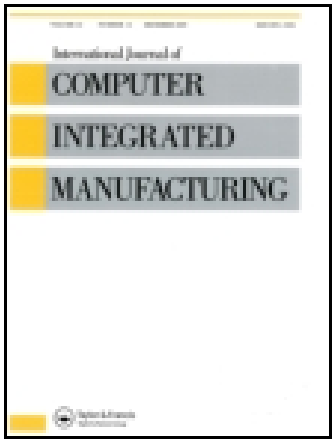
This article was downloaded by: [North Dakota State University]
On: 02 December 2014, At: 22:03
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



International Journal of Computer Integrated Manufacturing

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcim20

Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach

M.E. Tayebi Araghi[a], F. Jolai[b] & M. Rabiee[c]

[a] Department of Industrial Engineering, University of Tehran, Kish International Campus, Kish Island, Tehran, Iran

[b] Department of Industrial Engineering, University of Tehran, Tehran, Iran

[c] Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran

Published online: 20 Sep 2013.

To cite this article: M.E. Tayebi Araghi, F. Jolai & M. Rabiee (2014) Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach, International Journal of Computer Integrated Manufacturing, 27:8, 733-746, DOI: 10.1080/0951192X.2013.834465

To link to this article: http://dx.doi.org/10.1080/0951192X.2013.834465

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the "Content") contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions

International Journal of Computer Integrated Manufacturing, 2014
Vol. 27, No. 8, 733–746, http://dx.doi.org/10.1080/0951192X.2013.834465

Taylor & Francis
Taylor & Francis Group

Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach

M.E. Tayebi Araghi[a], F. Jolai[b]* and M. Rabiee[c1]

[a]*Department of Industrial Engineering, University of Tehran, Kish International Campus, Kish Island, Tehran, Iran;* [b]*Department of Industrial Engineering, University of Tehran, Tehran, Iran;* [c]*Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran*

(*Received 8 August 2012; accepted 19 July 2013*)

This paper considers flexible job-shop scheduling (FJSP) where set-up times were sequence-dependent, learning effect and deterioration in jobs are concurrently considered too. The optimisation criterion is the minimisation of the maximum completion time. Since the genetic algorithm (GA) and the variable neighbourhood search (VNS) are successfully applied to some NP-hard problems, we have been motivated to employ both of them to solve the considered FJSP. To further enhance and improve the hybrid algorithm, a well-known procedure called affinity function is used in our proposed algorithm genetic variable neighbourhood search with affinity function (GVNSWAF). In order to evaluate the performance of the proposed algorithm, some experiments are designed where the proposed method is compared against some new and successful algorithms found in the literature (i.e. GA and VNS). In view of the fact that the performances of meta-heuristic algorithms are considerably influenced by the proper setting of their parameters, we employed Taguchi's robust design method to define the best parameter values. Related results are analysed by statistical tools. The experimental results and statistical analyses demonstrate that the proposed GVNSWAF is effective for the problem.

Keywords: flexible job-shop scheduling problem; sequence-dependent set-up time; learning effect; deterioration; Taguchi experimental design

Downloaded by [North Dakota State University] at 22:03 02 December 2014

1. Introduction

Scheduling is one of the most prominent issues in the short-term planning for manufacturing systems and service companies. Many kinds of scheduling problems are among the complex combinatorial optimisation problems and are thus very difficult to solve and this type of optimisation problems have been recently considered by researchers (Desprez, Chu, and Chu 2009; Luo et al. 2010; Mohammadi, Ghomi, and Jafari 2011; Solimanpur and Elmi 2011). One of the most complicated problems in this area is the job-shop scheduling problem (JSP). JSP deals with sequencing the operation of a set of jobs on a set of machines by optimising a criterion or some criteria. The main assumption of the problem (which differs from that of the flexible JSP or FJSP) is that each operation of a job is to be processed only on one predetermined machine, i.e. route of operations of each job is unique (Wang and Yu 2010). In FJSP, each machine may be capable of performing more than one type of operations, i.e. for a given operation, there is at least one machine capable of performing it. FJSP has been highly concentrated on by investigators in recent years (Fattahi, Saidi, and Jolai 2007; Fattahi, Jolai, and Arkat 2009; Amiri et al. 2010; Bagheri et al. 2010; Wang and Yu 2010; Rabiee, Zandieh, and Ramezani 2012).

The problem of scheduling jobs in FJSP could be decomposed into two sub-problems: a routing sub-problem, which consists of assigning each operation to a machine out of a set of capable machines, and a scheduling sub-problem, which consists of sequencing the assigned operations on all selected machines in order to attain a feasible schedule with optimised objectives (Xia and Wu 2005). In this literature, two approaches have been employed for solving the FJSP. The first one is the hierarchical approach and second one is the integrated approach.

In hierarchical approach, the assignment of operations to machines and the sequencing of operations on the machines are considered separately, i.e. assignment and sequencing are considered independently. In integrated approach, assignment and sequencing are not differentiated.

The JSP has been proven to be NP-hard (Garey, Johnson, and Sethi 1976) and thus computationally intractable. So, FJSP is NP-hard because it is an extension of JSP. The NP-hardness of an optimisation problem reminds us that finding an optimal solution in a satisfactory computational time is not always possible (Mastrolilli and Gambardella 2000). Hence, many approaches, especially meta-heuristic algorithms (e.g. tabu search (TS), simulated annealing (SA), beam search (BS), variable

*Corresponding author. Email: fjolai@ut.ac.ir
[1]Current affiliation: Department of Industrial Engineering, Tuyserkan's Engineering Faculty, Bu–Alisina University, Hamedan, Iran

© 2013 Taylor & Francis

Downloaded by [North Dakota State University] at 22:03 02 December 2014

neighbourhood search (VNS) and genetic algorithm (GA) etc.) have been employed to solve this problem.

Brucker and Schlie (1990) and Brandimarte (1993) published the earliest papers on FJSP. Brucker and Schile (1990) developed a polynomial graphical algorithm for a two-job problem. Brandimarte (1993) used a hierarchical approach (i.e. decomposition) for the FJSP. He solved the assignment problem using some existing dispatching rules and then focused on the resulting job-shop sub-problems, which were solved using a TS heuristic. This approach is followed by Paulli (1995) and Chambers and Barnes (1998). In the integrated approaches, the assigning sub-problem and the sequencing sub-problem are solved simultaneously. Integrated approach is much more difficult, but in general renders better results, as reported in Dauzére-Pérés and Paulli (1997), Hurink, Jurisch, and Thole (1994) and Mastrolilli and Gambardella (2000).

Dauzère-Pérès and Paulli (1997) defined a new neighbourhood structure for the problem, where there is no distinction between reassigning and re-sequencing an operation, and proposed a TS algorithm based on the integrated approach. Mastrolilli and Gambardella (2000) improved Dauzère-Pérès TS techniques and presented two neighbourhood functions. Chen, Ihlow, and Lehmann (1999) developed a GA for FJSP. They split the chromosome representation into two parts, the first one defining the routing policy, and the second one delineating the sequence of operations on each machine. Kacem, Hammadi, and Borne (2002a, 2002b) proposed a localisation approach to solve the resource assignment problem, and an evolutionary approach controlled by the assignment model for FJSP.

Mastrolilli and Gambardella (2000) proposed some neighbourhood functions for the FJSP. Pezzella, Morganti, and Ciaschetti (2008) integrated different strategies for generating the initial population, selecting the individuals for reproduction, and reproducing new individuals. Fattahi, Saidi, and Jolai (2007) proposed a mathematical model and two meta-heuristic algorithms (SA and TS), for solving FJSP. Moreover, considering two developed meta-heuristics and concerning the integrated and hierarchical approaches, they presented six different algorithms.

Ho, Tay, and Lai (2007) proposed an architecture where a learning module is introduced into an evolutionary algorithm for solving FJSSP. The makespan is thus minimised and the authors show that their method outperforms the existing ones (Brandimarte 1993; Kacem, Hammadi, and Borne 2002a) for some instances mentioned in the literature. Rajkumar, Asokan, and Vamsikrishna (2010) solved FJSSP with preventive maintenance (non-fixed availability) constraints using a greedy randomised adaptive search procedure (GRASP) algorithm and compared their results with those of hybrid GA (hGA).

