\ \ machines\ availability\ time\ (AT_s)\ to\ zero.$
- $Always\ choose\ the\ minimum\ available\ time\ and\ its\ machine\ at\ each\ stage\ for\ each\ job.$
- $For\ stage\ s{:}\ AT_s = min(AT_{sm_i}), and\ M_s = m_i.$
- $Starting\ time\ for\ each\ job\ (ST_{sj}) will\ be, AT_s\ at\ each\ stage.$

Step 2:

- $For\ each\ stage\ s, find\ AT_s\ and\ m_i(s).$
- $Start\ with\ the\ first\ job\ in\ the\ starting\ sequence.$
- $Set\ \ J^C = J^C - \{j\}\ \ and\ \ add\ \ the\ \ job\ \ j\ \ to\ \ the\ set\ \ \ J.$

Step 3:

- $For\ each\ job\ j, set\ AT_s\ Starting\ time\ (ST_{sj})\ and\ m_i(s)\ as\ the\ machine\ that\ processing$
- $the\ job\ j.$
- $For\ the\ first\ stage{:}\ The\ completion\ time\ (CT_{1j}) = ST_{1j} + P_{1j}.$
- $For\ the\ rest\ stages$:
  - $If\ \ \ CT_{(s-1)j} \geq ST_{sj} \Longrightarrow ST_{sj} = CT_{(s-1)j} \quad and \quad CT_{sj} = CT_{(s-1)j} + P_{sj}$
  - $else \Longrightarrow CT_{sj} = ST_{sj} + P_{sj} \quad and \quad CT_{(s-1)j} = ST_{sj}$
  - $And\ for\ the\ machine\ that\ process\ this\ job\ j\ in\ stage\ (s-1)$:
    - $The\ CT_{(s-1)j}\ for\ all\ the\ jobs\ that\ following\ this\ job\ and\ has\ been\ processed\ by\ the\ same\ machine\ will\ be\ updated\ by\ this\ value\ (ST_{sj} - CT_{(s-1)j}).$
- $Repeat\ this\ step\ until\ \ J^C = \emptyset\ \ then\ go\ to\ step\ 4.$

Step 4:

- $If\ \ s = S\ \ then\ stop\ \ and\ \ compute\ \ the\ \ value\ \ of\ \ Makespan.$
- $Otherwise,\ \ go\ to\ Step2.$

The developed constructive heuristic will help in constructing the generated solutions (schedules) and computing the value of the makespan. This constructive heuristic is also used for the starting solution and for the randomly generated solutions during the solution process in the designed metaheuristics. Moreover, ensuring the feasibility of the generated solutions (Schedules). Since the proposed algorithms are improvement type algorithms, a starting solution is necessary to be the start out complete schedule for the used metaheuristics. Then an iterative improvement approach will be applied to the starting solution until a high quality feasible solution is obtained. The starting solution in the

single-based metaheuristics (SA and TS) was generated using the longest processing time (LPT) dispatching rule, while in the population-based metaheuristic (GPSO) a set of swarms was generated randomly.

3.2 Simulated Annealing (SA)

Simulated annealing (SA) is a search process that has its origin in the fields of material science and physics (Pinedo 2012). SA is an approach for solving combinatorial optimization problems (Moraga *et al.* 2006). This approach a based on the early research of Metropolis et al. (Metropolis *et al.* 1953) in 1953. Basically, Metropolis's algorithm simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state, this approach point to a straight analogy in which liquids freeze and crystallize or that metal cool and anneal (Moraga *et al.* 2006). The SA technique was first proposed in 1982, and published in 1983 by (Kirkpatrick and Vecchi 1983), as a new iterative method that can avoid the local optima (Petrowski and Taillard 2006). A similar work, developed independently at the same time by (Černý 1985), was published in 1985. (Kirkpatrick and Vecchi 1983) and (Černý 1985) proposed this approach (SA) to deal with highly nonlinear problems and after that, they proposed it for solving combinatorial optimization problems. Since then, SA had a major impact on the field of heuristic search for its simplicity and efficiency in solving combinatorial optimization problems (Moraga *et al.* 2006).

SA explores the set of all possible solutions, minimizing the chance of being cohesive to local optima by accepting moves that may worsening the value of the objective function to escape from that local optima and move toward a new area in the solution space. A better move is always accepted (Moraga *et al.* 2006). Algorithm 1 illustrates the template of the SA algorithm as in Table 1.

Table 1: Algorithm 1, the template of the SA algorithm (Talbi 2009).

Algorithm 1: SA algorithm template.

