Journal of Industrial and Systems Engineering
Vol. 10, No. 3, pp 92-112
Summer (July) 2017

Hybrid algorithms for jobshop scheduling problem with lot streaming and a parallel assembly stage

Parviz Fattahi[1*], Fatemeh Daneshamooz[2]

[1]*Department of Industrial Engineering, Alzahra University, Tehran, Iran*
[2]*Department of Industrial Engineering, Bu-Ali Sina University, Hamedan, Iran*
p.fattahi@alzahra.ac.ir, f.daneshamooz@gmail.com

Abstract

In this paper, a Job shop scheduling problem with a parallel assembly stage and Lot Streaming (LS) is considered for the first time in both machining and assembly stages. Lot Streaming technique is a process of splitting jobs into smaller sub-jobs such that successive operations can be overlapped. Hence, to solve job shop scheduling problem with a parallel assembly stage and lot streaming, decision makers not only need to determine the processing sequences on machines in first stage, but also need to assign each product to a machine and determine the assembly sequences of the products in second stage and the sub-lot sizes of all jobs and products to minimize the makespan. At first, this problem is modeled as a mixed integer linear programming and GAMS software is applied to solve small problems. Since this problem is classified as NP-hard, four hybrid algorithms based on iterative procedures are suggested to solve the problem in medium and large dimensions. In order to verify the effectiveness of the proposed algorithms, a statistical analysis is used along with Relative Percentage Deviation (RPD) factor. Computational results revealed that the hybrid genetic and parallel simulated annealing algorithm (HGAPSA) and the hybrid genetic and parallel variable neighborhood search algorithm (HGAPVNS) perform better than the other proposed algorithms with respect to the objective function. Also, considering lot streaming for both stages instead of applying it only to the first stage leads to achieve better solutions. Finally, the HGAPSA algorithm is compared with a hybrid genetic algorithm (HGA). Experimental results showed that the HGAPSA outperforms the HGA in terms of solution quality.

Keywords: Job shop, Parallel Assembly, Lot streaming, HGAPSA, HGAPVNS.

1- Introduction

Scheduling is one of the most important issues in the planning of production systems (Cummings and Egbelu, 1998). Because of strong competition and limitation of resources in our environment, scheduling is a very important decision-making process in production and service industry (Maleki-Darounkolaei et al., 2012). The jobshop scheduling problem (JSP) has been considered as a notoriously stubborn combinatorial optimization problem since the 1950s (Zhang and Cheng, 2011). In the job shop scheduling problem, a finite set of jobs is processed on a finite set of machines. Each job is characterized by a fixed order of operations, each of which is to be processed on a specific machine for a specified duration. Each machine can process at most one job at a time and once a job initiates processing on a given machine it must complete processing uninterrupted.

*Corresponding author.

The objective of the JSP is to find a schedule that optimizes a specified objective function such as makespan.

Although scheduling for the parts machining and planning for the assembly operations have been independently considered (Yokoyama and Santos, 2005), but this may not lead to best results for the total production system. Hence, scheduling for machining parts should be simultaneously with planning for assembly operations. Adding an assembly stage to the jobshop scheduling problem makes it closer to the real environment. In this regard, the bill of material (BOM) needs to list the components that must be used to assemble products. On the other hand, in most manufacturing systems according to increase factory productivity and efficiency (the number of products produced in a given period of time), the assembly stage can be parallel. A parallel assembly stage is able to reduce bottlenecks and in case of a machine breakdown it can avoid assembly line stoppage. So, considering the parallel stages of production simultaneously in planning and scheduling is very important. Furthermore, in classic jobshop scheduling systems a lot is usually invisible and the entire lot must be completed before being transferred to its successor machine (chenand steiner, 1997). Firstly introduced by Reiter (1966), LS technique is a methodology of splitting a job into a number of smaller sub-jobs such that its successive operations can be overlapped on different stages which increases the material flow between machines and so reduces the makespan. It is clear that under a condition when more than one of each product is needed, determining the batch sizes must be considered.

Lot streaming in jobshop scheduling problem is very useful as many practical production systems can be modeled as JSP. The researches about the jobshop scheduling problem with lot streaming in the literature are not as extensive as the JSP. According to the studies around this problem, most of them presented an iterative procedure to solve the considered problem. This procedure first computes the sub-lot sizes and then determines the sequence of sub-lots on the machines. Dauzere-Peres and Lasserre(1993) presented a model and an iterative procedure in a general jobshop environment with lot steaming in order to improve the makespan. In their suggested procedure, in a first step, optimal sub-lot sizes computed and in the second step, a better sequence computed by solving a standard jobshop scheduling problem with fixed sub-lot sizes. Wagner and Ragatz (1994) studied the jobshop scheduling problem with lot splitting to minimize the number of jobs tardy. They also considered the impact of setup times and the size of the transfer batch on lot splitting performance. Jeong et al. (1999) presented an iterative approach that minimizes the makespan of jobshop scheduling problem with lot streaming. Their formulated problem reflected the real manufacturing environment by considering setup times and due dates. They proposed a modified shifting bottleneck procedure to solve the problem. Buscher and Shen (2009) focused on solving the lot streaming in a jobshop environment. This research is one of the pioneer studies that tried to solve this kind of problems with meta-heuristic algorithms. They presented three-phase algorithm incorporated the predetermination of sub-lot sizes, determination of schedules and the variation of sub-lot sizes based on tabu search. Because iterative procedure is a common method used in order to solve the jobshop scheduling problem with lot streaming, they used an iterative procedure in the first phase and a hierarchical approach in the second and third phase. Lei and Guo (2013) considered lot streaming problem in a jobshop with consistent sub-lots and transportation. A modified artificial bee colony algorithm is proposed to minimize makespan. They also applied an effective two phase decoding procedure.Demir and Isleyen (2014) presented a flexible jobshop scheduling problem with overlapping in operations. They developed a new mathematical model. Also a genetic algorithm with an effective chromosome representation and new decoding methodology is proposed.

The first paper about the two-stage scheduling problems, published in less than 30 years ago by lee et at. (1966). This problem has many applications in industry, and therefore has received increasing attention from many researchers (Seyedi, Maleki-Daronkolaei and Kalashi, 2012; Fattahi, Hosseini and Jolai, 2013; Al-Anzi and Allahverdi, 2013; Navaei et al., 2014; Xiong, Xing and Wang, 2015). Chan et al. (2008), for the first time, extended the application of lot streaming to assembly job shop problem. In their model the objective function is to minimize inventory and lateness costs. Also they presented an efficient algorithm based on the genetic algorithm and simple dispatching rules. In the other paper presented by Chan et al. (2009), the previous study of LS to AJSP extended by allowing part sharing among distinct products. The objective function of the proposed model is to minimize the lateness cost. In addition to the use of simple dispatching rules, they proposed an evolutionary approach with genetic algorithm to solve the problem. Wong et al. (2009) investigated a resource-constrained assembly jobshop scheduling problem with lot streaming technique. The objective of the presented model is to minimize the total lateness cost of all final products. They also used common part ratio and workload index to enhance the model usefulness. Since the complexity of AJSSP is NP-hard, an innovative approach with genetic algorithm is suggested in order to solve the problem. Wong and Ngan (2013) studied the assembly jobshop scheduling with lot steaming. In this study, objective function is to minimize the makespan time. They also considered part sharing ratio (PSR) and system congestion index (SCI) to differentiate product-specific components from general components and creating different starting conditions of the shop floor. In order to solve the problem, they proposed a hybrid genetic algorithm (HGA) and a hybrid particle swarm optimization (HPSO). Daneshamoz et al. (2013) proposed a mixed integer linear programming model for jobshop scheduling with a parallel assembly stage in order to minimize makespan. Also they suggested a particle swarm optimization algorithm to solve problem. Their results showed that suggested algorithm can reach to near optimal solutions in various dimensions of problem. Yao and Sarin (2014) addressed a lot steaming problem for a two-stage assembly system involving multiple lots with the objective of minimizing the makespan. They proposed a branch and bound based methodology for this problem that relies on effective lower bounds and dominance properties. Mohammadi (2016)addressed a multi-objective job shop scheduling problem with considering an assembly stage and lot streaming. The objective function is to minimize the makespan and total weighted earliness and tardiness penalties. A mixed integer linear programming model is presented in this research. Also, a meta-heuristic algorithm based on a Harmony Search is proposed to solve the problem. Nejati et al. (2016) considered a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint and lot streaming to minimize the sum of weighted completion times of products in each shift. A mixed integer non-linear programming model is introduced to describe the problem. Also, they used a genetic algorithm and simulated annealing to determine the best sequence and scheduling for the problem.

To the best of our knowledge according to the literature review, no one has considered a job shop scheduling problem with lot streaming technique (in both machining and assembly) and a parallel assembly stage so far. Hence, the primary research goal of this paper is to fill the gap. In this paper, a mathematical model for the considered problem is presented with the objective function of minimizing makespan. Since the JSSP is known to be a NP-hard optimization problem (Garey, Johnson and sethi, 1976), so this problem will be more complex by adding a parallel assembly stage and lot streaming technique to the classical job shop scheduling. Therefore, it is necessary to solve the medium and large size instances with effective meta-heuristics to find the optimal or near optimal solutions in a reasonable amount of time.GA has been shown to perform well in mixed (continues and discrete) combinatorial problems. But it easily becomes trapped in local optima (Manar and Shameem 2011). So, this algorithm is used along with algorithms like SA, VNS, PSA and PVNS in order to avoid the local minima problem under iterative procedures. These meta-heuristics enhance the performance of GA by better local searches.

The paper is organized as follows. In section 2 a literature review of the problem is presented. In section 3, the problem is described in detail and the assumptions and mathematical model are presented. Section 4 describes the proposed algorithms. Computational results are reported in section 5. Finally, conclusions and recommendations for future research are described in Section.