Bagheri and Zandieh (2011) proposed a VNS algorithm for the FJSP with sequence-dependent set-ups. Their algorithm was aimed at minimising an aggregate objective function of makespan and mean tardiness. They compared VNS with parallel variable neighborhood search (PVNS) (Yazdani, Zandieh, and Amiri 2009) and GA (Pezzella, Morganti, and Ciaschetti 2008). Results of their study showed that the proposed VNS outperformed PVNS and GA.

In real manufacturing systems, many assumptions can be considered, but in most of studies of this field, investigators tried to solve the problems with a few practical assumptions. In contrast to the existence of many research results on FJSP, there has not been any attempts to study the type of the FJSP that concurrently involves sequence-dependent set-up times, learning effect and deterioration.

Machine set-up time is an important factor for production scheduling in all flow type manufacturing environments. Set-up times can be either sequence-independent or sequence-dependent. Sequence-independent set-up times can be ignored or added to the processing times. As for the sequence-dependent set-up times (SDST), the value of the set-up time for each job on a specific machine depends on the previous job operated on the same machine.

Production scheduling problems with deteriorating jobs and/or learning effects have been given more attention in recent years. Extensive surveys related to scheduling deteriorating jobs can be found in Alidaee and Womer (1999) and Cheng, Ding, and Lin (2004). An extensive survey of different scheduling models and problems involving jobs with learning effects can be found in Biskup (2008). More recent papers that have considered scheduling jobs with deteriorating jobs and/or learning effect include Wu, Lee, and Shiau (2007), Shiau et al. (2007) and Eren and Güner (2009).

Review of the studies reveals that the FJSP with the four realistic assumptions addressed above has not been studied yet. Hence, in this paper, we propose a novel, hybrid, efficient meta-heuristic algorithm for solving a FJSP, taking into account SDST, partial flexibility, learning effect and deterioration and aiming at minimising maximum completion time. The remaining sections of this paper are organised as follows. In Section 2, the problems with their respective assumptions are described. Section 3 presents the structure of the proposed meta-heuristic algorithm. In Section 4, we present data generation, parameter tuning and the evaluation of the performance of algorithms. Finally, in Section 5, the results of the study are presented and discussed. This is followed by giving concluding remarks and proposing potential areas for further research.

2. Problem definition

The problem is to organise the execution of n jobs, noted J, $J = \{J_1, ..., J_n\}$, with each job J_i consisting of a sequence of n_i operations, $O_{i,1}, O_{i,2}, ..., O_{i,n_i}$ on a given machine from a machine set named $M = \{M_1, ..., M_m\}$.

In general, there are two kinds of FJSP, i.e. total FJSP problem (TFJSP) and partial FJSP (PFJSP). For the TFJSP, each job can be operated on every machine

Downloaded by [North Dakota State University] at 22:03 02 December 2014

from the set M; for the PFJSP, there is a problem constraint for the operating process, namely, one operation of a job must be processed by a set of machines in $M' \subseteq M$. The detailed definition of the FJSP is as follows (Liouane et al. 2007; Saidi-mehrabad and Fattahi 2007):

Moreover, we shall assume the following conditions:

- All machines are available at time 0;
- All jobs are released at time 0;
- The order of operations for each job is predefined and cannot be modified;
- The processing time of an operation $O_{i,j}$ on machine k (i.e. O_{ij}^k) is predefined and denoted by p_{ij}^k and starting time for operation $O_{i,j}$ on machine k denoted by t_{ij}^k;
- Each operation cannot be interrupted during its performance (non-pre-emptive condition).
- Each machine can perform at most one operation at any time (resource constraint).
- The precedence constraints of the operations in a job can be defined for any pair of operations and
- The applied position-dependent learning effect model (Biskup 1999) is as follows:

$$S_{ijk}^{'} = S_{ijk} \times r^{\alpha} \tag{1}$$

where $S_{ijk}^{'}$ is actual SDST for jth operation of ith jobs on kth machine and S_{ijk} is normal SDST. Also, r is position of jth operation in operation sequence and $\alpha$is non-positive learning index.

- The applied time-dependent deterioration rate is as follows:

$$P_{ijk}^{'} = P_{ijk} \times (1+\beta)^{\sum P_{ijk}} \tag{2}$$

where $P_{ijk}^{'}$ is actual processing time for jth operation of ith jobs on kth machine and P_{ijk} is normal processing time. Also, β is non-negative deterioration index.

3. Genetic variable neighbourhood search with affinity function (GVNSWAF)

GAs have a wide range of applications in solving scheduling and other operational research problems (Ponnambalam, Ramkumar, and Jawahar 2001; Mansouri, Moattar-Husseini, and Zegordi 2003; Hao et al. 2005; Lei 2009). They are better with respect to the objective of interest and are especially suitable when the search space is unknown (Vahdani and Zandieh 2010). Meta-heuristic algorithms are utilised to obtain the best sequence of jobs and the best combination of machine assignments that minimise makespan. This is known as a solution of the problem. At the beginning of describing of our proposed meta-heuristic algorithm, a representation scheme of solutions in our proposed algorithm and benchmark algorithms is elaborated in Subsection 3.1. Also other segments of our proposed algorithm are described in the following subsections.

3.1. *Solution representation*

The scheduling of orders in a job shop is a multifaceted problem (Uckun et al. 1993). In order to represent the chromosome of the problem, a vector of integer numbers with a length of two times the total number of operations is randomly generated. It is formed in two parts. The first part represents operations sequence and the second one represents the machines that are assigned to the operations in symmetric order.

The first part is a vector of random values that is generated discretely in a range between 1 and the total number of operations. In the second part, for each operation, a number between 1 and the number of available machines for the related operation is generated. For simplifying the representation of the solution, it is assumed that there is a schedule with four jobs and four machines. Jobs 1, 2, 3 and 4 have two, three, one and four operations, respectively. Hence, there are a total of 10 operations which will be scheduled on four machines. Further information about this problem is shown in Figure 1. Figure 2 depicts a solution for this problem. It is clear that in this solution, precedence constraint between

Jobs	1		2		
operations	1	2	3	4	5
Available machines	M_1-M_2-M_4	M_2-M_3-M_4	M_1-M_2-M_3-M_4	M_1-M_2-M_3	M_2-M_3
Jobs	3	4			
Operations	6	7	8	9	10
Available machines	M_2-M_3-M_4	M_1-M_2-M_3-M_4	M_1-M_3-M_4	M_1-M_2-M_3	M_1-M_3

Figure 1. An example with four jobs and 10 operations and four machines.

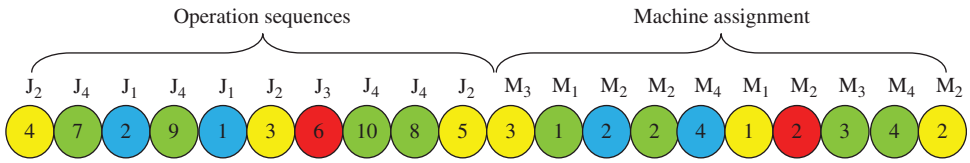


Figure 2. An example of solution representation.

operations of each job is not considered. For solving this problem, this solution requires a repair procedure. In this procedure, operations related to each job change their positions based on their priority in precedence constraints. Similar to the first part of the chromosome, the second part is repaired too. Repaired chromosome for this solution is demonstrated in Figure 3.

3.2. Initial population

GVNSAF requires a population of individuals which is the potential solution to a given problem. The initial population of individuals is randomly generated as a of population size (popsize). In other words, we have a primary population which is called pop1 (i.e. pop1= popsize).

3.3. Objective evaluation

In this method, an objective function value is assigned to the chromosome as soon as it is generated. In optimisation problems, fitness function is the value of the objective function. In this study, the objective function for a chromosome is defined as minimising the maximum completion time.

