BOHR International Journal of Operations Management Research and Practices
2026, Vol. 5, No. 1, pp. 1–10
DOI: 10.54646/bijomrp.2026.38
www.bohrpub.com

CASE STUDY

Application of GATS, a hybrid meta-heuristic model of genetic algorithm and tabu search, to solve a real-size problem of flow-shop scheduling with changeover times in operations: A case study in the flushing-kit-manufacturing industry

Phong Nguyen Nhu[1]*, Kim Ngan Nguyen Thi[1] and Tu Anh Nguyen Nhu[2]

[1]University of Technology – VNU-HCM, Ho Chi Minh City, Vietnam
[2]Monash University, Melbourne, Australia

***Correspondence:**
Phong Nguyen Nhu,
nnphong@hcmut.edu.vn

Received: 06 May 2025; **Accepted:** 08 October 2025; **Published:** 09 January 2026

Flow shop scheduling (FSS) problems are nondeterministic polynomial (NP)-hard combinatorial optimization problems. It is quite difficult to achieve an optimal solution for real-size problems with mathematical modeling approaches. Meta-heuristics algorithms, like genetic algorithm (GA) and tabu search (TS), play a major role in searching for near-optimal solutions for NP-hard optimization problems. The scheduling method in the case study is not effective; the total tardiness time of orders is rather high. This paper develops a genetic algorithm and tabu search (GATS) algorithm for solving the real FSS problem, with the objective to schedule orders more effectively than the current earliest due date (EDD) model. The GATS algorithm is a hybrid meta-heuristic model, combining GA and TS. In the model, GA is used as the platform for global search, and TS is used to support GA in local search. The performance of the algorithm is compared with the heuristic EDD model being used. The result shows that the algorithm is a good approach for FSS problems. However, the factors of the algorithm are chosen empirically, so the results are only better than the results of the current approach, and not really satisfactory. Future research is to use the experimental design to identify the algorithm's factors to obtain better results.

Keywords: hybrid meta-heuristic, genetic algorithm, tabu search, flow shop scheduling problems, changeover times.

Introduction

Scheduling is the allocation of resources to perform a collection of tasks over a period of time. Scheduling problems determine the order or sequence for processing a set of jobs through several machines in an optimal manner. Flow shop scheduling (FSS) problems consider m different machines and n jobs; each job consists of m operations, and each operation requires a different machine, and all the jobs are processed in the same processing order.

The problem to be solved is a FSS problem with the assumption that the orders are ready at the start of the scheduling process. The objective of the problem is to minimize the total weighted tardiness of orders. The constraints are on the sequence of orders, on the sequence of operations in the orders, and on machine changeover time.

The model of the problem is built based on the above assumptions, objective, and constraints. The genetic algorithm and tabu search (GATS) algorithm is built based on the combination of genetic algorithm (GA) and tabu search (TS) with the foundation of GA. Based on the model of the problem, the GATS algorithm will find a good solution for the problem; this solution will be compared with the solution of the currently used heuristic model to evaluate the effectiveness of the algorithm.

Literature review

FSS problems are nondeterministic polynomial (NP)-hard combinatorial optimization problems. For such problems, heuristics play a major role in searching for near-optimal solutions. Etiler et al., developed a genetic algorithm-based heuristic for the flow shop scheduling problem with makespan as the criterion (1). Complex GA algorithms have also been researched to solve the FFS problem effectively. Engin et al., developed an efficient GA for hybrid flow shop scheduling with multiprocessor task problems (2). Phong et al., developed a GA model to solve a flow shop scheduling problem, with the objective to reduce total weighted tardiness time and a constraint on changeover time in operations (3). Pezzella et al., developed a GA for the flexible job-shop scheduling problem. The algorithm integrates different strategies for generating the initial population, selecting the individuals for reproduction, and reproducing new individuals (4).

Gupta et al., designed a heuristic model based on TS for a two-stage flow shop problem (5). Phong et al., developed a TS model for a FSS problem (6). Burduk et al., applied TS and GA to solve production process scheduling problems (7). Umam et al., combined the TS process with a GA to solve the flow shop scheduling problem with the objective of minimizing makespan (8). Ben-Daya and Al-Fawzan proposed a TS approach for solving the permutation flow shop scheduling problem. The approach suggested simple techniques for generating neighborhoods of a given sequence and a combined scheme for intensification and diversification (9).

Phong et al., considered a pilot FSS problem with four different machines and 10 different jobs. The problem was solved by two different hybrid meta-heuristic models, GATS (10) and TSGA (11). Phong et al., developed a hybrid meta-heuristic model to solve FFS problems with nine orders, scheduling on four stations. The problem had the objective to minimize the total tardiness time and the constraints on workstation set-up times and production batch sizes (12).

