Journal of Engineering Science and Technology
Vol. 8, No. 5 (2013) 603 - 622

TAGUCHI METHOD FOR THREE-STAGE ASSEMBLY FLOW SHOP SCHEDULING PROBLEM WITH BLOCKING AND SEQUENCE-DEPENDENT SET UP TIMES

AREF MALEKI-DARONKOLAEI[1,*], IMAN SEYEDI[2]

[1]Department of Business and Accounting, University of Applied Science and Technology, Golestan, Gorgan, Iran
[2]Department of Industrial Engineering University of Payame Noor19395-4697, Tehran, Iran
*Corresponding Author: a.maleki.1983@gmail.com

Abstract

This article considers a three-stage assembly flowshop scheduling problem minimizing the weighted sum of mean completion time and makespan with sequence-dependent setup times at the first stage and blocking times between each stage. To tackle such an NP-hard, two meta-heuristic algorithms are presented. The novelty of our approach is to develop a variable neighborhood search algorithm (VNS) and a well-known simulated annealing (SA) for the problem. Furthermore, to enhance the performance of the (SA), its parameters are optimized by the use of Taguchi method, but to setting parameters of VNS just one parameter has been used without Taguchi. The computational results show that the proposed VNS is better in mean and standard deviation for all sizes of the problem than SA, but on the contrary about CPU Time SA outperforms VNS.

Keywords: Three-stage assembly flowshop scheduling, Sequence-dependent setup times, Blocking times, Simulated annealing, Variable neighborhood search, Taguchi method.

1. Introduction

Today, because of strong competition and limitation of resources in our environment, scheduling is an important decision-making process in production and service industries. There are many papers on flow shop scheduling problem since 1954 when Gupta [1] published his article. Assembly-type production systems have unfolded partially as an answer to the market pressure for larger product diversity [2]. Assembly flowshop is a hybrid flowshop presented for the first time by Lee et al. [3]. In common flowshop scheduling, we have two main

Nomenclatures

At_i	Assembly time of job in position i
C_i	Completion time of job j in position i at the end of third stage
$C_{i,1}$	Completion time of job j in position i at the end of first stage
$C_{i,2}$	Completion time of job j in position i at the end of second stage
C_{max}	Completion time of the job in the last sequence that is equal to C_n
$D_{i,h}$	Departure time of job in position i at stage h
$e_{i,h}$	Starting time of job in position i at stage h
m	Number of machines
n	Number of jobs
$S_{i\text{-}1,i,k}$	Set up time on machine k from job in position i-1 to job i at the first stage
T_i	Time of collecting and transferring job in position i to third stage
$t_{j,k}$	Processing time of job j on machine k at first stage
w_j	Assigned weight to job j
$x_{i,j}$	If job j is in position i of sequence$x_{i,j} = 1$; otherwise, it is 0.

elements, namely a group of M machines and a set of N jobs to be processed on this group of machines [4]. Assembly flowshop scheduling is a type of flowshop that at first each job has to be processed at the first stage which consists of m different parallel machines and then assembled at the second stage including only one assembly machine [5]. Most of studies considered a two-stage assembly flowshop scheduling problem (AFSP) defined as follows. There are m machines at the first stage while there is only one machine at the second stage. Therefore, each job consists of m+1 operations. Also, there are n jobs available at time zero and no preemption is allowed. The first m operations of a job are performed at the first stage in parallel by m machines and the final operation is conducted at the second stage. Each of m operations of a job at the first stage is performed by a different machine and the last operation on the machine at the second stage may start only after all m operations at the first stage are completed. Each machine can process only one job at a time. It should be noted that when there is only one machine at the first stage the addressed problem became to two stage flowshop. In the two-stage AFSP, assumed collecting and transferring time of components from the first stage to assemble is negligible. But to have more realistic environments of a production system, it is required that the intermediate operation is devoted to collect and transport the fabricated parts from the various production areas to the assembly area. This stage is important especially when parts are manufactured in multiple production sites. The three-stage AFSP is the extended model of two-stage assembly flowshop that the collecting and transferring actions are regarded as the second stage, and assembly machine is in the third stage.

In the three-stage AFSP there are n jobs which each of them includes m components. At the first stage, there are m parallel and independent machines, each machine can process just one component. When all of m components of each job are processed on the first stage machines, they will be collected and transferred to the assembly machine (i.e., third stage) by passing the second stage (i.e., transportation stage). Then the machine at the third stage assembles m components of job that are transferred from the first stage together for completing a job [2].

Lee et al. [3] studied a two-stage AFSP with considering two machines at the first stage. Potts et al. [6] presented this problem with an optional number of machines in the first stage, the objective in this problem was to minimize makespan. They proved that these problems are NP-hard. Allahverdi and Alanzi [7] addressed a two-stage AFSP with setup time by minimizing the total completion time and they proposed a dominance relation and three heuristics, such as Ntabu, SDE (Self-Adaptive Differential Evolution) and NSDE. Alanzi and Allahverdi [8] considered a two-stage AFSP with the objective of minimizing the weighted sum of makespan and maximum lateness and presented heuristics namely TS, PSO, and SDE. Cheng et al. [9] studied two-stage differentiation flowshop consisting of a common critical machine in stage one and two independent dedicated machines in stage two by minimizing the weighted sum of machine completion times. Ng [4] proposed a branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. Ruiz and Allahverdi [10] minimized the bi-criteria of makespan and maximum tardiness with an upper bound on maximum tardiness of the flowshop scheduling problem. Sun [11] addressed powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. Yokoyama and Santos [12] presented a branch-and-bound method for three-stage flowshop scheduling with assembly operations to minimize the weighted sum of product completion times where there is only one machine in each stage. Koulamas and Kyparisis [2] introduced three-stage assembly scheduling problem by considering transportation as second stage and minimizing the makespan and analyzed the worst-case ratio bound for several heuristics for this problem.

In some environments, there are limited buffers or zero buffers between stages. Generally, blocking scheduling problems arise in modern manufacturing environments with finite intermediate buffers between machines, such as just-in-time production systems or flexible assembly lines, and those without intermediate buffers, such as surface mount technology lines in the electronics industry for assembling printed circuit boards, which includes three different stages in the following sequence: solder printing, component placement and solder reflow [13].

