Jr. of Industrial Pollution Control 33(2)(2017) pp 1579-1588
www.icontrolpollution.com
Research Article

HYBRIDIZATION OF GENETIC ALGORITHM & VARIABLE NEIGHBORHOOD SEARCH (VNGA) APPROACH TO CRACK THE FLEXIBLE JOB SHOP SCHEDULING PROBLEM: A FRAMEWORK

RAJAN [1*] AND VINEET KUMAR [2]

[1]Research Scholar, Department of Mechanical Engineering, University Institute of Engineering & Technology, Maharishi Dayanand University, Rohtak (Haryana), India

[2]Professor, Department of Mechanical Engineering, University Institute of Engineering & Technology, Maharishi Dayanand University, Rohtak (Haryana), India

(Received 07 August, 2017; accepted 20 September, 2017)

Key words: Flexible job shop scheduling, Makespan, Genetic algorithm (GA), Variable neighborhood search (VNS), Differential evolution algorithm, Meta-heuristics

ABSTRACT

It is well recognized that for finding the finest or precised solution of a flexible job shop scheduling problem (FJSSP) which is NP-hard in nature, one needs to connect different aspects of many optimization approaches. This paper proposes differential evolution algorithm (DE) to show hybrid combination of genetic algorithm & variable neighborhood search (VNGA) approach for solving FJSSP and assess the effect of flexibility with an objective to minimize makespan. DE algorithm is a latest population based evolutionary meta-heuristic. Simulation provides a meaningful understanding of real phenomenon of a system nature. A regeneration scheme as suggested in literature is also embedded into the framework. For solving the FJSSP an algorithm and its main representation techniques (GA & VNS) are presented in order to have optimized output.

INTRODUCTION

In modern era scheduling quandary plays a crucial part due to increasing customer requirement of variety; shorten life span of commodity, transmuting mart by ecumenical race and brisk growth of incipient technical methods. Job shop scheduling (JSS) is one of the trendiest processes subsisting amongst the toughest conjunctional optimization quandaries (Garey, *et al*., 1996). Flexible job shop scheduling (FJSS) is the extension of job shop scheduling (JSS) (Vinod and Sridharan, 2008). It is assumed that there be an option to choose from different machines which process has to be performed under prescribed set of process plans. For cracking the single objective FJSSP, various techniques are being used in the literature. (Bruker Schlie, 1990) suggested a polynomial graphical design of two jobs for reducing makespan. (Tay and Ho, 2008) employed suitable parameter and operator spaces for evolving composite dispatching rules using genetic programming for achieving greater scalability and flexibility. (Kacem, *et al*., 2002) described pareto base technique on hybrid combination of fuzzy logic (FL) & evolutionary algorithm (EA) for solving flexible job shop scheduling problem (FJSSP). (Vinod and Sridharan, 2008) presented seven scheduling rules in which setup times are sequence dependent. A hybrid algorithm of simulated annealing (SA) with particle swarm optimization (PSO) to solve the FJSSP was proposed by (Xia and Wu, 2005; Ruiz and Maroto, 2006; Gao, *et al*., 2007) proposed a meta-heuristic algorithm that incorporates four new

***Corresponding authors email: rajan@gmail.com**

crossover operators in genetic algorithm (GA) for an intricate FJSSP. The hybrid combination of genetic algorithm & variable search descent (VND) has also been recommended by (Goncalves, *et al*., 2005) for the FJSSP. (Pezzella, *et al*., 2008) offered a genetic algorithm (GA) for FJSSP to integrate the diverse schemes for engendering initial population, culling individual for reproduction and for reprlicating incipient individual. Hybridization of particle swarm optimization & tabu search for calculating FJSSP was proposed by (Zhang, *et al*., 2009). The organization of rest of the paper is as follows. Section-3 & Section-4 introduces "Problem Description" & "Problem Formulation" of the FJSSP with the objective function to minimise the makespan in NP hard atmosphere. The "Proposed Hybridized Algorithm VNGA" is presented in Section-5 & Section-6. The comprehensive demonstration of genetic operator and variable neighbourhood search has been introduced. Various VNS algorithms are also proposed in Section-6. Several finishing comments have been also made in Section-7 "Conclusions".

SEQUENCING AND SCHEDULING

Sequencing means arranging jobs in distinct series. Different kinds of sequencing are pursued by industries on the basis like: first in first out, priority, job size and processing time. Dissimilar processing times can be attained during processing time based sequencing. The sequence which offers minimum processing time is accepted (Baker, 1974).