In general, there are many hybrid meta-heuristic models, and the former models are rather complicated. This research proposes a simple combination of GA and TS. In the algorithm, GA is used as the platform for global search, and TS is used to support GA in local search. GA diversifies the search by exploring the search space, and TS intensifies the search by exploiting the best solutions found.

In each iteration, GA generates the elite population from the current population by using selection operators and generates genetic populations from the elite population by using cross-over and mutation operators. Then TS generates the neighborhood population from the genetic population by using neighborhood operators. Finally, GA generates the next population from the current population, the elite population, and the neighborhood population by using replacement operators. The detailed research methodology is shown in the next section.

Research methodology

The FSS problem is a NP-hard problem with a large size of solution space. The methodology used in this research to solve the problem includes four phases:

1) Phase A: Construct the model of the FSS problem.
2) Phase B: Construct the GATS algorithm.
3) Phase C: Use the GATS algorithm to solve the pilot FSS problem.
4) Phase D: Use the GATS algorithm to solve the real FSS problem.

Phase A: construct the model of the FSS problem

In phase A, the model of the problem is formulated with the objective of minimizing the total weighted tardiness time and constraint on the sequence of orders, on the sequence of operations in the orders, and on machine changeover time. The FSS model has indexes, parameters, and variables as in **Table 1**.

In the case, there are four machines (N = 4), M_1, M_2, M_3, and M_4. Each order has three parts, P1, P2, and P3, processed in eight operations, O_j, j = 1÷8, distributed on the four machines as in **Figure 1**.

The constraints on the sequence of operation on each order are as follows.

$TS_{i4} \geq TC_{i1}$,
$TS_{i5} \geq TC_{i2}$,
$TS_{i6} \geq TC_{i3}$,
$TS_{i7} \geq TC_{i5}$,

TABLE 1 | The indexes, parameters, and variables of the FSS model.

i	Order index, i = 1÷M
j	Machine index, j = 1÷N
M	Number of orders
N	Number of machines
k	The index of the next order of order i
h	The index of the previous order of order i
W_i	The weight of order i
D_i	The due date of order i
P_{ij}	The processing time of order i on machine j
S_{ijk}	The changeover times from order i on machine j to order k
TS_{ij}	The start time of order i on machine j
TC_{ij}	The completion time of order i on machine j
T_i	The tardiness time of order i
T	The total weighted tardiness time
Tbest	The best total weighted tardiness time

	M1	M2	M3	M4
P1	O1	O4	-	O8
P2	O2	O5	O7	
P3	O3	O6	-	

FIGURE 1 | The production process.

$TS_{i8} \geq \max(TC_{i4}, TC_{i6}, TC_{i7})$.

The start time of order i at operation j, TS_{ij} depends on the completion time of the previous order h, TC_{hj} and the changeover time S_{hji} between the orders on operation j.

$$TS_{ij} = TC_{hj} + S_{hji}$$

The completion time of order i on operation j, TC_{ij} is determined by the start time and processing time of the order.

$$TC_{ij} = TS_{ij} + P_{ij}$$

The tardiness time of order i, T_i is determined by the completion time of the last operation and due time of the order.

$$T_i = Max\,(0, TC_{i8} - D_i)$$

The total weighted tardiness is defined as follows:

$$T = \sum_{i=1}^{M} w_i T_i$$

The objective function that minimizes the total weight tardiness is defined as follows:

$$Tbest = Min\ T,\ T = \sum_{i=1}^{M} w_i T_i$$

The model of the problem is summarized as follows:

Min T

St.

$TS_{i4} \geq TC_{i1}$,

$TS_{i5} \geq TC_{i2}$,

$TS_{i6} \geq TC_{i3}$,

$TS_{i7} \geq TC_{i5}$,

$TS_{i8} \geq \max(TC_{i4}, TC_{i6}, TC_{i7})$.

$TS_{ij} = TC_{hj} + S_{hji}$

$TC_{ij} = TS_{ij} + P_{ij}$

$T_i = \text{Max}\ (0, TC_{i8} - D_i)$

$TS_{ij}, TC_{ij} \geq 0$.

Phase B: construct the GATS algorithm

The GATS algorithm combines GA and TS; GA is used to perform a global search of the solution space, and TS is used to perform a local search to refine the solution found by GA. The GATS procedure is as follows:

Step 1: Initialize the GATS model.

Step 2: Generate the initial population $\mathbf{P}^{(0)}$. Set k = 0.