Hall and Sriskandarajah [13] reviewed machine scheduling problems with blocking and no wait in process. Qian [14] presented an effective hybrid algorithm based on deferential evolution (DE) for multi-objective flow shop scheduling with limited buffers. Liu [15] solved flow shop scheduling with limited buffers with an effective hybrid PSO-based algorithm to minimize the maximum completion time. Wang [16] introduced a hybrid genetic algorithm (GA) for flowshop scheduling with limited buffers with the objective to minimize the total completion time. Grabowski [17] developed a fast tabu search (TS) algorithm to minimize the makespan in a flow shop problem with blocking. Ronconi [18] analyzed the minimization of the makespan criterion for the flowshop problem with blocking by proposing constructive heuristics, namely MM, MME and PFE. Tavakkoli-Moghaddam [19] presented an efficient memetic algorithm (MA) combined with nested variable neighborhood search (NVNS) to solve the flexible flow line with blocking (FFLB). Sawik [20] addressed a new mixed integer programming for the FFLB. Ronconi [21] introduced a GRASP-based (greedy randomized adaptive search Procedure) heuristic method for a scheduling problem with blocking to minimizing the total tardiness. Tozkapan [22] developed a lower bounding procedure and a dominance criterion incorporated into a branch-and-bound

procedure for the two-stage AFSP to minimize the total weighted flowtime. Yagmahan and Yenisey [23] offered a multi-objective ant colony algorithm for flowshop scheduling to minimizing the makespan and total flow time. Sung and Kim [24] developed a branch-and-bound algorithm for two stage multiple assembly flowshop to minimize the sum of completion times. Yokoyama [25] considered flowshop scheduling with setup and assembly operation and to solve used pseudo-dynamic programming and a branch-and-bound. Liu and Kozan [26] studied scheduling flowshop with combined buffer condition considering blocking, no-wait and limited-buffer. Lee et al. [27] brought the concept of blocking into the deteriorating job scheduling problem on the two-machine flow- shop. They suggested a branch-and-bound algorithm incorporating with several dominance rules and a lower bound also several heuristic algorithms. Gong et al. [28] studied two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. Wang et al. [29] proposed a novel hybrid discrete differential evolution (HDDE) algorithm for solving a flowshop scheduling with blocking to minimize the makespan.

In addition, in most of real manufacturing systems, sequence dependent setup times are an ineluctable factor. Norman [30] explored a flowshop scheduling problem with finite buffer and sequence-dependent setup times and proposed a TS method. Hatami et al. [31] extended the three-stage assembly flowshop model presented in [2] with sequence-dependent setup time by minimizing the mean flow time and maximum tardiness and they proposed two meta-heuristics, namely simulated annealing (SA) and tabu search (TS).

In this paper, two meta-heuristics proposed to solve the three-stage assembly flow shop with blocking and sequence dependent setup time by minimizing makespan (C_{max}) and weighted mean completion time ($\bar{C}_w$). Problem description and mathematical model are in next section. The proposed meta-heuristics are introduced in section 3 and the performances of them are evaluated in section 4 after using Taguchi experimental design which analysed suggestive algorithms. Finally, summary of results and some possible further researches are mentioned in section 5.

2. Problem Description

The problem considered in this paper is a three-stage AFSP with sequence-dependent setup times at the first stage and blocking time between stages minimizing the weighted mean completion time and makespan. In this problem, there are n jobs available at zero time. Job preemption is not allowed and each job includes m parts or components. We have m independent parallel machines at the first stage and every part of a job should be processed on just one machine at the first stage. Each machine can process only one part. Setup time of a job on machines depends on job and its previous job. After completing all m parts of a job at the first stage, if the next stage machine is available, they will be collected and transferred by an automatic transportation system. We assume that transfer is done in the second stage. At the second stage and third stage, there is only one machine so each job needs to m+2 operations ,m operations at first stage and 2 operations for second and third stage, to be completed. There is no buffer storage between stages so if the downstream machine is not available, the job stays on the current machine and blocks, as long as, the machine in the next stage will be available [32].

To compute the objective function we give weight to each of objects which showed in this way [8]:

$$\text{OF} = \alpha(\sum_{j=1}^{n}\sum_{i=1}^{n} w_j C_i / W) + (1-\alpha)C_{max}$$

where $0 < \alpha < 1$. Note that when α is equal to 0 or 1, the problem is reduced to the single criterion of C_{max} or the weighted mean weighted completion time, respectively. The objective is to find a schedule which yields a minimum objective function value (OFV). In addition, we use the following notations in the presented model.

To calculate makespan the completion time of each stage calculated as follow:

$$C_{i,1} = e_{i,1} + \{max_{k=1,..,m}(\sum_{j=1}^{i}(t_{j,k} + S_{j-1,j,k}))\} \times x_{i,j} \qquad \forall \tag{1}$$

Above formulation describe completion time at first stage and the second and third stage's is as two next equations respectively: and makespan is equal completion time of last job in third stage.

$$C_{i,2} = e_{i,2} + \sum_{j=1}^{n}(T_i \times x_{i,j}) \qquad ;\forall i \tag{2}$$

$$C_{max} = C_{i,3} = e_{i,3} + \sum_{j=1}^{n}(At_i \times x_{i,j}) \quad ;\forall i \tag{3}$$

As this problem considered blocking between stages so it's time is explained as follow and it is noteworthy that blocking doesn't accrue for first job and last job in optimum sequence.

$$B_{i,h} = \begin{cases} D_{i,h} - C_{i,h}, & D_{i-1,h+1} > C_{i,h} \\ 0, & D_{i-1,h+1} \leq C_{i,h} \end{cases} \tag{4}$$

3. Proposed Meta-heuristics

The considered model is belong to the (AF (m, 1, 1)//)[1] group and this kind of problem is a general form of (AF (m, 1)//)[2] and (F3//)[3], it is accepted that the model is NP-hard [3]. Accordingly, the considered problem is NP-hard in a strong sense. As regards this problem is strongly NP-hard, it is strongly suggested to focus on heuristic or meta-heuristic based approaches to solve such a hard problem. Hence to solve this model, two more applicable meta-heuristics are suggested, namely variable neighborhood search (VNS) and simulated annealing (SA). These meta-heuristics are explained in the following two subsections.

