Computers & Operations Research 155 (2023) 106222

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A MIP model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting

Busra Tutumlu [a,*], Tugba Saraç [b]

[a] Kütahya Dumlupınar University, Faculty of Engineering, Department of Industrial Engineering, 43100 Kütahya, Turkey
[b] Eskişehir Osmangazi University, Faculty of Engineering and Architecture, Department of Industrial Engineering, 26480 Eskişehir, Turkey

ARTICLE INFO

Keywords:
Flexible Job Shop Scheduling Problem
Job-Splitting
Mixed Integer Programming
Hybrid Genetic Algorithm
Local Search Algorithm

ABSTRACT

In the scheduling literature, it is generally assumed that jobs are not split into sub-lots, or that the number and size of sub-lots are limited or predetermined. These assumptions make the problem more manageable. However, they may prevent more successful schedules. For many businesses, considering the splitting of jobs while scheduling them can create significant improvement opportunities. This study addresses the Flexible Job-Shop Scheduling Problem (FJSP) with job-splitting, determining how many sub-lots each job should be split into and the size of each sub-lot. A MIP model is proposed for the considered problem. In the model, the size and number of sub-lots of a job are not predefined or bounded. The objective function of the model is to minimize the makespan. Feasible solutions could not be found for large-sized problems by the mathematical model. So, a Hybrid Genetic Algorithm (HGA) is also proposed. In the proposed HGA, a Local Search Algorithm (LSA) that determines the size of sub-lots has been included in the GA to improve the efficiency. To show the success of the proposed HGA, its performance is compared with the classical GA.

1. Introduction

Today, many businesses increase their product range by developing new products according to customer demands to survive competitively. For this reason, businesses expanding their product range prefer job shop manufacturing among manufacturing processes. In job shop-type manufacturing, a job may require more than one operation. The operations of a job are performed on certain machines. Flexible job shop manufacturing is a type of job shop manufacturing in which more than one machine can do the same job, and thus the job routes become flexible. It provides essential services to businesses, such as shortening the completion times of jobs. However, the scheduling of jobs is much more complicated than in traditional job shop manufacturing.

Job shop scheduling can be described as 'the workload distribution between the machines and the determination of the process sequences of these jobs on their machines' (Stevenson, 1996). In a job shop scheduling problem where there are n jobs and m machines, there are $(n!)^m$ schedules. As the number of jobs and machines increases, schedules increase exponentially, making it difficult to obtain the optimum schedule. Job shop scheduling problems are NP-hard problems (Garey et al., 1976).

In most scheduling problems, it is assumed that jobs cannot be split. However, this assumption is not always very realistic. When businesses cannot deliver their orders on time, they can experience huge costs. Jobs may be split into machines that perform the same operations to avoid these costs. Thus, orders can be completed earlier.

In this study, FJSPs with job-splitting are discussed. It is aimed to examine the effect of splitting the jobs on the makespan. Therefore, a MIP model is proposed to minimize the makespan. The number and size of sub-lots in the model are not predefined or bounded. To show the performance of the proposed model, randomly generated test problems of different sizes are solved using the proposed model with GAMS/Cplex. Feasible solutions of large-sized problems has not been found within a time limit. For this reason, GA has been applied to solve these problems. However, GA may not sufficient for determining the successful size of sub-lots even in the small-sized test problems with the

Abbreviations: ABC, Artificial Bee Colony Algorithm; C_{max}, Minimization of makespan; CP, Constraint Programming; DE, Differential Evolution Algorithm; GA, Genetic Algorithm; HHA, Hyper-Heuristic Algorithm; HS, Heuristic Search Algorithm; ICA, Imperialist Competitive Algorithm; ILS, Iterated Local Search; LSA, Local Search Algorithm; PSO, Particle Swarm Optimization; SAA, Simulated Annealing Algorithm; SLS, Maximization of sub-lot sizes; TEC, Total energy cost; TFT, Minimization of total flow time; TPC, Minimization of total processing cost; TWT, Total weight tardiness; TS, Tabu Search; VNS, Variable Neighborhood Search.

* Corresponding author.
E-mail addresses: busra.tutumlu@dpu.edu.tr (B. Tutumlu), tsarac@ogu.edu.tr (T. Saraç).

https://doi.org/10.1016/j.cor.2023.106222
Received 7 May 2022; Received in revised form 16 March 2023; Accepted 16 March 2023
Available online 24 March 2023

known optimum solution because it randomly searches in a very large space. Therefore, a Local Search Algorithm (LSA) has been developed to successfully determine the size of sub-lots and search the space intelligently. The proposed LSA is included in the GA and is applied instead of the mutation operator in certain cases or to the elite solution of the population at determining iterations. The GA, which includes LSA, has been called a Hybrid Genetic Algorithm (HGA).

The remainder of this study is organized as follows: In Sect. 2, a literature review is given. The problem definition and proposed mathematical models are presented in Sect. 3. The proposed HGA is introduced in Sect. 4. Test problems and computational results are represented in Sect. 5. Finally, in Sect. 6, results and recommendations are presented.

2. Literature review

FJSP is one of the most complex scheduling problems. This problem consists of two sub-problems. These are assigning jobs to machines and sequencing them. Many studies have been published after Brucker and Schlie (1990) first studied the FJSP. The literature of FJSP is so comprehensive and varied. In this respect, a detailed analysis study by Chaudhry and Khan (2016) can be viewed. As seen in this study (Chaudhry and Khan, 2016), the objective function is to minimize the makespan, and the solution method is GA in most studies. Today, this objective function and algorithm are still widely used (see, e.g., (Liang et al., 2019; Yan et al., 2022; Wang and Zhu, 2021; Liu et al., 2021). Amjad et al. (2018) present a comprehensive literature review of the studies using GA in this area. Especially in the last 20 years, the study has grown even more with the addition of research that uses hybrid approaches, such as GA.

In recent years, studies addressing the problem of scheduling in the literature have allowed job-splitting (lot-streaming). The splitting of jobs greatly contributes to increasing the velocity of the flow of material over machines and reducing makespan, cycle time, and work-in-process inventory. Cheng et al. (2013) reviewed the literature on job-splitting, including flow shops, job shops, open shops, and parallel machines, and studies were analyzed in detail. In recent years, Pan et al. (2023), Chen et al. (2020), and Meng et al. (2018) studies can be given as examples of the flow shop scheduling problem. Studies by Saraç and Tutumlu (2022), Zheng et al. (2022), Lee et al. (2021), Kim and Lee (2021), and Liu et al. (2018) can be given as examples of the parallel machine scheduling problem. The studies carried out in the last 10 years on the problem of job-splitting in job shop scheduling are given in Table 1. While giving the problem types in the second column of Table 1, Sarin and Jaiprakash (2006)'s classification method for splitting jobs addressed in scheduling problems is used. In this classification scheme, the categories are, respectively, {machine configuration}{number of machines}/{number of product types}/{sub-lot type}/{idling}/{number of sub-lots}/{sub-lot sizes}/{setup}/{transfer time or removal time}/{objective function}. Machine configuration refers to the arrangement of the machines (flow shop (F), job shop (J), open shop (O), parallel machines (P), etc.), the number of product types shown as a single product (1) or multiple products (n), sub-lot types show as consistent sub-lots (C), variable sub-lots (V) and equal sub-lots (E), idling refers to whether or not idle time is allowed (intermittent idling (II) and no-idling (NI)), the number of sub-lots refers to the number of sub-lots may be known a priori (FixN), or is to be determined (FlexN), sub-lot sizes are either continuous (CV) or discrete (DV) valued and setups show as lot-attached setup (L(a)) or lot-detached setup (L(d)).

As can be seen from Table 1, there are a few studies on the FJSP with job-splitting. Li et al. (2022) proposed a hybrid algorithm to divide the problem into two stages and solve them respectively. It is found by assignment and sequencing in the first stage, and the batch arrangement in the second stage. Li (2022) proposed a mixed-integer linear programming model and hyper-heuristic algorithm for FJSP with variable sublots. In the studies of Li et al. (2022) and Li (2022), the upper and lower number of the sub-lots are determined. Fan et al. (2022) established a monolithic mixed integer linear programming model for FJSP with lot-streaming and machine reconfigurations (FJSP-LSMR). Moreover, a matheuristic method with a variable neighbourhood search component is developed. The number of sublots is bounded to the maximum number. Daneshamooz et al. (2022) decomposed the problem into two sub-problems and proposed two algorithms based on VNS. Abderrabi et al. (2021) proposed a mathematical model for FJSP with job-splitting. Novas (2019) proposed a CP model. The CP enables both the lots' splitting and the scheduling of production tasks. Zhang et al. (2018) proposed a modified binary particle swarm optimization algorithm to tackle the equal-size lot-splitting FJSP problem involving large-scale batch production. Bozek and Werner (2017) proposed a two-stage optimization process for FJSP with lot streaming and lot sizing of variable sub-lots. In the first stage, the makespan is minimized, and in the second stage, sub-lot sizes are maximized. The upper and lower sizes of the sub-lots are determined. Lei and Guo (2013) proposed a modified ABC to minimize makespan. They applied an effective two-phase decoding procedure in which a schedule is first built, and then transportation tasks are dispatched. Liu et al. (2013) examined the influence of a fixed number of sub-lots on the performance of the lot stream by systematically varying this parameter using the fixed number job splitting approach and proposed a GA-based job splitting approach. Wang et al. (2012) proposed an evolutionary algorithm that allows splitting into a certain number of lots in a dyeing facility. Defersha and Chen (2012) proposed a model for FJSP with lot streaming and developed an island-model parallel GA to solve the proposed model. In most of the studies in the table, the properties related to job-splitting are as follows; sub-lot types are shown as consistent (C), the number of sub-lots is known a priori, and sub-lot sizes are discrete (DV) valued.