Input: Cooling schedule.
$s = s_0$; %Generation of the initial solution
$T = T_{max}$; %Starting temperature
Repeat
  Repeat %At a fixed temperature
    Generate a random neighbor s';
    $\Delta E = f(s') - f(s)$;
    If $\Delta E \leq 0$ **Then** $s = s'$ %Accepting the neighbor solution
    Else Accept s' with a probability $e^{-\frac{\Delta E}{T}}$;
  Until Equilibrium condition
  %e.g. a given number of iterations excused at each temperature T
  $T = g(T)$; %Temperature update
Until Stopping criteria satisfied %e.g. $T < T_{min}$
Output: Best solution found.

SA still simple and easy to implement if compared with other local search techniques (Talbi 2009). (Reinelt 1994) stated that SA can give very good-quality solutions, but that it may take a considerable amount of time. In addition, an important issue that should not be underestimated is the proper choice of the cooling schedule, which is highly problem dependent and subject to numerous experiments (Moraga *et al.* 2006). A good survey on SA can be found in the following references (Laarhoven 1987, Aarts and Korst 1988, Suman and Kumar 2006).

3.3 Tabu Search (TS)

Tabu search algorithm was proposed by (Glover 1989). The particular feature of tabu search is the use of memory to store the information related to the search process (Talbi 2009). In the 1990s, the tabu search algorithm became very popular in solving optimization problems. Nowadays, it is one of the most widespread metaheuristics (Glover and Kochenberger 2003, Moraga *et al.* 2006, Talbi 2009). In TS, and as in SA, the current solution will be replaced if a better neighbor is found. The search will carried on by accepting a candidate worse than the current solution, when local optima is reached. The new current solution will be the best solution found in the neighborhood even if it is not improving the current solution. This search policy may be viewed as dynamic transformation of the neighborhood, which may generate cycles (previous visited solutions could be selected again) (Glover and Kochenberger 2003, Talbi 2009, Pinedo 2012). To avoid cycles, TS uses a memory of the solutions or moves recently accepted, which is called the tabu list. TS discards the neighbors that have been previously visited. The tabu list usually contains a constant number of tabu moves. Usually, the attributes of the moves are stored in the tabu list (Moraga *et al.* 2006, Talbi 2009).

TS in many ways similar to SA in that escaping local optima by accepting worse moves, and we can find that the acceptance-rejection criterion is based on a probabilistic process in SA while it is based on a deterministic process in TS (Pinedo 2012). The risk of rejecting solutions that have not yet been generated may be occur regarding the restrictions in the tabu list. Thus, for some conditions, called aspiration criteria, tabu solutions may be accepted (Glover and Kochenberger 2003, Talbi 2009). The most common used aspiration criteria is that if the forbidden solution objective value better than the best objective value found so far, this forbidden solution will be accepted and its move will be removed from the tabu list (Talbi 2009, Pinedo 2012). Algorithm 2 illustrates the template of the TS algorithm as in Table 2.

Table 2: Algorithm 2, the template of the TS algorithm (Talbi 2009).

Algorithm 2: TS algorithm template.
$s = s_0$; %Initial solution Initialize the tabu list, medium-term and long-term memories; **Repeat** Generate a set 'A' of solutions; Find best neighbor s' of 'A'; %non tabu or aspiration criterion holds $s = s'$; Update tabu list, aspiration conditions, medium and long term memories; **If** $intensification\_criterion$ holds **Then** intensification; **If** $diversification\_criterion$ holds **Then** diversification; **Until** Stopping criteria satisfied **Output:** Best solution found.

TS is the most widely recognized of the modern heuristics and has been extensively applied to a number of combinatorial problems (Moraga *et al.* 2006). According to (Laporte *et al.* 2000), TS-based methods have been the most successful metaheuristics for a wide variety of problems. In general, (Reinelt 1994) stated that the basic TS difficulties include the tabu list design, the mechanism of list management, and the non-prohibited move selection. This is not a straightforward task for some optimization problems. Moreover, TS may be very sensitive to some parameters such as the size of the tabu list (Talbi 2009). In their book, (Glover and Laguna 1999) present a very good discussion of TS applications on scheduling and many other problems.

3.4 Particle Swarm optimization for the NWFFSP: The Geometric PSO (GPSO)

Unlike discrete optimization algorithms, PSO algorithms are applied traditionally to continuous optimization problems. Some adaptations must be made for discrete optimization problems. They differ from continuous models in "mapping between particle positions and discrete solutions" and "velocity models" (Talbi 2009). According to (Ghaleb *et al.* 2015) GPSO was the superior PSO approach for solving the two-stage NWFFSP. Geometric PSO (GPSO) is an innovative discretization of PSO, which is based on the geometric framework of recombination operators (Moraglio *et al.* 2008). The location of each particle i is represented by a vector of integer numbers (permutation) $x_i = (x_{i1}, x_{i2}, \dots, x_{iN})$ where each element x_{ij} (with j in $(1, N)$) takes an integer value within $(1, N)$. The key issue of the GPSO is the concept of particle movement (Talbi 2009).