3.4. Selection

xSelections of chromosomes in GVNSAF are performed for the crossover procedure and neighbourhood procedure. The momentous purpose of this operator is to choose the better chromosomes among the individuals in iteration, in order to reach better results. For this purpose, chromosomes among the individuals are chosen to create the offspring in the crossover procedure. There are many approaches to this selection procedure, such as roulette wheel selection, tournament selection and elitism. In this paper, the tournament selection procedure has been used for selecting the chromosome for crossover and neighbourhood mechanism. In this procedure, the first M numbers for the population size are selected and then the corresponding fitness functions are calculated. Afterwards, the chromosomes with the minimum cost among the M members are chosen.

3.5. Crossover

Crossover is the process of taking two parent chromosomes from the mating pool and producing offspring by combining them, thus aiming to find better solutions. In this paper, we use uniform crossover and one-point crossover for our proposed algorithm. This operator is one of the popular operators which have been studied in many researches (Jeong, Lee, and Cho 2005; Moon, Lee, and Bae 2008). This operator first produces a random integer in the range (1, (total operations) − 1), then the right-hand side part of chromosome one is transferred to the right-hand side of chromosome two and the left-hand side part of chromosome two transferred to the left-hand side of chromosome one. Also, for the second section, a random number in the range (total operations + 1, 2 (total operations) − 1) is generated and an implementation similar to operation sequences is performed. In this procedure, three scenarios can be considered and chosen randomly for the crossover in each generation. These are as follows:

- Scenario one: based on this scenario, crossover that is carried out only in the first section is called operation sequence (refer Figure 2).
- Scenario two: based on this scenario, crossover that is performed only in the second section is called machine sequence (refer Figure 2).

Crossover operator. The goal of the crossover operator is to obtain better chromosomes to improve the result by exchanging information contained in the current good ones.

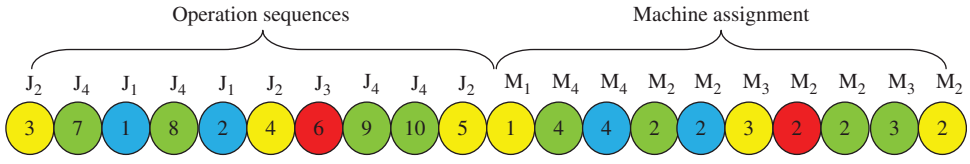


Figure 3. An example of repaired solution representation.

Downloaded by [North Dakota State University] at 22:03 02 December 2014

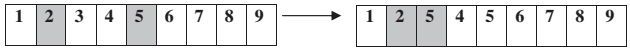
Downloaded by [North Dakota State University] at 22:03 02 December 2014

In this study, two kinds of crossover operator are carried out for machine selection (MS) and job sequence (JS).

Job sequence part. The each value in the JS is equal to the index of each job.

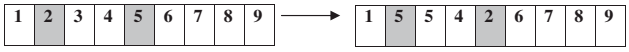
Insertion

- Pick two allele values at random.
- Move the second to follow the first, shifting the rest along to accommodate.
- Note that this preserves most of the order and the adjacency information.



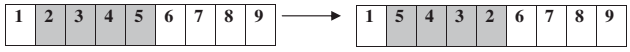
Swap

- Pick two alleles at random and swap their positions.
- Preserves most of adjacency information (four links broken), disrupts order more.



Inversion

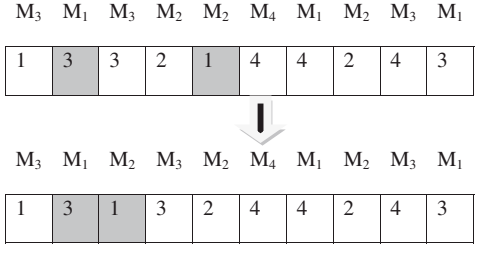
- Pick two alleles at random and then invert the sub-string between them.
- Preserves most adjacency information (only breaks two links), but disruptive of order information.



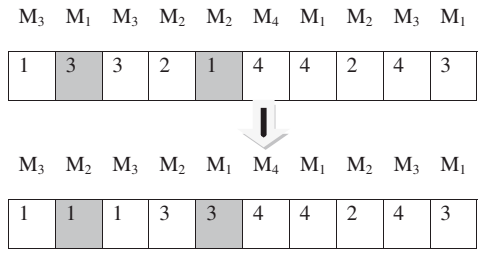
In machine assignment, a problem that may occur is that assigning the machine that is not available for job; for solving this problem, we used the random selection of available machine without last machine assigned to the job for remain job.

Example:

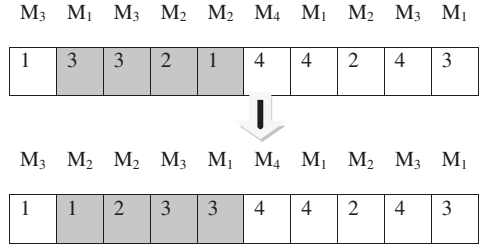
Insertion



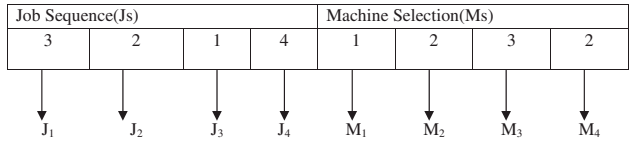
Swap



Inversion



For example:



- Available machines for j_1 is [m_1, m_3, m_4]
- Available machines for j_2 is [m_2, m_3, m_4]
- Available machines for j_2 is [m_1, m_3, m_4]
- Available machines for j_2 is [m_1, m_3]

Machine selection part. The crossover operator of MS only performs on machine selection parts. In order to ensure all feasible individuals after crossover operation, the two-point crossover operator was adopted in this paper.

Mutation operator. Mutation usually works on a single chromosome and creates another chromosome through alteration of the value of a string position or exchange of the values of two string positions, in order to maintain the diversity of population.

MS part. MS mutation operator only changes the MS part of the chromosome. The gene selected randomly is replaced by the machine that its processing time is the shortest processing time from its alternative machine.

JS part. JS mutation operator only acts on the JS part and the mutation probability is equal to the MS mutation probability (refer to job crossover).

- Scenario three: crossover is carried out in both parts simultaneously.

Downloaded by [North Dakota State University] at 22:03 02 December 2014

It is obvious that the solutions generated by this procedure may require repairing procedures. The chromosomes obtained through crossover operation have been called pop2 in this paper.

3.6. Neighbourhood structure

For neighbourhood operation, we used a VNS algorithm, which is a well-known local search method for solving combinatorial optimisation problems. We employed this method instead of mutation in GA. VNS with the capability of utilising different neighbourhood structures explores increasingly distant neighbourhoods of the current incumbent and supplants a new solution if and only if the new one has a better quality (Hansen and Mladenovic 2001).

For better exploitation of the solution space in different regions, we use five types of neighbourhood structures, denoted by N_k (k = 1, 2, …, k_{max}). In the main step of the algorithm, the stopping condition is optional (In this case, we set it as a maximum number of iterations or l_{max}). With the aim being further improvement in the quality of the solutions, we considered the following five neighbourhood structures. It is noticeable that these mutation operators are applicable for the two parts of the chromosome in the following ways:

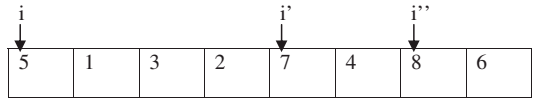
- *Swap*: In this case, mutation/neighbourhood is performed based on exchange mutation; two different arbitrary genes of the parent chromosome are chosen and their allele values swapped (Dell'Amico and Trubian (1993); Nowicki and Smutnicki (1996).
- *Inversion*: based on this operator, two genes are randomly chosen each from one of the two parts of the chromosome; then the values of their positions are exchanged. In addition, the sequences of genes between these two selected positions are inversed.
- *Insertion*: in this operator, similar to the inversion operator, first, two positions on the chromosome are randomly selected; then the value of the selected position in the right-hand side of the solution is inserted to the left-hand side of the other chosen position and the rest of the genes maintain their sequence.
- *Three-point heuristic*: in this neighbourhood search strategy, at first three operations in the permutation are randomly selected and then all possible permutation of these operations are performed. After this, the new solution with the highest fitness (minimum objective function) among the three produced solutions is selected.
- *Four-point heuristic*: this method is quite similar to the previous neighbourhood search strategy, but here the number of randomly selected points is four. Similar to the crossover operation, this operation can be performed in three scenarios. After producing a new solution, and similar to the crossover, it may require a repairing procedure. The chromosomes obtained from the neighbourhood procedure are called pop3 here.