Table 1
Studies on FJSP with job-splitting in the last 10 years.

Addressed by	Problem	Method
This Study	Jm/n/V/II/FlexN/DV/-/-/C_{max}	GA, LSA
Li et al. (2022)	Jm/n/V/-/FlexN/DV/-/T/C_{max}, TEC	ICA, SAA
Li (2022)	Jm/n/V/-/FlexN/DV/-/T/C_{max}	HHA
Fan et al. (2022)	Jm/n/V/-/FlexN/DV/-/-/TWT	Matheuristic, GA
Daneshamooz et al. (2022)	Jm/n/V/ /FixN/ DV/-/-/C_{max}	VNS
Abderrabi et al. (2021)	Jm/n/V/-/FixN/DV/ L(a)/-/ TFT	GA, ILS
Novas (2019)	Jm/n/C/NI/FlexN/DV/ L(a)/T/ C_{max}	CP
Zhang et al. (2018)	Jm/n/E/-/FixN/DV/-/T/ C_{max}, TPC	PSO
Bozek and Werner (2017)	Jm/n/V/II/FlexN/CV/-/T/ C_{max}, SLS	TS, CP
Lei and Guo (2013)	Jm/n/C/II/ FixN/-/-/T/C_{max}	ABC
Liu et al. (2013)	Jm/n/C/II/FixN/DV/-/-/C_{max}	HS
Wang et al. (2012)	Jm/n/C/II/FixN/-/-/T/TPC	DE
Defersha and Chen (2012)	Jm/n/C/II/FixN/CV/-/-/C_{max}	GA

Explanations of the objective function and solution method abbreviations used in Table 1 are given in Appendix A.

An FJSP with job-splitting is discussed in this study. Unlike in the literature, the numbers and sizes of sub-lots are not known beforehand. The sub-lot properties are variable (V), flexible number (FlexN), and discrete value (DV). Considering the studies in Table 1, there are a few studies that addresses these three properties simultaneously. However, the number of sub-lots is bounded in those studies. In this study, there are not any bounds. In the literature, there are two main approaches to solving FJSP; the hierarchical approach and the integrated approach. Since the hierarchical approach reduces the complexity of the problem

as it approaches the solution by dividing the problem into sub-problems, it is widely used in the literature. Although the integrated approach makes the solution more difficult, this study is preferred because it has great advantages, such as making it possible to reach the optimal solution. This study makes a contribution by proposing for the first time an integrated mathematical model for FJSP with job-splitting which has no bounds on size and number of sub-lots, as well as an HGA in which the LSA decides how jobs are split.

3. Problem definition and mathematical model

In this study, the FJSP, which has m machines, n jobs, and r operations of these jobs, has been addressed. In this problem, a job consists of operations. The order of these operations is certain, which is called the route of that job. The routes should be considered when determining the order in which the operations will be processed on each machine. In classic FJSP, the operations can only be processed to a machine. In FJSP with job-splitting, some operations can be processed on multiple machines and jobs can be assigned to these machines with different sizes of sub-lots. For example, a job with three operations and a lot size of 100 is processed in a job shop with five machines. Machines M2, M3, and M4 are parallel that can process the same operations. Fig. 1 shows a sample job route in the classic FJSP. The job's first operation was completed on the M1. The second operation can be processed on M2, M3, or M4. As can be seen in Fig. 1, the second operation is processed on M2. The final operation is processed on M5.

Fig. 2 shows a sample job route in the FJSP with job-splitting. Since the number and size of sub-lots are not bounded, the job in the second operation may be split into different or equal sizes on M2, M3, or M4. In Fig. 2, the job is split into two sub-lots with sizes 30 on M2 and 70 on M3.

A mathematical model with the objective function of minimizing the makespan is proposed for the FJSP with job-splitting. The assumptions for the proposed model are given as follows:

- All jobs are available for processing at time zero.
- A machine can process at most one job at a time.
- The priorities of all jobs are the same.
- All processing and setup times of the jobs are deterministic.
- Jobs may have more than one operation.
- Each operation needs a setup, and the setup times of operations are different on each machine.
- The routes of jobs are known.
- The job can be split and assigned to different parallel machines with different lot sizes. Lot sizes may be variable.
- If a job is split into sub-lots, a separate setup time is needed for each.

The sets, indices, parameters, decision variables, objective functions, and constraints of the proposed mathematical model are below.

Sets:

$I = \{1, 2, ..n\}$ the job set.

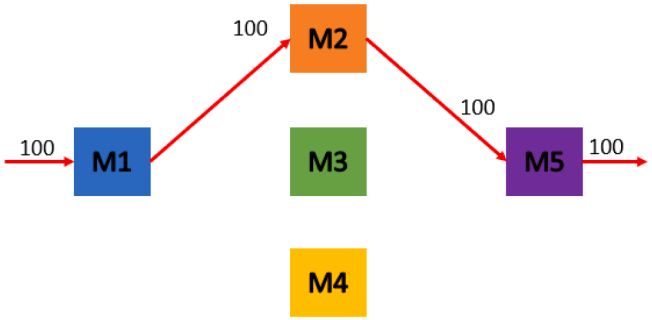

Fig. 1. A sample job route in the classic FJSP.

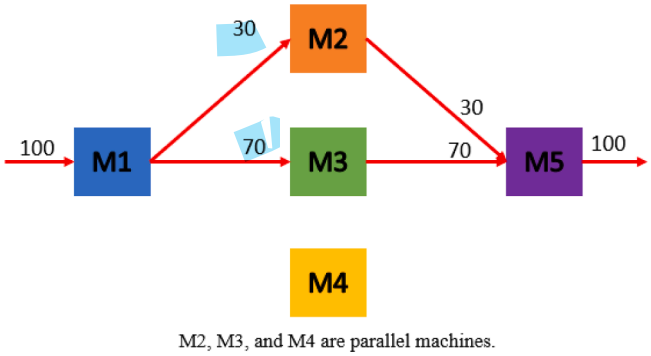

Fig. 2. A sample job route in the FJSP with job-splitting.

$J = \{1, 2, ..r\}$ the operation set.

$K = \{1, 2, ..m\}$ the machines set.

$L = \{1, 2, ..\delta\}$ the sequences set.$\delta = \max_k \left(\sum_i \sum_j u_{ijk}\right)$

Indices:

$i, w \in I$are indices to denote a job.

$l, b \in L$are indices to denote the position of a job in the sequence of its machine.

$j, q \in J$are indices to denote an operation of a job.

$k, a \in K$are indices to denote a machine.

Parameters:

g_i : Lot-size of job i.

p_{ijk} : Unit processing time for operation j of job i on the machine k.

h_{ijk} : Setup time for operation j of job i on the machine k.

o_i : Total number of operations of job i.

$$u_{ijk} = \begin{cases} 1, & \text{if machine } k \text{ is capable of processing operation } j \text{ of job } i \\ 0, & \text{otherwise} \end{cases}$$

M: A large enough positive number.

s: The smallest value for the sub-lot size. (When this value equals zero, there is no bound for the sizes.).

Decision Variables:

C_{ijk} : Completion time of job i in operation j on the machine k.

C_{max} : Makespan.

α_{ijk}: Sub-lot size of job i in operation j on machine k

$$x_{ijkl} = \begin{cases} 1, & \text{if job } i \text{ in operation } j \text{ is processed on machine } k \text{ in the position } l \\ 0, & \text{otherwise} \end{cases}$$

Objective Functions:

$$minf = C_{max} \tag{1}$$

Constraints:

$$C_{ijk} + M\left(1 - x_{ijkl}\right) \geq h_{ijk} + p_{ijk}\alpha_{ijk} \quad \forall_{i,j,k,l;l=1,j=1} \tag{2}$$

$$C_{ijk} + M\left(2 - x_{ijkl} - x_{wqk(l-1)}\right) \geq C_{wqk} + h_{ijk} + p_{ijk}\alpha_{ijk} \quad \forall_{i,j,w,q,k,l;l>1,j=1,i\neq w} \tag{3}$$

$$C_{ijk} + M\left(2 - x_{ijkl} - x_{i(j-1)ab}\right) \geq C_{i(j-1)a} + h_{ijk} + p_{ijk}\alpha_{ijk} \quad \forall_{i,j,k,a,l,b;l=1,j>1,k\neq a} \tag{4}$$

$$C_{ijk} + M\left(2 - x_{ijkl} - x_{wqk(l-1)}\right) \geq C_{wqk} + h_{ijk} + p_{ijk}\alpha_{ijk} \quad \forall_{i,j,w,q,k,l;l>1,w\neq i,j\leq o_i,q\leq o_w} \tag{5}$$

$$C_{ijk} + M\left(2 - x_{ijkl} - x_{i(j-1)ab}\right) \geq C_{i(j-1)a} + h_{ijk} + p_{ijk}\alpha_{ijk} \quad \forall_{i,j,a,b,k,l;j>1,j\leq o_i} \tag{6}$$

$$l - b \geq 1 - M\left(2 - x_{ijkl} - x_{iqkb}\right) \quad \forall_{i,j,q,k,l,b;j\neq q,j\leq o_i,l\neq b} \tag{7}$$

$$\sum_i \sum_{j;j\leq o_i} x_{ijkl} - \sum_w \sum_{q;q\leq o_w} x_{wqk(l-1)} \leq 0 \quad \forall_{k,l;l>1} \tag{8}$$