Scheduling is concerned with the assignment of time to a set of jobs for processing through a group of machines in order to best satisfy the objective function. Whenever there is an option so that a number of jobs can be conducted in parallel, scheduling subsists (Conway, *et al*., 1967). In order to conduct a set of duties for allocating the several resources in a prescribed tenure, scheduling is required. Therefore, it is a governing practice that assists to boost the presentation of organization (Tay and Ho, 2008) Scheduling is defined in general as the process of accrediting tasks to the available limited resources with goal of meeting the settled aspiration.

Types of Scheduling

Broadly, the manufacturing organizations classify the scheduling as job shop & flow shop. Job shop is very difficult kind of scheduling. Exploring the job shop scheduling problem gives paramount picture of the solution encountered during genuine and complex system. On the basis of routing of different jobs on the shop floor job shop scheduling can be categorize as open shop and closed shop. During open shop no boundations are there on the routing i.e., each job can have a separate routing. While on the other hand the number of routings available is limited in a closed shop, and the job has to chase one of the existing routings.

Complications arise during FJSS can be reviewed as under:

- How engagement of a process on a suitable machine will took place.
- How sequencing of different processes will be done on individual machine.
- How a work piece will travel on a machine repeatedly (called recirculation).

For finding the precised optimal solutions above mentioned aspects extensively increase the intricacy of the problem

PROBLEM DESCRIPTION

Being NP-Hard in nature, job-shop scheduling comes out to be very intensive and complicated problems. The chief aspire of the job shop scheduling is to minimize the makespan during a set of random moves. The term makespan means, maximum finishing time needed to complete the set of random moves for n jobs on m machines. The problem of job shop scheduling will turn to NP-hard only when iff there will be increase in both number of jobs and machines. During job shop, orders of the goods to be manufactured are low in quantity. Every machine on the floor can be distinguished by the flow of the work that's why of flow of the operation is not in one direction. It is assumed that individual machine is available for individual process and there exists separate process plan for individual job. Although in flexible job shop scheduling there is an option to choose from different machines that which process has to be performed under which process plan. Because of choosing amongst the various available process plan FJSS becomes cumbersome quandary than JSS. FJSS is also NP-hard in nature as the same way of JSS is (Mitchell, 2002).

For obtaining an optimum solution, accurate techniques such as branch & bound and dynamic programming, consume bunch of hours. Because of complexity, scientists have concentrated on variety of meta-heuristics techniques which are offered in the literature as reference (Kolonko, 1999; Pezzella and Merelli, 2000; Goncalves, *et al*., 2005; Huang and Liao, 2008; Fonseca and Navaresse, 2002; Tanev, *et al*., 2004) & various different heuristic techniques (Chen and Luh, 2003; Huanh and Yin, 2004; Jansen, *et al*., 2005) to solve the FJSS problem.

This paper suggests a hybrid combination of genetic algorithm & variable neighborhood search (VNGA) approach for solving the FJSS problem with an objective to minimize the makespan. (Storn and Kenneth, 1997) recommended a differential evolution (DE) algorithm which is the newest density dependent progressive meta-heuristic amongst all.

PROBLEM FORMULATION

Consider set of 'j' number of autonomous jobs, j = $\{j_1, j_2, j_3, \ldots, j_n\}$ and a set of 'm' number of available multiple process plans (MPP) i.e., m = $\{m_1, m_2, m_3, \ldots, m_n\}$. Job is composed with a series of process O, o = $\{o_1, o_2, o_3, \ldots, o_n\}$ one by one as per to the available series. Every action should be performed on selected machine selected out off available machines. Time to carry out the action depends on machine. For every action processing time of any desired machine is already recognize & out coming plans are subjected to constraints that single machine will process single operation and repetition of process is not allowed. Pair of troubles exists during scheduling namely routing & sequencing. Routing means to allocate every action on a suitable machine and sequencing establishes series of processes on everyone machine. Reduction in makespan F (C_{max}) (Zhang) is the main agenda for locating a perfect schedule. The notations which are used to develop a mathematic model of the designing of Flexible job shop scheduling problem are defined as follows in Table 1:

Objective Function

Minimize makespan F (C_{max})

Minimize F (Cmax) = $C_{n,m}$

Conjunctive constraints;

