



# A hybrid genetic local and global search algorithm for solving no-wait flow shop problem with bi criteria

Kenan Keskin<sup>1</sup> · Orhan Engin<sup>2</sup>

Received: 26 November 2020 / Accepted: 28 April 2021

Published online: 13 May 2021

© The Author(s) 2021

## Abstract

This paper addresses the  $m$ -machine no-wait Flow Shop Scheduling with Setup Times (NW-FSSWST). Two performance measures: total flow time and makespan are considered. The objective is to find a sequence that minimizing total flow time ( $\sum C_j$ ) and makespan ( $C_{\max}$ ) simultaneously. A Hybrid Genetic Local and Global Search Algorithm (HGLGSA) is proposed to solve the NW-FSSWST for two performance criteria. The hybrid genetic algorithm is constructed by insert-search and self-repair algorithm with self-repair function. The proposed HGLGSA is tested on 192 benchmark problems of NW-FSSWST in the literature. A full factorial experimental design is made for determined the best parameter sets that improve the performance of the proposed algorithm. The computational results are compared with the benchmark solutions from the literature. The experimental results demonstrate the effectiveness and efficiency of the proposed HGLGSA for solving NW-FSSWST.

**Keywords** Hybrid genetic algorithm · No-wait flow shop scheduling · Total flow time · Makespan · Local search · Global search · Self-repair · Insert-search

## 1 Introduction

Scheduling is a highly effective method for modeling industrial production processes [1]. No-wait flow shop problems are specific forms of flow shop problems. In flow shop problems, parts flow through serial machines. While some parts are processed in a certain machines, others might be skipped those machines. In a no-wait flow shop scheduling problem, a certain process of a job needs to be conducted uninterruptedly from the beginning of the process to the end of the process [2]. No-wait scheduling has many applications in real-world such as flight, train, and surgery problems.

Minimizing total flow time in a no-wait flow shop environment attracts the attention of many researchers. To minimize the work-in-process inventory by using the

total flow time as a criterion in no-wait scheduling problems is meaningful for today's production environment due to stabilizing usage of resources [3]. Besides, the makespan criterion is directly related to resource utilization. The makespan performance is the most commonly used as a criterion for the no-wait scheduling problems. In this paper, two performance measures: total flow time and makespan are considered. The makespan is defined as the completion time of the last job on the last machine. The total flow time is defined as a sum of the completion times of all jobs.

The no-wait flow shop scheduling is an NP-hard problem with a single objective if the number of machines is more than two. In this research, total flow time and makespan performance measures are considered for NW-FSSWST problems. For solving the NW-FSSWST problems, the

✉ Orhan Engin, orhanengin@yahoo.com; Kenan Keskin, kenankeskin@gmail.com | <sup>1</sup>New York, USA · <sup>2</sup>Industrial Engineering Department, Konya Technical University, Konya, Turkey.



genetic algorithm is hybridized. A genetic algorithm (GA) is an effective population-based meta-heuristic method for solving combinatorial optimization problems. But, GA has got some disadvantages steps during the run time. For example, GA can sticks into local optima, after a few solution steps. To eliminate this disadvantage of GA, improve its performance, and reducing search time a hybrid method is proposed. The GA is hybridized by the insert-search and self-repair method. These hybridize methods are refining the search space and performing local exploration. Insertion-search is used as a local search procedure. It searches a small neighborhood. The self-repair is used as a global search method. Also, it searches a large neighborhood in a search space. The aims of the insert-search and self-repair method are to find out better fitness value by inserting a chosen gene to other possible locations on the chromosome and repairing the chromosome itself.

It can be seen in the literature review section, there is no published paper about the no-wait flow shop scheduling problem with setup times to minimizing total flow time and makespan simultaneously. The contributions of this paper are summarized as follows.

- This is the first study to use the total flow time and makespan performance criteria simultaneously for no-wait flow shop scheduling problems with setup times.
- The proposed hybrid genetic local and global search algorithm is first used to solve the no-wait flow shop scheduling with setup times problems for two criteria.
- A full factorial experimental design is done for determining the best parameter sets of the proposed hybrid genetic algorithm for solving the no-wait flow shop scheduling problems with setup times.
- The proposed HGLGSA is calibrated for NW-FSSWST by some benchmark problems in the literature.
- The NW-FSSWST benchmark problems in the literature are solved by the proposed HGLGSA with two criteria.

The experimental results showed that the proposed HGLGSA provides better makespan and total flowtime values for the NW-FSSWST problems.

The remaining contents of the paper are organized as follows. The literature review related to no-wait scheduling problems is given in Sect. 2. Section 3 defines the NW-FSSWST formulation. Section 4 provides the hybrid genetic local and global search algorithm. Section 5 gives the computational results. Section 6 gives the conclusion and future researches.

## 2 Literature review

This section is devoted to the literature review related to the no-wait flow shop scheduling problems. The no-wait flow shop scheduling (NW-FSS) problem has been addressed widely in the literature. Some of them are given as follows. Hall and Sriskandarajah [4] provided literature reviews with no-wait in process from 1970 to 1993. Allahverdi [5] provided second literature reviews with no-wait in process from 1993 to 2016. Grabowski and Pempera [6] suggested and compared different local search algorithms like descending search algorithms with multi-moves and tabu search algorithms for minimizing  $C_{\max}$ . They also proposed dynamic tabu lists to avoid trapped at a local optimum. Wang et al. [7] compared simulating annealing and genetic algorithm in an environment of 10–300 jobs on 2–3 machines on account of scheduling efficiency for no-wait scheduling problems. Liu et al. [8] suggested a hybrid particle swarm optimization algorithm for the no-wait scheduling problem. In their study, a meta-lamarckian simulating annealing algorithm was used as a local search algorithm for hybridizing their particle-swarm optimization algorithm. Laha and Chakraborty [9] suggested a heuristic for minimizing  $C_{\max}$  in case of job insertion. They also compared their algorithm with well-known methods in the literature. Pan et al. [10] proposed a hybrid discrete particle swarm optimization to solve NW-FSS problems with the criterion of  $C_{\max}$ . Tseng and Lin [11] improved a hybrid genetic algorithm to minimize total completion time. They hybridized the genetic algorithm and a novel local search schema. An orthogonal-array-based crossover operator was designed to enhance the capability of intensification in the genetic algorithm. They proposed a local search schema that combines two local search methods: the insertion search and a novel local search method for makespan performance criterion. Engin and Gunaydin [12] studied an adaptive learning approach to obtain total completion time in NW-FSS problems with setup times. Chaudhry and Khan [13] presented a general GA methodology for no-wait flow shop scheduling and compared it with other well-known meta-heuristics. Allahverdi and Aydilek [14] tried to minimize total completion time in NW-FSS with constrained  $C_{\max}$  value which must be less than a certain time. Panwalkar and Koulamas [15] considered two-machine no-wait open shop scheduling with minimum makespan objective. An optimal algorithm was proposed to solve two-machine no-wait open shop scheduling problems. Lin and Ying [16] designed a no-wait flow shop as an asymmetric traveling salesman problem and suggested two metaheuristics algorithms to solve it for minimizing the makespan criterion. Ye et al. [17] proposed a heuristic that was named “average departure

time" to minimize makespan. Ye et al. [18] introduced an average idle time heuristic to minimize makespan in no-wait flow shop production. An initial sequence algorithm was proposed in which current and future idle times were taken into consideration. They used insertion and neighborhood exchanging methods to improve solution quality. Shao et al. [19] suggested an iterated greedy algorithm for distributed no-wait flow shop scheduling problems with the makespan criterion. Engin and Güçlü [20] developed a Hybrid Ant Colony Optimization (HACO) algorithm for no-wait flow shop scheduling problems with setup times. Zhao et al. [21] proposed a discrete water wave optimization algorithm to solve no-wait flow shop scheduling problems.