In this approach, instead of using the concept of velocity, a three-parent mask-based crossover ($3PMBCX$) operator is applied to each particle to "move" it. The position of the particle sequence at iteration t is updated according to the following equation (Moraglio *et al.* 2008):

$$x_i(t) = CX\left((x_i(t-1), w_1), (p_g(t-1), w_2), (p_i(t-1), w_3)\right) \qquad (1)$$

Permutation $\pi_i(t)$ corresponds to the particle $x_i(t)$. The factor t represents the iterative generation number, CX is crossover denotation that denotes three particles with crossover operator ($3PMBCX$). For a given particle i, three parents take part in the $3PMBCX$ operator: the current position x_i, the social best position p_g, and the historical best position found p_i of this particle. The weight values $w_1, w_2, and\ w_3$ indicate for each element in the crossover mask the probability of having values from the parents x_i, p_g, or p_i, respectively (Moraglio *et al.* 2008, Talbi 2009). Algorithm 3 illustrate the GPSO approach as in Table 3.

3.5 SA, TS and GPSO Design Parameters

The definition of the neighborhood and the generation of the initial solution are among the common design issues in single-based metaheuristics. However, and specific to SA, there are design issues considered as the main design issues,

which are he acceptance probability function and the cooling schedule (Initial temperature, Equilibrium state and Cooling function).

Table 3: Algorithm 3, the template of the GPSO algorithm.

Algorithm 3: GPSO algorithm template.

Random initialization of the whole swarm (as permutations);
Repeat
  Evaluate $f(x_i)$;
  For all particles i;
    Update and move to the new position using the 3PMBCX:
    $x_i(t) = CX\left((x_i(t-1), w_1), (p_g(t-1), w_2), (p_i(t-1), w_3)\right)$;
    Mutate x_i;
    If $f(x_i) < pbest_i$, ***Then*** $p_i = x_i$;
    If $f(x_i) < gbest$, ***Then*** $g_i = x_i$;
    Update (x_i);
  End For
Until Stopping criteria
Output: Best solution found.

For TS, we have to define the following concepts that compose the search memory of TS: tabu list size, inner step size. In GPSO, in addition to the swarm size the crossover operator and mutation operators were defined. Table 4 summarizes the main parameters of TS and PSO algorithms.

Table 4: Summary of the main parameters and levels for the designed metaheuristics.

Algorithm	Parameters	Levels
SA	Initial Temperature (t_0)	100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000
	Final Temperature (t_{min})	0.01, 0.001, 0.0001, 0.00001, and 0.000001
	Neighborhood Structure	Swap and Insertion
	Cooling Function	Linear, Geometric, Logarithmic, and Very slow decrease
	Equilibrium Condition	Static*, Adaptive1**, and Adaptive2***
	Stopping Condition	Reaching t_{min}, or a Predefined number of Replications ($nrep$)
TS	Inner Step Size (Number of Neighbors)	2, 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50
	Neighborhood Structure	Swap and Insertion
	Size of the Tabu list	5, 7, 10, 13, 15, 17, 20, 23 and 25
	Number of Iterations	50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700
GPSO	Swarm Size	30, 40, 50, 60, 70, 80, 90, and 100
	Crossover Operators	3PMBCX
	Mutation Operator	Swap, Insertion, and Inversion
	Mutation Probability (p_m)	0.001, 0.003, 0.005, 0.007, and 0.01
	Number of Iterations	50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700

*In a static strategy, the number of transitions is determined before the search starts. ** The number of generated neighbors will depend on the characteristics of the search (here we used a non-equilibrium condition). ***As in adaptive1 we use in adaptive2 a predefined number of runs without any improvement in the best solution found so far.

4. Experimental Study

In this section, before conducting the tuning process and the simulation runs, a set of 45 instances have been randomly generated to investigate the performance of the designed metaheuristics. The instances were in three categories (small, medium, and large) and are solved to a number of 7 stages. A tuning process is conducted according to the chosen parameters values in Table 4. A wide rang simulation runs have been established to find the best parameters setting for each metaheuristic. Table 5 summarizes the best-found setting of parameters for each metaheuristic.

Table 5: The best-found sittings of parameters for the designed metaheuristics.