$$\sum_{k} \alpha_{ijk} = g_i \quad \forall_{i,j\,:\,j \le o_i} \tag{9}$$

$$\alpha_{ijk} \le g_i \sum_{l} x_{ijkl} \quad \forall_{i,j,k;j \le o_i} \tag{10}$$

$$\alpha_{ijk} \ge s \sum_{l} x_{ijkl} \quad \forall_{i,j,k;j \le o_i} \tag{11}$$

$$\sum_{l} x_{ijkl} \le 1 \quad \forall_{i,j,k;j \le o_i} \tag{12}$$

$$\sum_{i} \sum_{j;j \le o_i} x_{ijkl} \le 1 \quad \forall_{k,l} \tag{13}$$

$$x_{ijkl} \le u_{ijk} \quad \forall_{i,j,k,l\,:\,j \le o_i} \tag{14}$$

$$C_{max} \ge C_{ijk} \quad \forall_{i,j,k} \tag{15}$$

$$C_{ijk} \ge 0 \quad \forall_{i,j,k} \tag{16}$$

$$C_{max} \ge 0 \tag{17}$$

$$x_{ijkl} \in \{0,1\} \quad \forall_{i,j,k,l} \tag{18}$$

$$\alpha_{ijk} \ge 0 \, and \, integer \quad \forall_{i,j,k} \tag{19}$$

The objective function is defined by Eq. (1), which minimizes makespan. Eq. (2) is the constraint that ensures the completion time to be calculated if the first operation of a job is assigned to the first sequence. Eq. (3) is the constraint that ensures the completion time of the first operation of a job that is not in the first sequence to be calculated. Eq. (4) is the constraint that ensures the calculation of the completion time of the job other than the first operations of it in the first sequence. Eq. (5) is the constraint that ensures the calculation of the completion time of jobs that are not in the first sequence in a machine. Eq. (6) is the constraint that ensures the calculation of the completion time of the job other than the first operations of it. Eqs. (7) and (8) are the constraints that ensure job operations are processed sequentially. Eq. (9) is the constraint that ensures that job i in operation j is assigned and that the sums of the size of sub-lots at which the jobs are assigned to the parallel machines are equal to g_i. Eqs (10) and (11) are the constraint that ensures the relationship between the α_{ijk} and x_{ijkl} decision variables. Eq. (12) guarantees that each job is assigned to each machine at most. Eq. (13) is the constraint to ensure that at most one job is assigned to each sequence of each machine. Eq. (14) is the machine eligibility constraint. Eq. (15) is the constraint that calculates the makespan. Eqs. (16), (17), (18) and (19) are sign constraints.

4. Proposed hybrid genetic algorithm

FJSP is strongly NP-hard (Fattahi et al., 2007). The splitting of jobs in this problem made it even more complicated. Therefore, a GA has been proposed to solve large-sized problems. Based on artificial selection and genetic mechanics, a GA combines the concept of survival of the fittest among the solutions with structured yet randomized information exchange and offspring creation (Xu et al., 2013). GA implements the operators of selection, crossover, and mutation. These operators can generate new solutions that hopefully retain parents' good features and enhance diversity.

Unlike a classical GA, a hybrid GA with an LSA is proposed in this study. Since the GA determines the size of sub-lots completely random and these sizes are handled without bounds, finding the optimal solution may be very difficult. For this reason, an LSA has been developed to be used both for obtaining elite individuals and using as a mutation operator. The flow chart of the proposed HGA is given in Fig. 3.

In the following sub-titles, the solution representation of the proposed HGA, the creation of the initial population, and the calculation of the fitness function, the LSA is used both as the mutation operator and to obtain elite individuals. Selection, crossover, and mutation operators are explained in detail, respectively.

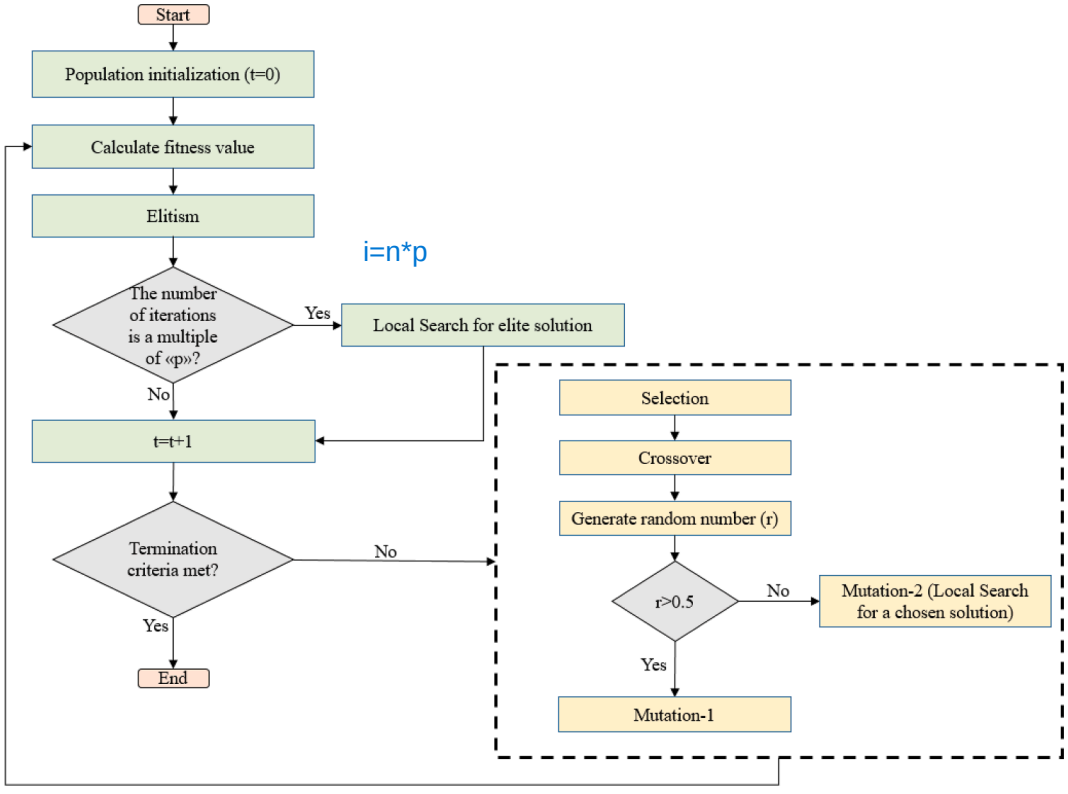

Fig. 3. Proposed HGA steps.

4.1. Solution representation

A solution representation is the chromosome structure that corresponds to a solution. It is formed by considering the constraints and characteristics of the problem. In this study, the developed solution representation includes the machines to which each job is assigned, the size of sub-lots, and sequence information on the machine to which it is assigned. It consists of two matrices. The first matrix shows the sub-lot sizes of jobs for each operation on each machine, and the second matrix shows which machine and sequence the operations are assigned to. The number of columns of the first matrix is the sum of all the operations of the jobs. The number of columns of the second matrix is determined by the sum of the number of machines to which all operations of the jobs can be processed. For the solution representation of a problem with two machines and two jobs with two operations, the first matrix is given in Fig. 4, and the second matrix is given in Fig. 5. O_{ij} is job i in operation j. O_{11} is only processed on machine 2. Other operations can be processed on machines 1 and 2. In this case, the number of columns of the second matrix is 7.

As seen from Figs. 4 and 5, in operation O_{11}, the job is not split, and it has been assigned on M2 with lot size of 100. In operation O_{22}, the job is split into two sub-lots. It has been assigned on M1 with sub-lot size of 63 and M2 with sub-lot size of 37.

4.2. Generating the initial population and calculation of the fitness function

The initial population of the HGA is generated by following the steps below so that it consists of feasible solutions.

Step 0. Population size (pb) = 200.

Step 1. $pb = pb$-1, a matrix of size ($m \times q$) is created for the first part of the chromosome. First, the number of machines to which all operations can be processed is determined. (m: number of machines, q: total number of operations of all jobs, TMS_{ij}: number of machines to which operation j of job i can be processed).

Step 2. If $TMSij = 1$, the value of g_i is assigned to the cell in the matrix corresponding to the machine to which the operation should be processed.

Step 3. If $TMSij > 1$, a random number is generated. If the generated random number is less than λ (=0.3), one of the machines to which that operation can be processed is selected. The value g_i is assigned to the cell corresponding to the selected machine in the matrix.

Step 4. A random machine is selected if the generated random number exceeds λ. The value α_{ijk} is determined randomly between s and (g_i -s) to split sub-lot size of job i in operation j into more than one machine. The corresponding cell in the matrix is assigned the value α_{ijk}; the selected machine is assigned the value α_{ijk} of job i in operation j.

Step 5. If $TMS_{ij} - 2 > 0$, μ is determined between s and (g_i-s-α_{ijk}), and the value of μ is assigned to the corresponding cell in the matrix. The α_{ijk} value is updated to $\alpha_{ijk} = \alpha_{ijk}+\mu$. This step is repeated until TMS_{ij}-2 ≤ 0.

Step 6. In the matrix, the value (g_i- α_{ijk}) is assigned to the cell where the machine is located, to which the operation can be processed last.

Step 7. *Steps 2, 3, 4, 5,* and *6* are repeated for all job in all operation.

Step 8. A ($2 \times h$) dimensional matrix is created for the second part of the chromosome. (h: the sum of the machines to which the operations of the jobs can be processed. For example, let a total of 2 jobs; the first one has two operations, and the second one has one operation. O_{11} can be processed on two machines, O_{12} on one machine, and O_{21} on two machines. In this case, $h = 5$.).