Step 3: Generate elite population $\mathbf{P}_E{}^{(k)}$.

Step 4: Generate the genetic population $\mathbf{P}_G{}^{(k)}$.

Step 5: Generate the neighborhood population $\mathbf{P}_N{}^{(k)}$.

Step 6: Generate the next population $\mathbf{P}^{(k+1)}$. Set k = k+1.

Step 7: Check the termination rule. If No, return to step 3. If Yes, finish the loop.

Step 8: Run the algorithm a number of times to choose the best scheduling result.

Step 1: Initialize the GATS model

This step sets up the factors of GATS model, including:

- The method of coding
- The GA factors
- The TS factors
- The termination rule
- The number of runs

The method of coding: Schedule alternatives are coded by chromosomes. With N orders, each chromosome is a string of N genes. Each gene is corresponding to an order. The orders are numbered from 1 to N. The sequence of genes represents the sequence of order scheduled:

$$C = [G_1, G_2, \ldots, G_N]$$

The GA factors: The GA factors include fitness function F, the population size P, and the parameters of GA operators. The GA operators include the following parameters:

- The selection method
- The crossover probability P_c
- The crossover method
- The mutation probability P_m
- The mutation method
- The replacement method.

In the GATS model, the fitness function F is defined as follows:

$$F_i = T_{max} - - T_i.$$

Where F_i, T_i are the fitness and objective values of chromosome i, T_{max} is the maximum objective value in the population.

The selection method uses selection probabilities, defined by fitness values, to choose chromosomes into the elite population. The crossover method is *POX (Precedence Operation Crossover),* the mutation method is SWAP, and

the replacement method is acceptance threshold with threshold factor K.

The TS factors: The TS factors include only the *neighborhood method*. The neighborhood method uses the method of permutation of adjacent genes in the chromosomes to find the neighborhood.

The termination rule: The termination rule uses iteration factor I to terminate the algorithm. The algorithm will be terminated if the best objective value *Tbest* of the population does not improve or decrease, after I consecutive iterations.

The number of runs: The GATS algorithm is basically a stochastic model; it will be run a number of times R to choose the best result among the runs.

Step 2: Generate the initial population $P^{(0)}$. Set k = 0

The initial population consists of P chromosomes. Some chromosomes are generated from three heuristic rules, earliest due date (EDD), shortest processing time (SPT), and longest processing time (LPT). The remaining chromosomes are randomly generated.

Step 3: Generate elite population $P_E{}^{(k)}$

This step uses the selection operator to generate elite population $\mathbf{P}_E{}^{(k)}$ from the current population $\mathbf{P}^{(k)}$. Each chromosome in the current population has a corresponding fitness value F_i and is selected for inclusion in the elite population $\mathbf{P}_E{}^{(k)}$, with selection probability P_i defined by fitness value as follows:

$$P_i = \frac{F_i}{F},\ F = \sum_{i=1}^{P} F_i$$

Step 4: Generate the genetic population $P_G{}^{(k)}$

This step uses the crossover and mutation operators to generate genetic population $\mathbf{P}_G{}^{(k)}$ from the elite population $\mathbf{P}_E{}^{(k)}$. The genetic population $\mathbf{P}_G{}^{(k)}$ includes the new chromosome generated from the crossover and mutation operators.

The chromosomes of $\mathbf{P}_E{}^{(0)}$) are selected to be included in the crossover list $\mathbf{P}_c$ with the crossover probability P_c. Then each pair of chromosomes in $\mathbf{P}_c$ is selected to cross over by the defined crossover method. The chromosomes of $\mathbf{P}_E{}^{(0)}$) are also selected to be included in the mutation list $\mathbf{P}_m$ with the mutation probability P_m. Then each chromosome in $\mathbf{P}_m$ is selected to mutate by the defined mutation method.

Step 5: Generate the neighborhood population $P_N{}^{(k)}$

This step uses the neighborhood operator to generate the neighborhood population $\mathbf{P}_N{}^{(k)}$ from the genetic population $\mathbf{P}_G{}^{(k)}$. Each chromosome in $\mathbf{P}_G{}^{(k)}$ will have a neighborhood defined by the neighborhood operator. In this neighborhood, the best chromosome will be selected to be included in $\mathbf{P}_N{}^{(k)}$ to go forward.