No-wait flow shop scheduling with the total flow time performance criteria has been addressed in the literature by a limited number of researchers. Some of these studies are given as follows: Filho et al. [22] improved an evolutionary algorithm for minimizing total flow time in the NW-FSS problem. Also, they compared their algorithm with Olivera and Lorana's evolutionary clustering search algorithm. Akhshabi et al. [23] proposed a hybrid particle swarm optimization algorithm for the NW-FSS problem to minimize the total flow time. Laha et al. [24] proposed a penalty-based construction algorithm for the NW-FSS problem with the minimum makespan and total flow time objectives. Originally, it was derived from Hungarian penalty method to generate an initial schedule of jobs. Koulamas and Panwalkar [25] offered an index priority rule for NW-FSS with makespan criterion for specially-structured job processing times. Moreover, they derived an index priority rule for two machine NW-FSS with the minimum weighted total completion time criterion. Zhu et al. [26] presented a novel quantum-inspired cuckoo co-search algorithm for solving the NW-FSS problem with the makespan criterion. Allahverdi and Allahverdi [27] developed a two-machine NW-FSS problem with the total completion time performance criteria. They developed eight algorithms for the processing and setup times on both machines, convert the two machine problem into a single machine one.

In the no-wait literature, multi-criteria scheduling problems have been addressed by using different criteria. There are a few studies in the literature. Some of these studies are given as follows: Allahverdi and Aldowaisan [28] suggested a hybrid genetic algorithm and hybrid simulated annealing algorithm to minimize just  $T_{\max}$  and both of  $T_{\max}$  and  $C_{\max}$  in no-wait flow shop scheduling. Tavakkoli-Moghaddam et al. [29] suggested an artificial immune system algorithm to minimize means of  $T_{\max}$  and  $C_{\max}$ . Rahimi-Vahed et al. [30] presented a search algorithm for minimizing weighted completion time and weighted mean tardiness time simultaneously. Pan et al. [31] improved a multi-objective particle swarm optimization algorithm for solving no-wait

flow shop scheduling problems with  $C_{\max}$  and  $T_{\max}$  criteria. Aydilek and Allahverdi [32] proposed several algorithms, including a simulated annealing algorithm for solving no-wait flow shop scheduling problems to minimize  $C_{\max}$  and Total Completion Time (TCT) performance criteria. Aydilek and Allahverdi [33] developed two improved simulated annealing algorithms for solving the no-wait flow shop scheduling problem to minimize  $C_{\max}$  and TCT criteria. Allahverdi and Aydilek [34] generated different versions of insertion in genetic algorithm in the no-wait flow shop to minimize TCT and  $C_{\max}$ . Allahverdi et al. [2] proposed an algorithm that was a combination of simulated annealing and insertion algorithm for solving the no-wait flow shop scheduling problem with two criteria: total tardiness and  $C_{\max}$ .

### 3 No-wait flow shop scheduling with setup times

This paper considers  $m$ -machine,  $n$ -job no-wait flow shop scheduling with setup times for total flow time and makespan performance criteria. The  $m$ -machine no-wait flow shop with setup times for makespan performance measure, denoted by  $Fm/\text{no-wait}, sj, i/C_{\max}$ , where  $Fm$  denotes the flow shop with  $m$  machines; no-wait indicates the no-wait processing characteristic,  $sj$  denotes the setup time of job  $j$  on machine  $i$ , and  $C_{\max}$  (makespan) indicates the performance criterion [20].

Some preconditions in the NW-FSSWST are as follow [20]:

- All jobs and machines are available,
- All jobs have a setup and processing times,
- The setup and processing times of each job on each machine are given before,
- The setup time is dependent on both job and machine,
- A job can be only processed on a single machine at a moment,
- Each machine can process only one job at a time,
- Before processing a job on a machine, its setup must be completed
- Jobs are independent,
- Jobs must be processed without waiting time,
- The target of the scheduling is to find out minimum total flow time and makespan values simultaneously.

The  $m$ -machine no-wait flow shop with setup times for total flow time performance measure, denoted by  $Fm/\text{no-wait}, sj, i/\sum C_j$ , where  $\sum C_j$  indicates the performance criterion, total flow time of job  $j$ .

In this study, two performance criteria: total flow time and makespan are considered. This problem is denoted by

$F_m$ /no-wait,  $s_{ji}/C_{max} \sum C_j$ . Several approaches have been developed for bi-criteria and multi-criteria scheduling problems in the literature. In this type of problem, some conflicting criteria have to be optimized simultaneously. The existing approaches for solving bi-criteria or multi-criteria no-wait scheduling problems do not integrate explicitly a manager's preferences [35, 36]. Thus, we proposed an aggregation procedure for two performance criteria. Total flow time and total completion time combined as a satisfaction function in the proposed procedure to solve no-wait flow shop scheduling problems. Also, this procedure was used by Allouche et al. [35] for multi-criteria flow shop scheduling problems in the literature. To the best of our knowledge, this is the first study that combined total flow time and total completion time performance criteria combined as the satisfaction function for the no-wait scheduling problem.

The NW-FSSWST consists of a set of  $n$  jobs ( $j=1,2,\dots,n$ ) to be processed on a set of  $m$  machines ( $i=1,2,\dots,m$ ) with an  $s_{ji}$  setup time of job  $j$  on machine  $i$ .

The NW-FSSWST is formulated as follow;

$n$	Job number
$m$	Machine number
$\pi$	Permutation of job $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$
$ST(\pi_j, i)$	Setup time of the job $j$ on machine $i$ .
$PT(\pi_j, i)$	Processing time of the job $j$ on machine $i$ .
$MinD(\pi_{j-1}, \pi_j)$	Minimum delay on the first machine between starting time of job $\pi_j$ and $\pi_{j-1}$
$MCT$	Minimum completion time.
$\sum C_j$	Total flow time (TFT) of job $j$ .

The  $MinD$  calculated from the following Eqs. 1, 2, 3 [6, 10, 20].

$$MinD(\pi_{j-1}, \pi_j) = ST(\pi_{j-1}, 1) + PT(\pi_{j-1}, 1) + \max \left[ 0, \max_{2 \leq i \leq m} \left\{ \sum_{i=2}^m (ST(\pi_{j-1}, i) + PT(\pi_{j-1}, i)) - \sum_{i=1}^{m-1} (ST(\pi_j, i) + PT(\pi_j, i)) \right\} \right] \tag{1}$$

$$Makespan = C_{max}(\pi) = MCT(\pi) = \sum_{j=2}^n MinD(\pi_{j-1}, \pi_j) + \sum_{i=1}^m (ST(\pi_n, i) + PT(\pi_n, i)) \tag{2}$$

$$MCT(\pi^*) \leq M(\pi_n, m) \forall \pi \in \pi \tag{3}$$

$$\sum C_j = \sum_{j=2}^{n-1} (n-j)MinD(\pi_{j-1}, \pi_j) + \sum_{i=1}^m (ST(\pi_n, i) + PT(\pi_n, i)) \tag{4}$$

$$TFT(\pi^*) \leq TFT(\pi_n, m) \forall \pi \in \pi \tag{5}$$

## 4 Hybrid genetic local and global search algorithm

### 4.1 Genetic algorithm

The genetic Algorithm which was proposed by Holland in 1975 firstly, is a method commonly used in the solution of combinatorial optimization problems [37]. A genetic algorithm is a population-based meta-heuristic method. It tries to find an optimum or closer solution for NP-hard problems [38, 39]. Fundamental parts of genetic algorithms are chromosome, population, and gen which is the smallest meaningful element of a chromosome. Every gen represents a job in a scheduling problem. Thus, multiple gens come together to generate a chromosome. Similarly, a population consists of a set of chromosomes [40]. The basic steps of a genetic algorithm are presented below [37, 41].