3.1. Variable neighbourhood search algorithm

One of the most recent meta-heuristics that has a systematic change of the neighborhood in search is variable neighborhood search (VNS) expanded for the problem solving in an easier way. It has been introduced as one of the very famous local search methods [33, 34] that takes more attention day-by-day, because of its ease of use and success in solving combinatorial optimization

[1]Three-stage assembly flowshop problem with m parallel machine at the first stage and single machine at the second and third stages

[2]Two-stage assembly flowshop problem with m parallel machine at the first stage and single machine at the second stage

[3]Three machine flowshop

problems. A common VNS algorithm starts with an initial solution, $x \in S$, where S is the whole set of search space, and manipulates it through a two-nested loop.

To develop an effective VNS algorithm, one need two kinds of neighborhood functions; $N_k^S(x)$ and $N_l^{LS}(x)$ resulting each with a particular neighborhood search (NS), where $N_k^S(x)$ and $N_l^{LS}(x)$ denote neighborhood functions for shake and local search functions, respectively. The NSs used may be more than one for each function (shake and local search) so as to achieve a valuable neighborhood change. For that purpose, the indices, k and l, are to be used for shake and local search functions, respectively, in order to ease switching from one to another neighborhood. Manifestly, both indices have upper boundaries, which are denoted with k_{max} and l_{max}. Hence, $1< k <k_{max}$ and $1< l <l_{max}$ are the ranges identified for each indices.

According to this basic scheme, a sequence of neighborhood structures, which explain neighborhoods around any point $x \in X$ of the solution area, are first selected. Then the local searches are used and eventuate in a local optimum x. A point x' is randomly selected within the first neighborhood $N_1(x)$ of x and a descent from x' is done with the local search routine. This leads to a new local minimum x''. At this point, three results are achievable:

- $x'' = x$, one is again at the bottom of the same valley. In this case, the procedure is iterated using the next neighborhood $N_k(x)$, $k \geq 2$;
- (II) $x'' = x$ but $f(x'') \geq f(x)$, i.e., another local optimum has been found, which is not better than the previous best solution(or current); in this case to the procedure is repeated using the next neighborhood;
- $x'' = x$ and $f(x'') < f(x)$, i.e., another local optimum, better than the incumbent has been found. In this case, the search is re-centered around x'' and begins again with the first neighborhood should the last neighborhood be obtained without a solution better than the incumbent being found, the search begins again at the first neighborhood $N_1(x)$ until a stopping condition (e.g., the greatest amount of time or the greatest amount of number of iterations or maximum number of iterations since the final improvement) is satisfied.

Initialization. Select the set of neighborhood structures N_k, for k=1,…,k_{max}, that will be used in the shaking phase, and the set of neighborhood structures $N_l$for l =1,…,k_{max}that will be used in the local search; find an initial solution x and make it better by using RVNS; choose a stopping condition;

Repeat the following algorithm until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2)Repeat the following sequence until $k = k_{max}$:

(a) Shaking: Generate a point x'at random from the $k$thneighborhood $N_k(x)$ of x;

(b) Local search: Apply some local search Procedure with x' as initial solution; denote with x'' the so obtained local optimum;

(c) Move or not: If this local optimum is better than the current, move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Steps of the primary VNS

3.1.1. Initial solution

Any method capable of generating a feasible solution is adequate for the first part of our algorithm. However, it seems that a good initial solution can reduce the computation time. In this paper, as the general variable neighborhood search scheme, we improve the initial solution found by the reduced VNS.

3.1.2. Neighbourhood structure

The NS with which the neighboring solutions are decided to move to is one of the very important elements of meta-heuristics that use NSs. That is why the performance of those meta-heuristics significantly depends on the efficiency of the NS. Although there are many other NSs reported in the literature, we prefer these three meta-heuristics due to simplicity and ease of use alongside a reasonable efficiency. The others maybe provide more efficiency; however, they definitely require much more computational time, experience and hard working. Every time a neighborhood is chose, a random procedure is called. This approach selects a random solution from the selected neighborhood structure. Therefore, for this paper, three procedures are explained in the following manner, one for each k:

1. For $N_1(S)$: Two pairs of different jobs swap together:

- Choose randomly four jobs.
- Swap each pair of jobs independently.
- Consider the following sequence of jobs: [5 2 6 8 7 1 3 9 4], For the corresponding digits numbers, 1, 2, 6, 9, are randomly selected, the jobs 1&2 and the jobs 6&9 must be swapped respectively:

Sequence: [52 6 8 7 1 3 9 4] → [25 6 8 7 4 3 9 1]

Jobs number: 1 2 3 4 5 6 7 8 9 2 1 3 4 5 9 7 8 6

2. For $N_2(S)$: Two different parts of sequence exchange together:

- Select randomly a number 'k' which is $3 \leq k < n/2$.
- Exchange the first k jobs with the last k jobs respectively in the sequence.
- For the sequence of [5 2 6 8 7 1 3 9 4], if k=3 jobs 1, 2, 3 (first part) must be swapped with jobs 7, 8, 9 (last part) respectively:

Sequence: [5 2 6 8 7 1 3 9 4] → [394 8 7 1 526]

Jobs number: 1 2 3 4 5 6 7 8 9 7 8 9 4 5 6 1 2 3

3. For $N_3(S)$: Inversing jobs in two parts separately:

Choose randomly a number 'k' which is $3 \leq k < n/2$. Inverse the first k jobs and inverse the last n-k jobs individually. For the sequence of [5 2 6 8 7 1 3 9 4], if k=3, then jobs 1, 2, 3 and jobs 4, 5, 6, 7, 8, 9 must inverse themselves respectively:

Sequence: [5 2 68 7 1 3 9 4] → [6 2 54 9 3 1 7 8]
Jobs number: 1 2 3 4 5 6 7 8 9 3 2 1 9 8 7 6 5 4

3.1.3. Local search

There are several variations of the VNS structure. In our proposed VNS, we use a simple local search for each neighborhood. In this local search, all of swaps between two jobs for a sequence are examined and store the best one in each swap, so that in the end of the search, best of them is selected. Note that the maximum number of iterations since the last improvement is equal to $3\times n$ as a starting condition.