Kick technique It reduces the occurrence of fragmented random structures by applying a coalescence procedure to each fragment (i.e. simultaneous push of each fragment towards the centre of mass).

Three-point and four-point To complete the procedure for this crossover (three-point crossover), we must select multiple positions (i.i′, i″) and generate results for all swapped positions (in this example, we have 3! positions); then we must choose the best child for the next generation.

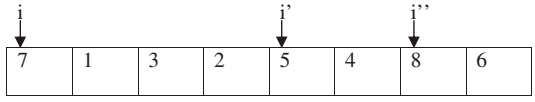
Example:

Parent 1:

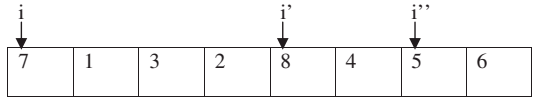


Then we get six children:

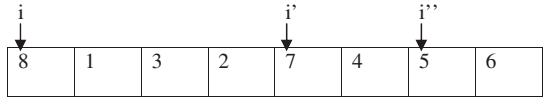
Child 1:



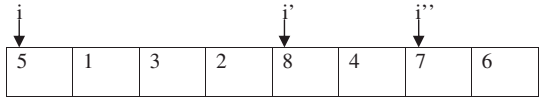
Child 2:



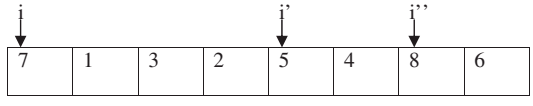
Child 3:



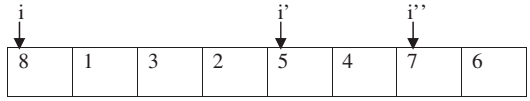
Child 4:



Child 5:



Child 6:



Downloaded by [North Dakota State University] at 22:03 02 December 2014

3.7. *Affinity function*

Affinity function was used for avoiding from premature convergence and increasing the diversification. Affinity function allows us the generated solutions with high diversity. We consider a parameter called percentage of affinity, which is denoted P_{AF} for defining the percentage of good sorted solutions, which remained at each iteration, and then the remained capacity of the population is selected from unique solutions existing among the present solutions. If unique solutions were not enough for filling the remained capacity of population, we have to use the repetitive solutions as follows:

First, we merge the population of mutation and crossover with the existing population and then separate PAF per cent of the unique individual results for the next generation; this method is used for prevention of premature convergence. The stochastic universal sampling (SUS) selection uses a single random value to sample all of the solutions by choosing them at evenly spaced intervals. This gives weaker members of the population (according to their fitness), a chance to be chosen and thus reduces the unfair nature of fitness-proportional selection methods.

3.8. *Merge*

We merge pop1, pop2 and pop3, and then fitness functions of them are calculated and sorted. After that by using the sorted solutions take npop solutions with considering the affinity function. Finally, this population is used in the next iteration as initial population.

A prominent feature of the GAVNS hybrid algorithm is the use of affinity function, as a result of which we separate PAF per cent of the unique, individual results (and not just good results). As seen in Figure 4, after some repetition of the algorithm, the results of GAVNS hybrid algorithm move away from the optimum local corner and towards the optimum point. In contrast, in GA standard algorithm and VNS algorithm, due to the affinity function not being used, the results remain in local optimum corners and thus remain inferior to those obtained through the GAVNS hybrid algorithm.

3.9. *Termination criterion*

The algorithm continues until all of the maximum iterations are satisfied. The framework of our proposed algorithm is illustrated in Figure 5.

In pseudocode, we used the selection (*select* n_{pop} *chromosome with considering affinity function*;) of the results by the affinity function is carried out from among all results pop1, pop2, and pop3 (whether good or bad), at a stable rate. This procedure prevents premature convergence.

4. Experimental evaluation

In this section, some experiments which are performed for the purpose of assessment of effectiveness and competitiveness of applied algorithms are presented. We test our proposed algorithm against recent successful reported meta-heuristic algorithm in literature including GA (Pezzella, Morganti, and Ciaschetti 2008) and VNS algorithm (Amiri et al. 2010).These algorithms belong to most popular evolutionary algorithm. Also, the computational tests which are performed to evaluate the effectiveness and efficiency of algorithms are described. Experimental tests are divided in two groups including small- and large-scale problems. The procedure of data generation is described in the next subsection.

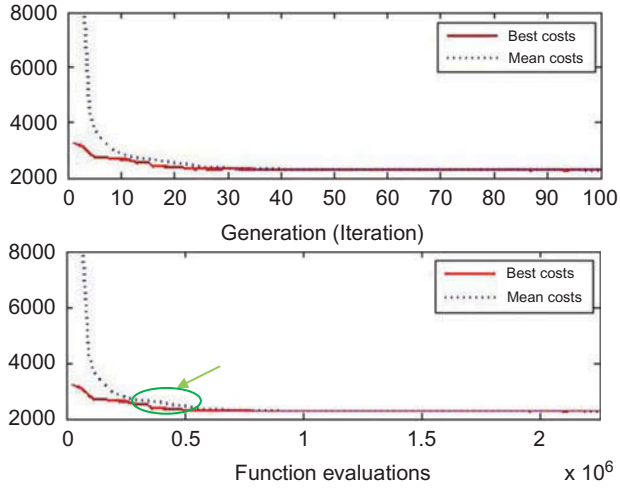


Figure 4. Problem 40-40-10 GAVNS algorithm generation.

4.1. *Data generation*

In this section, for compensating the lack of benchmark problems in considered FJSP, it is forced us to use random test problems to applied algorithms. In order to evaluate algorithms, 15 instances in small-scale and 15 random instances in large-scale problems are generated. Number of jobs is classified to two scales, namely small and large. In each scale, five jobs are considered. Also for each job, three combinations of maximum operation (OpMax) and machines are considered which illustrated in Table 1. The processing times are generated by the uniform distribution ranging [1, 50]. The duration of SDST, as same as processing times, is generated by the uniform distribution ranging [1,25]. Also the generated instances have partial flexibility. For implementation of partial flexibility, a variable ratio between 70% and 90% is defined. So, 10–30% of machines per operations are not available for performing the operations. Moreover, multiplier of learning effect (α) and deterioration (β) are produced by a uniform distribution function, as maximum impact of learning effect is 0.5 and maximum impact of deterioration is 1.5. Further information about random test problems is shown in Table 1. It is noticeable that totally there are 30 class problems. We

Downloaded by [North Dakota State University] at 22:03 02 December 2014

```
Begin
Input : MaxIt, n_pop, M, P_C, P_NS, P_AF, L_max
Generate Pop1;
Repair Pop1;
Evaluate fitness values of the Pop1;
for i = 1 to MaxIt do
    for j = 1 to round [(P_C × n_pop)/2]
        Select two individuals: (X_1, X_1);
        Randomly select one scenario for crossover operation;
        Randomly select uniform crossover or one point crossover;
        Perform selected crossover : (X_1, X_2) → (X_1', X_2') ;
        Repair Pop2;
        Evaluate fitness values of the Pop2;
    endfor
    for j  1 to round[(P_NS × n_pop) / 2]
        Select an individual: X ;
      for L = 1 to L_max
          for K = 1 to k_max
            if f (X_LK ) < f (X_Best ) then
              X_Best = X_LK ;
              Break ;
            endif
          endfor
      endfor
        Repair Pop3;
        Evaluate fitness values of the Pop3;
    endfor
    Merge all of population {Pop1 ∪ Pop2 ∪ Pop3};
    Select n_pop chromosome with considering affinity function;
end for
Output : Extract the best solution with minimum completion time;
end
```

Figure 5. Pseudo code of GVNSWAF.

generated 10 random problems in each class, i.e. we have 300 test problems for comparison among the algorithms.