*Step 1* All possible solutions must be presented as an array of gens in the solution space. So, all solutions are coded as a set of arrays in this study.

*Step 2* An initial population is created by random generally.

*Step 3* A fitness value is calculated for every array.

*Step 4* A certain number of arrays are selected for the process of breeding randomly.

*Step 5* Fitness values of every individual in the new generation are calculated. Then, they are crossed over and mutated.

*Step 6* Step 3 to Step 5 is repeated up to reach a pre-determined generation number.

*Step 7* The array which has the best fitness value is selected.

Operators of genetic algorithm are explained in below;

#### 4.1.1 Breeding operator

The intention of breeding operators in genetic algorithms is that the individuals that have better fitness values are more likely to be in the next generation. Hence, the fitness value of those individuals is normalized by the mean of the population.

#### 4.1.2 Crossover operator

Crossover operator exchanges information between arrays and produces better individuals. Randomly (generally) selected at least two gens from two arrays are interchanged. In other words, it generates two possible solutions. Although, there are many types of crossover in the literature, single-point crossover, and multi-point crossover are used widely [42]. Position-based, order-based, partial-mapped crossover, linear order-based crossover are well-known multi-point crossovers.

#### 4.1.3 Mutation operator

The mutation is a sudden change in a parent's genotype which is caused by an alteration in a gene or a chromosome. The number of genes is constant in a process of mutation. Mutation occurs just on one array (chromosome). Reverse mutation, exchanging randomly chosen two genes, exchanging randomly chosen three genes, and gene insertion are widely used mutation operators in the literature.

#### 4.1.4 Elitism strategy

After breeding, crossover, and mutation operators, the individuals which have the best fitness value in their population might not be transferred to the next generation. To avoid this undesirable condition, the individuals which have better fitness value are exchanged with other individuals those have worse fitness value in their population by the elitism process [43].

#### 4.1.5 Crossover and mutation probability

Crossover and mutation probabilities are used as parameters of the genetic algorithm. These may have a value

between 0 and 1. Thus, they are considered probabilities. If the "0" value is given to an operator, it means none of the individuals will be changed in this population as a result of that operator. Similarly, if "1" is given as a parameter, that means all the individuals, in other word, all populations will be changed. Both of those values are not to be given generally since they don't contribute efficiency of the algorithm. So, we choose a value between 0 and 1 for crossover and mutation probabilities in this study.

#### 4.2 Hybrid genetic local and global search

The genetic algorithm uses a selection and recombination operator for global exploration in the search space. During the run time, the genetic algorithm converges slowly toward the local optimum. After a few solution steps, the genetic algorithm sticks to a local optimum. Therefore, the proposed algorithm hybridizes the genetic algorithm for refining the search space and performing local exploration. The insertion search is used as a local search algorithm. The local search method plays important role in the search spaces. The insertion search is used for searching a small neighborhood. Also, the self-repair algorithm is used as a global search algorithm. The self-repair algorithm is responsible for searching a large neighborhood in a search space.

The flow chart of the HGLGSA is given in Fig. 1. Local and global search procedures were likewise utilized in addition to a hybrid genetic algorithm. After initialization, insert-search and self-repair algorithms are integrated into 3rd step. While insert search algorithm works as a local search algorithm, self-repair algorithm addresses to repair the chromosomes itself. Additionally, when the self-repair algorithm is executed, it sends  $a_2$  cut point parameter to the insert-search algorithm instead of  $a_1$ . We avoid designing a too greedy algorithm for preventing the trap of local optimal point. Thus,  $P_1$  and  $P_2$  chromosomes are chosen randomly rather than utilizing the best pair of chromosomes in the initial population.

The proposed insert search and self-repair algorithm are given in Figs. 2 and 3, respectively. Also, flow chart of the self-repair function is given in Fig. 4.

The insert-search is a kind of local search method which aims to find better fitness values by inserting a chosen gene to other possible location on the chromosome. It doesn't check all possible neighborhoods of all genes in the chromosome, but it searches different neighborhoods of a chosen genes in the chromosome.

Self-repair method, repeats to run self-repair function and insert-search methods up to reach a pre-assigned loop number. The main purpose of the self-repair function is to find out better fitness value by repairing the chromosome itself.