$C_{i,j,k} \geq C_{i,j+1} - d_{i,j+1}$ for,

Table 1. Flexible job shop scheduling problem

Notation	Representation
i	Part type index, i = 1, 2, 3…n
j	Quantity of job j =
m	Quantity of machine m =
O	Sequence of process O =
(C_{max})	Makespan (Maximum completion time) in minutes
($C_{i,j,k}$)	Partial makespan without predecessors.
($C_{i,j+1,k}$)	Enhanced makespan with predecessors.
$d_{i,j}$	Crashed tenure of activity from node i to node j.
$d_{i,j+1}$	Tenure of activity from node i=1 to jth node
$O_{i,k}$	Job action of parallel machine.
($f_{i,j,k}$)	Parallel action on job of particular machine.
R_{oij}	Most work remaining (MWKR) in minutes.
C_r	Correction Ratio.
P_r	Probability (Randomly taken).

i= 1, 2…n;

j= 1, 2…p;

k=1, 2…m.

$C_{i,j,k} \geq 0$ for,

i= 1, 2…n;

j= 1, 2…p;

k=1, 2…m

Resource constraints

$O_{i,j,k} = 1$

If job j scheduled before the same job on specified machine = 0

Otherwise for,

i= 1, 2…n;

j= 1, 2…p;

k= 1, 2…m.

THE PROPOSED HYBRID ALGORITHM

The proposed hybrid algorithm is a combination of two algorithms, namely, GA and VNS. Two different conditions are provided in order to terminate the algorithm entirely or partially (Yuan and Xu, 2013; Riza, *et al*., 2014). The first termination condition is the achievement of best known solution and second termination condition is the attainment of highest number of generations in the main loop of hybrid combination of genetic algorithm & variable neighborhood search (VNGA). If algorithm reaches either at first or at second condition, the whole algorithm will terminate at once.

The advantage of hybrid algorithm is employed in the proposed framework in order to find the optimum solutions for the JSSPs. (Fig. 1), proposes a flowchart of differential evolution (DE) algorithm to show hybrid combination of genetic algorithm & variable neighborhood search (VNGA). It starts with the initialization of the population at random. The created population is evaluated based on the fitness function, and a new knowledge-based operator is applied in this step to improve the solution quality of the individuals. In addition, this knowledge-based operator is merged with the function evaluation phase, and it works with machines' idle times. If condition 1 is reached then apply 1st termination. In the reproduction phase, a selection operator is applied to select the parents for the mating pool, and then a crossover operator is performed to produce the offspring. In addition, a mutation operator is carried out on randomly selected individuals

to create the mutants. The created offspring and mutants are evaluated. VNS will be started with the best individual of GA. If condition 2 is reached then 2nd termination is considered in order to guide the algorithm to be on the right path. The termination conditions are described in following sections.

Encoding and Decoding

In any algorithm, the first and the most important step is to find appropriate encoding and decoding procedures in order to represent the problem. In this proposed algorithm, an operation-based representation is adopted to represent the operations of different jobs (Yusof, *et al*., 2011; Cheung and Zhou, 2001). In this approach, a chromosome consists of m x n genes in which each of the genes represents the sequence of the operations (1st thread) that should be executed on the machines (2nd thread).

Proposed algorithm starts with the initialization of the population at random. Initialization of differential evolution algorithm aims to set the control parameters and initial population vector (m x n). Parameter includes size of the population 'Np' (Non deterministic polynomial). The created population is evaluated based on the fitness function, and a new knowledge-based operator is applied in this step to improve the solution quality of the individuals. The size of the population (m x n) doesn't change during the evaluation process, and it is one of the controlled parameter of the algorithm. Initially, the population is randomly generated and may cover the complete space with consistent probability (Zhang, *et al*., 2009). Initialization process is chased by the process of evaluation, i.e., the cost of each vector is calculated and stored for the future reference.

Fitness Function and the Knowledge-Based Operator

The fitness function usually determines probability of the solution that can be shifted to subsequent generation. In other words, quality of solution is determined by applying the knowledge-based operator and the chromosomes of higher quality is having greater chance for surviving; however, the less fitted chromosomes must be discarded from the population. In FJSSPs, many different performance evaluators exist for defining the fitness functions. In proposed research makespan (C_{max}) is used as fitness function in order to evaluate each chromosome. Programmer presents the score based fitness function to each & every chromosome and then judges how well each individual performs at the specified time. In the proposed work, a well-known fitness function is taken into account for calculating the performance of individual chromosomes which is illustrated by (Goldberg, 1989) as follows:

if 'X_c' is an individual chromosome having the makespan value as '$C(X_c)$' and

if 'C_{max}' is the highest value of makespan amongst the entire population of chromosomes,

then value of fitness function of individual chromosome 'X_c' can be shown as under:

$$\text{Fit. } f^n(X_c) = C_{max} - C(X_c)$$

In this proposed algorithm and in the context of the fitness function, an incipient cognizance-predicated operator is framed predicated on the quandary characteristics. This operator is designed based on the machines idle times that exist in the job shop environments. In addition, this knowledge-based operator is applied during the evaluation of function for decreasing computational time of algorithm and to cover all the chromosomes that need to be evaluated.