3.2. Simulated annealing (SA)

The simulated annealing (SA) algorithm was initially proposed by Metropolis et al. [35] in order to simulate the annealing process. SA begins with a high temperature. After generating an initial solution, it tries to move from the current solution to one of its neighborhood solutions. The variations in the objective function values (i.e., ΔE) are calculated. If a new solution gives a better objective value, it is accepted. If a new solution yields a poor value, it can be accepted according to the probability function P, where k_B Boltzmann's constant and T is the current temperature. By accepting poor solutions, SA can avoid being trapped on local optima. It repeats this process L times at each temperature to gain the thermal equilibrium, where L is a control parameter, usually called the Markov chain length. The parameter T is gradually reduced by a cooling function as SA proceeds until the stopping condition is met.

Algorithm: Simulated annealing

```
Procedure Simulated Annealing
  C = Choose an initial solution
  T = Choose an initial temperature
  REPEAT
    S' = Generate a neighbor of the solution C
    ΔE = objective( S' ) – objective( C )
    IF (ΔE > 0 ) THEN // S' better than C
      C = S'
    ELSE with probability EXP(ΔE/ T )
      C = S'
    END IF
    T = lower the T using linear/ non-linear techniques
  UNTIL meet the stop criteria
  End
```

4. Setting Parameters

To have better performance of the proposed algorithms it's necessary to set parameters properly. We used the Taguchi method to set the parameters for SA. It should be noted that, for the VNS, because there is just one parameter which is stopping criteria, we do not use this method. Setting parameters for the SA is considered in the following section.

4.1. Taguchi experimental design

To achieve better robustness of the algorithms by not producing functional variance under the external environment influence, the "parameter design" can be applied to process design. The Taguchi method describes that the optimal operator combination is to minimize variances of quality characteristics resulted from the S/N ratio, which explains the reason why the parameter design is called robust design. A long with the S/N ratio utilized for decreasing the variances, the mean of quality characteristics is also used for deciding the adjustment factors that are utilized for causing to approach the quality characteristic to the objective point. In addition, quality characteristic of this research is the mean objective function value (OFV), which prefers the lower is better principle. Thus, the smaller S/N ratio has the better characteristic result. The S/N ratio is computed by:

$$Z = -10 \times \log\left(\sum_{i=1}^{n} \frac{y_i^2}{n} \right) \tag{5}$$

Earlier formula is derived from the loss function with a lower is a better principle, which is given by

$$L(Y) = K \times Y^2 \tag{6}$$

4.2. Parameter setting for SA

In the following, the pattern of generation of test data and the appropriate Taguchi scheme are explained. Subsequently, the analysis of the Taguchi experimental design is calculated on SA meta-heuristic.

In this paper, the processing times of the first and third stages are integers and randomly generated from the uniform distribution (1, 100) on all *m* first stage machines and single third stage machine, second stage processing times are integers as well and randomly generated from the uniform distribution (1, 10) on single second stage machine [2]. Setup times are integers and randomly generated from uniform distribution (1, 20) on all m machines [36], also problems divided to three groups: small, medium and large size problems. The problems that had investigated are shown in Table 1.

Table 1. Solved Problems to Setting Parameter.

Dimension of problem					
Small		**Medium**		**Large**	
N	**M**	**N**	**M**	**N**	**M**
15	2	30	2	60	4
15	4	30	4	60	2
20	2	30	2	65	6
20	4	30	4	70	8
20	6	40	6		
		55	4		

The data required for the SA consists of the initial temperature (T_0), cooling pattern (α), final temperature (T_f) and finally number of repetition (F). Table 2 shows the factors and their levels.

Table 2. Factor Levels.

Factor	Level index	Level
Initial temperature T_0	1	0.85
	2	0.88
	3	0.91
Final temperature T_f	1	0.00095
	2	0.00100
	3	0.00105
Cooling pattern α	1	0.95
	2	0.96
	3	0.97
Number of repetition	1	40
in each temperature F	2	45
	3	50

The full factorial experiment design for the aforesaid four factors requires 3^4=81experiments for the algorithm. To determine optimum parameters levels these experiments are conducted for three levels of problems: (small, medium, and large). For each small, medium and large size of problem is considered 5 different problems and each of them is repeated 4 times. Therefore total number of required experiments for doing factorial designs is, 81×5×4×3=4860 but considering cost and time, this type of experimental design is not economical. On the other hand, considering statistical theories, it is not required experimenting all the combinations of the factors. Hence, we use fractional replicated designs. To select an appropriate orthogonal array, it is required to calculate the number of the degree of freedom. In this paper, a degree of freedom for the total mean and two degrees of freedom for each factor with three levels (2×4=8) are required for each algorithm. Thus, the sum of the required degree of freedom equals to 1+2×4=9. Therefore, the appropriate array at least must have 9 rows. Table 3 shows the orthogonal array $L_9(3^4)$, where control factors are assigned to the columns of the orthogonal array, and the corresponding integers in these columns indicate the actual levels of these factors. In the foregoing scheme only main effects are estimated.

Table 3. Orthogonal Array L_9 (3^4).

Trial	T_0	T_f	α	F	Trial	T_0	T_f	α	F	Trial	T_0	T_f	α	F
1	1	1	1	1	**4**	2	1	3	3	**7**	3	1	2	2
2	1	2	3	2	**5**	2	2	2	1	**8**	3	2	1	3
3	1	3	2	3	**6**	2	3	1	2	**9**	3	3	3	1

4.2.1. Analysis of performance ratios of S/N for quality characteristics

S/N value for each quality characteristic of each level in small, medium and large sized problems is calculated by Eq. (6) and has shown in Table 4.

To consider the effect of varying levels on averaged S/N ratio we use main effect plot, therefore total amount of S/N ratio and averaged this amount is calculated by Eqs. (7) and (8) then results are shown in Table 5 and Figs. 1 to 3.

$$S_{xi} = \sum_{j=1}^{n} \left(F(\delta_{xj}) \times \eta_j \right) \tag{7}$$

$$A_{xi} = \left(\frac{S_{xi}}{\sum_{j=1}^{n} F(\delta_{xj})} \right) \tag{8}$$

$$F(\delta_{xj}) = \begin{cases} 1 & \delta_{xj} = i \\ 0 & \delta_{xj} \neq i \end{cases}$$

where S_{xi} is total S/N ratio in level i of factor x, δ_{xj} is the index of level of facto x in trail j, η_j is the S/N ratio in trial j, and is the total trial.

Table 4. S/N Value for Each Quality Characteristic.