Step 6: Generate the next population $P^{(k+1)}$. Set k = k+1

This step uses the replacement operator to generate the next population $\mathbf{P}^{(k+1)}$ from the populations $\mathbf{P}_G{}^{(k)}$ and $\mathbf{P}_N{}^{(k)}$. The chromosomes from $\mathbf{P}_G{}^{(k)}$ and $\mathbf{P}_N{}^{(k)}$ will be added to the current population $\mathbf{P}^{(k)}$ to make the next population $\mathbf{P}^{(k+1)}$, if their fitness values exceed the acceptable threshold, selected as the fitness value of the Tth chromosome of $\mathbf{P}^{(k)}$ in the ranking.

$$T = P/K$$

To keep the next population size constant, the chromosomes with the lowest values are removed from the next population.

Step 7: Check the termination rule

This step checks the termination rule. If the best objective value *Tbest* of the population improves or decreases. It will return to step 3. If the best objective value *Tbest* of the population does not improve or decrease after I consecutive iterations. It finishes the run.

Step 8: Run the algorithm a number of times to choose the best scheduling result

The algorithm will be run for R times; the best scheduling result among the runs will be chosen. The factors of the GATS algorithm are summarized in **Table 2**.

TABLE 2 | The factors of the GATS algorithm.

Factors	Values
Population size	P
Fitness function F	$F_i = T_{max} - T_i$
Selection method	Selection probability
Crossover probability	Pc
Crossover method	POX
Mutation probability Pm	0.2
Mutation method	SWAP
Replacement method	Acceptance threshold
Threshold coefficient	K
Neighborhood operator	Permutation
Iteration factor	I
Number of runs	R

TABLE 3 | The weight W_i, i = 1÷10 and the due date D_i, i = 1÷10 of order i.

i	1	2	3	4	5	6	7	8	9	10
W_i	3.70	3.40	3.30	4.65	3.90	2.35	2.70	4.55	4.65	4.30
D_i (h)	24	36	40	60	68	80	88	88	96	96

TABLE 4 | The processing time P_{ij} of order i, i = 1÷10, on operation j.

j	P_{1j}	P_{2j}	P_{3j}	P_{4j}	P_{5j}	P_{6j}	P_{7j}	P_{8j}	P_{9j}	P_{10j}
1	2.34	6.17	6.20	7.09	2.47	9.56	3.44	14.74	5.26	1.39
2	2.25	0.94	7.05	3.22	1.07	5.26	1.81	7.49	5.26	0.66
3	0.00	3.99	4.34	4.38	2.60	9.52	0.00	16.75	14.74	0.44
4	2.63	1.33	1.08	9.00	2.60	12.56	1.32	17.54	16.52	1.29
5	1.06	1.00	6.58	9.00	4.21	12.64	0.65	17.54	16.52	1.29
6	0.00	8.33	6.58	3.60	3.95	12.64	0.00	13.33	15.79	0.40
7	2.67	3.29	1.13	2.21	0.68	4.00	1.11	5.26	1.60	0.44
8	4.08	3.31	3.42	12.00	3.95	6.12	2.21	8.22	5.26	1.32

Phase C and D: use the GATS algorithm to solve the FSS problem

Based on the model of the problem and the GATS algorithm, the pilot problem of small size is solved in phase C to get experience in defining the parameters of the GATS algorithm to solve the real problem of real size in phase D. Phase C and phase D will be shown in the following sections.

The pilot FSS problem

The pilot problem to be solved is a FSS problem with 10 orders, O_i, i = 1÷10. The weight W_i and the due date D_i of order i are estimated in **Table 3**.

The processing time P_{ij} of order i, i = 1÷10, on operation j, j = 1÷8, are estimated in **Table 4**.

The changeover times on operation j, j = 4÷8 are equal to 0.

$$S_{ijk} = 0,\ j = 4 \div 8.$$

The changeover times in hours on operation j, j = 1÷3 are the same and depend on the current order i = 1÷10, and the next order, k = 1÷10, as shown in **Table 5**.

The company is currently using the EDD dispatching method. The sequence of dispatching **S**, the value of the objective function T are as follows.

$$S = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10);$$
$$T = 215.95(h)$$

The Gantt chart are shown in **Figure 2**.

The GATS model for the pilot FSS problem

The pilot FSS problem is a NP hard problem with the size of the solution space of 10! or 3,628,800. The GATS model is used to solve the problem as follows:

Step 1: Initialize the GATS model

This step sets up factors of GATS model, including the method of coding and all factors of the algorithm. Each chromosome is a string of 10 genes. Each gene corresponds to an order. The orders are numbered from 1 to 10. The sequence of genes represents the sequence of order scheduled:

$$C = [G1, G2, G3, G4, G5, G6, G7, G8, G9, G10]$$

The factors of the algorithm for the pilot problem are shown in **Table 6**.