To design this operator more efficiently, the following steps are applied.

- The idle points of each machine must be found. Then, for each of these idle points, the idle start time, idle finish time, and length of idle time must be calculated.
- Based on the position of the idle point on the machine sequence list, a candidate operation is chosen from the right side of the machine sequence list in order to be shifted to the idle position by considering the duration of idle time and the processing time of the candidate operation.
- If the length of idle time is bigger than or equal to the processing time of the candidate operation, it will be accepted conditionally. Otherwise, it will be rejected.
- If the candidate operation is rejected for transfer, the operator goes back to the second step and chooses the subsequent operation.
- For the conditionally accepted candidate operation, all of the processing constraints must be considered in order to reject the shift or accept it. For instance, the previous operation of the candidate operation must be completed.
- If all of the constraints are satisfied, the candidate operation will be transferred to its new position. Otherwise, the candidate operation will remain in its own position.
- For each of the machines, Steps 2–6 should

be continued until the last operation on the machine sequence list.

Selection Operator

During reproduction stage of algorithm, a machine dependent superiority, order dependent crossover and mutation operators are evolved for producing the offspring & mutants after selection. While employing the genetic algorithm (GA), selection is another significant parameter to be considered. Through selection an offspring of the chromosomes can randomly be chosen from base to the next hierarchy level amongst the entire population according to their evaluation function. Therefore, probability of selection is very high for the parent chromosome which has better fitness function so as it could shift to the next hierarchy level. There are two type of renowned selection methods during the reproduction phase namely Roulette Wheel selection and Tournament selections.

A good selection technique can increase GA's performance in terms of reaching faster to the optimal solutions. In this paper, the Roulette Wheel selection technique which is the most commonly used operator is proposed for the selection of parents [26]. In addition, the elitism approach is applied in this selection technique in order to retain the fittest chromosomes for the next generation and to prevent the solution from getting worse from one generation to another. In the Roulette Wheel selection, Boltzmann Probability is used for calculating every chromosome.

$P(x) = \exp(-\beta' \times f(x)) / f'(x)$

Where:

$P(x)$ is the probability of individual chromosome

β' is the selection pressure

$f(x)$ is the fitness of each individual, and

$f'(x)$ is the fitness of the worst individual in the generation

It should be noted that $f'(x)$ is added to the original equation of Boltzmann Probability for making the selection process independent of problem scale. In addition, the normalized probability of each selected individual is given as

Crossover

Being the main genetic operator, crossover works on two chromosomes simultaneously by exchanging the genes out of two and hence generates the offspring's as a result of two incipient chromosomes. It is the backbone of the algorithm. The crossover operator is performed by combining the information of the first and second parents and the produced offspring with the features of both parents can be either better or worse than their parents. As a consequence of exchanging of genes, incipient chromosomes begin at a cut-point and culminate at another point. In addition, the main goal of this operator is to produce better and feasible offspring from the parental information. The crossover can incorporate an arbitrary separation of gene for connecting the cut-point. Rate of exchange of gene during crossover can be determined as follows:

Above equation is extremely significant since it helps to conclude how many times crossover takes place i.e., number of crossover 'Cc', which governs the number of offspring formed during the exchanging process. In this paper, a machine dependent superiority and order dependent crossover (POX) is presented for generating the feasible offspring (Lee, *et al*., 1998).

The following detailed steps are taken in order to implement the POX operator.

• First, two individuals are selected as parents (P_1, P_2) by applying the Roulette Wheel selection.

• Then, two sets of sub jobs are selected and called sj_1 and sj_2. One of the sub jobs is selected from the bottleneck machines and the other one is selected randomly amongst the n remaining jobs.

• In this step, elements of first sub job (sj_1) are copied from first parent (P_1) to the exact positions in the first child (O_1), and same goes for second sub job; that is, elements of second sub job (sj_2) are copied from second parent (P_2) to exact alleles in the second child (O_2).