Algorithm	Parameter	Values for Small-Scale Problems	Values for Medium-Scale Problems	Values for Large-Scale Problems
SA	Initial Temperature (t_0)	200	400	800
	Final Temperature (t_{min})	0.001	0.001	0.001
	Neighborhood Structure	Swap	Swap	Swap
	Cooling Function	Very slow decrease	Very slow decrease	Very slow decrease
	Equilibrium Condition	Static	Adaptive2	Static
TS	Inner Step Size	15	45	50
	Neighborhood Structure	Swap	Swap	Insertion
	Size of the Tabu list	5	7	7
	Number of Iterations	600	700	700
GPSO	Swarm size	50	60	70
	Crossover operators	3PMBCX	3PMBCX	3PMBCX
	Mutation Operator	Insertion	Swap	Swap
	Mutation Probability (p_m)	0.001	0.01	0.005
	Number of Iterations	400	500	600

We studied the performance of the proposed algorithms according to the best-found setting resulted from the tuning process (Table 5). All runs are done on set of 3 identical computers, all computer packages are DELLTM version OPTIPLEX 7010, each with a 64-based Intel® Core™ i7-3770 CPU running at 3.40GHz with 8 GB RAM. The designed metaheuristics are compiled in MATLAB R2013a and run under windows 7. The performance of the proposed algorithms is testified by solving the 90 generated instances (15 small-scale, 15 medium-scale and 15 large-scale instances), for the 3-stage and 7-stage problems. For each instance, the proposed metaheuristics have been run ten times (10 replications).

The obtained results will be compared and evaluated using the relative percentage deviation RPD of the makespan over the best found solution as a performance measure, RPD will measure the deviation from the best found metaheuristic, which calculated as follows:

$$RPD = \frac{|ProposedMets_{sol} - BestFound_{sol}|}{BestFound_{sol}} x100\%$$

Where:

- $ProposedMets_{sol}$ is the value of C_{max} found by each designed metaheuristic.
- $BestFound_{sol}$ is the best found value of C_{max} among all the designed metaheuristics.
- RPD nearer to zero gives the best results.

The minimum, average, and maximum values of the Makespan for each designed metaheuristic have been summarized in Table 6 and 7 (45 randomly generated instances, with three different machine sets for each instance a, b, and c, have been solved). These tables represent the obtained results of the designed metaheuristics. Applying the chosen performance measure (RPD) to the obtained results will allow us to easily compare the designed metaheuristics with each other, by measuring the deviation from the best found solution.
As mentioned before, the results are for 10 replications, which means that we have the best case scenario (the minimum of that 10 replications), the average, and worst case scenario (the maximum of the 10 replications). We will consider in the analysis and comparisons the worst case results, because with the worst case scenario any other performance for the compared metaheuristics will be for sure better than the performance already compared. On the other hand, the minimum found values (best case scenario) by the compared metaheuristics will be highlighted. Figures 2 and 3 interpret the interval plots for the calculated RPD of the designed metaheuristics at each problem set at number of stages 3 and 7, for the worst case scenario (maximum).

After reviewing the results in Table 6 and Figures 2 and 3, we can find that average deviation from the best found solution (according to the RPD) for each used metaheuristic (as an average of the 3 and 7 stages sets) is as follows: 1.473% for SA, 0.164% for TS, and 1.355% for GPSO. Therefore, in term of RPD performance measure the results show that TS has the lowest variation as it range is the narrowest one. This leads to the conclusion that TS results in less percentage deviation, and have been better than SA and GPSO in term of C_{max} value. Results also showed that

TS has the minimum value of the average RPD (0.164%), which confirm the superiority of the TS compared to other metaheuristics.

Table 6: Results of C_{max} for all type scale problems for S = 3.