2- Problem description

The jobshop scheduling problem with lot streaming and a parallel assembly stage can be defined as a problem which is consists of assignment, sequencing and determining lot sizes of jobs in a 2-stage production system. At the first stage, a set of J jobs $(J_1, J_2, \dots, J_{n_p})$ of P products $(P_1, P_2, \dots, P_p)$are processed on a set of M machine $(M_1, M_2, \dots, M_m)$in the Work Station, then they are stored in the Inventory Station until all lots from the BOM of the same product are completed and at least one unit of the final product can be produced. Each job of each product has $h_{j,p}$ operations and $s_{j,p}$ sub-lots that must be processed with processing time $ps_{j,p,h}$.Then the assembly operation will be started in a Parallel Assembly Stage for final product assembly if at least one machine is unemployed at the second stage. Otherwise they will be sent to the Inventory Station. Each sub-lot of each product must be assembled with assembly time A_p on a machine from set of M' machines $(M'_1, M'_2, \dots, M'_m)$. Figure 1 shows the layout of the considered production system.

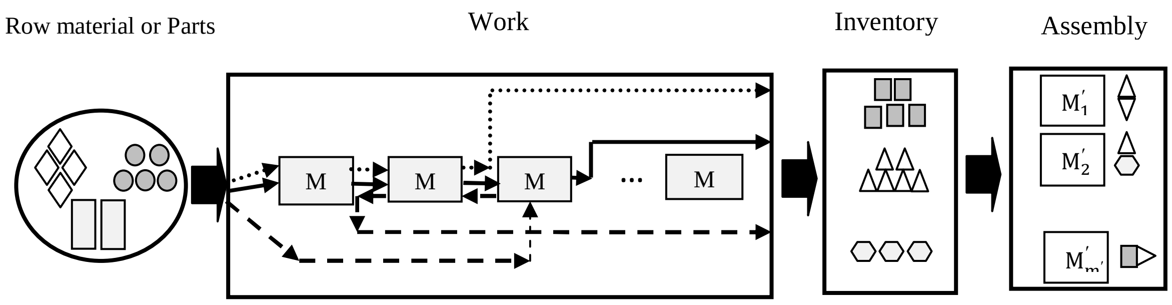


Figure 1. The layout of the considered problem

Basic assumptions of this problem are provided below:

- ✓ There is no machine breakdown.
- ✓ No preemption of operations (sub-lots) is allowed.
- ✓ Each machine can process only one job (sub-lot) at a time.
- ✓ Each job can be processed by only one machine at a time. Each job has a fixed processing route which traverses all the machines in a predetermined order.
- ✓ Demand for final products is specified and all jobs are available at time zero.
- ✓ A typical BOM of a product defines the assembly relationship between the root components (jobs or lots) in the system.
- ✓ Time processing of jobs and assembly of products is deterministic.
- ✓ When all lots from the BOM of the same product are completed in the Work Station, the assembly operation can be started for a product.

In this paper, the following notations are used to formulate the problem:

Parameters

P	Total number of products
p	Products index ($p = 1, 2, \dots, P$)
n	Total number of parts
j	Part index ($j = 1, 2, \dots, n_p$)
n_p	Number of sub-jobs of product p
m	Total number of machines at stage 1
i	Machine index at stage 1 ($i = 1, 2, \dots, m$)
m'	Total number of machines at stage 2
i'	Machine index at stage 2 ($i' = 1, 2, \dots, m'$)
s	Sub-lots index ($j = 1, 2, \dots, S_{j,p}$)
$O_{s,j,p,h}$	Operation h of sub-lot s of product p's job j
k_i	Number of operations dedicated to machine i
$ps_{\text{j,p,h}}$	Processing time of operation $h$of job $j$of product p
A_p	Assembly time of product$p$
$S_{j,p}$	Number of sub-lots of job j of product p
S'_p	Number of sub-lots of product p
$Q_{j,p}$	Lot size of job j of product$p$
DM_p	Demand for product p
$R_{j,p}$	Ratio of job j to the product p

L	A large number
$a_{i,j,p,h}$	1 if $O_{s,j,p,h}$can performed on machine i; 0 otherwise

Decision variables

C_{max}	Makespan
$F_{s,j,p}$	Completion time of sub-lots of job j of product p
$F'_{s,p}$	Maximum completion time of sub-lots of product p
$\mathrm{C_p}$	Completion time of product p
$t_{s,j,p,h}$	Start time of the processing of operation$\mathrm{O_{s,j,p,h}}$
$\mathrm{St_{s',p}}$	Starting assembly time of sub-lot s' of product p
$\mathrm{Tm_{i,k}}$	Start of working time for machine $i$in priority k
$\mathrm{Sm_{i',k'}}$	Start of working time for machine i'in priority k'
$\mathrm{x_{i,s,j,p,h,k}}$	1 if operation $\mathrm{O_{s,j,p,h}}$ is preformed on machine i in priority k ; 0,otherwise
$\mathrm{Z_{i',s',p,k'}}$	1 if sub-lot s' of product p is assembled on machine $\mathrm{i'}$ in priority $\mathrm{k'}$; 0, otherwise
$Q'_{\mathrm{s,j,p,h}}$	Size of sub-lot s of job j of product p
$V_{s',p}$	Size of sub-lot s' of product p
$\mathrm{\delta_{s,j,p,h}}$	1 if $Q'_{\mathrm{s,j,p,h}}$ be more than zero(positive) ; 0, otherwise
$\mathrm{\gamma_{s',p}}$	1 if $V_{s',p}$ be more than zero(positive) ; 0, otherwise

Model formulation

$$Min\, Z = (C_{max}) \qquad (1)$$

Subject to:

$$c_{max} \geq C_p \qquad \forall p \qquad (2)$$

$$t_{s,j,p,h} + ps_{j,p,h}.Q'_{s,j,p,h} \leq t_{s,j,p,h+1} \qquad \forall p,j,s; h = 1,2,\ldots,H-1 \qquad (3)$$

$$t_{s,j,p,h} + ps_{j,p,h}.Q'_{s,j,p,h} \leq t_{s+1,j,p,h} \qquad \forall p,j,h; s = 1,2,\ldots,s_{j,p}-1 \qquad (4)$$

$$Tm_{i,k} + ps_{j,p,h}.Q'_{s,j,p,h}.x_{i,s,j,p,h,k} \leq Tm_{i,k+1} \qquad \forall p,j,s,h,i;\ \mathrm{k} = 1,2,\ldots,k_i-1 \qquad (5)$$

$$Tm_{i,k} \leq t_{s,j,p,h} + \left(1 - x_{i,s,j,p,h,k}\right).L \qquad \forall p,j,s,h,k,i \qquad (6)$$

$$Tm_{i,k} + \left(1 - x_{i,s,j,p,h,k}\right).L \geq t_{s,j,p,h} \qquad \forall p,j,s,h,k,i \qquad (7)$$

$$\sum_p \sum_j \sum_s \sum_h x_{i,s,j,p,h,k} \leq 1 \qquad \forall k,i \qquad (8)$$

$$\sum_k x_{i,s,j,p,h,k} = a_{i,j,p,h}\,.\,\delta_{s,j,p,h} \qquad \forall p,j,h,i,s \qquad (9)$$

$$t_{s,j,p,h} + ps_{j,p,h}.Q'_{s,j,p,h} \leq F_{s,j,p} \qquad \forall p,j,s,h \qquad (10)$$

$$F_{s,j,p} \leq F'_{s,p} \qquad \forall j,p,s \qquad (11)$$

$$F'_{s,p} \leq \mathrm{St_{s',p}} \qquad \forall p,s,s' \qquad (12)$$

$$\mathrm{St_{s',p}} \leq \mathrm{St_{s'+1,p}} \qquad \forall p,s' \qquad (13)$$

$$A_p.V_{s',p} + \mathrm{St_{s',p}} \leq C_p \qquad \forall p \qquad (14)$$

$$Sm_{i',k'} + A_p.\mathrm{Z_{i',s',p,k'}}.V_{s',p} \leq Sm_{i',k'+1} \qquad \forall p,i',s';\ k'=1,2,3,\ldots,k'_{i'}-1 \qquad (15)$$

$$Sm_{i',k'} \leq \mathrm{St_{s',p}} + \left(1 - \mathrm{Z_{i',s',p,k'}}\right).L \qquad \forall p,k',i',s' \qquad (16)$$

$$Sm_{i',k'} + \left(1 - \mathrm{Z_{i',s',p,k'}}\right).L \geq \mathrm{St_{s',p}} \qquad \forall p,k',i',s' \qquad (17)$$

$$\sum_{i'}\sum_{k'} \mathrm{Z}_{\mathrm{i}',s',\mathrm{p},\mathrm{k}'} = \gamma_{s',p} \qquad \forall p, s' \tag{18}$$

$$\sum_{s=1}^{s_{j,p}} Q'_{s,j,p,h} = Q_{j,p} \qquad \forall p, j \tag{19}$$

$$Q'_{s,j,p,h} \leq Q_{j,p}.\delta_{s,j,p,h} \qquad \forall p, j, s, h \tag{20}$$

$$\delta_{s,j,p,h} \leq Q'_{s,j,p,h} \qquad \forall p, j, s, h \tag{21}$$

$$Q_{j,p} = DM_p.R_{j,p} \qquad \forall p, j \tag{22}$$

$$\delta_{1,j,p,h} = 1 \qquad \forall p, j, h \tag{23}$$

$$\delta_{s+1,j,p,h} \leq \delta_{s,j,p,h} \qquad \forall p, j, s, h \tag{24}$$

$$\sum_{s'} V_{s',p} = DM_p \qquad \forall p \tag{25}$$

$$V_{s',p} \leq \gamma_{s',p}.DM_p \qquad \forall p, s' \tag{26}$$

$$\gamma_{s',p} \leq V_{s',p} \qquad \forall p, s' \tag{27}$$

$$Q'_{1,j,p,H} \geq V_{1,p}.R_{j,p} \qquad \forall p, j, s, s' \tag{28}$$

$$\sum_{ss=1} Q'_{ss,j,p,H} - \sum_{ii=2} V_{ii-1,p}\,.R_{j,p} \geq V_{s',p}.R_{j,p} \qquad \forall p, j, s' = 2, \dots, S'_p, s' = s \tag{29}$$

$$\sum_{ss=2} Q'_{ss-1,j,p,H} - \sum_{ii=2} V_{ii-1,p}\,.R_{j,p} < V_{s',p}.R_{j,p} \qquad \forall p, j, s' = 2, \dots, S'_p, s' = s \tag{30}$$