Control parameters					S/N Ratio		
Trial	**T_0**	**T_f**	**α**	**F**	**Small**	**Medium**	**Large**
1	1	1	1	1	-20.19	-24.0955	-32.9024
2	1	2	3	2	-19.5493	-23.6962	-30.3919
3	1	3	2	3	-19.9319	-25.5952	-33.592
4	2	3	3	1	-19.9822	-23.719	-31.1143
5	2	1	2	2	-20.1281	-25.2343	-31.4367
6	2	2	1	3	-20.424	-25.6668	-33.0655
7	3	2	2	1	-20.0482	-24.6435	-30.8041
8	3	1	1	2	-20.7331	-24.6621	-33.5705
9	3	3	3	3	-20.3104	-24.6515	-35.1058

Table 5. Average Levels of S/N for Quality Characteristic in Problems.

Size	Levels	Factors			
		T_0	**T_f**	**α**	**F**
Small	**1**	-19.89	-20.073	-20.449	-20.210
	2	-20.178	-20.137	-20.036	-20.007
	3	-20.346	-20.222	-19.947	-20.216
Medium	**1**	-24.462	-24.153	-24.808	-24.660
	2	-24.873	-24.531	-25.158	-24.669
	3	-24.652	-25.305	-24.022	-24.659
Large	**1**	-32.295	-31.607	-33.179	-33.148
	2	-31.872	-31.800	-31.994	-31.420
	3	-33.160	-33.921	-32.204	-32.759

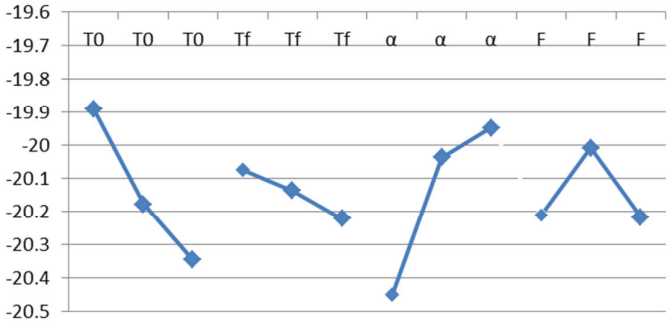

Fig. 1. Effects of Factors on Performance S/N of Quality Characteristic in Small Size Problem.

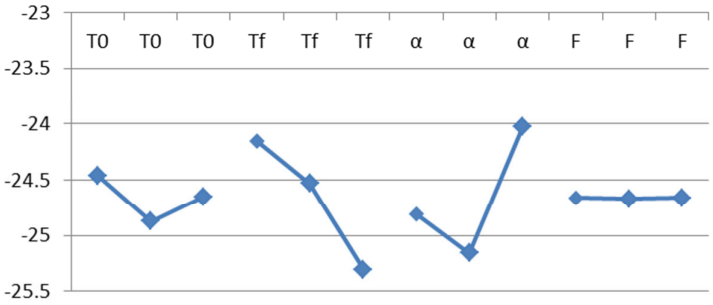

Fig. 2. Effects of Factors on Performance S/N of Quality Characteristic in Medium Size Problem.

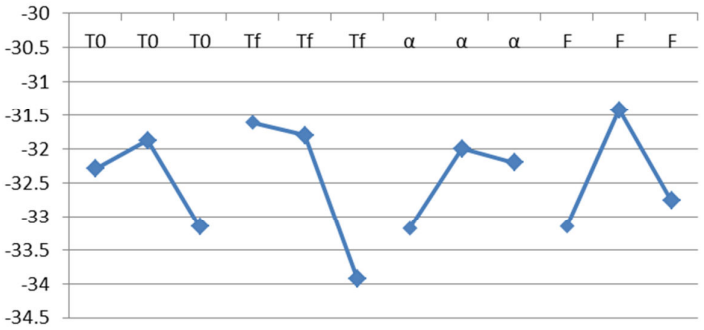

Fig. 3. Effects of Factors on Performance S/N of Quality Characteristic in Large Size Problem.

4.2.2. Analysis of quality characteristics

To determine the optimum levels of control parameters, after the analysis of S/N ratio the average of characteristics must be considered. Whenever the S/N ratio for some levels of parameter has negligible difference, the optimum level for parameter is determined by usage of average of data in levels. In Table 6 the average of quality characteristics for each trial is shown.

Table 6. Average Value for the Quality Characteristic.

Control parameters					Average ratio		
Trial	**T_0**	**T_f**	**α**	**F**	**Small**	**Medium**	**Large**
1	1	1	1	1	690.6153	1413.204	2179.742
2	1	2	3	2	689.478	1412.375	2169.466
3	1	3	2	3	689.576	1416.457	2183.584
4	2	3	3	1	689.567	1413.587	2175.646
5	2	1	2	2	689.947	1415.867	2174.784
6	2	2	1	3	690.784	1416.548	2183.577
7	3	2	2	1	689.385	1415.368	21.80.339
8	3	1	1	2	690.325	1415.976	2191.456
9	3	3	3	3	690.024	1416.137	2195.874

The average ratios of mean among all levels are calculated and are shown in Table 7 and Figs. 4 to 6.

Table 7. Average Levels of Mean Ratio for Quality Characteristic in Problems.

Size	levels	Factors			
		T_0	T_f	α	F
Small	1	690.6153	690.9511	691.9288	690.4596
	2	691.5153	691.1896	690.4893	691.2494
	3	692.0386	691.7822	690.2618	691.7562
Medium	1	1413.204	1412.191	1415.209	1414.449
	2	1414.856	1412.928	1415.756	1414.523
	3	1415.246	1416.866	1414.798	1414.56
Large	1	2179.742	2177.861	2188.486	2189.564
	2	2180.491	2177.995	2175.526	2173.612
	3	2180.819	2178.452	2180.565	2187.88

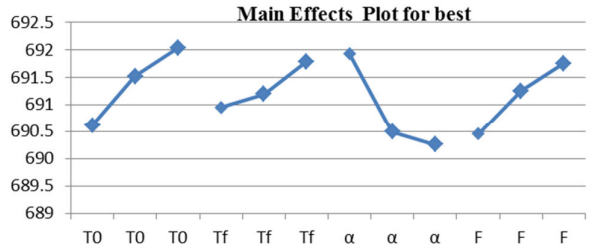

Fig. 4. Effects of Factors on Mean Criteria for Quality Characteristic in Small Size Problem.