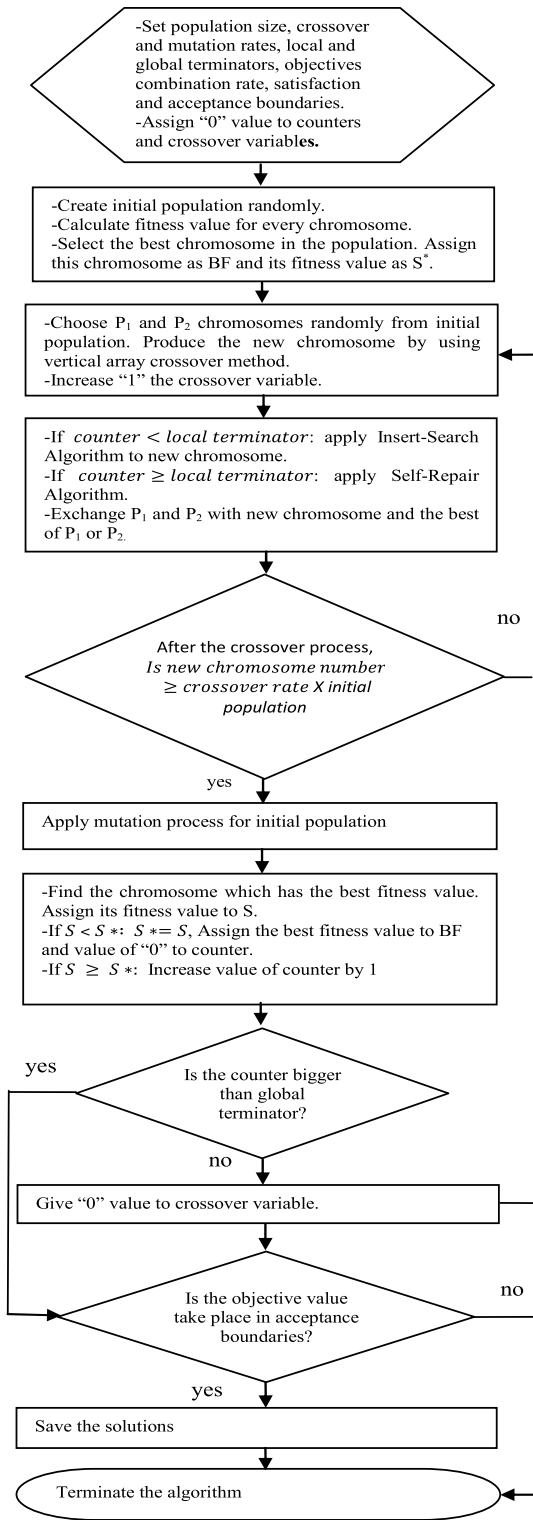


Fig. 1 Flow chart of proposed HGLGSA

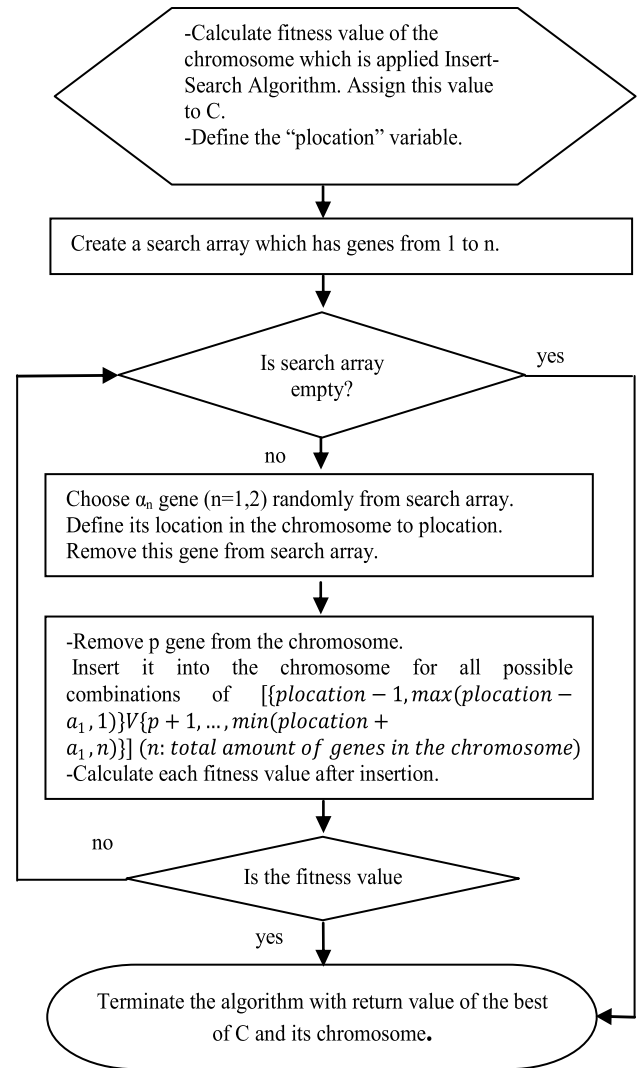


Fig. 2 Insert-search algorithm

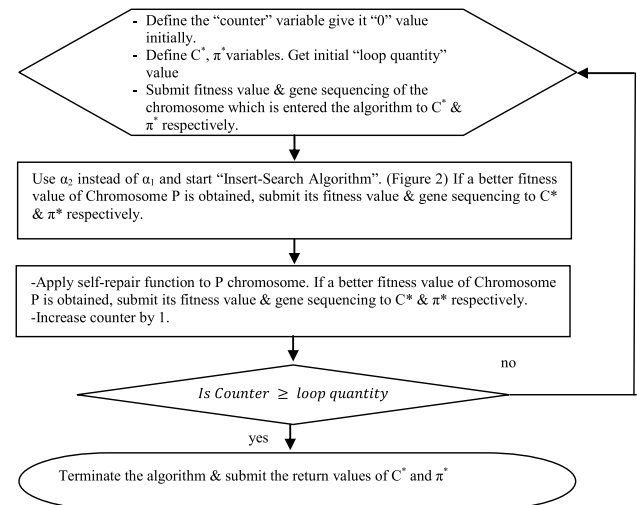


Fig. 3 Self-repair algorithm

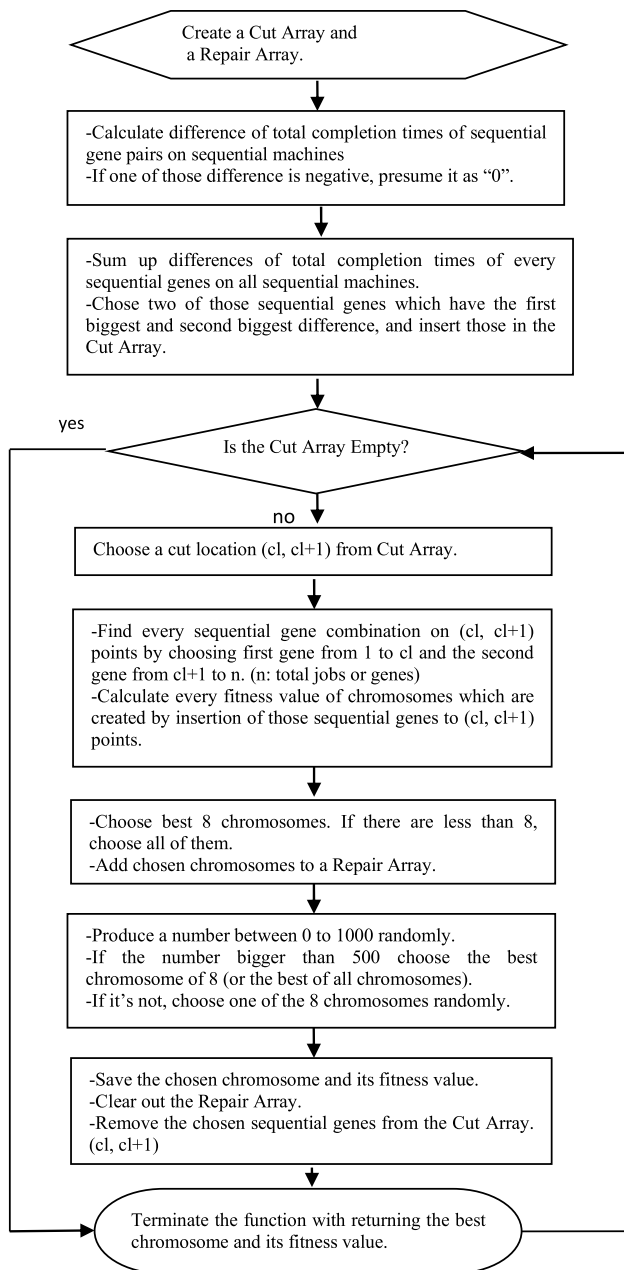


Fig. 4 Self-repair function

The function chooses the most problematic gene pair, where makes the biggest difference in total completion time as a repair location. Then, it explores to find better fitness values by inserting those genes into several locations on the chromosome. Also, it has a random chromosome selection process to prevent overpressure of elitist strategies.

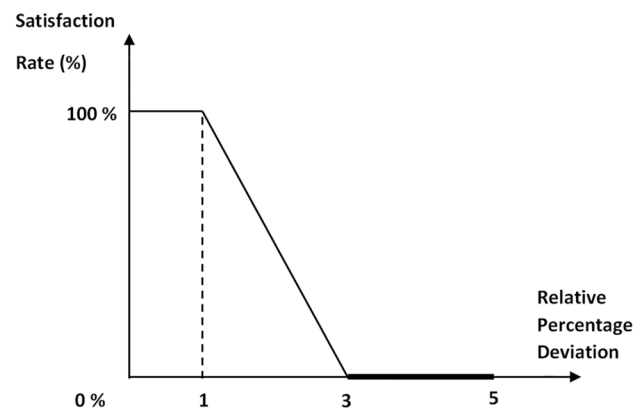


Fig. 5 The satisfaction function

## 5 Computational results

In this study, we suggested a self-repaired multi-objective hybrid genetic algorithm for NW-FSSWST. Moreover, we have tried to optimize the goals in Tseng and Lin [3]'s study which are  $C_{max}$  and  $\sum C_{ji}$  in a flow shop environment. Total flow time and total completion time were combined as a satisfaction function of Allouche et al. [35] to reflect the manager's preferences. The manager's preferences can explicitly express any deviation of the achievement from the aspiration level of each objective. The model that explicitly incorporates the manager's preferences through the satisfaction functions for objectives of makespan and total flow time is given in Eq. 6 [35].

$$\text{Maximize } \sum_{c=1}^2 SR_c(RPD_c) \quad (6)$$

where  $c$  is defined as the number of criteria (Makespan and total flow time);  $SR_c$  is defined as satisfaction rate for criterion  $c$ , and  $RPD_c$  is defined as relative percentage deviation for criterion  $c$ .