$$\gamma_{1,p} = 1 \qquad \forall p \tag{31}$$

$$\gamma_{s'+1,p} \leq \gamma_{s',p} \qquad \forall p, s' \tag{32}$$

$$x_{i,s,j,p,h,k}\ ,\mathrm{Z}_{\mathrm{i}',s',\mathrm{p},\mathrm{k}'}\ ,\delta_{s,j,p,h}, \gamma_{s',p} \in \{0,1\} \tag{33}$$

$$C_p \geq 0 \qquad \forall p \tag{34}$$

The objective function (1) indicates minimization of completion time for all products. Constraint (2) expresses that the makespan is not smaller than the completion time of any product. Constraint (3) enforces each job to follow a predefined processing sequence. Constraint (4) ensures precedence relationship between sub-lots. Constraint (5) expresses that a machine at stage one, is not able to process an operation in priority k+1, before processing the operation in priority k. Constraints (6) and (7) force each operation$O_{s,j,p,h}$, can be start after its assigned machine is idle and previous operation $O_{s,j,p,h-1}$ is completed. Constraints (8) and (9) define the sequence restrictions. Constraint (10) shows the maximum processing time of each sub-lot of each of job for each product. Constraints (12) and (13) define the earliest start time of assembly stage. Constraint (14) shows the completion time of products. Constraint (15) expresses that a machine in the assembly stage, first assembles the parts of product in priorityk', then it starts assembling the parts of product in priority k'+1.Constraints (16) and (17) imply that assembling each sub-lot of a product in the second stage, can start when its assigned machine is idle and previous product is completed. Constraint (18) assigns each sub-lot of a product to a priority at second stage. Constraint (19) shows that, the total amounts dedicated to the sub-lots of a product's job, is equal to that job's lot size. Constraint (20) ensures that, while the amount of $Q'_{\mathrm{s,j,p,h}}$ is not zero, the binary variable $\delta_{\mathrm{s,j,p,h}}$ will be equal to one. But if $Q'_{\mathrm{s,j,p,h}}$ become zero, the binary variable $\delta_{\mathrm{s,j,p,h}}$ will be equal to zero according to the constraint (21). Constraint (22) defines the amount of each product's jobs (parts) according to the order and the number of jobs needed in each product. Constraints (23) and (24) define the presence of sub-lots in the first stage. Constraint (25) shows that, the total amounts dedicated to the sub-lots of a product, is equal to that product's demand. Constraint (26) forces the binary variable $\gamma_{s',p}$to take the value 1 if the sublot size $V_{s',p}$is greater than zero. If the sublot size$V_{s',p} = 0$, the binary variable $\gamma_{s',p}$ is forced to take the value 0 by the constraint in equation (27).Constraints (28-30) also define the size of sub-lots in the assembly stage. Constraints (31) and (32) define the presence of sub-lots in the second stage. Constraint (33) enforces the binary requirements of the decision variables. Constraint (34) implies that the completion time of products must be positive. According to the constraints (5) and (15) this model is nonlinear. In these two constraints, a binary variable is multiplied in a continuous variable. So a linearization method is used as below.

$$Tm_{i,k} + V_{i,s,j,p,h,k} \leq Tm_{i,k+1} \qquad \forall p, j, s, h, i;\ \mathrm{k} = 1,2,\ldots,k_i - 1 \tag{35}$$

$$V_{i,s,j,p,h,k} \leq ps_{j,p,h}.\, Q'_{s,j,p,h} \qquad \forall p, j, s, h, i, \mathrm{k} \tag{36}$$

$$V_{i,s,j,p,h,k} \leq L.\, x_{i,s,j,p,h,k} \forall p, j, s, h, i, \mathrm{k} \tag{37}$$

$$V_{i,s,j,p,h,k} \geq ps_{j,p,h}.\, Q'_{s,j,p,h} - L.\,(1 - x_{i,s,j,p,h,k}) \qquad \forall p, j, s, h, i, \mathrm{k} \tag{38}$$

$$V_{i,s,j,p,h,k} \in \{0,1\} \forall p, j, s, h, i, \mathrm{k} \tag{39}$$

3- Proposed Algorithms

According to the proposed model and its several constraints and also due to the complexity of this type of problem, it requires much time to be solved by exact algorithms. So, utilization of new approaches and meta-heuristics is necessary to solve the problem in medium to large scales.The proposed algorithms are based on GA, SA,VNS, PSA and PVNS to benefit from their advantages.The iterative optimization procedure of the proposed algorithms can be seen in Figure 2. According to this procedure, the primary problem determines lot sizes and the secondary problem specifies the sequences.

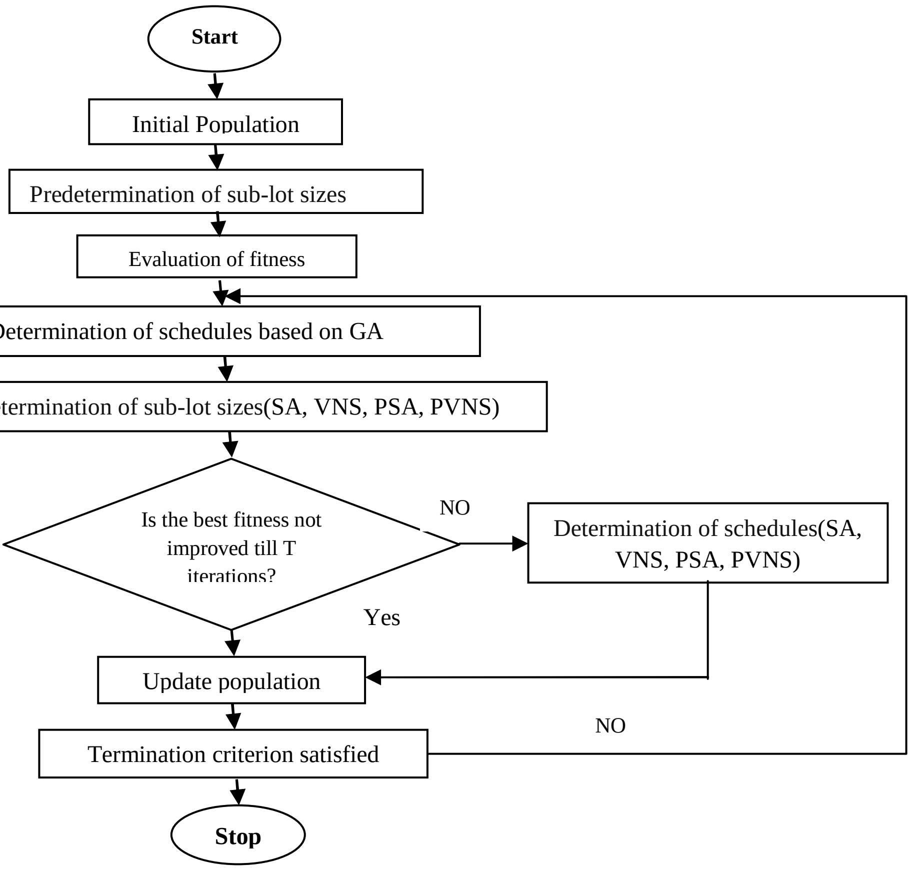

Start
Initial Population
Predetermination of sub-lot sizes
Evaluation of fitness
Determination of schedules based on GA
Determination of sub-lot sizes(SA, VNS, PSA, PVNS)
Is the best fitness not improved till T iterations?
NO
Determination of schedules(SA, VNS, PSA, PVNS)
Yes
Update population
NO
Termination criterion satisfied
Stop

Figure 2. Optimization procedure of the proposed algorithms.

3-1- Hybrid genetic algorithm and parallel variable neighborhood search

Variable neighborhood search (VNS) is a recent meta-heuristic which exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them. In order to reduce the computational time or increase the exploration in the search space, a parallelization strategy for VNS is applied (Yazdani, Amiri and Zandieh, 2010). Therefore, according to the exploitation ability of genetic algorithms and exploration capability of parallel variable neighborhood search, a new hybrid algorithm is presented in this paper. In this proposed algorithm, each member of the population must be produced as chromosomes in GA. Then, in each generation of GA after computing the fitness of all solutions, a specified percentage of solutions which have the best fitness function, will be saved. Then they will be improved by PVNS. In the first step of PVNS a set of neighborhood structures and the sequence of their implementations are determined. Also, the current solution ỹequals to the initial solution y. The main part of the algorithm is comprised of internal and external loops. The internal loop is responsible for searching the solution space, whereas the external loop controls the stop condition of algorithm. There are a number of processors in the internal loop which are used in search process. In each iteration of the internal loop, every processor performs a single iteration of a sequential VNS method including shake and local search procedures. After all processors complete a set of runs, one generation gets finished and updating step is then commenced. In the updating step, the best solution (y'') is identified among obtained solutions of processors.If y'' is better than $\tilde{y}$,y'' replaces with $\tilde{y}$($\tilde{y} \leftarrow y''$). And search process begins again at the first iteration of the internal loop ($k \leftarrow 1$). Otherwise, one unit is added to the iteration counter. On the condition that all iterations of internal loop are considered, one iteration of PVNS algorithm is accomplished. In this case, stop condition is checked by the external loop. Figure 3 shows the steps of the PVNS algorithm. The variable neighborhood search algorithm performs like PVNS but it has only one processor ($h = 1$).