Type	Instances	Designed Metaheuristics								
		SA			TS			GPSO		
		Min	Avg.	Max	**Min**	Avg.	Max	**Min**	Avg.	Max
Small-Scale	10a	**124**	124	124	**124**	125.1	126	**124**	124	124
	10b	**72**	72.3	73	**72**	72	72	**72**	72	72
	10c	**96**	96	96	**96**	96	96	**96**	96	96
	15a	**138**	139.1	141	**138**	140	141	**138**	139.5	141
	15b	**85**	87.4	89	**85**	85.7	87	**85**	85.5	87
	15c	**138**	138.8	139	**138**	138.9	141	**138**	139	140
	20a	**189**	189	189	**189**	189	189	**189**	189	189
	20b	113	114.1	116	**110**	110.6	111	111	112.3	115
	20c	**221**	221	221	**221**	221	221	**221**	221	221
	25a	**248**	248	248	**248**	248	248	**248**	248	248
	25b	121	123.6	126	**118**	118.9	120	121	123.4	126
	25c	**240**	240	240	**240**	240	240	**240**	240.2	241
	30a	249	249.1	250	**248**	248	249	249	249.1	250
	30b	130	133.2	136	**128**	128.8	130	131	132.9	136
	30c	**287**	287	287	**287**	287	287	**287**	287	287
Medium-Scale	40a	**467**	472.1	480	469	475.5	483	471	475.4	481
	40b	496	499.2	506	**486**	490	495	487	495.6	502
	40c	**578**	580.7	585	579	581.5	585	**578**	581.6	585
	45a	**496**	499.7	506	500	503	507	499	503.7	511
	45b	527	533.9	544	**523**	524.8	529	525	529.2	537
	45c	**620**	622.4	627	623	623.6	626	621	625	631
	50a	**535**	540.3	543	539	540.7	545	**535**	541.6	545
	50b	582	585.8	590	**574**	576.3	579	579	586.9	604
	50c	**665**	669	675	668	671.2	674	668	672	676
	55a	**582**	588.3	594	583	588.8	596	586	588.8	595
	55b	638	644.8	658	**631**	634.1	638	635	641.7	647
	55c	**725**	727.9	732	727	732	740	727	732.2	742
	60a	**620**	628.4	635	624	630.2	637	622	631.2	636
	60b	672	678.4	685	**665**	668.8	674	673	677.9	683
	60c	**780**	784.8	788	783	786.4	790	782	786.5	791
Large-Scale	70a	**1762**	1774.7	1794	**1762**	1778.8	1800	1766	1774.2	1789
	70b	1780	1797.7	1845	1752	1763.3	1780	**1748**	1771.4	1790
	70c	**1648**	1652.3	1664	1663	1669.6	1682	1675	1690.9	1712
	85a	1984	2008	2029	1992	2005.4	2019	**1981**	2004.6	2022
	85b	2007	2032	2068	2006	2014.7	2039	**1994**	2021	2046
	85c	**1826**	1847	1864	1847	1863.5	1875	1854	1874	1905
	100a	2346	2366.6	2390	**2342**	2355.6	2371	2349	2365.1	2383
	100b	2352	2374.3	2394	**2334**	2356	2368	2340	2354.6	2374
	100c	**2078**	2097.3	2120	2087	2096.2	2121	2091	2120.3	2146
	115a	2546	2561.9	2581	**2542**	2553.8	2571	2546	2561.2	2577
	115b	2618	2642.1	2680	2609	2627	2647	**2608**	2624.3	2644
	115c	**2403**	2426.1	2448	2406	2421.4	2429	2429	2454.3	2478
	130a	3000	3015	3034	2988	3000.6	3016	**2987**	3008	3024
	130b	2966	2999.9	3032	**2943**	2959.3	2971	2955	2984.4	3004
	130c	2808	2829.7	2844	**2802**	2812.1	2820	2833	2857.3	2887

5. Conclusions and Recommendations

In this research, a flexible flow shop scheduling problem introduced and a special case of such environment, which is the No-Wait flexible flow shop scheduling problem, that considered as NP-hard problems are dealt in detail using efficient metaheuristics. An extensive computational study has been carried out on a set of 225 randomly generated

instances by the help of efficient SA, TS, and GPSO metaheuristics. Simulation tests of the generated instances demonstrates that the proposed metaheuristics are capable of solving FFSP and NWFFSP effectively and efficiently due to the balance of global exploration and local exploitation capability. Comparison of simulation results of the used metaheuristics in terms of relative percentage deviation showed that TS has the minimum deviation from the best found values, which confirm the superiority of the TS compared to the other metaheuristics. The results indicate that the proposed TS can be considered as an effective and efficient algorithm for solving the NWFFSP with minimum makespan.

Table 7: Results of C_{max} for all type scale problems for S = 7.