1. Matrix

	O_{11}	O_{12}	O_{21}	O_{22}
Machine 1	0	20	0	63
Machine 2	100	80	100	37

Fig. 4. The first matrix of the representation structure.

2. Matrix

	O_{21}	O_{21}	O_{11}	O_{22}	O_{12}	O_{12}	O_{22}
Job	2	2	1	2	1	1	2
Machine	2	1	2	2	1	2	1

Fig. 5. The second matrix of the representation structure.

Step 9. A list is created in which the numbers of jobs repeat until $\sum_j TMS_{ij}$ and the jobs in this list are randomly shuffled. For example, considering the example given in *Step 8*, a list (Stevenson, 1996; Stevenson, 1996; Stevenson, 1996; Garey et al., 1976; Garey et al., 1976) is created, and then the elements in the list are shuffled to obtain a new list like (Garey et al., 1976; Stevenson, 1996; Garey et al., 1976; Stevenson, 1996; Stevenson, 1996). The created list represents the first row of the second matrix.

Step 10. A list of machines to which all operations can be processed is created. For the second row of the second matrix, a machine is selected from the list of machines, starting with the first operation on the first row, and the operation is processed on the selected machine. This machine is removed from the list.

Step 11. If $pb = 0$, STOP. If $pb > 0$, go to *Step 1*.

To calculate the HGA's fitness function value, the machine information which the operation for the job is processed in the first column of the second matrix is recorded. The element in the cell corresponding to the machine to which the operation for the job in the first matrix is processed is α_{ijk}. If $\alpha_{ijk} > 0$, the completion time of the last job on the same machine and the completion time of the previous operation of the related job are considered. Whichever time is greater is called the *Previous Time*. Then it is calculated by the formula $C_{ijk} = Previous\ Time + p_{ijk}$ $\alpha_{ijk} + h_{ijk}$. This process is applied to all columns in the second matrix. The C_{max}, the fitness value, is equal to the greatest C_{ijk}.

4.3. Developed local search algorithm

Since this study considers job-splitting, it is complex to determine at which the sizes of jobs is processed on which machine. It becomes difficult to find good solutions when the sizes of jobs in the GA are determined randomly. It also failed other classical methods used in the literature, as it is the first time that size of sub-lots are handled without bounds. Therefore, an LSA is developed to find a successful solution in the large solution space. This algorithm is applied to the randomly selected solution in the mutation process and to the elite solution periodically. The steps to be followed for the developed LSA are given below:

Step 1. Selection. Select a chromosome from the current population to which the LSA will be applied and keep that as the first solution (S_1).

Step 2. Recording the value of α_{ijk}. Starting from the first operation of the first job (O_{11}) records the first α_{ijk} which the value is greater than zero.

Step 3. Splitting of jobs assigned to a single machine into machines. If $\alpha_{ijk} = g_i$and the operation of the job can be processed to more than one machine, reduce α_{ijk} by s. Select the machine with less time than the other machines to which it can be processed and change the value in the corresponding row to s (S_2). Calculate the fitness value of the obtained solution (F_{S2}). If $F_{S2} \leq F_{S1}$, update S_1. $S_1 = S_2$.

Step 4. Decreasing the size of sub-lots. If $s+\beta \leq \alpha_{ijk} \leq g_i$-$s$, reduce α_{ijk} by β (β: The amount of increase or decrease in the size of sub-lot of jobs in the algorithm). Increase the α_{ijk} by β from one of the other machines where O_{ij} is processed (S_2). Calculate the fitness value of the obtained solution (F_{S2}). If $F_{S2} \leq F_{S1}$, update S_1 and repeat this step until $F_{S2} > F_{S1}$.

Also, if $F_{S2} < F_{S1}$, define $c = 1$.

Step 5. Increasing the size of sub-lots. If $s \leq \alpha_{ijk} \leq g_i\text{-}s\text{-}\beta$, increase α_{ijk} by β. Reduce the α_{ijk} by β from one of the other machines where O_{ij} is processed (S_2). Calculate the fitness value of the obtained solution (F_{S2}). If $c = 1$ and $F_{S2} < F_{S1}$, update S_1 and repeat this step until $F_{S2} \geq F_{S1}$. If $c \neq 1$ and $F_{S2} \leq F_{S1}$, update S_1 and repeat step 5 until $F_{S2} > F_{S1}$.

Step 6. Repeat *Steps 3, 4,* and *5* for all columns in the first matrix, the operations of all jobs.

Step 7. Merging of splitting jobs. To examine whether it is more advantageous not to split a job, the split job is merged and C_{max} is calculated when it is assigned to each of the possible machines. The smallest C_{max} value is equal to F_{S2}. ($\alpha_{ijk} = g_i$) (S_2). If $F_{S2} \leq F_{S1}$, update S_1. Repeat this step for each O_{ij}.

4.4. *Selection, crossover, and mutation operators of proposed HGA*

Selection is the selection of chromosomes according to certain selection criteria and creating a new generation. Binary tournament selection and the elitism method are used as selection operators. In binary tournament selection, two chromosomes are randomly selected from the population and compared according to their fitness functions. The chromosome with the better fitness function is passed on to the next generation. Elitism makes sure that the best solution is passed down from one generation to the next.

Crossover is the process of producing two new offspring by exchanging genes between two parents. As the crossover operator, the one-point crossover method is used for the first matrix, which shows the size of sub-lots of the jobs to be split. The JBX crossover operator is used for the job row of the second matrix. The JBX steps in Li and Gao (Li and Gao, 2016) are used in this study. Fig. 6 shows the application of the JBX crossover operator.

For the second matrix's machine row, a list of machines to which all operations of jobs can be processed is created. Starting from the O_{ij} in the job row, a machine is selected sequentially from the machine list, and the operation is processed on the selected machine. This machine is removed from the list. After using the mutation operators, this process is also applied for machine row correction. This process ensures that the changes made in the job row are made without changing the operational priorities of the jobs. Fig. 7 shows how the machine row is corrected after a change in the job row.

As shown in Fig. 7, when O_{22} and O_{11} are replaced, the job row is decoded again, as it is impossible to process O_{22} before O_{21}. Thus, the column that was previously O_{22} is changed to O_{21}. However, O_{21} may not be assigned to Machine 1. For this reason, the correction process is performed on the machine row.

The mutation is the process of facilitating the search for a new solution by modifying a gene to change the direction of the search. Two different methods are used as mutation operators in this study. One of the methods is the application of the developed local search algorithm to the selected individuals to mutate. The other is to change the size of sub-lots of randomly selected jobs assigned to the machines for the first matrix of the selected individuals and randomly use the swap or inversion operators for the job row of the second matrix. Fig. 8 shows the flow chart of mutation processes.

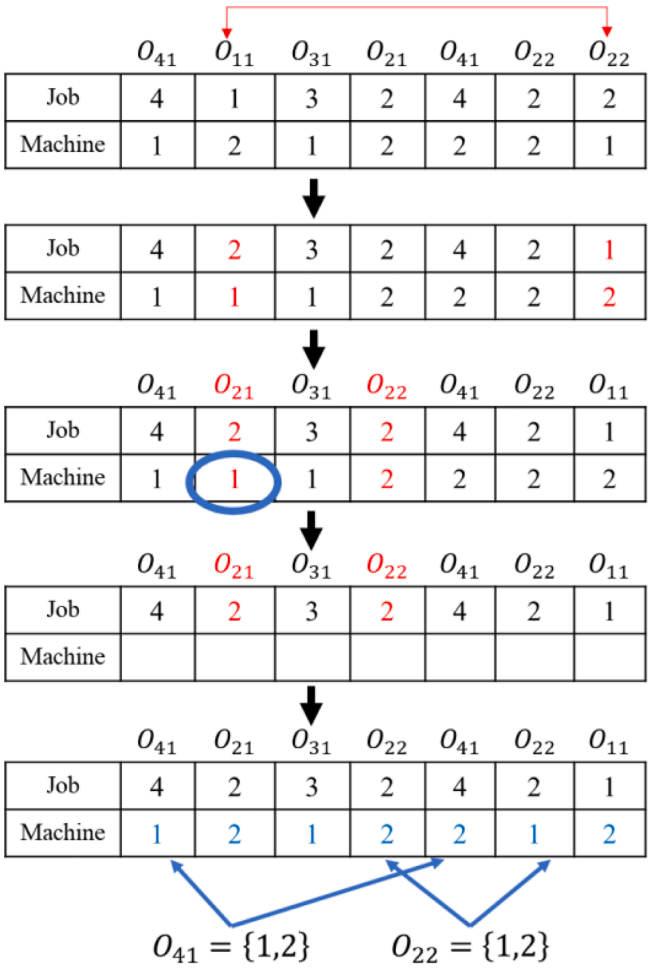

Fig. 7. Applying correction for the machine row of the second matrix.

The steps of the mutation operator are as follows:

Step 1. Generate a random number (r_1).

Step 2. If $r_1 < 0.5$, apply Mutation-2 (LSA). End the mutation process.