4.2. Parameter setting

Taguchi's design of experiment (TDOE) approach offers a way for the researcher to determine, before the runs are made, which specific configurations to simulate so that the desired information can be obtained with the least amount of simulation (Kelton and Law 2000). It is very effective to deal with responses influenced by many factors. The difference between a traditional full factorial design of experiment and TDOE is the significant reduction in the size of the experiment. Taguchi method involves the determination of a large number of experimental situations, described as orthogonal arrays, to reduce errors and enhance the efficiency and reproducibility of the experiments. Orthogonal arrays are a set of tables of numbers, which can be used to efficiently accomplish optimal experimental designs by considering a number of experimental situations (Roy 2001). Taguchi method uses signal-to-noise ratio (SNR), instead of the average value to interpret the data (experimental results) into a value for the characteristic which is being evaluated, within the optimum setting analysis (Ross 1996). This is because the SNR can reflect both the average and the variation of the quality characteristic.

Taguchi categorises objective functions into three groups: the smaller-the-better type, the larger-the-better type and nominal-is-the best type. Since almost all objective functions in scheduling are categorised in the smaller-the-better type, its corresponding SNR (Phadke 1989) is:

$$\mathrm{SNR} = -\log_{10}\left(\frac{1}{n}\sum_{i=1}^{n}(\text{objective function})_i^2\right) \qquad (3)$$

Before calibration of GVNSWAF, the algorithm is subjected to some preliminary tests to obtain the proper parameter levels to be tested in the fine-tuning process.

In order to achieve more accurate and stable results for our proposed algorithm, we considered seven parameters for tuning. These parameters are MaxIt, npop, M, P_C, P_{NS}, P_{AF} and L_{max}. These are shown, along with their levels, in Table 2. The resulting orthogonal array with seven factors and three levels in Taguchi method is L_{27}. The orthogonal array L_{27} is presented in Table 3.

By analysing all of experimental results using Taguchi method, the average SNR and average maximum completion time were obtained for the considered experiments. Figure 6 displays the average SNR obtained. So, as

Table 1. Factors and their levels.

Factors	Levels
Number of jobs (N)	Small: 5, 10, 15, 20 and 25 Large: 40, 60, 80, 100 and 120
Combination of maximum operation and number of machines ($OpMax \times M$)	Small: 5×2 , 10×3 , 15×4 Large: 20×6 , 30×8 , 40×10
Processingtimes ($P_{i,j}$)	U(1, 50)
SDST (S_{jki})	U(1, 25)

Downloaded by [North Dakota State University] at 22:03 02 December 2014

Table 2. Parameters and their levels.

	Parameters						
	A	B	C	D	E	F	G
Level	MaxIt	n_{pop}	M	P_C	P_{NS}	P_{AF}	L_{max}
Low (1)	80	40	2	0.2	0.1	0.3	6
Medium (2)	100	50	3	0.3	0.2	0.4	8
High (3)	120	60	4	0.4	0.3	0.5	10

illustrated in Figure 6, optimal levels are A(3), B(3), C(2), D(2), E(2), F(3) and G(1). Furthermore, results computed in terms of mean of makespan in Taguchi experimental analysis confirmed the optimal levels obtained using SNR (see Figure 7). Table 4 exhibits the effectiveness rank of factors in minimising makespan.

4.3. Computational evaluation

In this section, performances of the suggested GVNSWAF, GA and VNS are evaluated. All the proposed algorithms were implemented in MATLAB 7.9 and run on 6 parallel PCs with Pentium IV processors @ 2.53 GHZ and 1GB memory.

Algorithm comparison is performed using a usual performance measure, which is known as relative percentage deviation (RPD). The best solutions for each instance are then obtained. *RPD* is computed by the formula given below:

$$\mathrm{RPD} = \frac{\mathrm{alg_{sol}} - \mathrm{min_{sol}}}{\mathrm{min_{sol}}} \times 100 \qquad (4)$$

Table 3. The orthogonal array L_{27}.

Experiments	A	B	C	D	E	F	G
1	A(1)	B(1)	C(1)	D(1)	E(1)	F(1)	G(1)
2	A(1)	B(1)	C(1)	D(1)	E(2)	F(2)	G(2)
3	A(1)	B(1)	C(1)	D(1)	E(3)	F(3)	G(3)
4	A(1)	B(2)	C(2)	D(2)	E(1)	F(1)	G(1)
5	A(1)	B(2)	C(2)	D(2)	E(2)	F(2)	G(2)
6	A(1)	B(2)	C(2)	D(2)	E(3)	F(3)	G(3)
7	A(1)	B(3)	C(3)	D(3)	E(1)	F(1)	G(1)
8	A(1)	B(3)	C(3)	D(3)	E(2)	F(2)	G(2)
9	A(1)	B(3)	C(3)	D(3)	E(3)	F(3)	G(3)
10	A(2)	B(1)	C(2)	D(3)	E(1)	F(2)	G(3)
11	A(2)	B(1)	C(2)	D(3)	E(2)	F(3)	G(1)
12	A(2)	B(1)	C(2)	D(3)	E(3)	F(1)	G(2)
13	A(2)	B(2)	C(3)	D(1)	E(1)	F(2)	G(3)
14	A(2)	B(2)	C(3)	D(1)	E(2)	F(3)	G(1)
15	A(2)	B(2)	C(3)	D(1)	E(3)	F(1)	G(2)
16	A(2)	B(3)	C(1)	D(2)	E(1)	F(2)	G(3)
17	A(2)	B(3)	C(1)	D(2)	E(2)	F(3)	G(1)
18	A(2)	B(3)	C(1)	D(2)	E(3)	F(1)	G(2)
19	A(3)	B(1)	C(3)	D(2)	E(1)	F(3)	G(2)
20	A(3)	B(1)	C(3)	D(2)	E(2)	F(1)	G(3)
21	A(3)	B(1)	C(3)	D(2)	E(3)	F(2)	G(1)
22	A(3)	B(2)	C(1)	D(3)	E(1)	F(3)	G(2)
23	A(3)	B(2)	C(1)	D(3)	E(2)	F(1)	G(3)
24	A(3)	B(2)	C(1)	D(3)	E(3)	F(2)	G(1)
25	A(3)	B(3)	C(2)	D(1)	E(1)	F(3)	G(2)
26	A(3)	B(3)	C(2)	D(1)	E(2)	F(1)	G(3)
27	A(3)	B(3)	C(2)	D(1)	E(3)	F(2)	G(1)

where alg_{sol} is the makespan value, obtained for a given algorithm in a given instance, and min_{sol} is the best value obtained throughout. It is clear that for each algorithm, the closer its upper and lower limits are to zero and the less overlapping with other algorithms it is, the more proper solutions we would expect it to yield.

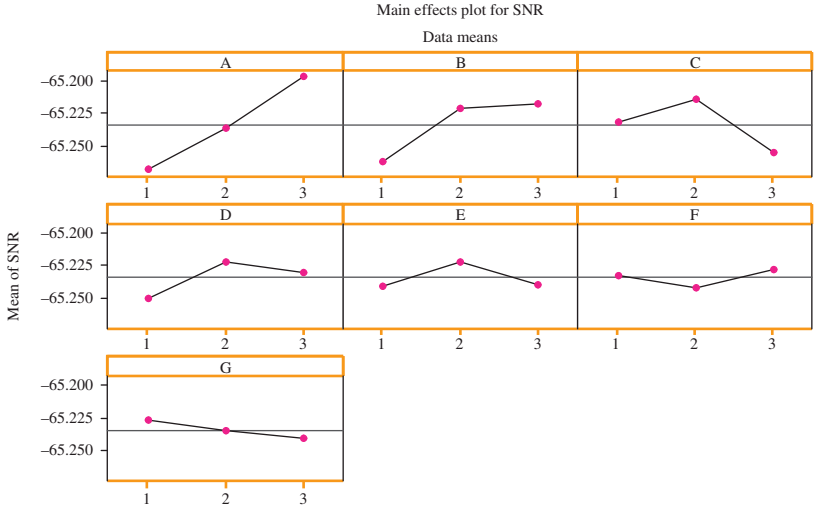


Figure 6. The SNR plot for experiments in Taguchi methodology.