• All of the alleles in the first sub job (sj_1) are deleted in the second parent (P_2), and the same goes for the second sub job; that is, all of the alleles in the second sub job (sj_2) are deleted in the first parent (P_1).

• The remaining alleles in the first and second parents (P_1, P_2) are transferred to the second and first offspring (O_2, O_1) from the most left to the most right, respectively.

The procedures that are used in implementing the proposed POX operator lead to the feasible solutions in which it does not need a repairing mechanism.

Mutation

Mutation is a process which engenders arbitrary transmutations in specific chromosomes. By reviving an incipient gene which disorients at the time of development of chromosome, this operator

plays decisive part in genetic algorithm. The rate of mutation regulates the number of genes mutated or launched.

In the reproduction phase, mutation is the second way of exploring the solution space. Mutation operator can prevent the algorithm from being trapped in the local optima, and it makes the algorithm faster in achieving better solutions. In addition, it could make perturbation in the chromosome in order to increase the population diversity. In the algorithm, two types of mutation operators, namely, swapping and insertion are proposed. Not only can these mutation operators increase the diversity of the population, but the insertion operator could carry out intensive search.

Machine Selection Part: It can only be reformed by mutation operator. Arbitrarily selected gene is superseded by the machine because the operational time is brief from the substitute machine.

Operation Sequence Part: It acts only about how the process succeeds. OS part incorporates following proposals namely exchanging (swapping) & insertion.

It should be noted that one of these OS mutation operators should be selected randomly in order to create an offspring, and they are described as follows.

Swapping Operator

To apply the swapping operator, first, two random numbers are generated which are the positions of two alleles in the chromosome (e.g., *R*= {2, 5}). Then, all of the parental information is copied to the exact positions in the offspring, except the randomly selected alleles which must be exchanged or swapped in the offspring. For instance, consider the parent as {2, 1, 1, 3, 2, 3} which is randomly selected from the population. The offspring of this chromosome, by applying the swapping operator, would be {2, 2, 1, 3, 1, 3}.

Insertion Operator

To apply the insertion operator, first, two random numbers are generated to be the positions of two alleles in the chromosome (e.g., *R*= {6, 4}). Then, all of the parental information is copied to the exact positions in the offspring, except the randomly selected alleles. The randomly selected allele with a smaller job-value is positioned on the left of the other random allele with a bigger job-value. For instance, given the randomly selected parent as {3, 1, 2, 3, 1, 2}, the offspring of this chromosome, by applying the insertion operator, would be {3, 1, 2, 2, 3, 1}.

In a situation when the algorithm continues through termination condition 2, that is, termination of GA's generations, VNS will be started with the best individual of GA. Moreover for improving the quality of the solution and up to some extent for improving population diversity which is obtained from GA, VNS is implemented. Reduction of makespan amongst available schedules is one of the important characteristics of the proposed hybridized algorithm. At last for proving the output of proposed hybrid algorithm only well-suited results will be standardized.

VARIABLE NEIGHBORHOOD SEARCH

It is very easy to solve the problems through most recent meta-heuristic technique developed like Variable neighbourhood search (VNS). Due to its easiness & user friendly behaviour in analytical analysis, local search is the best-known method among rest of all the methods (Mladenovic and Hansen, 1997). Exploration of entire problem can be done inside a partial region through LS algorithm. Before moving to advance explorations LS can expedite a stipulation of improved solutions. Because of efficient investigation procedure, systematic change of neighbourhood can be made through VNS. There are two types of loops in VNS algorithm the (i) inner loop and (ii) outer loop. Following are the steps to carry out the loops:

- Inner loop alter and explore the problem through shake and local search.
- Outer loop re-iterates the first loop after diversification.
- Major search can be carried out with first loop.
- For obtaining superior solutions, Local search, explores the problem inside the local neighbourhood space.
- Solution can be shifted to different local neighbourhood through diversification of Shake.
- Iteration of the inner loop will continued until we got the better solutions.
- Loop length is get controlled by integer 'k'.

After completion of the inner loop, re- iteration process of the outer loop is continued till termination condition is encounter.

While implementing the VNS, heuristic functions should be chosen very carefully in order to achieve a competent algorithm. For developing an efficient technique two types of neighbourhood operators are required namely: $N_k^s(x)$ & $\mathrm{N}_L^{LS}(\mathrm{x})$

Where: $N_k^s(x)$ represent for 'Shake' and

$N_L^{LS}(x)$ represent for Local search.