Step 3. If $r_1 > 0.5$, change the size of sub-lots by randomly selecting

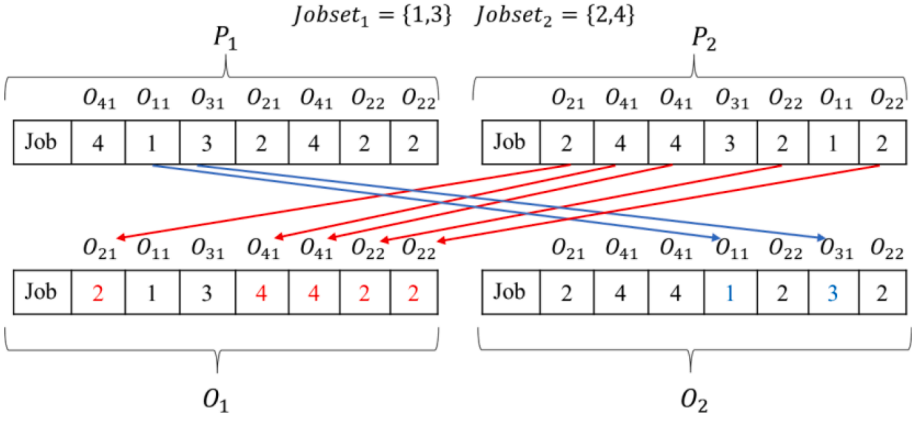

Fig. 6. JBX crossover operator.

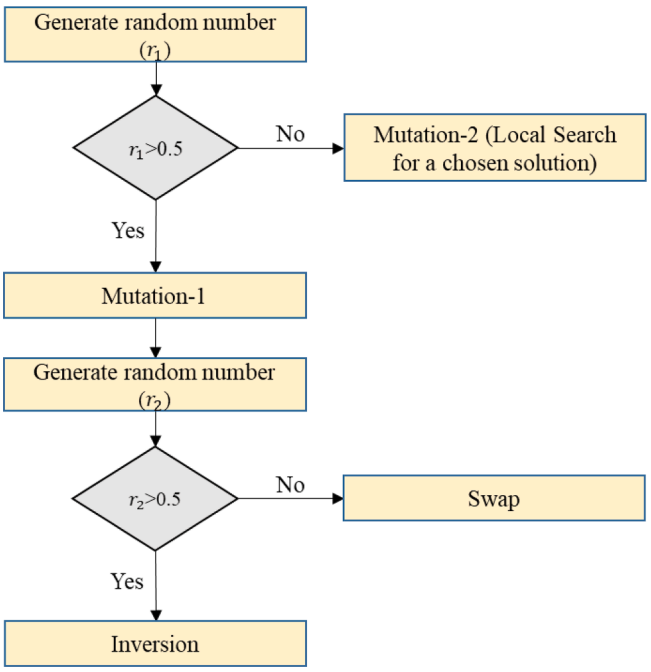

Fig. 8. The flow chart of mutation processes.

from the jobs assigned to the machines for Mutation-1, the first matrix of the selected chromosomes.

Step 4. Generate a random number (r_2) for the second matrix.

Step 5. If $r_2 > 0.5$, apply the inversion operator; otherwise, use the swap operator.

Step 6. Apply correction for the machine row of the second matrix.

5. Experimental results

This section generates different-sized test problems to test the performance of the proposed mathematical model, GA, and HGA. The CPLEX solver of GAMS 23.7 solves the proposed model to compare with the case where splitting the job is not allowed. Classical GA and HGA have been solved by Python 3.7. All tests have been performed on a desktop with a 2.40 GHz Intel Core i5 and 8 GB of RAM.

5.1. Generation of test problems

The proposed mathematical model and HGA performance are tested using randomly generated different sizes of problems. A total of 30 test problems were generated, 10 for each size. The name of the problem has two parts. The first part is the number of operations, and the second is the example number. For example, *12-1* is the first example of problems with the *12-operation*. In Table 2, the properties of test problems of three different sizes are given, where *x* represents the sample number, and its value can range from (Stevenson, 1996; Cheng et al., 2013). As seen from the table, in the problems with *12-operation*, the number of jobs is 4, the number of operations for each job is 3, the number of machines is 5, and the maximum number of machines on which the operation of a job can be processed is 2. In 24-operation problems, the number of jobs is 6, the number of operations for each job is 4, the number of machines is 8, and the maximum number of machines on which the operation of a job can be processed is 3. In 50-operation problems, the number of jobs is 10, the number of operations for each job is 5, the number of machines is 11, and the maximum number of machines on which the operation of a job can be processed is 3.

In addition, three different large-sized problems with a single sample have been generated to represent real-life problems better. Table 3 presents the properties of large problems of three different sizes. In the three problems, the number of machines is 20, and the maximum number of machines on which the operation of a job can be processed is 3. In the first problem, there are a total of 150 operations, including 50 jobs and 3 operations of each job. In the second problem, there are a total of 200 operations, including 50 jobs and 4 operations of each job. In the last problem, there are a total of 250 operations, including 50 jobs and 5 operations of each job.

The test problems are generated according to the uniform distribution in the range of $[\frac{1}{g_i}, \frac{99}{g_i}]$ for unit processing time.

(p_{ijk}) and in the range of (Stevenson, 1996; Wang et al., 2012) for setup time (h_{ijk}). The value of parameters s and g_i are 10 and 100, respectively.

5.2. The detailed solution of the problem 12-1

The solution to problem *12-1*, one of the generated test problems, is shown in detail. This problem consists of 5 machines and 12 operations, including 4 jobs and 3 operations of each job. In this problem, the maximum number of machines on which the operation of a job can be processed is 2. The unit processing time.

(p_{ijk}), setup times (h_{ijk}) and machine eligibility parameter values (u_{ijk}) of problem *12-1* are given in Table 4.

To show the contribution of job-splitting, solutions where jobs are not split, are also found by adding just a constraint given in Eq. (20) to the proposed mathematical model.

$$\sum_k \sum_l x_{ijkl} = 1 \quad \forall_{i,j;j\le o_i} \tag{20}$$

In this study, *Scenario 1 (S1)* represents the case without job-splitting, and *Scenario 2 (S2)* represents the case with job-splitting. The problem is solved first with the proposed mathematical model for *S1* and *S2* and then classical GA and HGA. Obtained solutions and CPU times are given in Table 5.

As seen in Table 5, the obtained C_{max} for *S1* is 217. On the other hand, it is 187.75 for *S2*. C_{max} improved by 13.48% by allowing jobs to be split. GA found better results than *S1* but worse than *S2*. It is remarkable that even for such a small problem, classical GA had difficulty in finding successful solutions. However, HGA found the same successful results as *S2* in a shorter time. Gantt Charts of obtained solutions are shown in Figs. 9, 10, and 11.

In Fig. 9, since the jobs are not split, the machines' idle times are longer. As seen from Fig. 10, job 3 in operations 1 and 3 and job 2 in operation 2 are split. For example, job 2 in operation 2 is split into two sub-lots with sizes 63 on machine 2 and 37 on machine 3. As shown in Fig. 11, the C_{max} value becomes worse when job 1 in operation 2 is split

Table 2
The properties of test problems of three different sizes.

Problem	*Number of operations (number of jobs X number of operations for each job)*	*Number of machines*	*Maximum number of machines to which operations of a job can be processed*
12-x	12 (4x3)	5	2
24-x	24 (6x4)	8	3
50-x	50 (10x5)	11	3

Table 3
The properties of large-sized problems of three different sizes.

Problem	*Number of operations (number of jobs X number of operations for each job)*	*Number of machines*	*Maximum number of machines to which operations of a job can be processed*
150-1	150 (50x3)	20	3
200-1	200 (50x4)	20	3
250-1	250 (50x5)	20	3

Table 4
p_{ijk}, h_{ijk}parameter values for $u_{ijk} = 1$of the problem *12-1*.

i	j	k	p_{ijk}	h_{ijk}
1	1	1	0.12	9
1	2	3	0.31	2
1	2	4	0.29	4
1	3	4	0.22	12
1	3	5	0.61	28
2	1	1	0.88	30
2	1	3	0.27	6
2	2	2	0.60	26
2	2	3	0.70	21
2	3	5	0.17	27
3	1	1	0.88	15
3	1	2	0.63	24
3	2	3	0.19	17
3	3	4	0.79	12
3	3	5	0.71	10
4	1	1	0.37	4
4	1	2	0.13	11
4	2	5	0.45	16
4	3	3	0.63	8
4	3	4	0.89	2

Table 5
Obtained solutions and CPU times.

Method	C_{max}	*Time* (sec.)
S1	217.00	87.20
S2	187.75	1616.53
GA	199.50	67.62
HGA	187.75	213.58

instead of job 3 in operation 1 and the size of sub-lots are changed. It has been noticed that there are idle times on machines 3, 4, and 5. In Fig. 12, the idle times' graph of the machines is given. Allowing jobs to be split, the machines' idle times are reduced.

5.3. Parameters of HGA

The basic parameters of the GA are as follows; population size (*pb*), crossover rate (P_c), mutation rate (P_m), number of iterations (t), and elite number of individuals (e). Population size and crossover rate were taken from the studies of Jiang and Du (Jiang and Du, 2015), and the number of iterations was taken from the study of Wang et al. (Wang et al., 2014). In this study, the number p should also be determined. This number is used to determine the iterations to which the developed LSA will be applied. When the number of iterations is a multiple of p, LSA is applied to the elite solution.

An experimental design is made to determine the parameter values (*pb*, P_c, P_m, *t*) that affect the success of the algorithms by using problems *50-7*. Factors and their levels are given in Table 6.

Analysis of variance results obtained with Minitab is given in Table 7.

In ANOVA, the terms are given the well-known p-values that display the probability of the term not being important, and a term with a p-value of 0.05 or less is taken as important (Stewardson and Whitfield, 2004). When the results in Table 7 are examined, *pb* and P_c are important according to p-values. Therefore, determining the value of *pb* and P_c is critical for this problem. The main effect plot for these two parameters is given in Fig. 13.

As can be seen from Fig. 13, *pb* and P_c should be 200 and 0.8, respectively. The parameter values used in the HGA are given in Table 8. An analysis of the convergence of problems *12-1*, *24-1,* and *50-1* is given in Fig. 14.