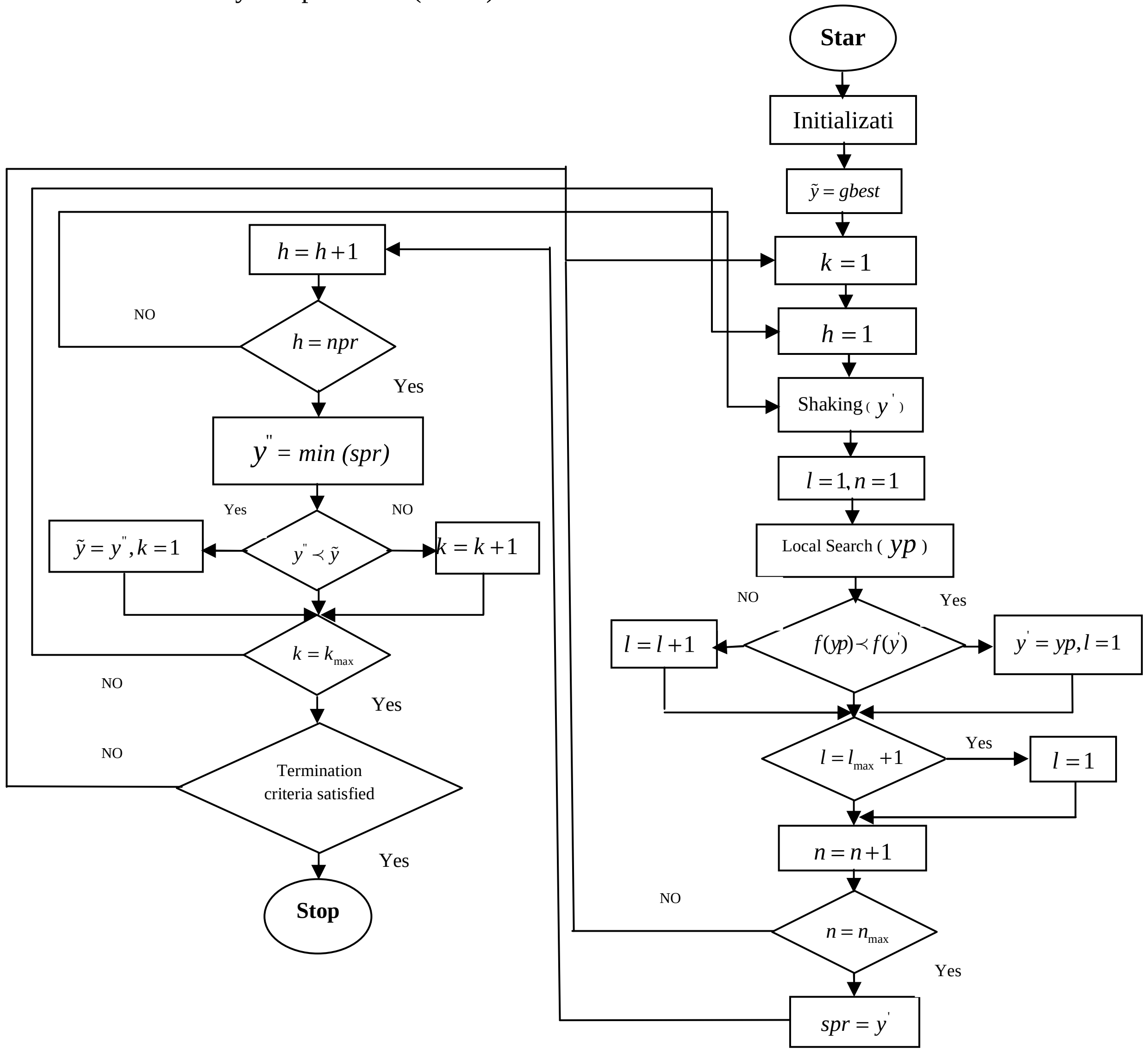


Figure 3. Flowchart of the proposed PVNS

3-1-1- Solution representation

The solution representation has a critical role on the performance of the algorithms to search the different parts of solution space. The representation of each member of population has two parts. First part represents the sequence of operations on machines in jobshop scheduling problem and the second part represents the product assignments to the machines in assembly stage. Figure 4 illustrates this representation.

According to the Figure 4, job 1 of product 1 has two operations and each operation has two sub-lots. So, numbers 1 and 2 are dedicated to job 1 and because this job has 2 operations, they occur exactly two times. So, the first and the second sub-lot of the first operation of job 1 are in the third and twelfth priority to be process and the first and the second sub-lot of the second operation are in the sixth and fifteenth priority. Also, in order to assign the products to the machines, a string containing numbers between $[1,(P \cdot S'_p + M' - 1)]$ will be generated. P represents the total number of products,$S'_p$the total number of sub-lots, and M' represents the number of assembly machines.When the generated number is larger than$(P \cdot S'_p)$, itstands forthe assembly machine.According to the figure 4, the first and the second sub-lots of product 2will be assembled in the first and third priority on the first assembly machine

j	2	2	1	2	1	1	1	2	3	2	3	1	2	1	1	1
S	1	1	1	1	1	1	1	2	1	2	2	2	2	2	2	2
p	2	2	1	1	3	1	2	2	1	1	1	1	2	2	1	3

Stage1	9	9	1	3	11	1	7	10	5	4	6	2	10	8	2	12

Work Station

Stage 2	2	3	2	4	1	1	3

Assembly Stage

Figure 4. Example of a solution representation

3-1-2- Initial solution

In order to generate the initial solutions a randomized procedure is used in the Work Station. Also in the assembly stage, any product which it's all corresponding jobs is processed sooner, will be assembled sooner. After selecting the product with high priority, it will be assigned to a machine with less workload. The machines workload will be updated after each assignment.

3-1-3- Selection operator

In the current study, a roulette wheel is used as a selection operator. Also a simulated annealing algorithm is used in case of selecting chromosomes and copying them to new population. In this method, each individual of new population and previous population are compared to each other. If the new individual's fitness function be better or equal to the previous one, it will be accepted. Otherwise, it may have a chance to participate in the next generation. The approach of this selection operator is to gradually move the selection criteria from exploration to exploitation. In other words, selection pressure increases by decreasing temperature.

3-1-4- Crossover operator

Spear and De Jong (1991) investigated different crossovers and indicated that uniform crossover gives the best results. Based on their study, the uniform crossover has more exploratory power than the n-point crossover. Therefore, the uniform crossover is used in this paper. In the uniform crossover, a random vector containing numbers between [0,1] is generated. The offspring 1 is produced by taking the bit from parent 1 if the corresponding mask bit is 1 or the bit from parent 2 if the corresponding mask bit is 0 and the offspring 2 takes the bit from parent 1 if the corresponding mask bit is 0.

3-1-5- Mutation operator

In genetic algorithm, mutation is as a random deformation of genes with a certain probability. The mutation operator preserves diversification in the search. In this study, two random position of the string are chosen and the bits corresponding to those positions are interchanged.

3-1-6- Neighborhood structure

Neighborhood structure is a mechanism which can obtain new neighborhood solutions. Each neighbor solution is generated from a given solution by a move. A good neighborhood structure should eliminate unnecessary moves. In this subsection, the neighborhood structures used in different phases of the considered problem are defined.

1) Neighborhood structures for sequencing phase

In this paper, seven neighborhood structures are used. In the first neighborhood structure, one of the critical blocks with more than one operation, will be chosen. If the first operation is selected, it will be replaced by second or third operation. And if the last operation is chosen, it will be replaced by k-1_{th} or k-2_{th} operation. K represents the total number of operations of the selected block. If the selected block has only two operations, they will be replaced with each other. In the second type of neighborhood structure, two operations will be chosen randomly and their values will be replaced. In addition, it will interchange value of the genes between these two selected operations. The third neighborhood structure selects a gene randomly and inserts it in a randomly selected place. In the next structure two genes are randomly chosen. Then their sub-lots will be defined and replaced with each other. As it is illustrated in figure 5, the fifth neighborhood structure attempts to shift an operation within a critical block. The new position this operation in the block is determined as the most distanced insert position that still yield feasible schedule. Also, the operations inside a critical block are only shifted to the first or last operation of the block. In the next structure, additional to the two operations at the beginning (or the end) of a critical block, the directly preceding (or directly succeeding) operations on a machine is also considered. A new neighbor will be generated by changing the order of three consecutive operations in a condition that two operations at the beginning (or at the end) are reversed. This neighborhood structure is illustrated in figure 6.The last neighborhood structure moves each operation of a critical block after the last or before the first operation of the block. This neighborhood structure is shown in figure 7.

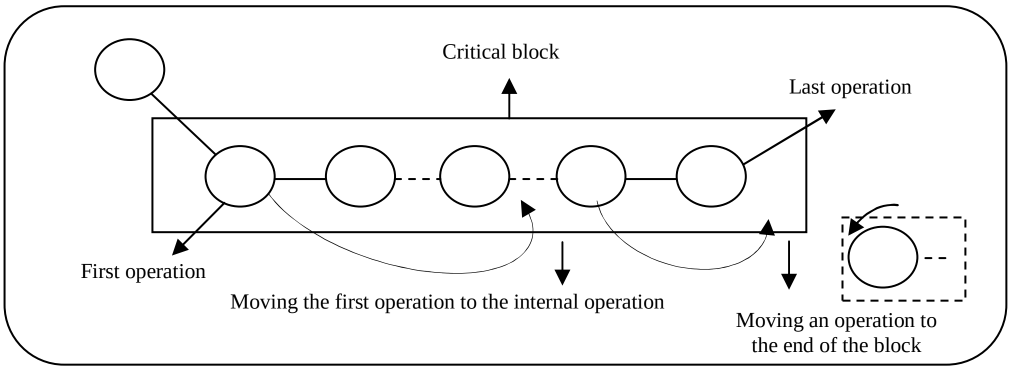

Critical block
Last operation
First operation
Moving the first operation to the internal operation
Moving an operation to the end of the block

Figure 5. The fifth neighborhood structure (Zhang et al., 2007)

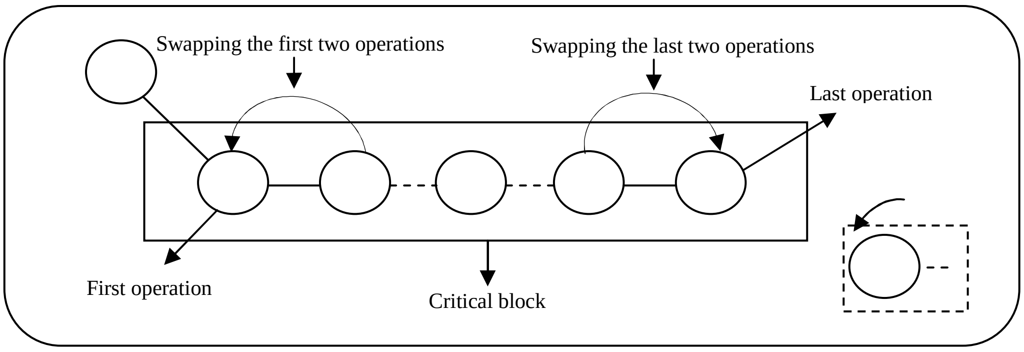

Swapping the first two operations
Swapping the last two operations
Last operation
First operation
Critical block