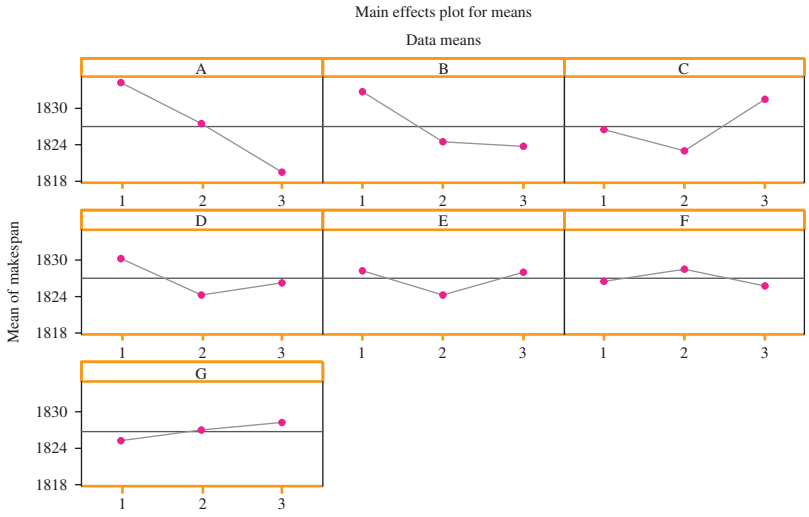


Figure 7. The plot of means of makespan for experiments in Taguchi methodology.

Table 4. SNR table for experiments.

Level	A	B	C	D	E	F	G
1	−65.27	−65.26	−65.23	−65.25	−65.24	−65.23	−65.23
2	−65.24	−65.22	−65.21	−65.22	−65.22	−65.24	−65.23
3	−65.2	−65.22	−65.26	−65.23	−65.24	−65.23	−65.24
Delta	0.07	0.04	0.04	0.03	0.02	0.01	0.01
Rank	1	2	3	4	5	7	6

As mentioned before, in order to evaluate performances of suggested algorithms, a number of random problems in two types of small and large size are generated. After running the algorithms, we converted raw data to RPD. RPD average values (i.e. average relative percentage deviation or ARPD), for small and large problems, are shown in Tables 5 and 6, respectively. In each type of problems, a value of the algorithm, which produced the better result, is bolded.

To signify the differences between algorithms with regard to statistical analyses, 95% confidence intervals for computed values in RPD are given for both types of the problem. Figures 8 and 9 illustrate 95% confidence intervals for the RPD related to each algorithm in small and large sizes, respectively. As it can be seen in both Figures 8 and 9, there are statistically significant differences between GVNSWA and singular approaches, including GA and VNS. Because there is not any overlapping between the upper and lower limits and GVNSWAF is closer to zero, hence GVNSWAF is the better algorithm in comparison with GA and VNS. For more statistical examination, a one-way ANOVA is employed for the both scales. Results of ANOVA indicated that there are statistically significant differences between mean values of algorithms (see Tables 7 and 8).

In order to further scrutinise the algorithms, a sensitivity analysis for ARPD values by considering variations of number of jobs is shown in Figure 10. According to Figure 10, GVNSWAF has a more proper behaviour and produces maximum job completion times with minimum values in all of the problems. In addition, interaction

Table 5. ARPD for the algorithms in small scale.

Job*OpMax*Machine	GA	VNS	GVNSWAF
5*5*2	0	0	**0**
5*10*3	2.401434	1.290323	**0.430107527**
5*15*4	5.296833	3.649109	**1.016912338**
10*5*2	3.14933	1.437996	**0.316455696**
10*10*3	1.688738	1.478757	**0.335937417**
10*15*4	3.130069	1.571185	**0.328173145**
15*5*2	2.315699	1.099958	**0.146755674**
15*10*3	3.945836	2.347377	**0.417149479**
15*15*4	4.388192	2.637472	**0.165662651**
20*5*2	3.840567	2.116419	**0.257009346**
20*10*3	4.44769	1.822527	**0.040983607**
20*15*4	2.103987	1.147794	**0.012399256**
25*5*2	2.035593	0.86905	**0.093095423**
25*10*3	1.956909	1.151666	**0.07800312**
25*15*4	5.67007	3.340472	**0.19791094**
Total	3.091396	1.730674	**0.255770374**

Downloaded by [North Dakota State University] at 22:03 02 December 2014

Table 6. ARPD for the algorithms in large scale.

	GA	VNS	GVNSWAF
40*20*6	3.537504	1.066952	**0.031540245**
40*30*8	4.392226	2.135049	**0.012520868**
40*40*10	4.419078	1.621024	**0.106910443**
60*20*6	5.032964	2.432487	**0.095768762**
60*30*8	8.079107	4.292187	**0.088080693**
60*40*10	12.13912	6.175009	**0.004918839**
80*20*6	8.110349	4.059062	**0.006127451**
80*30*8	5.562571	2.003129	**0.010301844**
80*40*10	10.02119	5.488695	**0.005700713**
100*20*6	6.951091	3.953938	**0**
100*30*8	7.882399	4.232691	**0**
100*40*10	8.03282	6.162923	**0**
120*20*6	6.152712	3.014634	**0**
120*30*8	9.176134	6.091948	**0**
120*40*10	6.930427	5.119459	**0**
Total	7.094646	3.856612	**0.024124657**

Table 7. ANOVA results for small-scale problems.

Source	DF	SS	MS	F	*P*-value
Factor	2	2319	1159.5	104.42	0
Error	447	4963.6	11.1		
Total	449	7282.6			

Table 8. ANOVA results for large-scale problems.

Source	DF	SS	MS	F	*P*-value
Factor	2	13,842.8	6921.4	556.38	0
Error	447	5560.7	12.4		
Total	449	19,403.5			

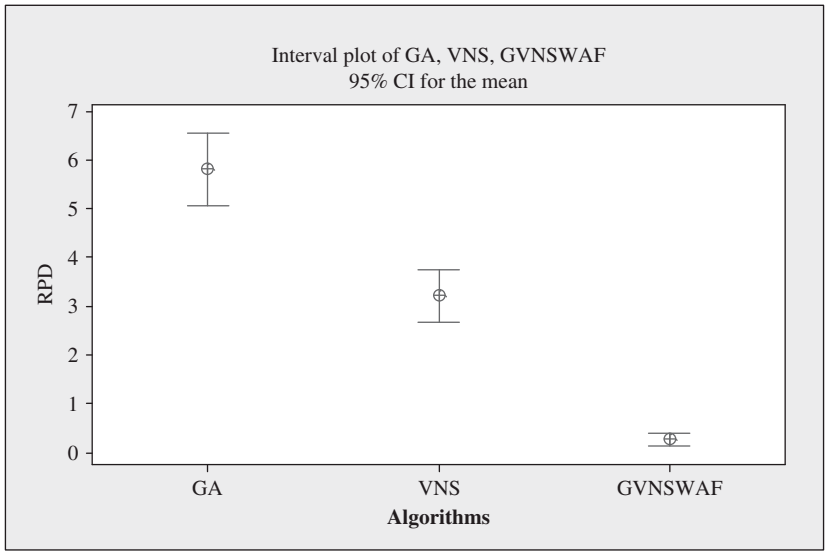


Figure 8. Means plot and LSD intervals (at the 95% confidence level) for algorithms in small scale.