Step 2: Generate the initial population $P^{(0)}$, set k = 0

The initial population consists of 10 chromosomes. There are three chromosomes generated from three heuristic rules: EDD, SPT, and LPT. The remaining chromosomes R1, . . ., R7 are randomly generated as shown in **Table 7**.

$$P^{(0)} = \{EDD, SPT, LPT, R1, R2, R3, R4, R5, R6, R7\}$$

TABLE 5 | Changeover time (h) S_{ijk}, j = 1÷3.

S_{ijk}	1	2	3	4	5	6	7	8	9	10
1	0.0	0.5	2.0	0.0	0.0	2.0	2.0	2.0	2.0	2.0
2	0.5	0.0	0.0	0.0	2.0	2.0	2.0	2.0	2.0	2.0
3	2.0	0.0	0.0	2.0	0.5	2.0	0.5	0.5	2.0	0.5
4	0.0	0.0	2.0	0.0	0.0	0.0	2.0	2.0	2.0	2.0
5	0.0	2.0	0.5	0.0	0.0	2.0	0.5	2.0	0.5	0.5
6	2.0	2.0	2.0	0.0	2.0	0.0	0.5	2.0	0.0	0.0
7	2.0	2.0	0.5	2.0	0.5	0.5	0.0	0.5	2.0	0.0
8	2.0	2.0	0.5	2.0	2.0	2.0	0.5	0.0	0.0	0.0
9	2.0	2.0	2.0	2.0	0.5	0.0	2.0	0.0	0.0	2.0
10	2.0	2.0	0.5	2.0	0.5	0.0	0.0	0.0	2.0	0.0

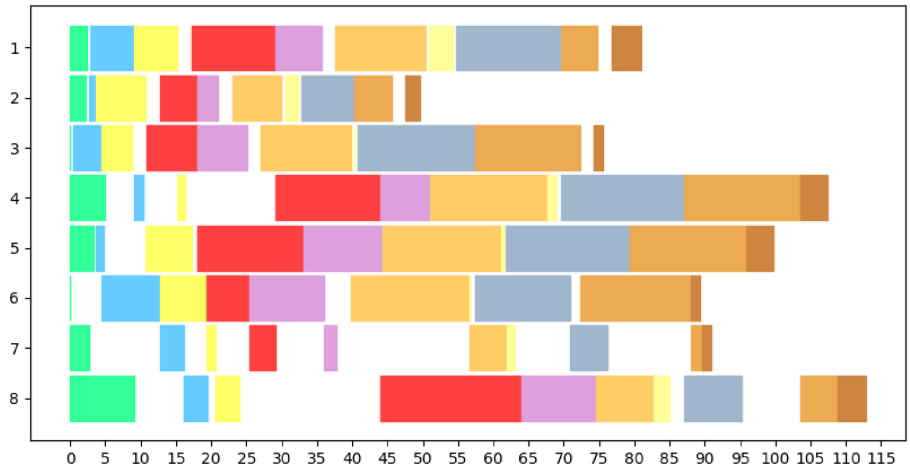

FIGURE 2 | The Gantt chart for the pilot problem by the EDD dispatching method.

TABLE 6 | The GATS factors for the pilot problem.

Factors	Values
Population size P	10
Fitness function F	$F_i = T_{max} - T_i$
Selection method	Selection probability
Crossover probability Pc	0.6
Crossover method	POX
Mutation probability Pm	0.2
Mutation method	SWAP
Replacement method	Acceptance threshold
Threshold coefficient K	2
Neighborhood operator	Permutation
Iteration factor I	10
Number of run R	5

Step 3: Generate elite population $P_E^{(0)}$

The step uses the selection operator to generate the elite population $\mathbf{P}_E^{(0)}$ from $\mathbf{P}^{(0)}$. Each chromosome in the current population has a corresponding fitness value F_i, and is selected for inclusion in the elite population $\mathbf{P}_E^{(0)}$, with selection probability P_i determined as follows:

$$P_i = \frac{F_i}{F}, \; F = \sum_{i=1}^{10} F_i$$

With population $\mathbf{P}^{(0)}$, the fitness values of F_i and selection probabilities P_i of chromosomes in $P^{(0)}$ are calculated as shown in Table 8.

Based on selection probabilities P_i, 10 random numbers are generated, and the chromosomes selected into the population $\mathbf{P}_E^{(0)}$ are as follows:

$$P_E^{(0)} = \{R3, R2, LPT, SPT, EDD, SPT, SPT, EDD, R2, R5\}$$

TABLE 7 | The initial population $P^{(0)}$.