The value of β and λ algorithm parameters have been determined by preliminary analysis and they are 1 and 0.3, respectively.

5.4. Test results

The randomly generated test problems are solved by the proposed mathematical model for *S1* and *S2*. Moreover, all test problems are solved by classical GA and HGA. CPU times are limited to 3600 s. In addition, the number of runs of the GA and HGA for each problem is 10.

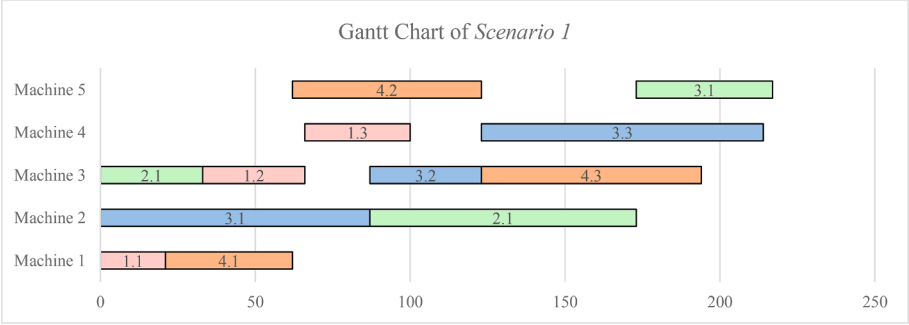

Fig. 9. Gantt Chart of *Scenario 1*.

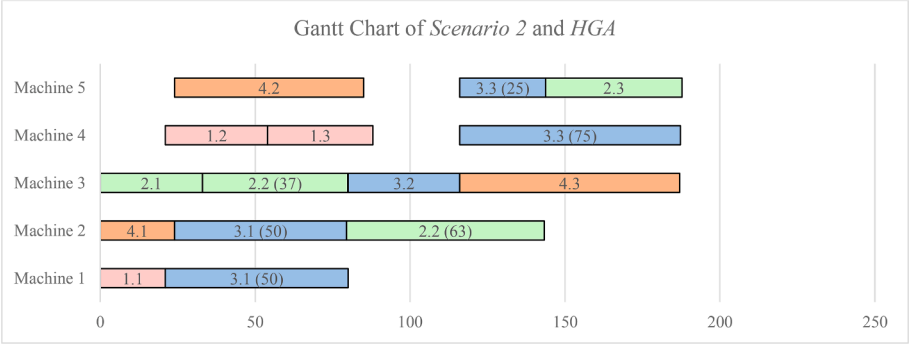

Fig. 10. Gantt Chart of *Scenario 2* and *HGA*.

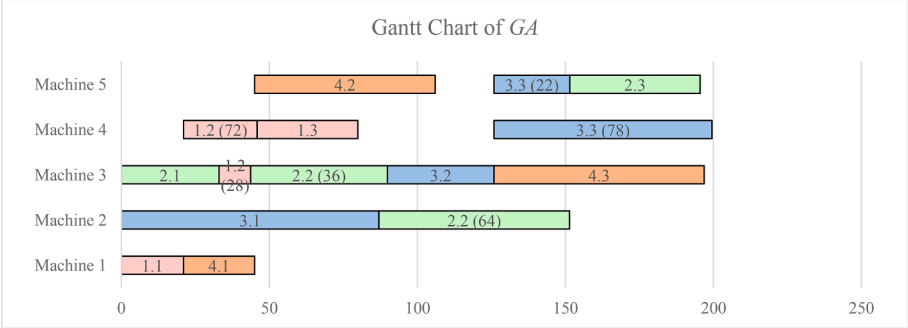

Fig. 11. Gantt Chart of *GA*.

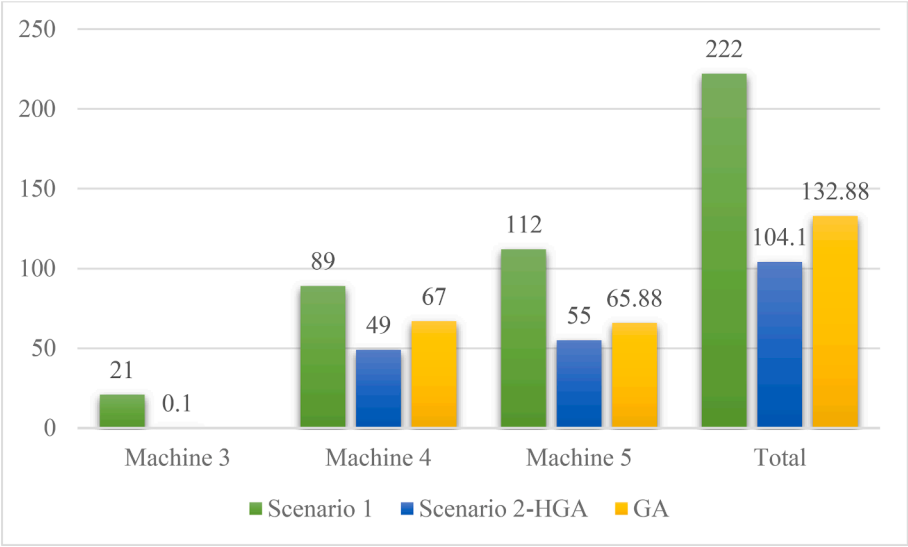

Fig. 12. The idle times' graph of machines.

Table 6
Factors and their levels.

Factors	Factor levels		
	Level 1 (−1)	*Level 2 (0)*	*Level 3 (1)*
pb	50	100	200
P_c	0.7	0.8	0.9
P_m	0.01	0.03	0.05
t	500	750	1000

Table 7
Analysis of variance results.

Analysis of variance					
Source	DF	Adj SS	Adj MS	F-Value	P-Value
pb	2	3496.8	1748.42	5.12	0.008
P_c	2	2961.5	1480.77	4.34	0.017
P_m	2	936.6	468.28	1.37	0.260
t	2	17.2	8.59	0.03	0.975
Error	72	24578.4	341.37		
Total	80	31990.5			

Elitism has been applied in both GA and HGA. In addition, when the number of iterations is a multiple of 25 ($p = 25$) in HGA, the LSA is applied. The percentage improvement value is used to compare the obtained solutions of S2 with S1, HGA with S2, and HGA with GA. Its formula is given in Eq. (21).

$$imp_{Method2-Method1} = \frac{f_{Method1} - f_{Method2}}{f_{Method1}} * 100 \quad (21)$$

The obtained C_{max} and CPU times for problems with *12-operation* are given in Table 9.

When Table 9 is examined, the splitting of jobs improves the C_{max} by 12.14% (an average of imp_{S2-S1} values). The same results have been obtained with HGA and *S2* in all problems with *12-operation*. When HGA and GA are compared, although these problems are small, HGA found 2.57% better results than GA. HGA obtained optimum solutions with 3% shorter CPU times than S2.

The obtained C_{max} and CPU times for problems with *24-operation* are given in Table 10.

When Table 10 is examined, feasible solutions have been obtained within the time limit by *S1* and *S2*. HGA and GA have also obtained feasible solutions in a shorter time. The C_{max} values of *S2* are 8.57% more successful than *S1*. The results of HGA are 6.79% more successful than *S2* and 13.53% than GA.

In problems with *50-operation*, the proposed mathematical model obtained no feasible solution for any test problem within the time limit. Therefore, no solution has been given for *S1* and *S2*. The obtained solutions and CPU times of HGA and GA in problems with *50-operation* are given in Table 11.

When Table 11 is examined, even if GA obtained solutions in a shorter time than HGA, on average, 22.53% more successful solutions have been obtained with HGA. In Table 12, the mean and coefficient of variation (CV) of 10 runs of HGA are given for each test problem. The CV

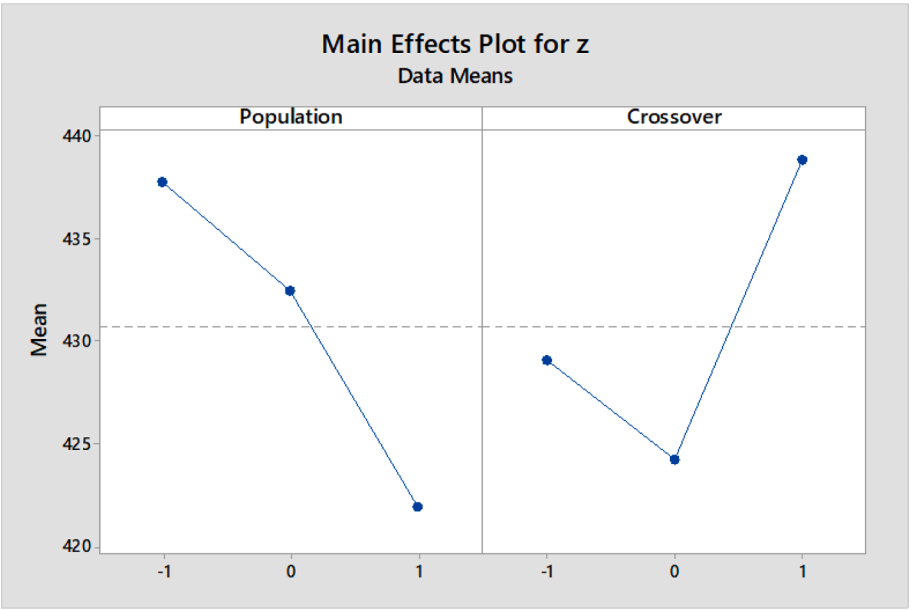

Fig. 13. The main effect plot.

Table 8
Parameter values of HGA.