According to Allouche et al. [35]'s study, the best value of every goal could deviate a certain rate in a multi-objective based algorithm. Similarly, we built a satisfaction function. When the RPD is between 0 and 1%, the manager is completely satisfied. If the rate goes between 1 and 3%, SR is reducing from 100 to 0%. When the satisfaction rate gets a "0" value, there is no satisfaction but the solution is still acceptable up to %5 of deviation. After "0" satisfaction and 5% RPD, the solutions of the multi-objective model will be rejected. The satisfaction function is presented in Fig. 5.

**Table 1** The SRMOHGA's parameter levels used for the parameter optimization

Parameters	Values										
Initial population	5	10	15	20	25	30	35	40	45	50	
Crossover rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
Mutation rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
Local terminator	1	2	3	4	5	6	7	8	9	10	
Global terminator	1	2	3	4	5	6	7	8	9	10	
Loop quantity	1	2	3	4	5	6	7	8	9	10	
N (cut point)	0	1	2	3	4	5	6	7	8	9	
$\alpha_1$ (local terminator)	2	4	6	8	10	12	14	16	18	20	
$\alpha_2$ (global terminator)	2	4	6	8	10	12	14	16	18	20	

**Table 2** Parameters of NW-FSP

Parameters	Values
Initial population	10
Crossover rate	0.3
Mutation rate	0.1
Local terminator	3
Global terminator	6
Loop quantity	1
N (cut point)	3
$\alpha_1$ (local terminator)	2
$\alpha_2$ (global terminator)	4

In this study, 192 no-wait flow shop,  $n$  jobs  $\{n=8, 10, 12, 50, 100, 150, 200, \text{ and } 250\}$  and a set of  $m$  machines  $\{m=2, 3, 5, 8, 10, 15, 20, \text{ and } 25\}$ , with setup times problems that were generated by Engin and Günaydın [12], were used as benchmark instances. According to the processing times on each machine, these problems are classified  $a$ ,  $b$ , and  $c$  type problems. These problems are commonly used in the literature [12, 20, 44–46].

Thus, 192 problems were solved by proposed HGLGSA with two-performance measures Total Flow Time (TFT) and makespan ( $C_{max}$ ).

$$\begin{aligned} &\text{Minimize TFT} \\ &\text{Subject to } C_{max} \end{aligned}$$

### 5.1 Parameter optimization of HGLGSA for NW-FSSWST

The operating parameters of the HGLGSA are playing important role in the quality of the NW-FSSWST problem's solution. The parameters of HGLGSA are initial population number, crossover and mutation rates, local and global terminators, loop quantity, cut point, a range for insert-search local terminator ( $\alpha_1$ ), and search range of global terminator ( $\alpha_2$ ). The parameter levels are shown in Table 1.

A full factorial experimental design was performed for obtaining a set of the best parameters, which are given in Table 2.

### 5.2 Calibration of proposed HGLGSA for NW-FSSWST

The proposed HGLGSA was calibrated by Engin and Günaydın [12]'s benchmark instances.

The performance of the HGLGSA is compared with the GEN-2 (Aldowaisan and Allahverdi's genetic algorithm for no-wait flow shop problems [44]), ALA (Engin and Günaydın's Adaptive Learning Algorithm [12]), and HACO (Engin and Güçlü's Hybrid Ant Colony Optimization [20]) as a Relative Percentage Deviation (RPD). The RPD is given in Eq. 7 [47].

$$RPD = \frac{\text{Heuristic Algorithm}(C_{max}) - \text{Best}(C_{max})}{\text{Best}(C_{max})} \times 100 \tag{7}$$

At Eq. 7,  $\text{Best}(C_{max})$  is the minimum makespan- $C_{max}$  of the Engin and Günaydın [12]'s benchmark instances the best values found by CPLEX solutions [12, 20, 44]. Also, the heuristic algorithms are defined as the proposed HGLGSA, GEN-2 [44], ALA [12], and HACO [20].

Solution performance of the proposed HGLGSA, GEN-2 [44], ALA [12], and HACO [20] is given in Table 3 for 48 problems totally.

Average Relative Percentage Deviation (ARPD) of proposed HGLGSA is compared to the GEN-2 [44], ALA [12], and HACO [20] algorithms. The ARPD is defined with Eq. 8. For each problem group, the number of instances is given as  $l$  ( $l = 1, \dots, l$ ) notation at Eq. 8 [20].

$$ARPD = \frac{\sum_{L=1}^l ARPD(L)}{l} \tag{8}$$



**Table 3** RPD of HGLGSA, GEN-2, ALA and HACO

Problem	RPD				Problem	RPD			
	GEN-2	ALA	HACO	HGLGSA		GEN-2	ALA	HACO	HGLGSA
8×2-a	0.0000	0.0000	0.0000	0.0000	10×2-a	0.0000	0.0000	0.0000	0.0000
8×2-b	0.0000	0.0000	0.0000	0.0000	10×2-b	0.2967	2.0771	1.4836	0.0200
8×2-c	0.0000	0.6451	0.6451	0.0100	10×2-c	2.0654	0.5163	0.3442	0.0100
8×3-a	0.0000	0.0000	0.0000	0.0000	10×3-a	1.7094	2.5641	0.8547	0.0300
8×3-b	0.0000	4.8278	4.1379	0.0500	10×3-b	4.6961	1.3812	0.0000	0.0100
8×3-c	1.2544	0.7168	0.3584	0.0100	10×3-c	3.8980	3.1484	0.7496	0.0300
8×5-a	0.7937	0.0000	0.0000	0.0000	10×5-a	2.8985	1.4492	0.0000	0.0000
8×5-b	0.0000	0.5050	0.5050	0.0100	10×5-b	6.9264	11.0389	8.4415	0.1100
8×5-c	1.0753	0.0000	0.0000	0.0000	10×5-c	4.4673	3.5509	2.4054	0.0400
8×8-a	0.0000	0.0000	0.0000	0.0000	10×8-a	1.1834	1.1834	0.5917	0.0100
8×8-b	0.0000	0.0000	0.0000	0.0000	10×8-b	8.2311	9.1068	7.5306	0.0900
8×8-c	0.0000	0.0000	0.0000	0.0000	10×8-c	9.6654	9.6654	8.3643	0.1000
8×10-a	1.3245	0.6622	0.6622	0.0100	10×10-a	3.4090	4.5454	1.1363	0.0500
8×10-b	0.0000	0.0000	0.0000	0.0000	10×10-b	6.8627	10.9477	7.1895	0.1100
8×10-c	0.0000	0.6024	0.0000	0.0100	10×10-c	11.3028	9.9534	2.8472	0.0600
8×15-a	0.0000	0.0000	0.0000	0.0000	10×15-a	3.7209	2.7906	0.0000	0.0200
8×15-b	0.5174	0.5174	0.2587	0.0100	10×15-b	4.4758	3.5335	0.8244	0.0400
8×15-c	0.0000	0.0000	0.0000	0.0000	10×15-c	3.1250	4.5955	1.5913	0.0500
8×20-a	0.4629	0.0000	0.0000	0.0000	10×20-a	2.7237	1.9455	1.1673	0.0300
8×20-b	0.5636	0.5636	0.5636	0.0100	10×20-b	4.5062	3.6433	1.8216	0.0400
8×20-c	0.0000	0.0000	0.0000	0.0000	10×20-c	4.4444	3.5555	1.8271	0.0400
8×25-a	0.0000	0.0000	0.0000	0.0000	10×25-a	3.6363	3.2727	1.0909	0.0300
8×25-b	2.6666	1.8095	1.1428	0.0200	10×25-b	4.6025	4.4351	1.6736	0.0400
8×25-c	2.4865	0.0000	0.0000	0.0000	10×25-c	3.6720	2.0068	0.0000	0.0200