Type	Instances	Designed Metaheuristics								
		SA			TS			GPSO		
		Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
Small-Scale	10a	**162**	162	162	**162**	162.10	163	**162**	162	162
	10b	**172**	174	180	**172**	172.6	175	**172**	172.10	173
	10c	**170**	170	170	**170**	171.60	174	**170**	170	170
	15a	**187**	188.20	189	**187**	187.40	188	188	188.30	190
	15b	216	218	222	**212**	215.50	218	213	216.20	220
	15c	**201**	203	205	**201**	202.20	203	202	203.40	205
	20a	214	216.60	220	**213**	214.60	216	216	218.20	220
	20b	**248**	256.10	267	**248**	251.20	254	252	256.90	263
	20c	**224**	225.50	229	**224**	224.60	226	226	228.10	230
	25a	240	242.40	246	**238**	239	240	244	245.60	248
	25b	314	318.70	327	**311**	312.40	316	316	324	336
	25c	272	272.70	274	**271**	272.10	273	273	278.30	286
	30a	288	289.70	291	**282**	285.90	288	289	293.10	299
	30b	**357**	365.80	372	359	363.10	368	371	378.90	386
	30c	332	334.60	337	332	333	334	337	342.60	348
Medium-Scale	40a	863	873.50	892	**859**	864.80	869	862	870.40	877
	40b	838	855.90	875	**835**	844.50	851	838	849.10	862
	40c	839	844.80	855	**836**	841.20	845	840	845.50	854
	45a	914	925.40	936	913	919.10	927	**910**	925.10	945
	45b	888	903.80	920	885	890.20	895	**883**	894.50	910
	45c	**885**	896.80	907	894	900.90	906	898	903.50	911
	50a	975	982.80	997	**954**	967.10	980	964	974.70	1002
	50b	972	984	997	958	962.50	969	**950**	975.10	996
	50c	957	967.60	977	**948**	955.5	962	959	968.20	987
	55a	1038	1050.80	1059	1037	1043.50	1054	**1029**	1052.60	1065
	55b	1010	1022.30	1034	**1001**	1012.40	1021	1008	1017.30	1033
	55c	1023	1030.10	1038	1030	1032.80	1037	**1022**	1030.10	1035
	60a	1076	1094.60	1122	**1071**	1080.30	1088	1082	1090.50	1108
	60b	1057	1075	1096	**1049**	1059.70	1066	1055	1065.60	1076
	60c	**1064**	1072.60	1081	**1064**	1074.60	1082	1073	1082.80	1090
Large-Scale	70a	2588	2636.20	2710	2580	2589.80	2603	**2574**	2602.90	2611
	70b	2514	2552.20	2583	**2489**	2508.60	2527	2502	2520.70	2537
	70c	2613	2639.90	2701	**2585**	2607.50	2648	2593	2613.30	2636
	85a	2915	2938.50	2974	**2829**	2851.20	2871	2868	2882.90	2911
	85b	2816	2855.70	2896	**2804**	2819.70	2832	2830	2890.70	2921
	85c	**2887**	2943.20	2995	2892	2910.50	2924	2922	2959.30	3007
	100a	3315	3370.40	3414	3277	3312.40	3364	**3275**	3319.50	3344
	100b	3268	3303.50	3341	**3250**	3280.60	3306	3267	3300.70	3356
	100c	3398	3424	3451	**3352**	3382.10	3409	3353	3394.90	3451
	115a	3601	3646.30	3696	**3513**	3535	3552	3550	3577.60	3610
	115b	3519	3567.30	3634	**3507**	3537	3553	3580	3605.10	3644
	115c	3637	3688.70	3738	**3592**	3634.50	3669	3617	3666.80	3723
	130a	3936	4013	4068	**3870**	3906.60	3944	3877	3906.70	3954
	130b	3898	3946.50	4022	**3863**	3908.80	3941	3896	3947.80	4013
	130c	4036	4090	4178	**4006**	4026.20	4042	4014	4042.50	4074

In spite of advantages obtained through proposed study, few limitations exist in the study because they have not been addressed. The solved problems considered in this study are static scheduling problems but the real industrial environment is dynamic in nature, for example, jobs to be processed continuously change over time. The dynamic nature of scheduling approach has not been considered in the study. Moreover, the work has been restricted to only one performance measures, the makespan. Other performance measures of scheduling like flow time, tardiness, lateness and number of tardy jobs can be considered to study the behavior of the proposed approaches.

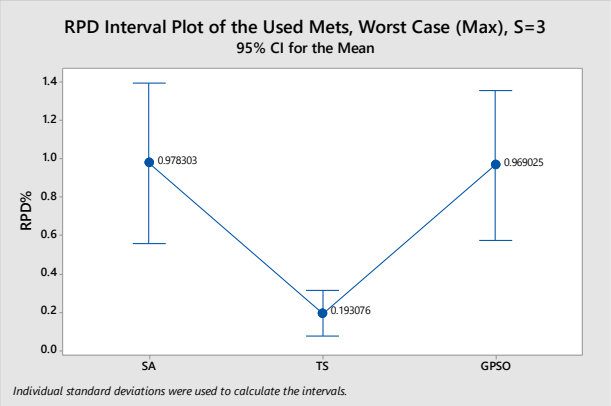

Figure 2: Means and interval plots for all instances with $\boldsymbol{S = 3}$ in terms of RPD.