$P^{(0)}$	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
EDD	1	2	3	4	5	6	7	8	9	10
SPT	5	3	2	1	4	10	7	8	9	6
LPT	10	7	1	5	2	3	4	6	9	8
R1	1	5	4	3	2	6	10	9	8	7
R2	5	3	4	1	2	10	8	9	6	7
R3	6	8	7	9	10	1	3	2	4	5
R4	10	6	9	7	8	5	1	4	2	3
R5	8	10	7	9	6	3	5	2	4	1
R6	7	8	10	9	6	2	3	5	4	1
R7	8	9	6	4	3	2	5	1	7	10

TABLE 8 | The values of F_i and P_i.

$P^{(0)}$	Fi	Pi
EDD	1217.6710	0.1754
SPT	1210.7427	0.1744
LPT	1120.0530	0.1613
R1	1094.8691	0.1577
R2	986.0005	0.1420
R3	455.6409	0.0656
R4	315.4120	0.0454
R5	283.1211	0.0408
R6	260.3407	0.0375
R7	0.0000	0.0000

Step 4: Generate the genetic population $P_G^{(0)}$

The genetic population $\mathbf{P}_G^{(0)}$ includes the new chromosome generated from the crossover and mutation operators.

TABLE 9 | The crossover population P^C.

P^C	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Fi
C1	2	1	3	4	5	10	7	8	9	6	1245.02
C2	5	3	1	2	4	6	7	8	9	10	1156.34
C3	10	1	5	2	3	6	7	4	9	8	1083.70
C4	1	7	2	3	4	5	8	6	9	10	1080.84
C5	7	9	10	1	3	6	2	8	4	5	603.65
C6	6	8	1	2	3	4	5	7	9	10	553.08
C7	10	7	1	5	4	2	3	9	8	6	1104.16
C8	5	3	2	1	10	7	4	6	8	9	1069.54
C9	8	7	10	1	3	2	4	5	9	6	806.04
C10	6	5	3	9	2	1	4	10	7	8	611.50
C11	6	8	7	5	10	1	3	2	9	4	617.67
C12	10	7	1	9	2	3	4	6	8	5	941.70

TABLE 10 | The mutation population P^M.

P^M	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Fi
M1	1	10	3	4	5	6	7	8	9	2	1030.69

TABLE 11 | The neighboring chromosomes of C1.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Fi
C1	2	1	3	4	5	10	7	8	9	6	1245.02
C11 (N1)	1	2	3	4	5	10	7	8	9	6	1285.22
C12	2	3	1	4	5	10	7	8	9	6	1282.16
C13	2	1	4	3	5	10	7	8	9	6	1259.76
C14	2	1	3	5	4	10	7	8	9	6	1268.09
C15	2	1	3	4	10	5	7	8	9	6	1212.12
C16	2	1	3	4	5	7	10	8	9	6	1248.52
C17	2	1	3	4	5	10	8	7	9	6	1265.44
C18	2	1	3	4	5	10	7	9	8	6	1228.27
C19	2	1	3	4	5	10	7	8	6	9	1205.94

The chromosomes of $\mathbf{P}_E{}^{(0)}$) are selected to be included in the crossover list $\mathbf{P}_c$ with the crossover probability of 0.6. After generating random numbers, the set $\mathbf{P}_c$ is determined as follows:

$$P_c = \{R3, LPT, SPT, EDD\}$$

Each pair of chromosomes in $\mathbf{P}_c$ is selected to cross over by the POX method, resulting in 12 new chromosomes in population $\mathbf{P}^C$ with corresponding fitness values as shown in **Table 9**.

The chromosomes of $\mathbf{P}_E{}^{(0)}$) are also selected to be included in the mutation list $\mathbf{P}_m$ with the mutation probability of 0.2. After generating random numbers, $\mathbf{P}_m$ is determined as follows:

$$P_m = \{EDD\}$$

Each chromosome in $\mathbf{P}_m$ is selected to mutate by the SWAP method, resulting in one new chromosome in population $\mathbf{P}^M$ as shown in **Table 10**.

TABLE 12 | The neighborhood population $P_N{}^{(0)}$.

$P_N{}^{(0)}$	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Fi
N1	1	2	3	4	5	10	7	8	9	6	1285.22
N2	5	3	1	2	4	6	7	8	10	9	1196.64
N3	10	1	5	3	2	6	7	4	9	8	1085.44
N4	1	7	3	2	4	5	8	6	9	10	1158.11
N5	7	9	10	1	3	6	2	4	8	5	707.71
N6	6	8	1	2	3	5	4	7	9	10	652.32
N7	10	7	1	5	2	4	3	9	8	6	1118.06
N8	5	3	2	1	10	7	4	8	6	9	1088.80
N9	8	7	10	1	3	2	5	4	9	6	879.98
N10	6	5	3	2	9	1	4	10	7	8	659.86
N11	6	7	8	5	10	1	3	2	9	4	657.54
N12	10	7	1	2	9	3	4	6	8	5	996.89
N13	1	10	3	4	5	6	7	8	2	9	1070.62

TABLE 13 | The next population $P^{(1)}$.