Figure 6. The sixth neighborhood structure (Zhang et al., 2007)

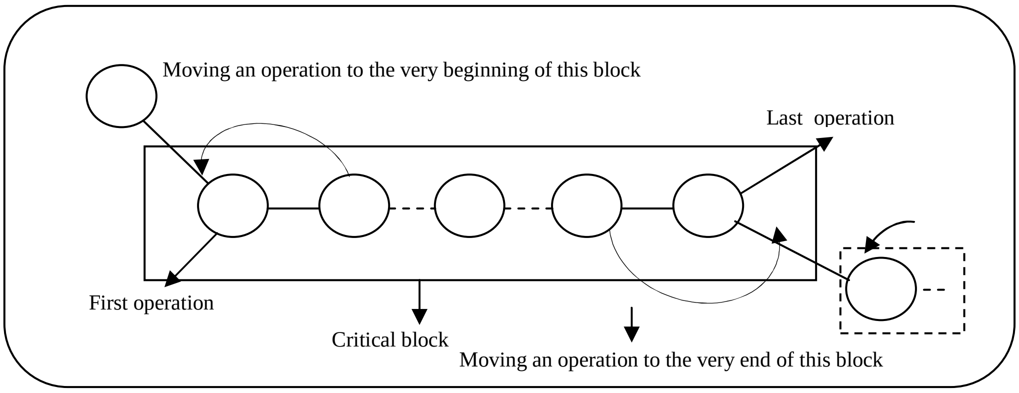

Moving an operation to the very beginning of this block
Last operation
First operation
Critical block
Moving an operation to the very end of this block

Figure 7. The last neighborhood structure (Zhang et al., 2007)

2) Neighborhood structures for assignment phase

In this algorithm, three neighborhood structures will be applied on assembly problem. In the first structure, one of the products corresponding to the assembly machine with more workload will be chosen randomly and will be added to a machine with less workload. The second neighborhood structure selects an assembly machine randomly and prioritizes its corresponding products based on the best completion time. The last structure selects a product with the largest processing time and assigns it to an assembly machine with less workload.

3) Neighborhood structures for sub-lot sizes

In the proposed algorithm, three neighborhood structures will be applied on sub-lot sizes. In the first neighborhood structure a block is selected among all of the critical blocks and an operation corresponding to that block will be chosen randomly. Then the selected operation’s sub-lots are randomly chosen and their values will be replaced. The second neighborhood structure selects an operation and one of its sub-lots randomly and then a specified percentage of that sub-lot’s volume (size) will be transferred to another sub-lot of the selected operation. The third neighborhood structure is based on a procedure which is presented by Bouscher and Shen (2009).

3-2- Hybrid genetic algorithm and simulated annealing

Simulated annealing algorithm was first proposed by Metropolis at 1953 and it was developed by Kirkpatricket al. (1983). The algorithm simulates the cooling process by gradually lowering the temperature of the system until it converges to a steady state. Simulated annealing is known as a significant algorithm to solve the combinatorial optimization problems due to its simplicity and high effectiveness. This algorithm avoids trapping in local optima. Also, it is amongst the most powerful neighborhood search techniques. Simulated annealing algorithm, first founds a neighbor around the initial solution. If the new neighbor solution has a smaller cost than the old solution, it will be saved as the base for its next generation. Otherwise, the algorithm calculates an acceptance probability (p) and then compares it to a uniform random number between 0 and 1.

$$p = \exp\left(-\frac{\Delta f}{KT}\right) \tag{40}$$

Δf is the difference of objective values between the new solution and the current one and T is current temperature. If the calculated value (p) be more than the random number, then the new neighbor solution should be accepted.

In this proposed algorithm, the fitness function of the individuals will be computed in each iteration of genetic algorithm and finally the solutions obtained by GA, will be improved by SA. This algorithm performs like the HGAPVNS in details.

3-3- Hybrid genetic algorithm and parallel simulated annealing

One of the main disadvantages of simulated annealing algorithm is its slowness. Parallelism in SA reduces computation times. There are different approaches to parallelize SA algorithm. Since the quality of SA's performance is usually affected by the starting solution (Onbasoglu et al, 2001), in this paper PSA starts the search from a group instead of a single initial solution. So it searches multiple reigns in the solution space. PSA algorithm founds a neighbor around each initial solution. The neighbors will be compared with their corresponding initial solutions. And in case of improving they will be replaced with the initials. But if no improvement in solution is detected, the algorithm will perform the same as simulated annealing. This is the only different between HGASA and HGAPSA algorithms.

4- Computational Results

In this section, the performance and effectiveness of the proposed algorithms are investigated. The mathematical model is solved by GAMS IDE (ver.23.5) software and the proposed algorithms are coded with MATLAB R2013a software. In order to evaluate the performance of proposed algorithms, 20 instances are generated randomly. The instances are categorized into three groups (small, medium and large-sized problems) based on the number of operations. For each instance the algorithms are replicated 10 times. The random samples are generated from the distributions in table 1.Since the parameters of meta-heuristic algorithms have a great influence on their effectiveness, choosing the best parameters for them is important. In this paper, Taguchi method has been used to determine the appropriate values for key parameters of the proposed algorithms.

Table 1. Data for generating test problems

Parameters	Ranges
P	{2,3,…,14}
n	{2,3,…,7}
m	{2,3,…,7}
m'	{2,3,4,5}
$ps_{j,p,h}$	Uniform Distribution
A_p	Uniform Distribution

For comparing the effectiveness of algorithms, Relative Percentage Deviation (RPD) factor is used. RPD factor is formulated as (41) and compares the performance of each procedure to other procedures for each instance:

$$\text{RPD} = \frac{\text{Alg}_{\text{sol}} - \text{Min}_{\text{sol}}}{\text{Min}_{\text{sol}}} * 100 \tag{41}$$

Also the improvement in initial solutions is measured using a well-known criterion.IMP factor is computed as equation (42).

$$Imp = \frac{Alg_{initialsol} - Alg_{finalsol}}{Alg_{finalsol}} * 100 \tag{42}$$

On the other hand, a factor named $D_{f^*}$is used to determine the mean deviation of the best solution. This factor is computed according to equation (43).

$$D_{f^*} = \frac{\sum_{i=1}^{n}(f_i - f^*)}{n.f^*} \tag{43}$$

Where $f^*$is the solution obtained by the mathematical model for small problems. Also for medium and large problems it is the best solution obtained by the algorithms.

4-1- Analysis

In order to validate the proposed model, the optimal solutions for small problems are obtained by GAMS and they are reported in table2.In small problems according to the RPD factor, the algorithms achieved the same results but the computational time of these algorithms are considerably lower than the time obtained by GAMS. According to computational times, the HGASA algorithm performs better than the others. Furthermore, GAMS was not able to obtain the optimal solution for medium and large problems, in an acceptable time. For this reason, an upper bound of the optimal solution is calculated for the medium size problems. It is obtained from GAMS software and presented in table 2.

According to the RPD values with 95% confidence interval for medium and large problems in Figures8 and 9, the HGAPSA gains the lowest average among the other algorithms in all the instances. Clearly, lower values of RPDs are preferred. Therefore, the HGAPSA performs better than the other algorithms statistically. Also the HGAPVNS outperforms HGASA and HGAVNS. Moreover, due to the figure 10, the HGAPVNS in all instances needs more computing time than the other algorithms.