For attaining remarkable amendment, more than one structure can be utilized despite of individual function among shake & local search. For getting simple switching index 'k' is used for shake and index 'l' is used for local search function respectively. Perceptibly, both indexes contain higher limits, indicated as 'k_{max}' & 'l_{max}'. Hence $1 \le k \le k_{max}$ & $1 \le l \le l_{max}$ are different scales recognized for every index.

Steps for comprising the VNS:

- Firstly, find the initial solution 'x' to initialize the problem.
- Until stopping condition is met, re-iterate the following steps:

I. Shake: Randomly generate an initial solution $X' \in N_k^s(x)$.

II. Local Search (LS): Using the base neighbourhood arrangement $N_L^{LS}(x)$, pertain a Local search from the initial solution x' till a local minimum $X'' \in N_L^{LS}(x)$ is found.

III. move 'yes or no': do $x \leftarrow x''$, if f x" is better than x.

At step (ii) iteration will test entire base moves and will provide the finest adjoining result till a least amount is procured when the local search (LS) carried ravenous approach. A random solution can be generated from $N_L^{LS}(x)$ during the Shake process.

Based on both previously mentioned neighbourhood structures, the Local search structure provides a very simple and descent algorithm. It will keep iterating until superior moves attained. If computation generates default quantity of worse runs repeatedly then it will stop iterating. The Shake function will diversify the exploration and will start working so as to switch another region for carrying out a new Local search, when the previous Local search will end up a move.

Following are the steps to adopt Local search (LS) process:

- Firstly, get the initialized solution, $X \in N_k^S(x)$.
- Set the length of the loop initially as, k = 1
- k should be less than & equal to k_{max} then start the iteration.

1st iteration…. for k = 1

I. if (k =1) then $x'' \in N_L^{LS}(x)$ i.e., local minimum is found, exchange the Local search by using base neighbourhood arrangement $N_L^{LS}(x)$ pertain from the initial solution x'

II. else if (k =2) then $x'' \in N_L^{LS}(x)$ i.e., for local minimum, insert the Local search by using base neighbourhood arrangement $N_L^{LS}(x)$ pertain from the initial solution x'.

III. if $x'' \in N_L^{LS}(x)$ is less than $x' \in N_k^S(x)$ then exchange Local search by local minimum i.e., set x = x" for length of the loop k = 1.

IV. else start the 2nd iteration…. for k = k + 1

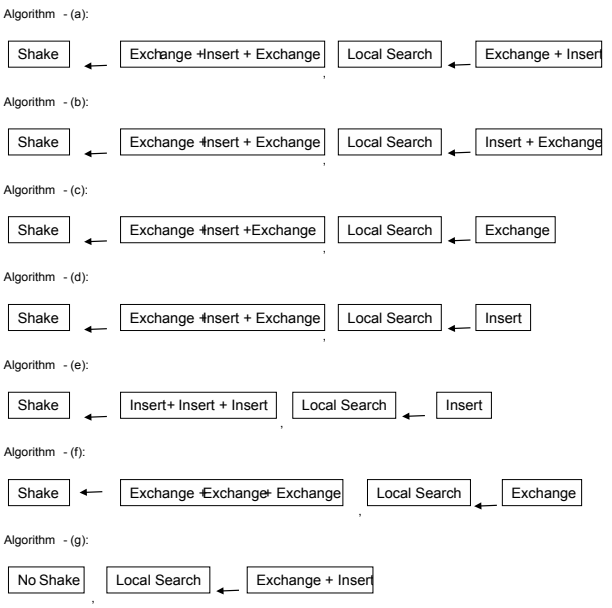
As mentioned above shake comprises of frequent arbitrary runs, in pair, organize with neighbourhood operator $N_k^S(x)$. Shaking is operational with exchange and Insert, sequentially. For getting x' any certain position x* is dealt with exchange and this is further activated by insert, one after the other. At last, for attaining 'x' exchange re-optimizes on the conclusion of insert x".

Finally, success of the algorithm has been considered regarding relative error (RE) index, up till superiority of the result is linked. Relative error (RE) index is calculated as under:

$$RE = \frac{best\ makespan - optimum\ boundary}{optimum\ boundary}$$

Various proposed Variable Neighbourhood Search (VNS) algorithms

Following are various proposed algorithms having utilitarian compositions for developing and proficient implementation of shake and local search functions. Each of them is different from the other (Fig. 1).