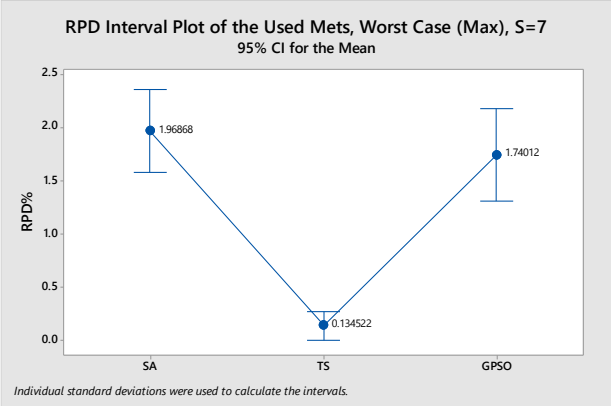

Figure 3: Means and interval plots for all instances with $\boldsymbol{S = 7}$ in terms of RPD.

Particularly, and for future research, the study may be extended to dynamic stochastic shop scheduling where the jobs arrive continuously in time and test the performance of proposed algorithms. In many real manufacturing environments, the nature of the job and machine is probabilistic in nature so stochastic processing time may be considered for the enhancement of the study. In certain applications, setup times cannot be ignored or integrated into the processing times particularly when setup times is sequence-dependent (setup time depends on the previous and next operation performed on the machine). The study can be extended integrating set up time as a critical parameter in the scheduling problem.

References

Aarts, E. and Korst, J. 1988. Simulated annealing and Boltzmann machines. *Wiley*.

Aldowaisan, T. and Allahverdi, A. 2004. New heuristics for m-machine no-wait flowshop to minimize total completion time. *Omega*, 32**,** 345-352.

Arthanari, T. and Ramamurthy, K. 1971. An extension of two machines sequencing problem. *Opsearch*, 8**,** 10-22.

Carpov, S., Carlier, J., Nace, D. and Sirdey, R. 2012. Two-stage hybrid flow shop with precedence constraints and parallel machines at second stage. *Computers & Operations Research*, 39**,** 736-745.

Černý, V. 1985. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45**,** 41-51.

Chang, J., Yan, W. and Shao, H. Scheduling a two-stage no-wait hybrid flowshop with separated setup and removal times. American Control Conference, 2004. Proceedings of the 2004, 2004. IEEE, 1412-1416.

Djellab, H. and Djellab, K. 2002. Preemptive hybrid flowshop scheduling problem of interval orders. *European Journal of Operational Research*, 137**,** 37-49.

Ghaleb, M. A., Suryahatmaja, U. S. and Alharkan, I. M. Metaheuristics for Two-stage No-Wait Flexible Flow Shop Scheduling Problem. Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management, 2015 Dubai, UAE.

Glover, F. 1989. Tabu search-part I. *ORSA Journal on computing*, 1**,** 190-206.

Glover, F. and Kochenberger, G. A. 2003. *Handbook of metaheuristics*, Springer Science & Business Media.

Glover, F. and Laguna, M. 1999. *Tabu search*, Springer.

Grabowski, J. and Pempera, J. 2000. Sequencing of jobs in some production system. *European Journal of Operational Research*, 125**,** 535-550.