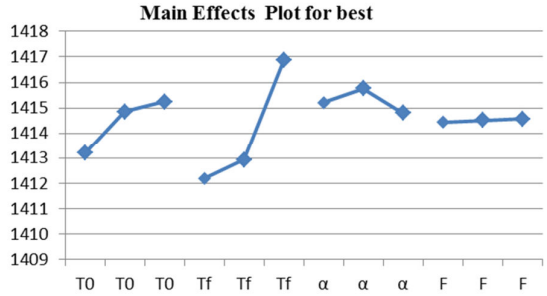

Fig. 5. Effects of Factors on Mean Criteria for Quality Characteristic in Medium Size Problem.

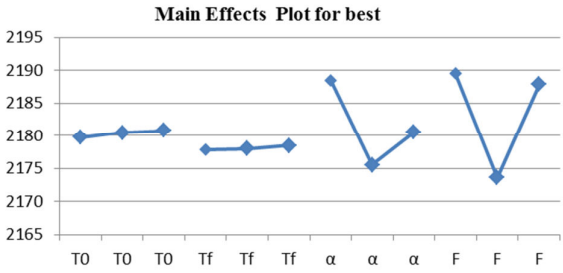

Fig. 6. Effects of Factors on Mean Criteria for Quality Characteristic in Large Size Problem.

4.2.3. Analysis of factor level effect on SA algorithm performance

As mentioned above, to determine the optimum levels of control parameters, after the analysis of S/N ratio the average of characteristics must be considered. Whenever the S/N ratio for some levels of parameter has negligible difference our decision is based on mean criteria. In this section, values of control parameters to obtain optimal objective function value are determined according to the results of S/N ratios and mean of data for small, medium and large sizes of problems. Results are shown as follow for problems in Table 8.

Table 8. Optimum Parameters for Different Size Problems.

Optimum factor	Small	Medium	Large
$\mathbf{T_0}$	0.98	0.85	0.88
$\mathbf{T_f}$	0.00095	0.00095	0.00095
$\boldsymbol{\alpha}$	0.97	0.96	0.96
F	45	All levels	45

According to Table 8 for medium size problem there is no significant difference among parameter levels of *F*, so we consider mean criteria in Fig. 5 and it shows that there is no significant difference among parameter levels too, therefore parameter *F* has no significant impact on quality characteristic.

4.3. Parameter setting for VNS

According to the one of the VNS algorithm parameters the only parameter that has considered is number of repetition for stopping criterion, we do not consider it for parameter setting by the Taguchi method , the levels that were considered are (40, 50, 60, and 70), then according to the variety experiments results have shown in Table 9.

Table 9. Levels for VNS.

Dimension	Number of repetition for stopping criteria
Small	50
Medium	50
Large	60

5. Computational Results

In this section, we now proceed to evaluate the proposed VNS against SA performance. The parameters that have set are based on results of algorithms setting parameter that has conducted before. These proposed meta-heuristics are accomplished in Delphi 10 in a PC with Core 2 Duo processors of 1.5 GHz running under the Windows 7 operating system with 2 GHz of RAM. Test problems are randomly generated for different number of jobs (n): {20, 40, 60, and 80}, a different number of machines at first stage (m): {2, 4, 6, and 8}, and (α) : {0, 0.3, 0.7, and 1}are considered. We divided problems into three groups

as small, medium and large size problem that shown in Table 10, so 64 problems are considered which each of them implemented 30 times by each algorithm. Processing time at first stage and third stage are integer and randomly generated by uniform distribution (1,100) on all m first stage machines and single machine at third stage. For second stage, transferring times are integer and randomly generated from the uniform distribution (1, 10) [2], and setup times are integer and randomly generated from the uniform distribution (1, 20) [36].

Table 10. Solved Test Problems By VNS and SA To Evaluate Performance of Algorithms.

	N	M			
Small	**20**	2	4	6	8
	40	2	-	-	-
Medium	**40**	-	4	6	8
	60	2	4	-	-
large	**60**	-	-	6	8
	80	2	4	6	8

The results of the computational tests are shown in Tables 11 to 13 and as well as in Figs. 7 to 9 for small, medium and large size of problem respectively.

Table 11. Comparison Result for Small Size Problems.

Size of problem	*N*	*m*	α	SA			VNS		
				Mean	**CPU time**	**Standard Deviation**	**Mean**	**CPU time**	**Standard Deviation**
Small	**20**	**2**	0	175.0279	60	1.259	173.2776	58	1.247
			0.3	394.76	62	1.353	390.8124	60	1.339
			0.7	686.6333	60	2.188	679.767	61	2.166
			1	904.2077	61	1.635	895.1656	62	1.618
		4	0	218.4269	58	1.974	214.0584	60	1.935
			0.3	463.0141	59	4.746	453.7539	62	4.651
			0.7	784.7123	59	2.310	769.0181	58	2.264
			1	1028.544	60	2.837	1007.973	62	2.780
		6	0	206.9108	63	2.718	198.6343	65	2.609
			0.3	471.5959	62	2.465	438.5842	66	2.367
			0.7	825.0581	62	2.095	767.304	64	2.011
			1	1087.733	63	2.002	1011.592	66	1.922
		8	0	206.9435	64	1.570	192.4575	64	1.491
			0.3	484.6809	67	1.709	445.9064	66	1.623
			0.7	846.3771	64	1.840	778.6669	68	1.748
			1	1122.267	60	3.039	1032.485	70	2.887
	40	**2**	0	289.4471	61	1.694	266.2913	62	1.52
			0.3	705.5953	60	1.988	627.9798	65	1.7900
			0.7	1257.563	62	2.025	1119.231	67	1.822
			1	1672.643	62	2.179	1488.652	69	1.961

Table 12. Comparison Result for Medium Size Problems.