**Table 4** The results of the t-test for proposed HGLGSA and HACO [20]

Problem machines (m)	HACO-RPD average	HGLGSA-RPD average
2	0.412	0.006
3	1.016	0.021
5	1.893	0.026
8	2.747	0.033
10	1.972	0.038
15	0.445	0.020
20	0.896	0.020
25	0.652	0.018
Mean	1.374	0.025
Variance	0.708	5.899E-05
Pearson correlation	0.858	
$P(T < = t)$ two-tail	0.0052	

The 48-benchmark instances, ARPD for proposed HGLGSA, GEN-2, ALA, and HACO are calculated as 0.0233; 2.3679; 2.3282, and 1.2543, respectively. The proposed HGLGSA has found a smaller ARPD than GEN-2, ALA, and HACO. These results showed that the proposed HGLGSA

has better solution performance on NW-FSSWST. Also to better understand the performance of the proposed HGLGSA, the t-test is done with 95% significance for m (2, 3, 5, 8, 10, 15, 20, 25)—machines instances. The results are given in Table 4. The average RPD values, in Table 4, are analyzed on a box plot in Fig. 6.

In Fig. 6, for average HACO-RPD and HGLGSA-RPD box plots are given. Also, minimum, first quartile (Q1), median, third quartile (Q3), maximum, mean, and range values were calculated. It can be seen in Table 4 and Fig. 6, the proposed HGLGSA performance is better than HACO.

The CPU time of the proposed HGLGSA is compared with the GEN-2 [44], ALA [12], and HACO [20] algorithms. The results are given in Table 5.

It can be seen in Table 5, the CPU time of the proposed HGLGSA is shorter than GEN-2, ALA, and HACO.

### 5.3 Computational results of two performance measures

The 192 NW-FSSWST benchmark instances, which were created by Engin and Gunaydin [12], were solved by proposed HGLGSA for minimizing  $C_{max}$  and  $\sum C_j$  values. Computational results are given in Table 6.

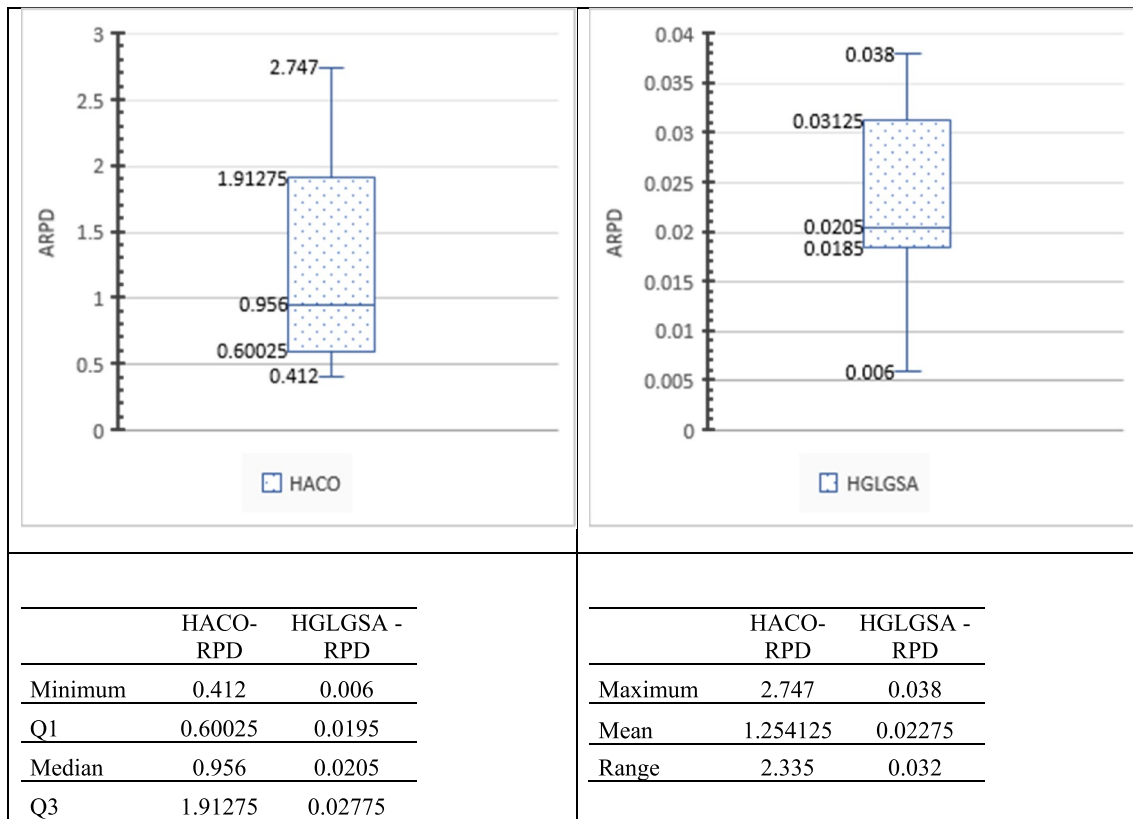


Fig. 6 Box plots for the average RPD values

**Table 5** The maximum CPU times of the proposed HGLGSA, GEN-2 [44], ALA [12], and HACO [20]

Heuristic algorithms	Configuration of the computer	Maximum CPU times (s)
Proposed HGLGSA	AMD turion 64 2.2 GHz—4 GB memory	87
GEN-2	PC pentium IV—3.0 GHz—1 GB memory	156
ALA	PC Pentium IV—3.0 GHz—1 GB memory	110
HACO	AMD Turion 64—2.0 GHz—1 GB memory	110

When we are analyzing Table 6, we see that deviation values of  $C_{max}$  and  $\sum C_j$  are raised generally if the amount of machines and jobs are increased. There are just 3 positive  $\sum C_j$  deviation values for 2 machines; on the other hand, there are 12 positive  $\sum C_j$  deviation values for 25 machines. Similarly, while there are 16 “0”  $C_{max}$  deviation values for 8 jobs, there is just one “0”  $C_{max}$  deviation value for 250 jobs.

Satisfaction status based on  $C_{max}$  and  $\sum C_j$  of HGLGSA is given in Table 7 for “completely satisfied” “satisfaction level” and “acceptance borders” to minimize  $\sum C_j$  and  $C_{max}$  values of every job-machine combination.

Also, the satisfaction status based on  $\sum C_j$  and  $C_{max}$  objectives is given in Fig. 7 by graphical results.

As seen in Table 7 and Fig. 7, full satisfaction is obtained in 156 of 192 problems based on  $\sum C_j$  objective deviation. While 29 of 192 deviations are satisfaction levels, 6 deviations are acceptance borders. One deviation is rejected.

Also, 56 of 192 fully satisfied deviations are found regarding to  $C_{max}$  objective value. While 11 of 192 are satisfaction levels, 6 of 192 are acceptance borders. 119 of 192 are rejected.