$P^{(1)}$	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Fi
N1	1	2	3	4	5	10	7	8	9	6	1285.22
C1	2	1	3	4	5	10	7	8	9	6	1245.02
EDD	1	2	3	4	5	6	7	8	9	10	1217.67
SPT	5	3	2	1	4	10	7	8	9	6	1210.74
N2	5	3	1	2	4	6	7	8	10	9	1196.63
N4	1	7	3	2	4	5	8	6	9	10	1158.11
C2	5	3	1	2	4	6	7	8	9	10	1156.34
LPT	10	7	1	5	2	3	4	6	9	8	1120.05
N7	10	7	1	5	2	4	3	9	8	6	1118.06
C7	10	7	1	5	4	2	3	9	8	6	1104.16

TABLE 14 | The results after 14 iterations.

Iteration	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Tbest
0	1	2	3	4	5	6	7	8	9	10	215.95
1	1	2	3	4	5	10	7	8	9	6	148.40
2	1	2	3	4	5	7	8	9	10	6	141.62
3	1	2	3	4	5	7	8	9	10	6	141.62
4	1	3	2	4	5	8	7	10	9	6	126.43
5	1	3	2	4	5	10	8	7	9	6	123.07
...	1	3	2	4	5	10	8	7	9	6	123.07
14	1	3	2	4	5	10	8	7	9	6	123.07

TABLE 15 | The results after four runs.

Run	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	Tbest	n
1	1	3	2	4	5	10	8	7	9	6	123.07	14
2	1	3	2	4	5	8	7	10	9	6	126.43	17
3	1	3	2	4	5	10	8	7	9	6	123.07	14
4	1	3	2	4	5	10	8	7	9	6	123.07	16
5	1	3	2	4	5	10	8	7	9	6	123.07	19

After crossover and mutation, 13 new chromosomes are created in the population $\mathbf{P}_G{}^{(0)}$:

$$P_G{}^{(0)} = \{C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, M1\}$$

Step 5: Generate the neighborhood population $P_N{}^{(0)}$

This step uses the neighborhood operator to generate the neighborhood population $\mathbf{P}_N{}^{(0)}$ from the genetic population $\mathbf{P}_G{}^{(0)}$. Each chromosome in $\mathbf{P}_G{}^{(0)}$ will have a neighborhood defined by the neighborhood operator. In this neighborhood, the best chromosome will be selected to go forward. For example, with C1, there are nine neighboring chromosomes, of which C11 is the best and is chosen, as shown in Table 11.

Same for the remaining chromosomes of $\mathbf{P}_G{}^{(0)}$. The neighborhood population $\mathbf{P}_N{}^{(0)}$ consists of the best neighbor chromosomes, as defined in Table 12.

$$P_N{}^{(0)} = \{N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13\}$$

Step 6. Generate the next population $P^{(1)}$

This step uses the replacement operator to generate the next population $\mathbf{P}^{(1)}$ from the populations $\mathbf{P}_G{}^{(0)}$ and $\mathbf{P}_N{}^{(0)}$. The chromosomes from $\mathbf{P}_G{}^{(0)}$ and $\mathbf{P}_N{}^{(0)}$ will be added to the current population $\mathbf{P}^{(0)}$ to make the next population $\mathbf{P}^{(1)}$, if their fitness values exceed the acceptable threshold. With the threshold coefficient K of 2 and the population size P of 10, the acceptable threshold is the fitness value of the 5th chromosome of $\mathbf{P}^{(0)}$ in the ranking.

To keep the next population size constant, the chromosomes with the lowest value are removed from the next population. After applying the replacement operator, the next population $\mathbf{P}^{(1)}$ is determined as in Table 13.

$$P^{(1)} = \{N1, C1, EDD, SPT, N2, N4, C2, LPT, N7, C7\}$$

Step 7. Check the termination rule

After iteration 1, N1 is the best chromosome with the best fitness value of 1285.22 and the best objective value of 148.40, appearing only once. The termination rule is not satisfied, so iteration 2 is executed. The results after 14 iterations are shown in Table 14.