Size of problem	*N*	*m*	α	SA			VNS		
				Mean	**CPU time**	**Standard Deviation**	**Mean**	**CPU time**	**Standard Deviation**
Medium	**40**	**4**	0	391.596	61	3.0001	348.520	67	2.700
			0.3	876.0355	60	1.943	770.9112	68	1.749
			0.7	1525.923	60	2.177	1342.812	70	1.959
			1	2012.16	60	1.981	1770.70	71	1.783
		6	0	325.231	62	1.713	286.203	74	1.456
			0.3	765.117	62	2.616	665.6518	76	2.223
			0.7	1349.276	63	2.357	1173.87	75	2.004
			1	1787.4	62	1.753	1555.038	78	1.490
		8	0	399.926	64	2.439	339.937	78	2.098
			0.3	969.5593	68	2.662	824.1254	79	2.289
			0.7	1727.103	64	2.657	1468.038	85	2.285
			1	2291.22	63	2.491	1924.62	84	2.142
	60	**2**	0	502.098	61	3.275	421.763	88	2.653
			0.3	1203.982	62	2.236	1011.345	92	1.811
			0.7	2139.484	62	1.862	1797.166	95	1.508
			1	2841.23	64	2.674	2358.22	92	2.165
		4	0	584.093	62	3.363	484.798	94	2.690
			0.3	1310.09	62	2.063	1087.375	98	1.651
			0.7	2275.599	62	3.059	1888.747	102	2.447
			1	2998.83	62	1.723	2459.04	100	1.378

Table 13. Comparison Result for Large Size Problems.

Size of problem	*N*	*m*	α	SA			VNS		
				Mean	**CPU time**	**Standard Deviation**	**Mean**	**CPU time**	**Standard Deviation**
Large	**60**	**6**	0	474.470	63	1.227	389.066	102	0.957
			0.3	1144.780	64	1.756	938.720	104	1.370
			0.7	2035.820	65	2.637	1649.014	107	2.057
			1	2704.533	65	1.613	2190.672	102	1.258
		8	0	568.753	78	2.538	460.690	105	1.955
			0.3	1402.602	64	3.417	1136.108	104	2.631
			0.7	2515.782	68	2.882	2012.625	100	2.219
			1	3345.067	63	2.876	2676.053	110	2.215
	80	**2**	0	728.919	62	2.447	583.135	120	1.836
			0.3	1757.249	61	3.165	1405.799	125	2.374
			0.7	3124.668	63	2.570	2437.241	126	1.927
			1	4152.733	61	2.664	3239.132	128	1.998
		4	0	777.856	63	2.261	606.728	132	1.718
			0.3	1802.253	64	3.050	1405.757	137	2.318
			0.7	3166.439	64	2.466	2438.158	139	1.874
			1	4192.467	65	3.014	3228.199	140	2.291
		6	0	617.588	64	2.745	475.543	142	2.114
			0.3	1524.292	63	2.128	1173.705	140	2.064
			0.7	2733.187	65	2.492	2049.890	147	2.193
			1	3639.433	66	3.540	2729.575	145	3.115
		8	0	688.100	64	1.862	516.075	156	1.638
			0.3	1806.956	65	1.902	1355.217	158	1.673
			0.7	3193.614	65	3.151	2395.211	154	2.867
			1	4245.667	67	2.670	3184.250	159	2.429

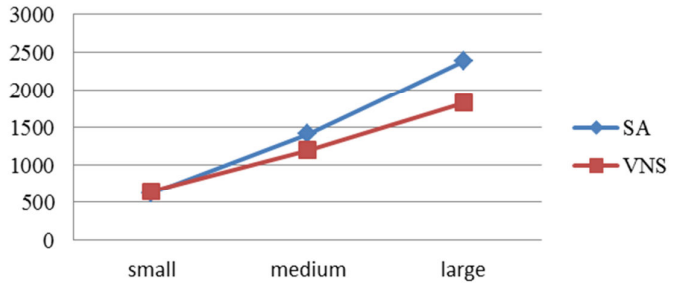

Fig. 7. Comparison Mean of Two Algorithms in Different Size Problem.

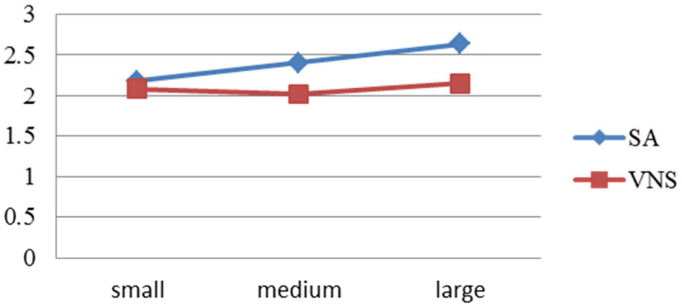

Fig. 8. Comparison Standard Deviation of Two Algorithms in Different Size Problem.

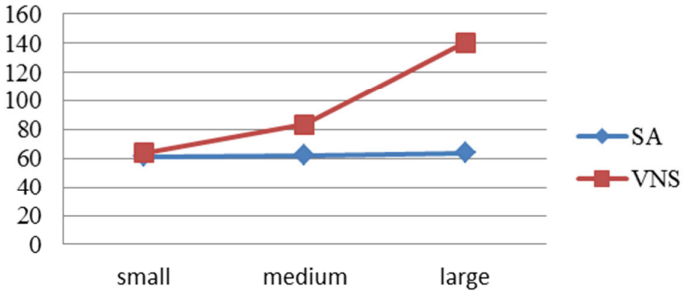

Fig. 9. Comparison CPU Time of Two Algorithms in Different Size Problem.

According to computational results we can find out from these figures that VNS outperforms SA in terms of mean and standard deviation in all of three size problems, but on the contrary about CPU Time SA is better.

6. Conclusion

This paper has presented a more realistic three-stage assembly flow shop scheduling problem with sequence-dependent setup times and no buffer intermediate simultaneously while minimizing weighted mean completion time and makespan. After presentation of the mathematical model, two meta-heuristics algorithms were

proposed as follow: variable neighborhood search (VNS) and simulated annealing (SA). The values of different parameters in the SA algorithm have been tuned based on different sizes of problems using the Taguchi method. A detailed statistical experiment based on the Taguchi experimental design cleared that except F for medium-sized problems, the levels of other parameters of SA have an important role in the efficiency of the addressed algorithm for all size of problems. The extensive computational experiments have been implemented in order to evaluate the performance of the proposed meta-heuristics. They have shown that the VNS outperforms SA in terms of mean and standard deviation. Moreover, the computational time of SA was better than VNS. To extend and make this model more realistic other constraints can be considered and different objective or other types of meta-heuristics would be investigated for solving the problem.