**Table 6** Computational results of HGLGSA in case of satisfaction function

Machine	Job	Problem type	Deviation from single objective		Machine	Job	Problem type	Deviation from single objective			
			$\sum C_j$	$C_{max}$				$\sum C_j$	$C_{max}$		
2	8	a	0	0	3	8	a	0	0		
		b	0	0			b	1	3		
		c	0	0			c	0	1		
	10	a	0	0		10	a	1	1		
		b	0	2			b	1	0		
		c	0	1			c	0	0		
	12	a	0	1		12	a	0	2		
		b	0	0			b	0	0		
		c	0	0			c	0	0		
	50	a	0	5		50	a	0	9		
		b	0	10			b	0	14		
		c	0	11			c	0	16		
	100	a	0	7		100	a	0	11		
		b	0	14			b	0	14		
		c	4	17			c	0	18		
	150	a	0	8		150	a	1	11		
		b	0	16			b	0	19		
		c	0	15			c	3	23		
	200	a	1	10		200	a	0	12		
		b	0	0			b	0	19		
		c	0	0			c	2	23		
	250	a	0	9		250	a	0	13		
		b	0	16			b	1	19		
		c	1	18			c	0	21		
	5	8	a	0		0	8	8	a	0	0
			b	0		1			b	1	0
			c	0		0			c	0	0
10		a	3	0	10	a		0	0		
		b	0	9		b		0	1		
		c	3	2		c		1	10		
12		a	1	6	12	a		1	1		
		b	0	5		b		4	0		
		c	2	7		c		0	5		
50		a	2	9	50	a		0	9		
		b	0	19		b		1	17		
		c	0	17		c		0	17		
100		a	0	13	100	a		1	0		
		b	4	24		b		0	0		
		c	1	25		c		1	0		
150		a	0	13	150	a		3	15		
		b	0	19		b		0	23		
		c	0	20		c		0	25		
200		a	1	15	200	a		0	16		
		b	0	23		b		0	24		
		c	0	0		c		1	26		
250		a	0	14	250	a		0	15		
		b	1	23		b		2	25		
		c	0	21		c		0	22		

**Table 6** (continued)

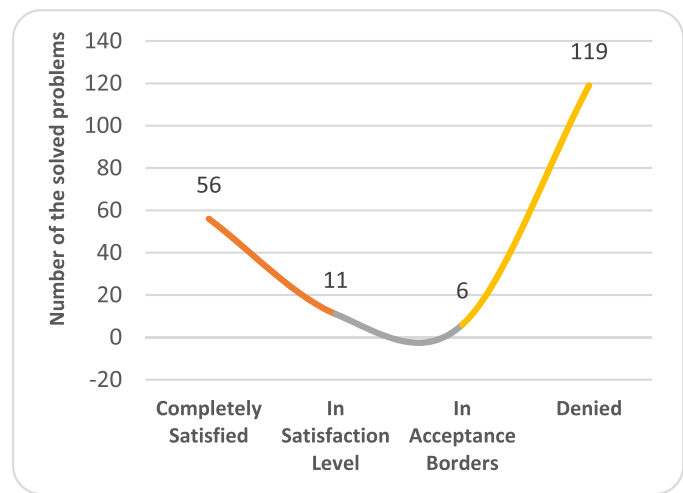
Machine	Job	Problem type	Deviation from single objective		Machine	Job	Problem type	Deviation from single objective	
			$\sum C_j$	$C_{max}$				$\sum C_j$	$C_{max}$
10	8	a	0	1	15	8	a	0	0
		b	0	0			b	0	1
		c	0	0			c	0	0
	10	a	0	3		10	a	0	0
		b	0	4			b	0	0
		c	0	0			c	0	2
	12	a	0	5		12	a	4	8
		b	3	0			b	0	0
		c	0	3			c	0	1
	50	a	1	11		50	a	0	11
		b	5	19			b	2	13
		c	3	20			c	0	15
	100	a	0	13		100	a	0	13
		b	0	21			b	2	0
		c	4	25			c	6	24
	150	a	0	16		150	a	2	17
		b	1	24			b	0	24
		c	0	26			c	3	23
	200	a	3	17		200	a	0	18
		b	0	23			b	0	26
		c	0	29			c	0	24
	250	a	1	17		250	a	1	20
		b	1	25			b	1	26
		c	0	27			c	0	27
20	8	a	0	0	25	8	a	0	0
		b	0	1			b	2	2
		c	0	0			c	0	10
	10	a	1	0		10	a	0	0
		b	0	4			b	0	0
		c	2	0			c	1	0
	12	a	0	0		12	a	1	7
		b	0	0			b	1	0
		c	0	0			c	2	8
	50	a	2	12		50	a	2	16
		b	0	12			b	2	0
		c	0	14			c	0	18
	100	a	3	18		100	a	0	13
		b	1	24			b	3	30
		c	2	27			c	0	0
	150	a	1	19		150	a	0	16
		b	0	22			b	0	26
		c	2	31			c	0	26
	200	a	0	17		200	a	2	21
		b	0	27			b	2	24
		c	1	30			c	1	29
	250	a	0	21		250	a	0	20
		b	2	28			b	0	29
		c	2	0			c	1	28

**Table 7** Satisfaction status based on  $\sum C_j$  and  $C_{max}$  objectives

Machines	Solved problems							
	Deviations based on $\sum C_j$				Deviations based on $C_{max}$			
	Com-pletely satisfied	In satis-faction level	In accept-ance borders	Denied	Com-pletely Satisfied	In satis-faction level	In accept-ance borders	Denied
2	23	–	1	–	10	1	1	12
3	22	2	–	–	7	2	–	15
5	19	4	1	–	5	1	1	17
8	21	2	1	–	10	–	1	13
10	19	3	2	–	2	5	2	15
15	18	4	1	1	6	1	–	17
20	17	7	–	–	9	–	1	14
25	17	7	–	–	7	1	–	16
Total	156	29	6	1	56	11	6	119



**a** For  $\sum C_j$  objective



**b** For  $C_{max}$  objective

**Fig.7** The satisfaction status based on  $\sum C_j$  and  $C_{max}$  objectives

According to those results, multi-objective problem structure does not affect  $\sum C_j$  values deeply. Calculated deviation values are distributed in a narrow interval.

Because of the multi-objective structure, we could have obtained better  $\sum C_j$  values in some cases. Also, the most of  $\sum C_j$  values have taken place in the full satisfaction area.

### 6 Conclusions and future research

In this study, we suggested an insert-search algorithm and self-repair algorithm with a self-repair function to develop our HGLGSA. HGLGSA was proposed to solve the NW-FSSWS, which are well-known NP-hard problems

in the literature. Firstly, the proposed HGLGSA was calibrated by GEN-2, ALA, and HACO in the literature for no-wait flow shop scheduling problems. Secondly, a full factorial design was performed to determine the best parameter sets in the HGLGSA for NW-FSSWST with two criteria that enhance the performance of the algorithm. Then Engin and Gunaydin [12]’s benchmark NW-FSSWST problems were solved by the proposed HGLGSA for two performance measures which are total flow time and makespan. The parameters of the proposed HGLGSA are the number of initial populations, the rates of crossover and mutation, execution times of local and global search, loop quantity, cut point (N), the levels of  $\alpha_1$  and  $\alpha_2$ . Finally, Engin and Gunaydin [12]’s 192 test benchmark instances were solved based on the satisfaction function

for bicriteria by using the best parameter set of the proposed HGLGSA. A full satisfaction was obtained in 156 of 192 problems based on  $\sum C_j$  objective deviation and 11 of 192 problems based on  $C_{\max}$  objective value.

Experimental results showed that the proposed HGLGSA is a highly effective meta-heuristic algorithm for solving NW-FSSWST with two performance criteria. Moreover, the multi-objective structure has not affected  $\sum C_j$  objective values dramatically but, it has caused to obtain longer  $C_{\max}$  values relatively.

Future studies are suggested to focus on different meta-heuristic methods to get fully satisfied  $C_{\max}$  values in the multi-objective environment for NW-FSSWST problems.