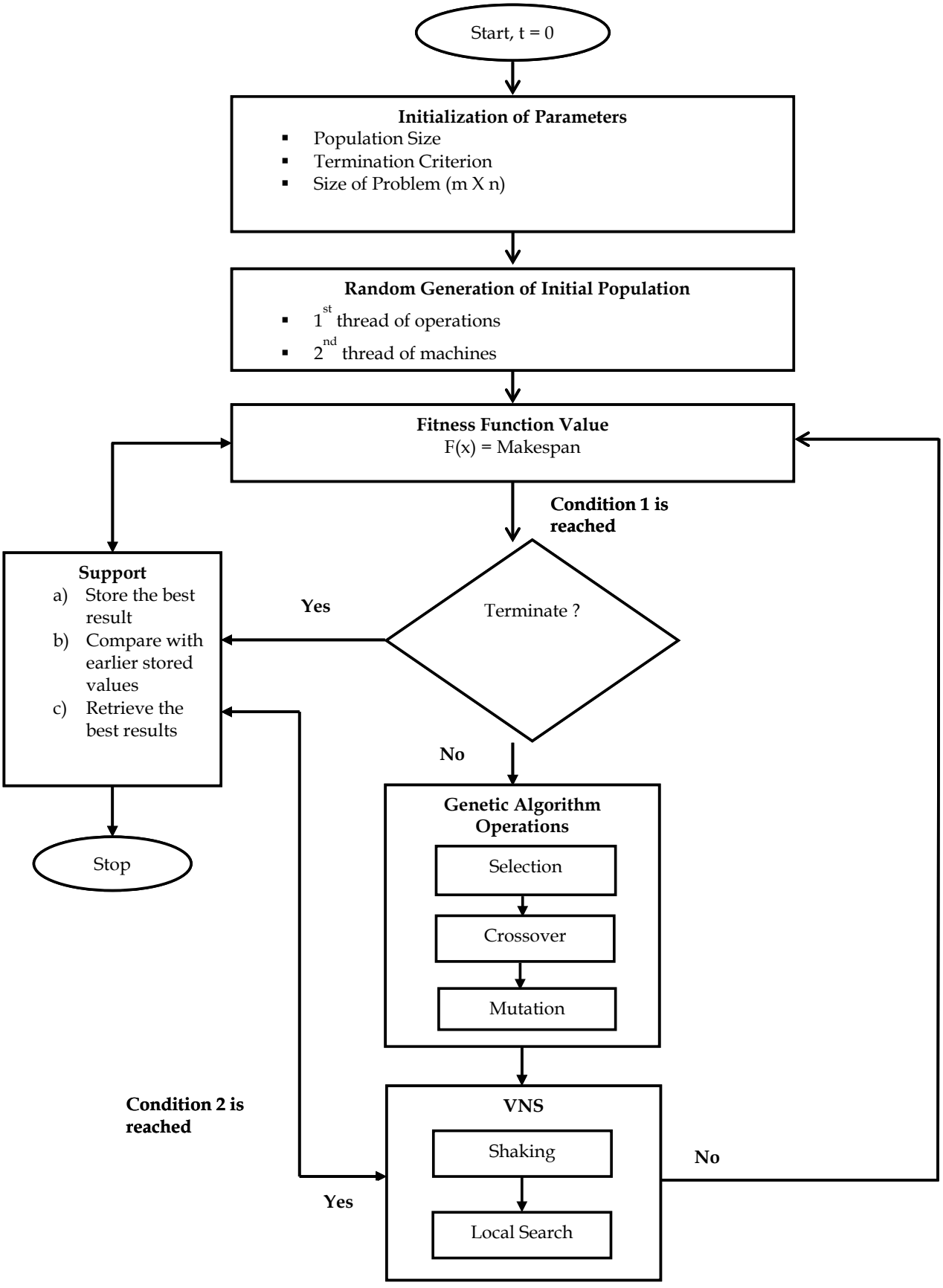


Fig. 1 Flowchart of the proposed hybrid algorithm of GA & VNS.

CONCLUSION

As scheduling problems are more stiff and hard, numerous algorithms have been investigated in literature. A cumulated genetic algorithm and variable neighbourhood search (VNGA) has been projected for solving the single objective FJSSP in the proposed work. For amending the ability of Local search (LS), VNS is rendered for transmuting the neighbourhood and to evade the algorithm for pre-maturing. Throughout search process, the hybridized algorithm neutralized the situation between the heterogeneity and augmentation. All through the optimization practice an external hard drive is installed as support so that non-dominated solutions can be saved and keep the backup updated. In the projected and framed algorithm, standard problems can be accounted to calculate the performance.