Gupta, J. N. 1988. Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society*, 359-364.
Hall, N. G. and Sriskandarajah, C. 1996. A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, 44, 510-525.
Hoogeveen, J., Lenstra, J. K. and Veltman, B. 1996. Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. *European Journal of Operational Research*, 89, 172-175.
Jolai, F., Asefi, H., Rabiee, M. and Ramezani, P. 2013. Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Scientia Iranica*, 20, 861-872.
Jolai, F., Rabiee, M. and Asefi, H. 2012. A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times. *International Journal of Production Research*, 50, 7447-7466.
Jolai, F., Sheikh, S., Rabbani, M. and Karimi, B. 2009. A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. *The International Journal of Advanced Manufacturing Technology*, 42, 523-532.
Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P. and Werner, F. 2008. Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *The International Journal of Advanced Manufacturing Technology*, 37, 354-370.
Kirkpatrick, S. and Vecchi, M. 1983. Optimization by simmulated annealing. *science*, 220, 671-680.
Laarhoven, P. J. 1987. Simulated annealing: theory and applications. *Mathematics and its Applications, Dordrecht: Reidel, 1987*, 1.
Laporte, G., Gendreau, M., Potvin, J. Y. and Semet, F. 2000. Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7, 285-300.
Liu, Z., Xie, J., Li, J. and Dong, J. 2003. A heuristic for two-stage no-wait hybrid flowshop scheduling with a single machine in either stage. *Tsinghua Science and Technology*, 8, 43-48.
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21, 1087-1092.
Michael, R. G. and David, S. J. 1979. Computers and intractability: a guide to the theory of NP-completeness. *WH Freeman & Co., San Francisco*.
Moraga, R. J., DePuy, G. W. and Whitehouse, G. E. 2006. Metaheuristics: A Solution Methodology for Optimization Problems. *Handbook of Industrial and Optimization Problems, Handbook of Industrial and Systems Engineering, AB Badiru (Ed.)*.
Moraglio, A., Di Chio, C., Togelius, J. and Poli, R. 2008. Geometric particle swarm optimization. *Journal of Artificial Evolution and Applications*, 2008, 11.
Morton, T. and Pentico, D. 1993. *Heuristic scheduling systems: with applications to production systems and project management*, John Wiley & Sons.
Petrowski, J. D. A. and Taillard, P. S. E. 2006. Metaheuristics for hard optimization. *Springer*.
Pinedo, M. L. 2012. *Scheduling: theory, algorithms, and systems*, Springer Science & Business Media.
Quadt, D. and Kuhn, H. 2007. A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178, 686-698.
Raaymakers, W. H. and Hoogeveen, J. 2000. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research*, 126, 131-151.
Rabiee, M., Ramezani, P. and Shafaei, R. 2011. An efficient simulated annealing algorithm for a No-Wait Two Stage Flexible Flow Shop Scheduling Problem. *International Journal of Advanced Information Technology*, 1.
Rajendran, C. 1994. A no-wait flowshop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society*, 472-478.
Reinelt, G. 1994. *The traveling salesman: computational solutions for TSP applications*, Springer-Verlag.
Ribas, I., Leisten, R. and Framiñan, J. M. 2010. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37, 1439-1454.
Röck, H. 1984. The three-machine no-wait flow shop is NP-complete. *Journal of the ACM (JACM)*, 31, 336-345.
Ruiz, R. and Vázquez Rodríguez, J. A. 2010. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205, 1-18.
Shafaei, R., Moradinasab, N. and Rabiee, M. 2011a. Efficient meta heuristic algorithms to minimize mean flow time in no-wait two stage flow shops with parallel and identical machines. *International Journal of Management Science and Engineering Management*, 6, 421-430.

Shafaei, R., Rabiee, M. and Mirzaeyan, M. 2011b. An adaptive neuro fuzzy inference system for makespan estimation in multiprocessor no-wait two stage flow shop. *International Journal of Computer Integrated Manufacturing*, 24**,** 888-899.

Shilong, W., Wei, Y. and Jie, Z. Single Hoist Scheduling in no-wait flexible flow shop system. International Symposium on Information, Huangshan, China, 2009. Citeseer, 381-386.

Sriskandarajah, C. and Ladet, P. 1986. Some no-wait shops scheduling problems: complexity aspect. *European Journal of Operational Research*, 24**,** 424-438.

Suman, B. and Kumar, P. 2006. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the operational research society*, 57**,** 1143-1160.

Talbi, E.-G. 2009. *Metaheuristics: from design to implementation*, John Wiley & Sons.

Vila, I. R. and Pascual, R. C. 2009. A hybrid flow shop model for an ice cream production scheduling problem. *Journal of Industrial Engineering and Management*, 2**,** 60-89.

Wang, H. 2005. Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems*, 22**,** 78-85.

Wang, S. and Liu, M. 2013. A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. *Computers & Operations Research*, 40**,** 1064-1075.

Wang, Z., Xing, W. and Bai, F. 2005. No-wait flexible flowshop scheduling with no-idle machines. *Operations Research Letters*, 33**,** 609-614.

Xie, J., Xing, W., Liu, Z. and Dong, J. 2004. Minimum deviation algorithm for two-stageno-wait flowshops with parallel machines. *Computers & Mathematics with Applications*, 47**,** 1857-1863.

Biography

Ibrahim M. Alharkan is an Associate Professor and Chairman of deanship of admission at King Saud University, Research interests are in the areas of production planning and control, modeling and simulation of industrial and service systems, and applied operations research. These areas of interest include production planning and control, inventory control, production sequencing, scheduling and lot sizing; expert system; simulated annealing algorithms; genetic algorithms; Tabu search; scatter search algorithms, and total quality management, quality control, maintenance planning and scheduling, project scheduling.

Mageed A. Ghaleb is a researcher and a master student at King Saud University (KSU), in industrial engineering department. His area of expertise is industrial and manufacturing systems. He received the BSc in Industrial Engineering from Taiz University, Taiz, Yemen.