Parameter	*Value*
pb	200
P_c	0.80
P_m	0.05
t	1000

is the ratio of the standard deviation to the mean, in other words, the CV of the data. The lower the CV, the lower the deviation and risk.

As can be seen from Table 12, the CV values of HGA are quite small. This means that the results do not vary and are consistent.

In summary, according to the results from all test problems, it has been seen that when the splitting of jobs is allowed, the makespan (C_{max}) is significantly reduced even if each split job requires additional setup time. No feasible solution has been found for problems with *50-operation* for *S1* and *S2*. Even if the problems are solved by GA, successful solutions cannot be obtained. The proposed HGA has shown its success by obtaining the optimal solutions for problems with *12-operation* and

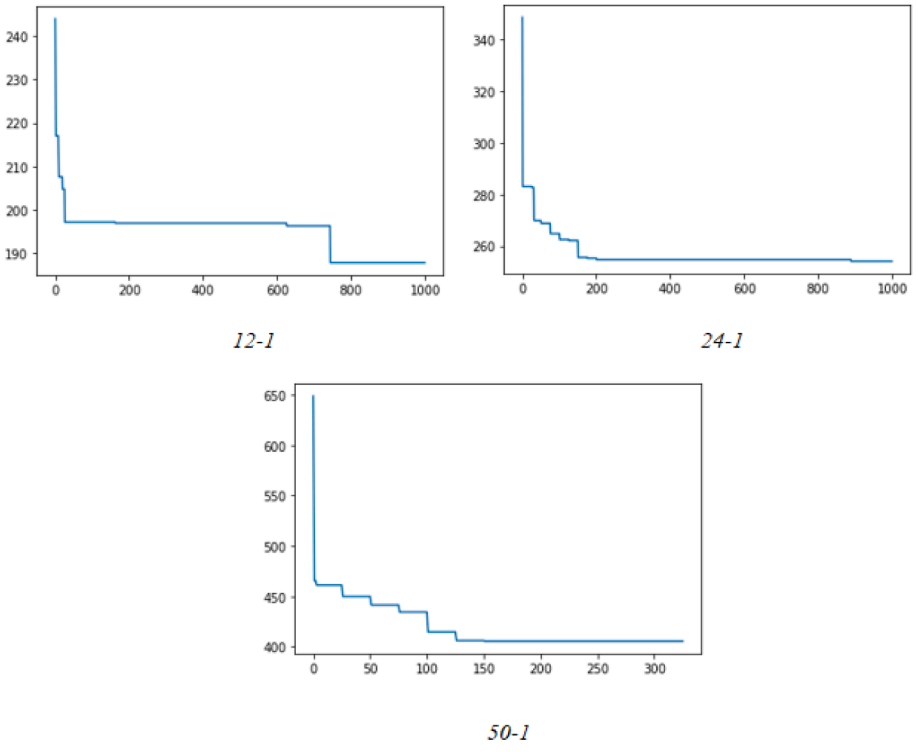

Fig. 14. An analysis of the convergence of problems *12-1*, *24-1*, and *50-1*.

Table 9
Results of 12-operation problems.

Problem	S1		S2		GA		HGA		imp_{S2-S1}	imp_{HGA-S2}	imp_{HGA-GA}
	f_{S1}	t_{S1} (sec.)	f_{S2}	t_{S2} (sec.)	f_{GA}	t_{GA} (sec.)	f_{HGA}	t_{HGA} (sec.)			
12-1	217	87.20	187.75	1616.53	199.50	67.62	187.75	213.58	13.48	0.00	5.89
12-2	194	20.94	175.59	177.28	183.18	51.96	175.59	162.64	9.49	0.00	4.14
12-3	293	96.42	257.72	697.84	257.72	3.05	257.72	20.60	12.04	0.00	0.00
12-4	239	59.78	202.42	117.53	228.78	43.67	202.42	341.38	15.31	0.00	11.52
12-5	292	6.45	224.70	42.17	224.95	18.26	224.70	15.76	23.05	0.00	0.11
12-6	237	10.84	223.00	24.73	223.00	2.26	223.00	14.13	5.91	0.00	0.00
12-7	254	7.84	224.56	36.14	228.20	32.63	224.56	84.63	11.59	0.00	1.60
12-8	172	21.61	155.90	141.42	158.05	59.25	155.90	158.78	9.36	0.00	1.36
12-9	284	6.06	255.12	7.80	255.12	7.05	255.12	8.15	10.17	0.00	0.00
12-10	271	15.13	241.25	69.03	244.00	24.41	241.25	19.37	10.98	0.00	1.13

Table 10
Results of problems with *24-operation*.

Problem	S1		S2		GA		HGA		imp_{S2-S1}	imp_{HGA-S2}	imp_{HGA-GA}
	f_{S1}	t_{S1} (sec.)	f_{S2}	t_{S2} (sec.)	f_{GA}	t_{GA} (sec.)	f_{HGA}	t_{HGA} (sec.)			
24-1	285	3600	250.54	3600	270.96	132.45	241.18	1987.01	12.09	3.74	10.99
24-2	286	3600	251.58	3600	286.00	138.51	234.56	2628.59	12.03	6.77	17.99
24-3	368	3600	329.12	3600	339.07	148.41	328.84	2344.02	10.57	0.09	3.02
24-4	347	3600	303.50	3600	324.88	123.09	286.20	1311.79	12.54	5.70	11.91
24-5	252	3600	235.83	3600	285.84	183.69	221.40	2406.82	6.42	6.12	22.54
24-6	302	3600	266.92	3600	304.60	175.71	262.00	2418.35	11.62	1.84	13.99
24-7	370	3600	349.86	3600	350.00	97.58	321.86	3284.66	5.44	8.00	8.04
24-8	296	3600	273.40	3600	314.22	174.30	242.23	1384.01	7.64	11.40	22.91
24-9	332	3600	321.09	3600	323.00	68.88	284.84	2190.64	3.29	11.29	11.81
24-10	379	3600	363.62	3600	360.03	82.03	316.50	1078.36	4.06	12.96	12.09

Table 11
Results of problems with *50-operation*.

Problem	GA		HGA		imp_{HGA-GA}
	f_{GA}	t_{GA} (sec.)	f_{HGA}	t_{HGA} (sec.)	
50-1	525.52	312.81	403.51	2512.68	23.22
50-2	546.05	155.94	414.30	2624.18	24.13
50-3	505.45	466.02	382.44	2390.04	24.34
50-4	597.65	208.21	477.00	2984.48	20.19
50-5	568.58	212.68	416.20	2805.16	26.80
50-6	601.90	230.95	473.00	2716.66	21.42
50-7	524.81	269.51	447.00	1300.77	14.83
50-8	515.07	416.55	392.00	3400.45	23.89
50-9	548.10	488.36	417.80	2580.11	23.77
50-10	544.95	164.15	421.00	2779.66	22.75

successful feasible solutions for problems with *24-operation* and *50-operation*.

To show that the proposed HGA and GA can solve real-life problems, large-sized problems with *150, 200,* and *250-operations* have been solved, and their results are given in Table 13.

When Table 13 is examined, in large-sized problems with *150, 200,* and *250 operations*, feasible solutions are obtained both with HGA and GA. However, HGA finds >17.62 % more successful results than GA.

6. Conclusion

This study addresses a flexible job-shop scheduling problem with job splitting. An integrated model is proposed for considered problem. Unlike in the literature, number and size of sub-lots are not known beforehand. In the proposed model to examine the effect of job splitting on makespan, Scenario 1, which represents the case without job-splitting, and Scenario 2, which represents the case with job-splitting, are compared. When splitting of jobs is allowed, the makespan is significantly reduced even if each sub-lot requires a separate setup time. Considering the results in which the optimal solution is obtained, job-

Table 13
Results of large-sized problems.

Problem	GA		HGA		imp_{HGA-GA}
	f_{GA}	t_{GA} (sec.)	f_{HGA}	t_{HGA} (sec.)	
150-1	984.14	2617.42	848.00	2306.17	13.83
200-1	1284.30	2541.49	1198.80	3547.90	6.66
250-1	1659.30	750.47	1122.00	1258.06	32.38

Table 12
The mean and standard deviation of 10 runs of HGA for each test problem.

Problem	f_{HGA}^{min}	f_{HGA}^{mean}	CV(%)	Problem	f_{HGA}^{min}	f_{HGA}^{mean}	CV(%)	Problem	f_{HGA}^{min}	f_{HGA}^{mean}	CV(%)
12-1	187.75	189.42	0.0061	24-1	241.18	243.14	0.0130	50-1	403.51	419.32	0.0141
12-2	175.59	175.67	0.0015	24-2	234.56	242.79	0.0124	50-2	414.30	438.04	0.0190
12-3	257.72	257.72	0.0000	24-3	328.84	332.41	0.0064	50-3	382.44	387.90	0.0129
12-4	202.42	203.22	0.0014	24-4	286.20	288.20	0.0106	50-4	477.00	486.57	0.0169
12-5	224.70	224.70	0.0000	24-5	221.40	227.90	0.0142	50-5	416.20	428.35	0.9966
12-6	223.00	223.00	0.0000	24-6	262.00	268.10	0.0131	50-6	473.00	495.21	1.9499
12-7	224.56	225.92	0.0043	24-7	321.86	323.98	0.0138	50-7	447.00	447.54	0.1161
12-8	155.90	155.90	0.0000	24-8	242.23	242.59	0.0019	50-8	392.00	393.78	0.2377
12-9	255.12	255.12	0.0000	24-9	284.84	291.65	0.0090	50-9	417.80	422.34	0.7610
12-10	241.25	241.27	0.0002	24-10	316.50	317.52	0.0074	50-10	421.00	437.23	1.9566

splitting reduces the makespan by 12.14%. In addition, the idle times of the machines are also reduced.