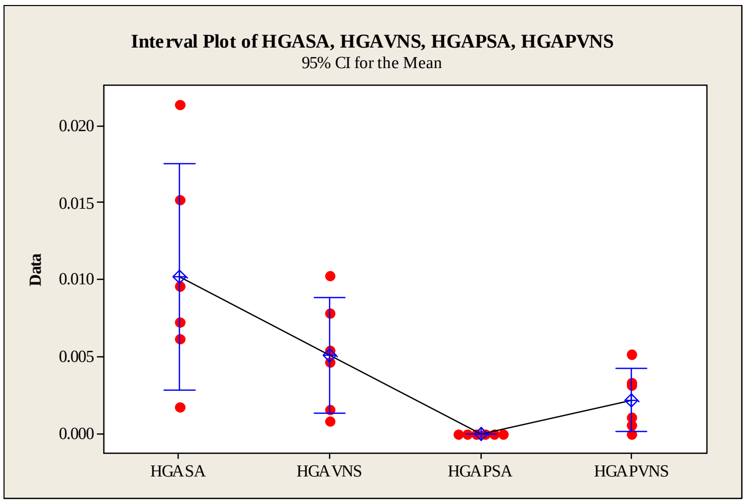


Figure 8. Interval plot of RPD value for medium problems

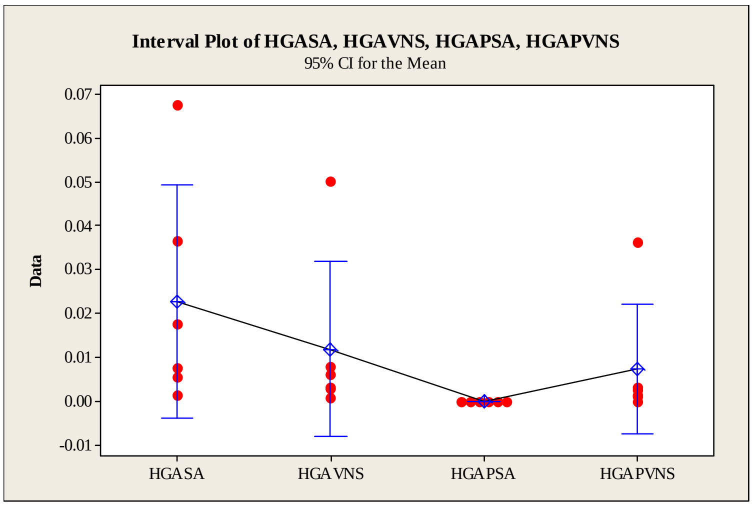


Figure 9. Interval plot of RPD value for large problems

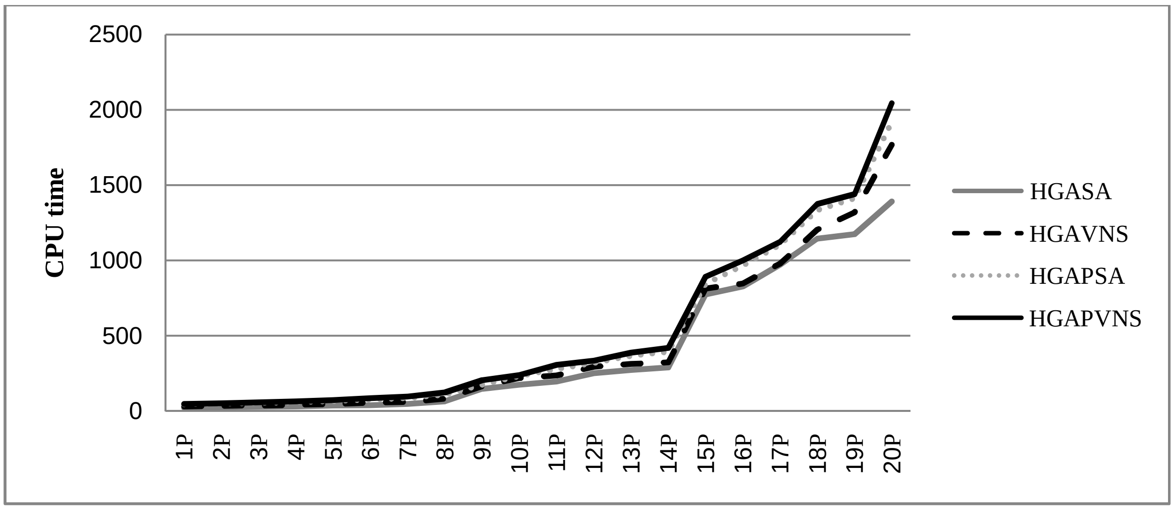
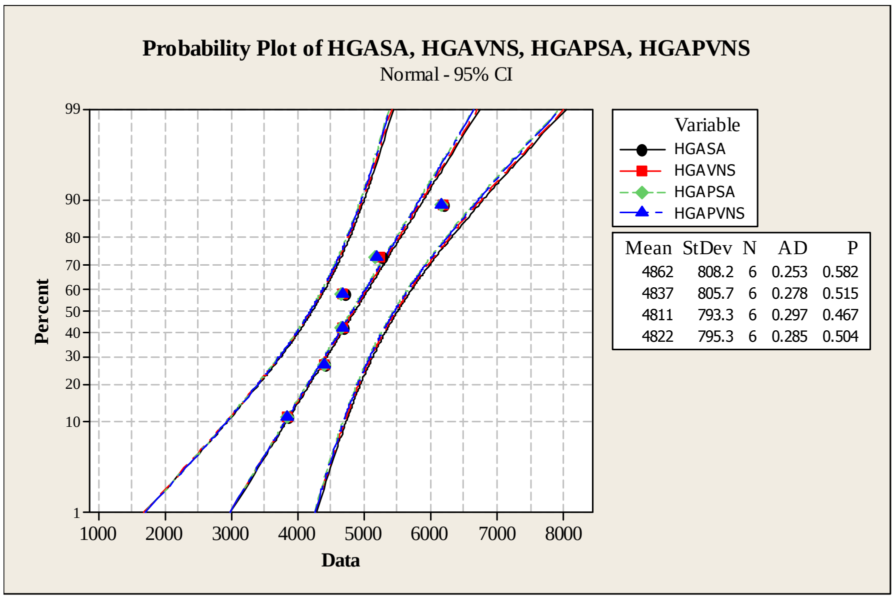


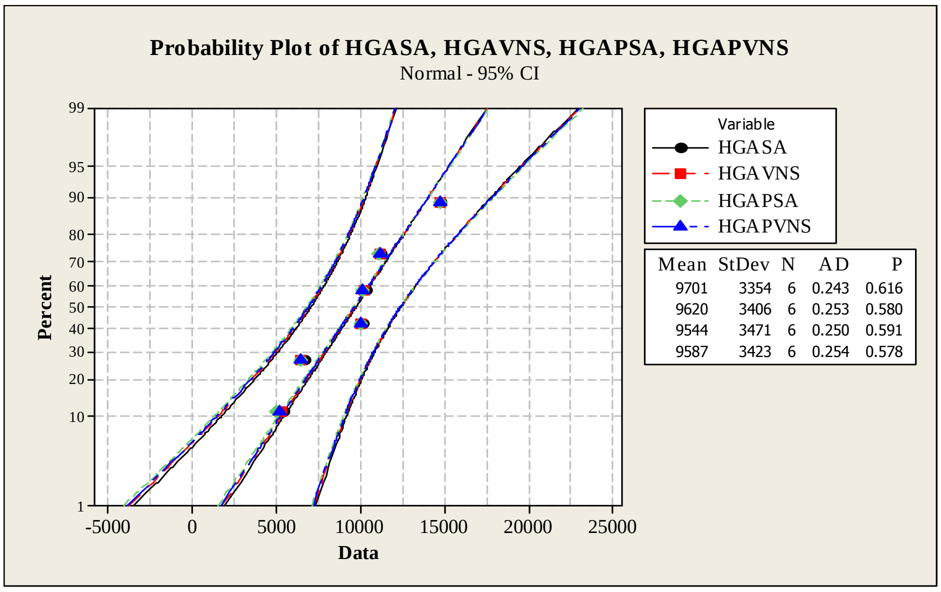
Figure 10. Computational time of HGASA, HGAVNS, HGAPSA and HGAPVNS algorithms

A statistical test is proposed in this paper to compare the ability of algorithms at finding the best solution. Since the size of the problems is different with each other and it will have some effects on the results, it must be controlled by including blocking factor in the experiment. Due to the considered problem, a randomized complete block design seems to be a proper test. First, the hypothesis of normality should be checked for HGASA, HGAVNS, HGAPSA and HGAPVNS values. The test is performed using Minitab software. Figures 11 and 12 show the result of normality test for HGASA, HGAVNS, HGAPSA and HGAPVNS in medium and large problems respectively. These tests are based on the ANDERSON-DARLING method with0.05 significance level. According to Figures11 and 12the significance level is less than the p-values. So, all algorithms for medium and large problems have normal distribution. Also, in order to select a proper test, the variances of HGASA, HGAVNS, HGAPSA and HGAPVNS are tested. The results revealed that the variances are equal and so it is allowed to use the randomized complete block design.



Mean	StDev	N	AD	P
4862	808.2	6	0.253	0.582
4837	805.7	6	0.278	0.515
4811	793.3	6	0.297	0.467
4822	795.3	6	0.285	0.504

Figure 11. Normality test for medium problems



Mean	StDev	N	AD	P
9701	3354	6	0.243	0.616
9620	3406	6	0.253	0.580
9544	3471	6	0.250	0.591
9587	3423	6	0.254	0.578

Figure 12. Normality test for large problems

The considered randomized complete block design for both medium and large problems is consist of 4 treatments and 6 blocks. The blocks are size of problems and the treatments are the algorithms. Table 3 illustrates the results of the test for medium and large problems. Since the p-value is less than 0.05, the null hypothesis is rejected. Therefore pairwise comparisons must be applied on the algorithms. The Least Significant Difference (LSD) test performs the paired comparisons between algorithms. The results of the LSD test in table 4 for both medium and large problems show that the p-value between the HGASA and other algorithms is less than 0.05, so they have significant differences. According to the means and standard deviations, all of the algorithms perform better than HGASA. Also, there is no convincing evidence that the HGAPVNS is different from the HGAVNS and HGAPSA. Moreover with considering the computational times, means and standard deviations, the HGAPSA is superior to the HGAPVNS. Also the HGAPVNS achieves better solutions than the HGAVNS. But it needs more computational time.

Table 3. Results of blocking test for medium and large problems

	Source	Sum of Squares	df	Mean Square	F	P-value
medium	factor	8641.125	3	2880.375	9.476	.001
	block	12816244.208	5	2563248.842	8432.433	.000
	Error	4559.625	15	303.975		
	Total	573409115.000	24			
large	factor	79612.611	3	26537.537	7.198	.003
	block	233005862.719	5	46601172.544	12640.038	.000
	Error	55301.856	15	3686.790		
	Total	2450923323.290	24			

Table 4. Results of pairwise comparisons on the algorithms (medium and large problems)

Algorithm	Algorithms	P-value (medium)	P-value (large)
HGASA	HGAVNS	.025	.035
	HGAPSA	.000	.000
	HGAPVNS	.001	.005
HGAVNS	HGASA	.025	.035
	HGAPSA	.023	.046
	HGAPVNS	**.157**	**.360**
HGAPSA	HGASA	.000	.000
	HGAVNS	.023	.046
	HGAPVNS	**.313**	**.239**
HGAPVNS	HGASA	.001	.005
	HGAVNS	**.157**	**.360**
	HGAPSA	**.313**	**.239**

The improvement in the initial solutions is another factor which is used to analyze the performance of the proposed algorithms. When the solution space is small, it is predictable that good algorithms can meet the optimal solution in the first iteration but by increasing the size of the problem, the probability of finding the best solution in the first iteration is very low. So, more improvement is a sign of a good algorithm. As it can be seen in figure 13, the IMP% of HGAVNS is more than the other algorithms in medium problems. But by expanding the search space, the improvement in initial of HGAPSA increases. Also, lack of improvements in small problems indicated that the algorithms can meet the optimal solution in the first iteration. Furthermore, in order to analyze the effectiveness and robustness of the proposed algorithms, the means and standard deviations are examined. Computational results in table 2 showed that, the scatter of the mean and standard deviation of the HGAPSA algorithm is less than the other algorithms. So, the HGAPSA has robust performance.