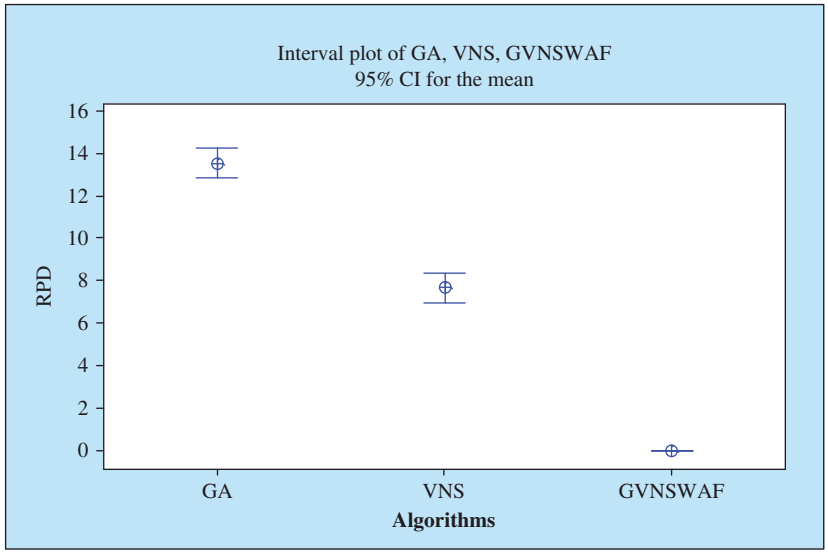


Figure 9. Means plot and LSD intervals (at the 95% confidence level) for algorithms in large scale.

Downloaded by [North Dakota State University] at 22:03 02 December 2014

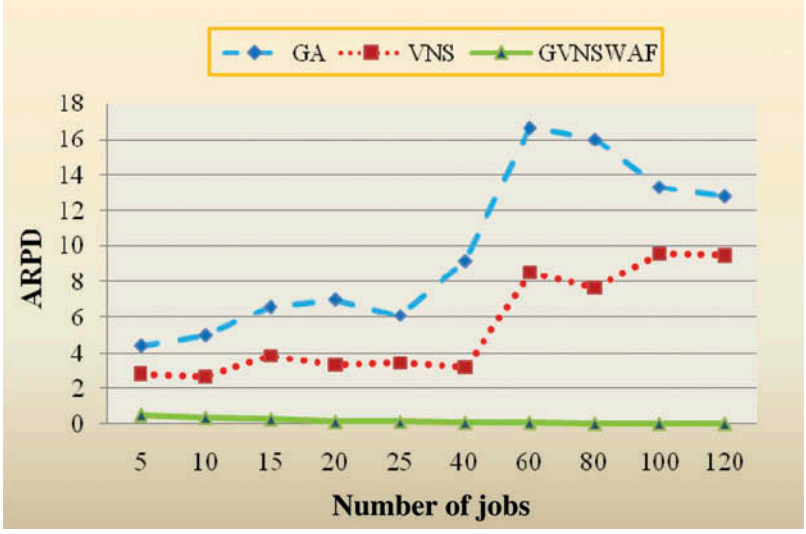


Figure 10. Interaction between performance of algorithms and number of jobs in terms of ARPD.

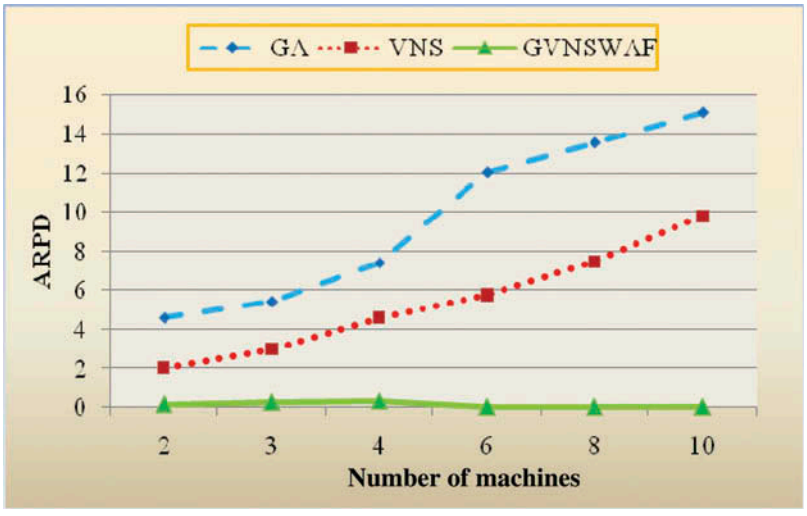


Figure 11. Interaction between performance of algorithms and number of machines in terms of ARPD.

between performances of algorithms versus number of machines is demonstrated in Figure 11. All in all, it could be concluded that GVNSWAF is the best algorithm among all those which we considered in this study.

5. Conclusion

This paper discussed the FJSP under the learning effect and deterioration with SDST. The studied objective was makespan minimisation. For solving the aforementioned problem, we proposed a hybrid approach with hybridisation of GA and VNS GA results showed that we could be faced with premature convergence. To verify the viability of the recommended method, we added the affinity function to the VNS algorithm, and saw the solutions obtained from the VNS algorithm significantly improve. In most of the cases that we tested (for example, 40-40-10 corresponding to Figure 4), we observed that the results moved away from the optimum local corner, thus demonstrating the efficacy of the hybrid algorithm for prevention of premature convergence. To improve the quality of our proposed algorithm, we added a segment of artificial immune system algorithm, which is known as the affinity function, for increasing the diversity in all iterations of the algorithm. GA was used in our proposed algorithm (GVNSWAF), because of its capability in global search optimisation, and VNS was used in order to increase the local search rate. Because of sensitiveness of our proposed algorithm to parameter values, a specific tool namely Taguchi

Downloaded by [North Dakota State University] at 22:03 02 December 2014

experimental method was employed. The computational results have shown that the proposed GVNSWAF statistically outperformed GA and VNS. The future work is to apply the GVNSWAF to other kinds of combinatorial optimisation problems and to develop some other hybrid algorithms for solving the aforementioned FJSP.

References

Alidaee, B., and N. K. Womer. 1999. "Scheduling with Time Dependent Processing Times: Review and Extensions." *Journal of the Operational Research Society* 50: 711–720.

Amiri, M., M. Zandieh, M. Yazdani, and A. Bagheri. 2010. "A Variable Neighbourhood Search Algorithm for the Flexible Job-Shop Scheduling Problem." *International Journal of Production Research* 48 (19): 5671–5689.

Bagheri, A., and M. Zandieh. 2011. "Bi-Criteria Flexible Job-Shop Scheduling with Sequence Dependent Setup Times – Variable Neighborhood Search Approach." *Journal of Manufacturing Systems* 30: 8–15.

Bagheri, A., M. Zandieh, I. Mahdavi, and M. Yazdani. 2010. "An Artificial Immune Algorithm for the Flexible Job-Shop Scheduling Problem." *Future Generation Computer Systems* 26: 533–541.

Biskup, D. 1999. "Single-machine Scheduling with Learning Considerations." *European Journal of Operational Research* 115 (1): 173–178.

Biskup, D. 2008. "A State-of-the-Art Review on Scheduling with Learning Effects." *European Journal of Operational Research* 188: 315–329.

Brandimarte, P. 1993. "Routing and Scheduling in a Flexible Job Shop by Taboo Search." *Annals of Operations Research* 41: 157–183.

Brucker, P., and R. Schlie. 1990. "Job-Shop Scheduling with Multipurpose Machines." *Computing* 45 (4): 369–375.

Chambers, J. B., and J. W. Barnes. 1998. *Reactive Search for Flexible Job Shop Scheduling*. Austin: The University of Texas, Department of Computer Sciences.

Chen, H., J. Ihlow, and C. Lehmann. 1999. "A Genetic Algorithm for Flexible Job-Shop Scheduling." In *IEEE International Conference on Robotics and Automation*, Detroit, May 10–15, 1120–1125.

Cheng, T., Q. Ding, and B. M. T. Lin. 2004. "A Concise Survey of Scheduling with Time-dependent Processing Times." *European Journal of Operational Research* 152: 1–13.

Dauzére-Pérés, S., and J. Paulli. 1997. "An Integrated Approach for Modeling and Solving the General Multiprocessor Job-Shop Scheduling Problem Using Tabu Search." *Annals of Operations Research* 70: 281–306.

Dell'Amico, M., and M. Trubian. 1993. "Applying Tabu Search to the Job-shop Scheduling Problem." *Annals of Operations Research* 41 (3): 231–252.