REFERENCES

Baker, K.R. (1974). Introduction to sequencing & scheduling. Wiley Publisher, NY, USA.

Bruker, P. and Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Com- putting.* 45(4) : 369-375.

Chen, H. and Luh, P.B. (2003). An alternative framework to Lagrangian relaxation approach for Job Shop Scheduling. *European Journal of Operational Research*. 149(3) : 499–512.

Cheung, W. and Zhou, H. (2001). Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. *Annals of Operations Research*. 107 : 65–81.

Conway, R.W., Maxwell, W.L. and Miller, L.W. (1967). Theory of Scheduling. Addison Wesley, Reading Massachusetts.

Fonseca, D.J. and Navaresse, D. (2002). Artificial neural networks for job shop simulation. *Advanced Engineering Informatics*. 16(4) : 241–246.

Gao, J., Sun, L.Y. and Gen, M. (2007). A hybrid genetic and variable neighbourhood descent algorithm for flexible job shop scheduling problems. *Computer & Operations Research*. 35 : 2892-2907.

Garey, M.R., Johnson, D.S. and Sethi, R. (1996). The complexity of flow shop and job-shop scheduling. *Mathematics of Operations Research*. 1(2) : 117-129.

Goldberg, D.E. (1989). Genetic algorithm in search optimization and machine learning. Addison Wesley.

Goncalves, J.F., Mendes, J.J.D.M. and Resende, M.G.C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research.* 167(1) : 77-95.

Huang, K.L. and Liao, C.J. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers and Operations Research.* 35(4) : 1030–1046.

Huanh, W.Q. and Yin, A.H. (2004). An improved shifting bottleneck procedure for the job shop scheduling problem. *Computers & Operations Research*. 31(12) : 2093–2110.

Jansen, K., Mastrolilli, M. and Solis-Oba, R. (2005). Approximation schemes for Job Shop Scheduling Problems with controllable processing times. *European Journal of Operational Research.* 167(2) : 297–319.

Kacem, I., Hammadi, S. and Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*. 60 : 245-276.

Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research.* 113(1) : 123-136.

Lee, K.M., Yamakawa, T. and Lee, K.M. (1998). Genetic algorithm for general machine scheduling problems. In Proceedings of the 2nd International Conference on knowledge-Based Intelligent Electronic Systems. IEEE, Adelaide, Australia, 60–66.

Mitchell, M. (2002). An introduction to genetic algorithms. Prentice-Hall of India, New Delhi.

Mladenovic, N. and Hansen, P. (1997). Variable neighbourhood search. *Computers and Operations Research.* 24 : 1097–1100.

Pezzella, F. and Merelli, E. (2000). A Tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research.* 120(2) : 297–310.

Pezzella, F., Morganti, G. and Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*. 35 : 3202-3212.

Riza, A., Rahman. and Santosa, B. (2014). Hybrid differential evolution and bottle neck heuristic algorithm to solve bi-objective hybrid flow shop scheduling unrelated parallel machines problems.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*. 169 : 781-800.

Storn, R. and Kenneth, P. (1997). Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization.* 11 : 341–359.

Tanev, I.T., Uozumi, T. and Morotome, Y. (2004). Hybrid evolutionary algorithm-based real-world Flexible Job Shop Scheduling Problem: application service provider approach. *Applied Soft Computing*. 5(1) : 87–100.

Tay, J.C. and Ho, N.B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Comp. & Industrial Engin*. 54 : 453–473.

Vinod, V. and Sridharan, R. (2008). Scheduling a dynamic job shop production system with sequence dependent setups: an experimental study. *Robotics and Computer-Integrated Manufacturing*. 24 : 435-449.

Xia W. and Wu. Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problem. *Computer & Industrial Engineering*. 48 : 409-425.

Yuan, Y. and Xu, H. (2013). Flexible job shop scheduling problem using hybrid differential evolution algorithms.

Yusof, R., Khalid, M., Hui, G.T., Yusof, S.M. and Othman, M.F. (2011). Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm. *Applied Soft Computing Journal*. 11 : 5782–5792.

Zhang, G.H., Shao, X.Y., Li, P.G. and Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computer & Industrial Engineering*. 56 : 1309-1318.

Zhang, R. A differential evolution algorithm for job shop scheduling problems involving due date determination decision.