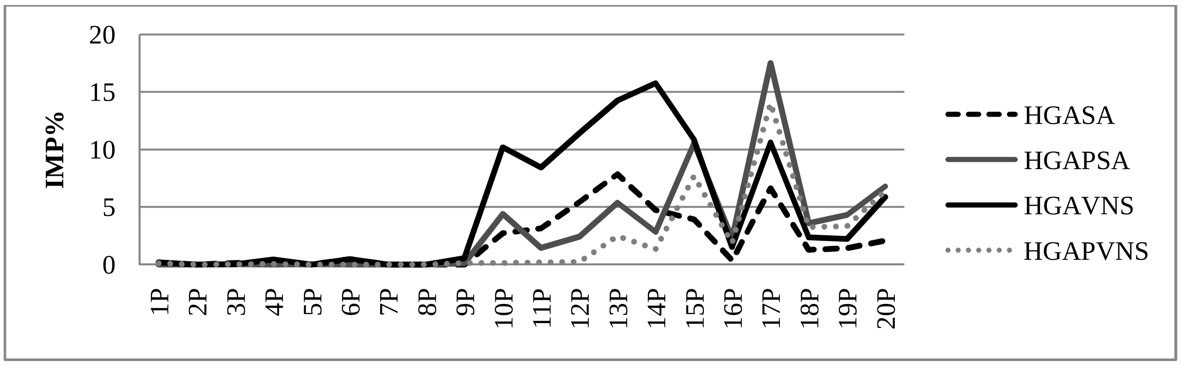


Figure 13. Improvements in initial solution obtained by the algorithms

For all of the algorithms, the summarized averages of $D_{f^*}$are shown in figures 14. It is concluded that, increasing in the size of problems leads to increaseD_{f^*}.Figure 14 indicates that for small problems, all of the algorithms are capable to attain the optimal solution. According to figure14 for the medium and large problems, the HGAPSA algorithm has better performance than the other algorithms and the HGASA has a poor performance. On the other hand, the convergence of HGASA, HGAVNS, HGAPSA and HGAPVNS is shown in figure 15. From the Figure, the best objective value found by the algorithms was compared across generations. The results revealed that the convergence of HGAPSA is relatively better than the other algorithms.

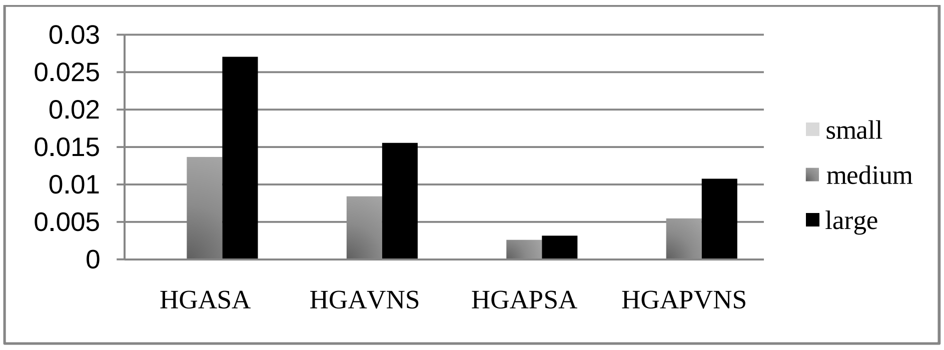


Figure 14. Averages of $D_{f^*}$for all of the problems

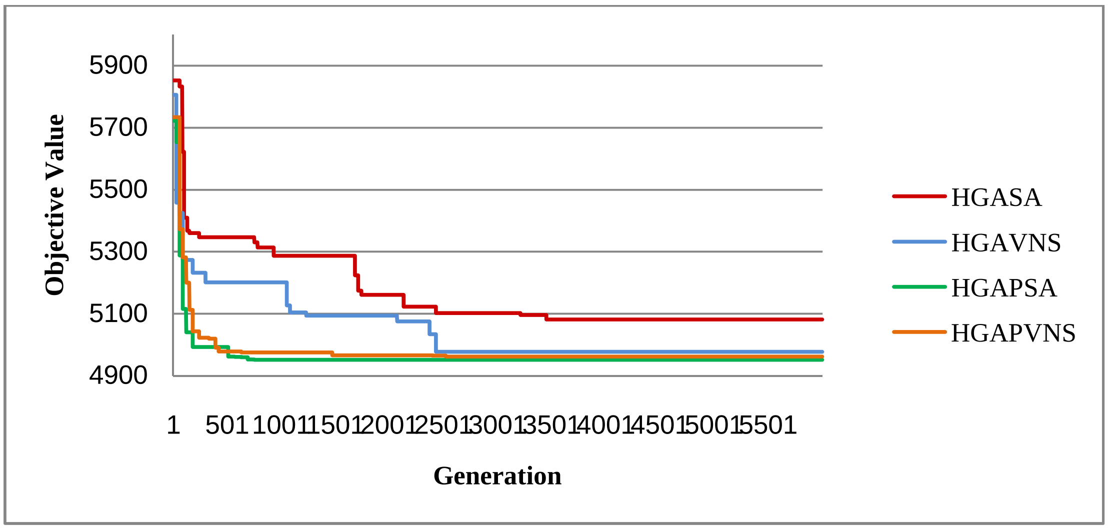


Figure 15. Convergence of the proposed algorithms

On the other hand, to investigate the impact of lot streaming on makespan, the results of this problem were compared to the results of problems without lot streaming and lot streaming only at the first stage. As illustrated in Figure 16, the problem with lot streaming at both stages outperforms the problems under the other considered conditions.

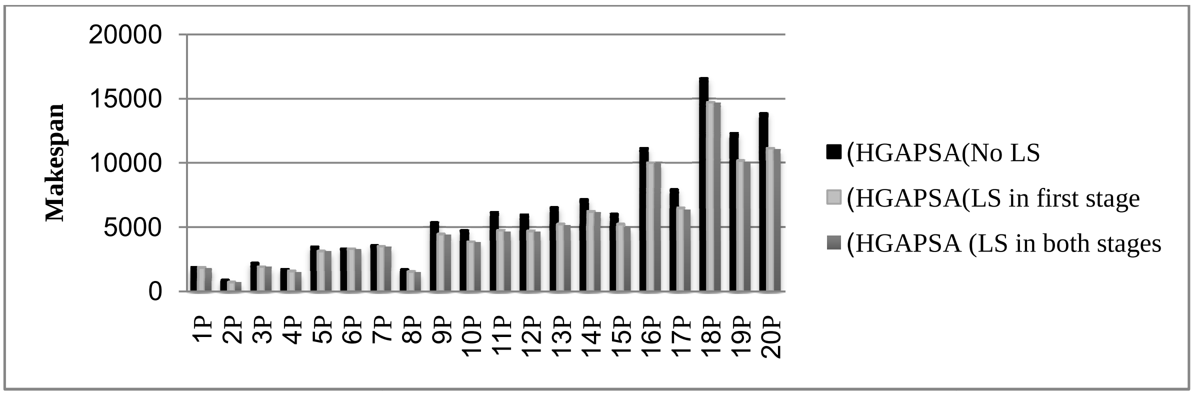


Figure 16. The HGAPVNS algorithm under different conditions

Finally in order to validate the performance of the HGAPSA algorithm, it is compared with a hybrid genetic algorithm (HGA) proposed by Wong and Ngan (2013).In this algorithm; a 3D solution representation is used. According to figure 17, by increasing the size the problem, the HGA achieves poorer solutions under the same conditions.

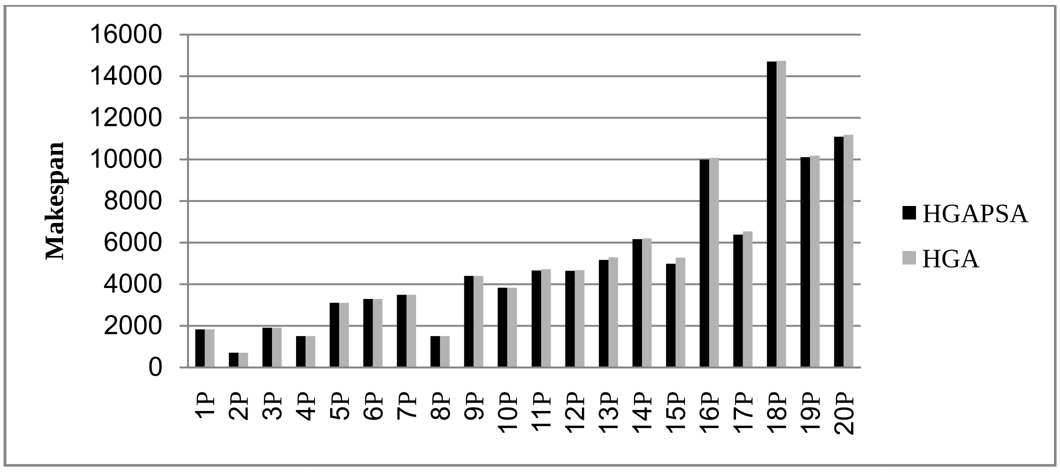


Figure 17. Makespan of the HGAPSA and HGA

5- Conclusion

This paper addressed a jobshop scheduling problem with a parallel assembly stage and lot streaming at machining and assembly stages. A mixed integer linear programming model is presented and it attempts to minimize the makespan time. According to the complexity of this problem, four hybrid algorithms based on Genetic algorithm, Simulated Annealing (SA), Variable Neighborhood Search (VNS), Parallel Simulated Annealing (PSA)and Parallel Variable Neighborhood Search (PVNS) were used to solve the problem in an iterative procedure. The results revealed that for small problems, the HGASA algorithm has approximately the same performance with the other algorithms but according to the computational times, the HGASA has better performance. Then a randomized complete block design considered comparing the ability of algorithms at finding the best solution for medium and large problems. Computational results revealed that for medium and large problems the HGAPSA algorithm outperforms the HGAVNS and HGASA algorithms and there is not a significant difference between HGAPSA and HGAPVNS. However, with considering the computational time, means and standard deviations the HGAPSA is superior to the HGAPVNS. Furthermore, in case of analyzing the robustness of the proposed algorithms, the means and standard deviations are examined. It is concluded that the HGAPSA has robust performance for the problem. Also, according to the RPD values with 95% confidence interval for medium and large problems, the HGAPSA performs better than the other algorithms statistically. On the other hand, to determine the mean deviation of the best solutions, a factor named$D_{f^*}$is used.The results revealed that by increasing the size of problem D_{f^*} increases.For small problems, all of the algorithms are capable to attain the optimal solutions. But for medium and large problems, the HGAPSA algorithm outperforms the other algorithms and the

HGAPVNS algorithm cannot guarantee the best results, but it is more suitable than HGASA and HGAVNS. Also, the result of the HGAPSA with lot-streaming at both stages was compared to the results of this problem without LS and lot streaming only at the first stage. The results indicated that the HGAPSA with lot-streaming at both stages performs better than the others. Finally in order to validate the HGAPSA algorithm, it is compared with a hybrid genetic algorithm. Results showed that the HGAPSA performs better than the HGA. Considering setup times, intermediate buffer and uncertainty for the parameters of the problem is suggested as future researches. Also, the results can be improved by developing the effective meta-heuristic algorithms and implementing more efficient and better solution representations and neighborhood structures.