Desprez, C., F. Chu, and C. Chu. 2009. "Minimising the Weighted Number of Tardy Jobs in a Hybrid Flow Shop with Genetic Algorithm." *International Journal of Computer Integrated Manufacturing* 22 (8): 745–757.

Eren, T., and E. Güner. 2009. "A Bicriteria Parallel Machine Scheduling with a Learning Effect." *The International Journal of Advanced Manufacturing Technology* 40 (11–12): 1202–1205.

Fattahi, P., F. Jolai, and J. Arkat. 2009. "Flexible Job Shop Scheduling with Overlapping in Operations." *Applied Mathematical Modelling* 33 (7): 3076–3087.

Fattahi, P., M. Saidi, and F. Jolai. 2007. "Mathematical Modeling and Heuristic Approaches to Flexible Job Shop Scheduling Problems." *Journal of Intelligent Manufacturing* 18 (3): 331–342.

Garey, M. R., D. S. Johnson, and R. Sethi. 1976. "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1 (2): 117–129.

Hansen, P., and N. Mladenovic. 2001. "Variable Neighborhood Search: Principles and Applications." *European Journal of Operational Research* 130: 449–467.

Hao, G. S., D. W. Gong, Y. Q. Shi, and L. Wang. 2005. "Interactive Genetic Algorithm Based on Landscape of Satisfaction and Taboos." *Journal of China University of Mining & Technology* 34 (2): 204–208.

Ho, N. B., J. C. Tay, and E. M. K. Lai. 2007. "An Effective Architecture for Learning and Evolving Flexible Job-shop Schedules." *European Journal of Operational Research* 179 (2): 316–333.

Hurink, J., B. Jurisch, and M. Thole. 1994. "Tabu Search for the Job-Shop Scheduling Problem with Multi-Purpose Machines." *OR Spectrum* 15: 205–215.

Jeong, B., J. Lee, and H. Cho. 2005. "Efficient Optimization of Process Parameters in Shadow Mask Manufacturing Using NNPLS and Genetic Algorithm." *International Journal of Production Research* 43 (15): 3209–3230.

Kacem, I., S. Hammadi, and A. Borne. 2002a. "Pareto-Optimality Approach for Flexible Job Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic." *Mathematics and Computers in Simulation* 60 (3): 245–276.

Kacem, I., S. Hammadi, and R. Borne. 2002b. "Approach by Localization and Multi-Objective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems." *IEEE Transactions on Systems, Man and Cybernetics, Part C* 32 (1): 408–419.

Kelton, W. D., and A. M. Law. 2000. *Simulation, Modelling and Analysis*. Boston, MA: McGraw-Hill.

Lei, D. 2009. "A Genetic Algorithm for Flexible Job Shop Scheduling with Fuzzy Processing Time." *International Journal of Production Research* 48 (10): 2995–3013.

Liouane, N., I. Saad, S. Hammadi, and P. Borne. 2007. "Ant Systems and Local Search Optimization for Flexible Job Shop Scheduling Production." *International Journal of Computers, Communications & Control* 2 (2): 174–184.

Luo, X., W. Li, D. Xue, J. Tang, and Y. Tu. 2010. "Optimal Resource Allocation for Hybrid Flow Shop in One-of-a-Kind Production." *International Journal of Computer Integrated Manufacturing* 23 (2): 146–154.

Mansouri, S. A., S. M. Moattar-Husseini, and S. H. Zegordi. 2003. "A Genetic Algorithm for Multiple Objective Dealing with Exceptional Elements in Cellular Manufacturing." *Production Planning & Control* 14 (5): 437–446.

Mastrollili, M., and L. M. Gambardella. 2000. "Effective Neighbourhood Functions for the Flexible Job Shop Problem." *Journal of Scheduling* 3: 3–20.

Mohammadi, M., S. F. Ghomi, and N. Jafari. 2011. "A Genetic Algorithm for Simultaneous Lotsizing and Sequencing of the Permutation Flow Shops with Sequence-Dependent Setups." *International Journal of Computer Integrated Manufacturing* 24 (1): 87–93.

Moon, I., S. Lee, and H. Bae. 2008. "Genetic Algorithms for Job Shop Scheduling Problems with Alternative Routings." *International Journal of Production Research* 46 (10): 2695–2705.

Downloaded by [North Dakota State University] at 22:03 02 December 2014

Nowicki, E., and C. Smutnicki. 1996. "A Fast Taboo Search Algorithm for the Job Shop Problem." *Management Science* 42 (6): 797–813.

Paulli, J. 1995. "A Hierarchical Approach for the FMS Scheduling Problem." *European Journal of Operation Research* 86: 32–42.

Pezzella, F., G. Morganti, and G. Ciaschetti. 2008. "A Genetic Algorithm for the Flexible Job-Shop Scheduling Problem." *Computers & Operations Research* 35 (10): 3202–3212.

Phadke, M. S. 1989. *Quality Engineering Using Robust Design*. Upper Saddle River, NJ: Prentice-Hall.

Ponnambalam, S. G., V. Ramkumar, and N. Jawahar. 2001. "A Multiobjective Genetic Algorithm for Job Shop Scheduling." *Production Planning & Control* 12 (8): 764–774.

Rabiee, M., M. Zandieh, and P. Ramezani. 2012. "Bi-objective Partial Flexible Job Shop Scheduling Problem: NSGA-II, NRGA, MOGA and PAES Approaches." *International Journal of Production Research* 50 (24): 7327–7342.

Rajkumar, M., P. Asokan, and V. Vamsikrishna. 2010. "A GRASP Algorithm for Flexible Job-shop Scheduling with Maintenance Constraints." *International Journal of Production Research* 48 (22): 6821–6836.

Ross, P. J. 1996. *Taguchi Techniques for Quality Engineering*. 2nd ed. New York: McGraw-Hill Inc.

Roy, R. K. 2001. *Design of Experiments Using the Taguchi Approach: 16 Steps to Product and Process Improvement*. New York: Wiley.

Saidi-mehrabad, M., and P. Fattahi. 2007. "Flexible Job Shop Scheduling with Tabu Search Algorithms." *International Journal of Advanced Manufacturing Technology* 32: 563–570.

Shiau, Y.-R., W.-C. Lee, C.-C. Wu, and C.-M. Chang. 2007. "Two Machine Flow Shop Scheduling to Minimize Mean Flow Time Under Simple Linear Deterioration." *International Journal of Advanced Manufacturing Technology* 34: 774–782.

Solimanpur, M., and A. Elmi. 2011. "A Tabu Search Approach for Group Scheduling in Buffer Constrained Flow Shop Cells." *International Journal of Computer Integrated Manufacturing* 24 (3): 257–268.

Uckun, S., S. Bagchi, K. Kawamura, and Y. Miyabe. 1993. "Managing Genetic Search in Job Shop Scheduling." *IEEE Expert* 8 (5): 15–24.

Vahdani, B., and M. Zandieh. 2010. "Scheduling Trucks in Cross-Docking Systems: Robust Metaheuristics." *Computers & Industrial Engineering* 58: 12–24.

Wang, S., and J. Yu. 2010. "An Effective Heuristic for Flexible Job-Shop Scheduling Problem with Maintenance Activities." *Computers & Industrial Engineering* 59: 436–447.

Wu, C.-C., W.-C. Lee, and Y.-R. Shiau. 2007. "Minimizing the Total Weighted Completion Time on a Single Machine Under Linear Deterioration." *International Journal of Advanced Manufacturing Technology* 33: 1237–1243.

Xia, W. J., and Z. M. Wu. 2005. "An Effective Hybrid Optimization Approach for Multi Objective Flexible Job-Shop Scheduling Problems." *Computers and Industrial Engineering* 48 (2): 409–425.

Yazdani, M., M. Zandieh, and M. Amiri. 2009. "Flexible Job-Shop Scheduling with Parallel Variable Neighborhood Search Algorithm." *Expert Systems with Applications* 37 (1): 678–687.

Downloaded by [North Dakota State University] at 22:03 02 December 2014