#### Declarations

**Conflict of interest** The authors declare that they have no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

#### References

- Baysal ME, Durmaz T, Sarucan A, Engin O (2012) To solve the open shop scheduling problems with the parallel kangaroo algorithm. *J Fac Eng Archit Gazi Univ* 27:855–864
- Allahverdi A, Aydilek H, Aydilek A (2018) No-wait flowshop scheduling problem with two criteria: total tardiness and makespan. *Eur J Oper Res* 269:590–601
- Tseng LY, Lin YT (2009) A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. *Eur J Oper Res* 198:84–92
- Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no wait in process. *Oper Res* 44(3):510–525
- Allahverdi A (2016) A survey of scheduling problems with no-wait in process. *Eur J Oper Res* 255:665–686
- Grabowski J, Pempera J (2005) Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Comput Oper Res* 32:2197–2212
- Wang TY, Yang YH, Lin HJ (2006) Comparison of scheduling efficiency in two/tree machine no wait flow shop problem using simulated annealing and genetic algorithm. *Asia-Pacific J Oper Res* 23:41–59
- Liu B, Wang L, Jin YH (2007) An effective hybrid particle swarm optimization for no wait flow shop scheduling. *Int J Adv Manuf Technol* 31:1001–1011
- Laha D, Chakraborty UK (2009) A constructive heuristic for minimizing makespan in no-wait flow shop scheduling problem. *Int J Adv Manuf Technol* 41:97–109
- Pan QK, Wang L, Qian B (2008) A novel multi objective particle swarm optimization algorithm for no wait flow shop scheduling problems. *Proc Inst Mech Eng Part B J Eng Manuf* 222:519–538
- Tseng LY, Lin YT (2010) A hybrid genetic algorithm for no-wait flowshop scheduling problem. *Int J Prod Econ* 128:144–152
- Engin O, Günaydın C (2011) An adaptive learning approach for no-wait flowshop scheduling problems to minimize makespan. *Int J Comput Intell Syst* 4:521–529
- Chaudhry IA, Khan AM (2012) Minimizing makespan for a no-wait flowshop using genetic algorithm. *Indian Acad Sci* 37:695–707
- Allahverdi A, Aydilek H (2014) Total completion time with makespan constraint in no-wait flowshops with setup times. *Eur J Oper Res* 238:724–734
- Panwalkar SS, Koulamas C (2014) The two-machine no-wait general and proportionate open shop makespan problem. *Eur J Oper Res* 238:471–475
- Lin SW, Ying KC (2016) Optimization of makespan for no-wait flowshop scheduling problems using efficient metaheuristics. *Omega* 64:115–125
- Ye H, Li W, Miao E (2016) An effective heuristic for no-wait flow shop production to minimize makespan. *J Manuf Syst* 40:2–7
- Ye H, Li W, Abedin A (2017) An improved heuristic for no-wait flow shop to minimize makespan. *J Manuf Syst* 44:273–279
- Shao W, Pi D, Shao Z (2017) Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. *Knowl-Based Syst* 137:163–181
- Engin O, Güçlü A (2018) A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Appl Soft Comput* 72:166–176
- Zhao F, Liu H, Zhang Y, Ma W, Zhang C (2018) A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. *Expert Syst Appl* 91:347–363
- Filho GR, Negano MS, Lorena LAN, (2007) Hybrid evolutionary algorithm for flow time minimization no wait flow shop scheduling. In: Mexican international conference on artificial intelligence, 1099–1109
- Akhshabi M, Tavakkoli-Moghaddam R, Rahnamay-Roodposhti F (2014) A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *Int J Adv Manuf Technol* 70:1181–1188
- Laha D, Gupta JND (2016) A Hungarian penalty-based construction algorithm to minimize makespan and total flow time in no-wait flow shops. *Comput Ind Eng* 98:373–383
- Koulamas C, Panwalkar SS (2018) New index priority rules for no-wait flow shops. *Comput Ind Eng* 115:647–652
- Zhu H, Qi X, Chen F, He X, Chen L, Zhang Z (2019) Quantum-inspired cuckoo co-search algorithm for no-wait flow shop scheduling. *Appl Intell* 49:791–803
- Allahverdi M, Allahverdi A (2020) Minimizing total completion time in a two-machine no-wait flowshop with uncertain and bounded setup times. *J Ind Manag Optim* 16(5):2439–2457
- Allahverdi A, Aldowaisan T (2004) No-wait flowshops with bicriteria of makespan and maximum lateness. *Eur J Oper Res* 152:132–147
- Tavakkoli-Moghaddam RT, Vahed AR, Mirzaei AH (2008) Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm. *Int J Adv Manuf Technol* 36:969–981

30. Rahimi-Vahed AR, Javadi B, Rabbani M, Tavakkoli-Moghaddam RT (2008) A multi objective scatter search for a bi-criteria no-wait flow shop scheduling problem. *Eng Optim* 40:331–346
31. Pan QK, Wang L, Tasgetiren MF, Zhao BH (2008) A hybrid discrete particle swarm optimization algorithm for no wait flow shop scheduling problem with makespan criterion. *Int J Adv Manuf Technol* 38:337–347
32. Aydilek H, Allahverdi A (2012) Heuristic for no-wait flowshops with makespan subject to mean completion time. *Appl Math Comput* 219:351–359
33. Aydilek H, Allahverdi A (2013) New heuristic for no-wait flowshops with performance measures of makespan and mean completion time. *Int J Oper Res* 10:29–37
34. Allahverdi A, Aydilek H (2013) Algorithms for no-wait flowshops with total completion time subject to makespan. *Int J Adv Manuf Technol* 68:2237–2251
35. Allouche MA, Aouni B, Martel JM, Loukil T, Rebaï A (2009) Solving multi-criteria scheduling flow shop problem through compromise programming and satisfaction functions. *Eur J Oper Res* 192:460–467
36. Martel JM, Aouni B (1990) Incorporating the decision-maker's preferences in the goal programming model. *J Oper Res Soc* 41:1121–1132
37. Engin O, Ceran G, Yilmaz MK (2011) An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Appl Soft Comput* 11:3056–3065
38. Kahraman C, Engin O, Kaya I, Yilmaz MK (2008) An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *Int J Comput Intell Syst* 1:134–147
39. Engin O, Figlali A (2002) To identify the best crossover operators in genetic algorithm for flow shop scheduling problems. *Dogus Univ J* 6:27–35
40. Engin O (2001) The parameter optimization for increasing solution performance of flow shop scheduling problems by using genetic algorithm", *ITU institute of science and technology*. Doctoral Thesis Dissertation, Istanbul
41. Croce FD, Tadei R, Volta G (1995) A genetic algorithm for the job shop problem. *Comput Oper Res* 22:15–24
42. Goldberg DE (1989) *Genetic algorithms in search. Optimization and machine learning*. addison wesley publishing company, USA
43. Mansfield RA (1990) *Genetic algorithms*. University of Wales. College of Cardiff. School of Electrical Electronic and Systems Engineering
44. Aldowaisan T, Allahverdi A (2003) New heuristics for no-wait flowshops to minimize makespan. *Comput Oper Res* 30:1219–1231
45. Rajendran C (1994) A no-wait flowshop scheduling heuristic to minimize makespan. *J Oper Res Soc* 45:472–478
46. Shyu SJ, Lin BMT, Yin PY (2004) Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. *Comput Ind Eng* 47:181–193
47. Engin BE, Engin O (2020) A new memetic global and local search algorithm for solving hybrid flow shop with multiprocessor task scheduling problem. *SN Appl Sci* 12:1–14

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.