Table 2. Computational results

Problem		**Cplex**		**HGASA**			**HGAVNS**			**HGAPSA**			**HGAPVNS**		
		CPU time	**RPD**	**Mean C_{max}**	**Std C_{max}**	**CPU time**	**Mean C_{max}**	**Std C_{max}**	**CPU time**	**Mean C_{max}**	**Std C_{max}**	**CPU time**	**Mean C_{max}**	**Std C_{max}**	**CPU time**
Small	P1	7.2	0	1825	0	21.3	1825	0	29.2	1825	0	44.1	1825	0	47.1
	P2	189.5	0	701	0	24.9	701	0	33.7	701	0	48.8	701	0	52.4
	P3	217.4	0	1910	0	27.6	1910	0	35.9	1910	0	53.5	1910	0	58.7
	P4	335.9	0	1504	0	30.3	1504	0	40.8	1504	0	59.4	1504	0	63.3
	P5	398.3	0	3115	0	35.5	3115	0	46.0	3115	0	65.3	3115	0	72.2
	P6	445.6	0	3300	0	38.1	3300	0	53.8	3300	0	76.7	3300	0	85.6
	P7	865.1	0	3500	0	46.4	3500	0	59.3	3500	0	84.3	3500	0	96.1
	P8	1203	0	1501	0	64.1	1501	0	80.1	1501	0	107.2	1501	0	122.9
Medium	P9	7200	0.06	4410	8	145.8	4406	6.1	162.4	4403	3.4	174	4403	5.6	203.8
	P10	7200	0.17	3850	17.5	174.9	3832	12.0	218.7	3827	8.9	231.9	3828	9.2	239.4
	P11	7200	0.12	4726	21.4	194.7	4676	30.4	234.8	4654	5.1	277.9	4669	9.3	306.8
	P12	7200	0.07	4694	12.3	251.4	4686	17.9	292.1	4650	6.5	317.6	4673	7.4	334.1
	P13	7200	0.19	5282	20.7	272.9	5223	24.3	311.5	5169	12.2	365.8	5187	14.8	387.9
	P14	7200	0.05	6209	34.1	290.0	6198	26.8	323.9	6165	10.7	392.4	6171	13.6	419.6
Large	P15	- - -	- - -	5349	53.4	773.9	5251	45.3	813.0	4989	27.0	843.4	5175	34.5	892.1
	P16	- - -	- - -	10044	36	827.5	10021	29.9	847.0	9989	14.9	965.5	10002	19.2	1000.2
	P17	- - -	- - -	6629	49.4	971.3	6438	31.3	979.2	6387	5.6	1104.0	6407	9.9	1122.7
	P18	- - -	- - -	14725	19.3	1143.8	14718	7.2	1203.3	14704	3.5	1332.9	14705	3.6	1374.7
	P19	- - -	- - -	10284	30.1	1173.5	10131	32	1318.7	10102	15.4	1411.6	10112	19.9	1440.0
	P20	- - -	- - -	11175	47.1	1391.3	11160	42.5	1767.7	11092	35.7	1920.6	11119	38.4	2043.9

References:

Al-Anzi, F.S. &Allahverdi, A. (2013). An artificial immune system heuristic for two-stage multi-machine assembly scheduling problem to minimize total completion time. Journal of Manufacturing Systems, 32 (4), 825– 830.

Buscher, U.,&Shen, L. (2009). An integrated tabu search algorithm for the lot streaming problem in job shops.European Journal of Operational Research, 199 (2), 385-399.

Cummings, D.H.,&Egbelu, P.J. (1998). Minimizing production flow time in a process and assembly jobshop. International Journal of Production Research, 36(8), 2315–2332.

Chan, F.T.S., Wong, T.C., &Chan, L.Y. (2008). Lot streaming for product assembly in job shop environment. Robotics and Computer-Integrated Manufacturing, 24 (3), 321–331.

Chan, F.T.S., Wong, T.C., &Chan, L.Y. (2009). An evolutionary algorithm for assembly job shop with part sharing. Computers & Industrial Engineering, 57 (3), 641–651.

Chen, J., &Steiner, G. (1997). Lot streaming with detached setups in three-machine flow shops. European Journal of Operational Research. 96(3), 591-611.

Daneshamooz , F., Jabbari, M., &Fattahi, P. (2013). A model for jobshop scheduling with a parallel assembly stage to minimize makespan. Journal of Industrial Engineering Research in Production Systems, 2(4), 39-53.

Dauzere-Peres, S.,&Lasserre, J.B. (1993). An iterative procedure for lot streaming in job-shop scheduling. Computers & Industrial Engineering, 25 (4), 231-234.

Demir, Y., &Isleyen, S.K. (2014). An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations. International Journal of Production Research, 52(13), 3905-3921.

Eschelman, L., Caruana, R., &Schaffer, D. (1989). Biases in the crossover landscape. Proc. Third international conference on genetic algorithms, Morgan Kaufman Publishing, 21-29.

Fattahi, P., Hosseini, S.M.H., &Jolai, F. (2013). Amathematical model and extension algorithm for assembly flexible flow shop scheduling problem. International Journal of Advanced Manufacturing Technology, 65 (5),787-802.

Garey, M.R., Johnson, D.S., &sethi, R. (1976). The Complexity of flow shop and job shop scheduling. Mathematics of Operation Research, 1 (2), 117-129.

Jeong, H., Park, J., &Leachman, R.C. (1999). A batch splitting method for a job shop scheduling problem in an MRP environment. International Journal of Production Research, 37 (15), 3583-3598.

Krikpatrick, S., Gelatt, C.D., &Vecchi, M.P. (1983). Optimization by Simulated Annealing. Science, 220 (4598), 671-680.

Lee, C.Y., Cheng, T.C.E., &Lin, B.M.T. (1993). Minimizing the makespan in the 3-machine assembly-type flow shop scheduling problem. Management Science, 39 (5), 616-625.

Lei, D., &Guo, X. (2013). Scheduling job shop with lot streaming and transportation through a modified artificial bee colony. International Journal of Production Research, 51(16), 4930-4941.

Maleki-Darounkolaei, A., Modiri, M., Tavakkoli-Moghadam, R.,&Seyyedi, I. (2012). A three-stage assembly flow shop scheduling problem with blocking and sequence depended setup times. Journal of Industrial Engineering International, 8:26.

Manne, A.S. (1960).On the job shop scheduling problem. Operational Research, 8 (2), 219-223.

Mohammadi,E. (2016).Multi objective job shop scheduling problem with an assembly stage and lot streaming .Master of Science Thesis, Buali-Sina University.

Navaei, J., Fatemi-Ghomi, S.M.T., Jolai, F., &Mozdgir, A. (2014). Heuristics for an assembly flowshop with non-identical assembly machines and sequence dependent setup times to minimize sum of holding and delay costs. Computers & Operations Research, 44, 52–65.

Nejati, M., Mahdavi, I., Hassanzadeh, R., &Mahdavi-Amiri, N. (2016). Lot streaming in a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint. International Journal of Production Research, 33(7), 459-471.

Reiter, S. (1966). A system for managing job-shop production. Journal of Business, 39 (3), 371–393.

Seyedi, I., Maleki-Daronkolaei, A., &Kalashi, F. (2012). Tabu search and simulated annealing for new three-stage assembly flow shop scheduling with blocking. Interdisciplinary Journal of Contemporary Research In Business, 4 (8), 394-402.

Spears, W. M., &De Jong, K. A. (1991). On the virtues of uniform crossover. Proceedings of the Fourth International Conference on Genetic Algorithms, 230-236.

Wagner, B.J.,&Ragatz, G. (1994). The impact of lot splitting on due date performance. Journal of Operations Management, 12 (1), 13-25.

Wagner, H. (1959). An integer linear-programming model for machine scheduling. Naval Research logistics Quarterly, 6 (2), 131-140.

Wong, T.C., Chan, F.T.S., &Chan, L.Y. (2009). A resource-constrained assembly job shop scheduling problem with Lot Streaming technique. Computers & Industrial Engineering, 57 (3), 983–995.

Wong, T.C.,&Ngan, S.C. (2013). A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop. Applied Soft Computing, 13(3), 1391–1399.

Xiong, F., Xing, K., &Wang, F. (2015). Scheduling a hybrid assembly-differentiation flow shop to minimize total flow time. European Journal of Operational Research, 240, 338-354

Yao L.,&Sarin, C.S. (2014). Multiple-Lot Lot Streaming in a Two-stage Assembly System. Essays inProduction Project Planning and Scheduling, Springer US.

Yazdani, M., Amiri, M., &Zandieh, M. (2010). Flexible job shop scheduling with parallel variable neighborhood search algorithm. Expert system with applications, 37(1), 678-687.

Yokoyama, M., &Santos, D.L. (2005). Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times. European Journal of Operational Research, 161, 754–770.

Zhang , C.Y. , Li, P., Guan, Z., Rao, Y. (2007). A tabu search algorithm with a newneighborhood structure for the job shop scheduling problem. Computers & Operations Research, 34 (11), 3229-3242.

Zhang, R.,&Cheng, W. (2011). A simulated annealing algorithm based on blocking properties for the jobshop scheduling problem with total weighted tardiness objective. Computer and operation research, 38 (5), 854-867.