References

1. Gupta, J.N.D.; and Jr. Stafford, E.F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3), 699-711.
2. Koulamas, S; and Kyparisis, G. (2001). The three-stage assembly flowshop scheduling problem. *Computers & Operations Research*, 28(7), 689-704.
3. Lee, C.Y.; Cheng, T.C.E.; and Lin, B.M.T. (1993). Minimizing the makespan in the 3-machine assembly type flowshop scheduling problem. *Management Science*, 39(5), 616-625.
4. Ng, C.T.; Wang, J.B.; and Cheng, T.C.E.; and Liu, L.L. (2010). A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. *Computers & Operations Research*, 37(1), 83-90.
5. Allahverdi, A.; and Al-Anzi, F.S. (2007). The two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time. *International Journal of Advanced Manufacturing Technology*, 37(1), 166-177.
6. Potts, C.N.; Sevast'janov, S.V.; Strusevich, V.A.; Van Wassenhove, L.N.; and Zwaneveld, C.M. (1995). The Two-Stage Assembly Scheduling Problem: Complexity and Approximation, *Operations Research*, 43(2), 346-355.
7. Allahverdi, A; and Al-Anzi, F.S. (2009). The Two-Stage Assembly Scheduling Problem to Minimize Total Completion Time with Setup Times. *Computers & Operations Research*, 36 (10), 2740-2747.
8. Al-Anzi, F.S; and Allahverdi, A. (2009). Heuristics for a two-stage assembly flowshop with bicriteria of maximum lateness and makespan. *Computers & Operations Research*, 36(9), 2682-2689.
9. Cheng, T.C.E.; Lin, B.M.T.; and Tian, Y. (2009). Scheduling of a two-stage differentiation flowshop to minimize weighted sum of machine completion times. *Computers & Operations Research*, 36(11), 3031-3040.
10. Ruiz, R.; and Allaherdi, A. (2009). Minimizing the bicriteria of makespan and maximum tardiness with an upper bound on maximum tardiness. *Computers & Operations Research*, 36(4), 1268-1283.
11. Sun, X.; Morizawa, K.; and Nagasawa, H. (2003). Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. *European Journal of Operational Research*, 146(3), 498-516.

12. Yokoyama, M; and Santos, D.L. (2005). Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times. *European Journal of Operational Research*, 161(3), 754-770.
13. Hall, N.G.; and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no wait in process. *Operations Research*, 44 (3), 510-525.
14. Qian, B.; Wang, L.; Huang, D.X.; Wang, W.; and Wang, X. (2009). An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. *Computers & Operations Research*, 36(1), 209-233.
15. Liu, B.; Wang, L.; and Jin, Y.H. (2008). An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research*, 35(9), 2791-2806.
16. Wang, L.; Zhang, L.; and Zheng, D.Z. (2006). An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research*, 33(10), 2960-2971.
17. Grabowski, J.; and Pempera, J. (2007). The permutation flowshop problem with blocking .A tabu search approach. *Omega*, 35(3), 302-311.
18. Ronconi, D.P. (2004). A note on constructive heuristics for the flowshop problem with blocking .*International Journal of Production Economics*, 87(1), 39-48.
19. Tavakkoli-Moghaddam, R.; Safaei, N.; and Sassani, F. (2009). A memetic algorithm for the flexible flow line scheduling problem with processor blocking. *Computers & Operations Research*, 36(2), 402-414.
20. Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, 31(13), 39-52.
21. Ronconi, D.P.; and Henriques, L.R.S. (2009). Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. *Omega*, 37(2), 272-281.
22. Tozkapan, A.; K,irca, O.; and Chung, C.S. (2003). A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. *Computers & Operations Research*, 30(2), 309-320.
23. Yagmahan, B.; and Yenisey, M.M. (2010). A multi-objective ant colony system algorithm for flow shop scheduling problem. *Expert Systems with Applications*, 37(2), 1361-1368.
24. Sung, C.S.; and Kim, H.K. (2008). A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times. *International Journal of Production Economics*, 113(2), 1038-1048.
25. Yokoyama, Y. (2008). Flow-shop scheduling with setup and assembly operations. *European Journal of Operational Research*, 161(3), 754-770.
26. Liu, S.Q.; and Kozan, E. (2009). Scheduling a flow shop with combined buffer conditions. *International Journal of Production Economics*, 117(2), 371-380.

27. Lee, W.C.; Shiuan, Y.R.; Chen, S.K.; and Wu, C.C. (2010). A two-machine flowshop scheduling problem with deteriorating jobs and blocking. *International Journal of Production Economics*, 124(1), 188-197.
28. Gong, H.; Tang, L.; and Duin, C.W. (2010). A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. *Computers & Operations Research*, 37(5), 960-969.
29. Wang, L.; Pan, Q.K.; Suganthan, P.N.; Wang, W.H.; and Wang, Y.M. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Computers & Operations Research*, 37(3), 509-520.
30. Norman, A.B. (1999). Scheduling flowshops with finite buffers and sequence-dependent setup times. *Computers & Industrial Engineering*, 36(1), 163-177.
31. Hatami, S.; Ebrahimnejad, S.; Tavakkoli-Moghaddam, R.; and Maboudian, Y. (2010). Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology*, 50(9-12), 1153-1164.
32. Maleki-daronkolaei, A.; Modiri, M.; Tavakkoli-moghaddam, R.; and Seyedi-badeleh, S.I. (2012). A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *Journal of Industrial Engineering International*, 8(26), 1-7.
33. Hansen, P.; and Mladenović, N. (1999). *An introduction to variable neighborhood search*. In: S. Voss et al. (Eds.). *Metaheuristics, advances and trends in local search paradigms for optimization*. Kluwer, Dordrecht, 433-458.
34. Hansen, P.; and Mladenović, N. (2001). *Developments of variable neighborhood search*. In: Ribeiro, C.C.; and Hansen, P. (Eds.). *Essays and surveys in metaheuristics*. Kluwer Academic Publishers, Boston/Dordrecht/ London. 415-439.
35. Metropolis, N.; Rosenbluth, A.W.; and Teller, A.H. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1087-1092.
36. Lin, S.W.; Ying, K.C.; and Lee, Z.J. (2009). Meta-heuristics for scheduling a non-permutation flow line manufacturing cell with sequence dependent family setup times. *Computers & Operations Research*, 36(4), 1110-1121.