Feasible solutions could not be found with the proposed mathematical model for large-sized problems. For this reason, a hybrid GA, including a local search for FJSP with job-splitting, has been proposed for the first time. The success of the proposed HGA is shown by solving the randomly generated test problems with both classical GA and HGA. In classical GA, successful sizes of sub-lots have not been found because searching for them in a very large space is difficult. However, HGA has obtained more successful results by searching for space intelligently. HGA shows its success by both finding the optimal solution to small problems and finding successful solutions to large-sized problems in a reasonable time.

As a future study, different heuristics/metaheuristics can be developed to solve the considered problem and compared with the HGA proposed in this study.

CRediT authorship contribution statement

Busra Tutumlu: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Tugba Saraç:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Abderrabi, F., Godichaud, M., Yalaoui, A., Yalaoui, F., Amodeo, L., Qerimi, A., Thivet, E., 2021. Flexible job shop scheduling problem with sequence dependent setup time and job splitting: hospital catering case study. Appl. Sci.-Basel 11 (4). https://doi.org/10.3390/app11041504.

Amjad, M.K., Butt, S.I., Kousar, R., Ahmad, R., Agha, M.H., Faping, Z., Anjum, N., Asgher, U., 2018. Recent research trends in genetic algorithm based flexible job shop scheduling problems. Math. Probl. Eng. 2018, 1–32.

Bozek, A., Werner, F., 2017. Flexible job shop scheduling with lot streaming and sublot size optimisation. Int. J. Prod. Res. 56 (19), 6391–6411. https://doi.org/10.1080/00207543.2017.1346322.

Brucker, P., Schlie, R., 1990. Job-shop scheduling with multi-purpose machines. Computing 45 (4), 369–375.

Chaudhry, I.A., Khan, A.A., 2016. A research survey: review of flexible job shop scheduling techniques. Int. Trans. Oper. Res. 23, 551–591. https://doi.org/10.1111/itor.12199.

Chen, T.L., Cheng, C.Y., Chou, Y.H., 2020. Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. Ann. Oper. Res. 290 (1–2), 813–836. https://doi.org/10.1007/s10479-018-2969-x.

Cheng, M., Mukherjee, N.J., Sarin, S.C., 2013. A review of lot streaming. Int. J. Prod. Res. 51, 7023–7046. https://doi.org/10.1080/00207543.2013.774506.

Daneshamooz, F., Fattahi, P., Hosseini, S.M.H., 2022. Scheduling in a flexible job shop followed by some parallel assembly stations considering lot streaming. Eng. Optim. 54 (4), 614–633. https://doi.org/10.1080/0305215X.2021.1887168.

Defersha, F., Chen, M., 2012. Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time. Int. J. Prod. Res. 50 (8), 2331–2352. https://doi.org/10.1080/00207543.2011.574952.

Fan, J., Zhang, C., Shen, W., Gao, L., 2022. A matheuristic for flexible job shop scheduling problem with lot-streaming and machine reconfigurations. Int. J. Prod. Res. https://doi.org/10.1080/00207543.2022.2135629.

Fattahi, P., Mehrabad, M.S., Jolai, F., 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J. Intell. Manuf. 18 (3), 331–342. https://doi.org/10.1007/s10845-007-0026-8.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Math. Oper. Res. 1 (2), 117–129.

Jiang, L.X., Du, Z.J., 2015. An improved genetic algorithm for flexible job shop scheduling problem. 2nd International Conference on Information Science and Control Engineering, 127-131. https://doi.org/10.1109/ICISCE.2015.36.

Kim, H.-J., Lee, J.-H., 2021. Scheduling uniform parallel dedicated machines with job splitting, sequence-dependent setup times, and multiple servers. Comput. Oper. Res. 126, 105115. https://doi.org/10.1016/j.cor.2020.105115.

Lee, J.H., Jang, H., Kim, H.J., 2021. Iterative job splitting algorithms for parallel machine scheduling with job splitting and setup resource constraints. J. Oper. Res. Soc. 72 (4), 780–799. https://doi.org/10.1080/01605682.2019.1700191.

Lei, D.M., Guo, X.P., 2013. Scheduling job shop with lot streaming and transportation through a modified artificial bee colony. Int. J. Prod. Res. 51 (16), 4930–4941. https://doi.org/10.1080/00207543.2013.784404.

Li, L., 2022. Research on discrete intelligent workshop lot-streaming scheduling with variable sublots under engineer to order. Comput. Ind. Eng. 165, 107928. https://doi.org/10.1016/j.cie.2021.107928.

Li, X.Y., Gao, L., 2016. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int. J. Prod. Econ. 174, 93–110. https://doi.org/10.1016/j.ijpe.2016.01.016.

Li, Y., Yang, Z., Wang, L., Tang, H., Sun, L., Guo, S., 2022. A hybrid imperialist competitive algorithm for energy-efficient flexible job shop scheduling problem with variable-size sublots. Comput. Ind. Eng. 172, 108641. https://doi.org/10.1016/j.cie.2022.108641.

Liang, J., Wang, Q., Xu, W., Gao, Z., Yan, Z., Yu, F., 2019. Improved Niche GA for FJSP. 2019 IEEE 6th International Conference on Cloud Computing and Intelligence Systems (CCIS), 190-194. https://doi.org/10.1109/CCIS48116.2019.9073748.

Liu, C.H., Chen, L.S., Lin, P.S., 2013. Lot streaming multiple jobs with values exponentially deteriorating over time in a jobshop environment. Int. J. Prod. Res. 51 (1), 202–214. https://doi.org/10.1080/00207543.2012.657255.

Liu, C.C., Wang, C.J., Zhang, Z.H., Zheng, L., 2018. Scheduling with job-splitting considering learning and the vital-few law. Comput. Oper. Res. 90, 264–274. https://doi.org/10.1016/j.cor.2017.02.011.

Liu, Z., Wang, J., Zhang, C., Chu, H., Ding, G., Zhang, L., 2021. A hybrid genetic-particle swarm algorithm based on multilevel neighbourhood structure for flexible job shop scheduling problem. Comput. Oper. Res. 135, 105431. https://doi.org/10.1016/j.cor.2021.105431.

Meng, T., Pan, Q.K., Li, J.Q., Sang, H.Y., 2018. An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem. Swarm Evol. Comput. 38, 64–78. https://doi.org/10.1016/j.swevo.2017.06.003.

Novas, J.M., 2019. Production scheduling and lot streaming at flexible job-shops environments using constraint programming. Comput. Ind. Eng. 136, 252–264. https://doi.org/10.1016/j.cie.2019.07.011.

Pan, Y., Gao, K., Li, Z., Wu, N., 2023. Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems. IEEE Trans. Autom. Sci. Eng. 20 (1), 361–371.

Saraç, T., Tutumlu, B., 2022. A bi-objective mathematical model for an unrelated parallel machine scheduling problem with job-splitting. J. Faculty Eng. Archit. Gazi Univ. 37 (4), 2293–2307. https://doi.org/10.17341/gazimmfd.967343.

Sarin, S.C., Jaiprakash, P., 2006. In: Flow Shop Lot Streaming. Springer, New York. https://doi.org/10.1007/978-0-387-47688-9.

Stevenson, J.W. 1996. Production/Operations Management (5th Ed.). Irwin.

Stewardson, D.J., Whitfield, R.I., 2004. A demonstration of the utility of fractional experimental design for finding optimal genetic algorithm parameter settings. J. Oper. Res. Soc. 55 (2), 132–138. https://doi.org/10.1057/palgrave.jors.2601703.

Wang, L., Wang, S.Y., Xu, Y., Zhou, G., Liu, M., 2012. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Comput. Ind. Eng. 62 (4), 917–926. https://doi.org/10.1016/j.cie.2011.12.014.

Wang, S.X., Zhang, C.Y., Jin, L.L., 2014. A hybrid genetic algorithm for flexible job-shop scheduling problem. Adv. Mat. Res. 889-890, 1179–1184. https://doi.org/10.4028/www.scientific.net/AMR.889-890.1179.

Wang, Y.L., Zhu, Q.W., 2021. A hybrid genetic algorithm for flexible job shop scheduling problem with sequence-dependent setup times and job lag times. IEEE Access 9, 104864–104873. https://doi.org/10.1109/ACCESS.2021.3096007.

Xu, X., Li, L., Fan, L., Zhang, J., Yang, X., Wang, W., 2013. Hybrid discrete differential evolution algorithm for lot splitting with capacity constraints in flexible job scheduling. Math. Probl. Eng. 2013, 1–10. https://doi.org/10.1155/2013/986218.

Yan, H., Du, X.M., Xu, L., Xu, S.C., Zhang, Y.F., Gong, P., 2022. Toward intelligent clothes manufacturing: a systematic method for static and dynamic task allocation by genetic optimization. Neural Comput. Appl. 34 (10), 7881–7897. https://doi.org/10.1007/s00521-022-06890-6.

Zhang, H.P., Ye, J.H., Yang, X.P., Muruve, N.W., Wang, J.T., 2018. Modified binary particle swarm optimization algorithm in lot-splitting scheduling involving multiple techniques. Int. J. Simulat. Model. 17 (3), 534–542. https://doi.org/10.2507/IJSIMM17(3)CO13.

Zheng, F.F., Jin, K.Y., Xu, Y.F., Liu, M., 2022. Unrelated parallel machine scheduling with job splitting, setup time, learning effect, processing cost and machine eligibility. Asia-Pacific J. Oper. Res. https://doi.org/10.1142/S0217595922500233.