Seeing that from the 5th iteration to the 14th iteration, the best objective value remains the same, the termination rule is satisfied, and the run ends. The scheduling result in this run is as follows:

$$S = (1, 3, 2, 4, 5, 7, 8, 9, 10, 6),$$
$$Tbest = 123.07(h)$$

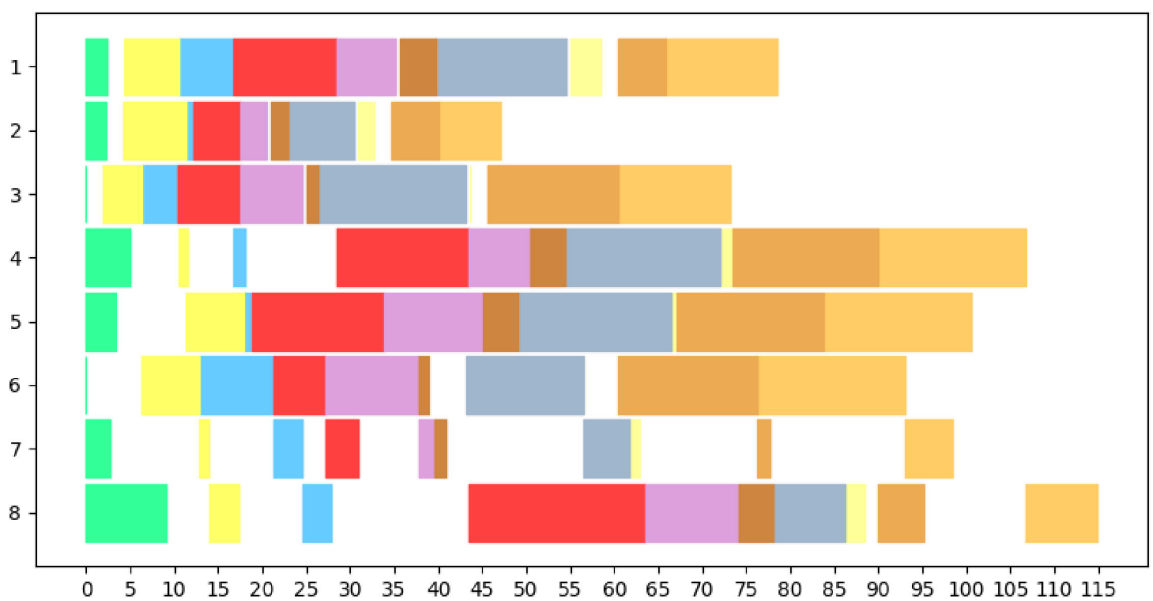

FIGURE 3 | The Gantt chart by the GATS model.

Step 8. Run the algorithm for five times to choose the best scheduling result

The algorithm is run five times with the results as shown in **Table 15**.

The best scheduling result is found on the 1st run, with a number of iterations n of 14. The sequence of dispatching S, the value of the objective function, and the Gantt chart are as follows:

$$S = (1, 3, 2, 4, 5, 7, 8, 9, 10, 6);$$
$$Tbest = 123.07(h)$$

Comparing to the currently used dispatching method, EDD, the value of the objective function T has been improved, reduced from 215.95 (h) to 123.07 (h).

The GATS model for the real FSS problem

The real problem to be solved is a FSS problem with 120 orders, scheduling on four machines. Each order has three parts, P1, P2, and P3, processed in eight operations, distributed on the four machines as in **Figure 1**. Problem data has been collected to estimate the parameters of the weight W_i, the due date D_i of order i, i = 1÷120, the processing time P_{ij} of order i, i = 1÷120, on operation j, j = 1÷8, and of the changeover times S_{ijk}, i = 1÷120, j = 1÷8, k = 1÷120.

The company is currently using the EDD dispatching method. The total weighted tardiness time T and the number of late delivery orders N of the real size problem are as follows:

$$T = 354.95\ (h),$$
$$N = 31.$$

The above GATS model has been coded on the Python platform and applied to solve the real problem. The real problem size is larger than the pilot problem size, so some factors of the GATS model have been changed to suit the real problem. The parameters or factors of the GATS model for the real problem are as in **Table 16**.

The sequence of the order to be scheduled by the GATS model is as follows: [1, 2, 3, 103, 5, 119, 6, 69, 8, 9, 11, 14, 12, 15, 13, 16, 17, 76, 18, 78, 21, 22, 24, 101, 25, 27, 10, 90, 28, 29, 31, 30, 33, 35, 32, 34, 37, 36, 38, 44, 39, 40, 41, 42, 19, 45, 47, 46, 49, 48, 52, 54, 51, 53, 55, 56, 57, 59, 91, 60, 61, 62, 63, 65, 64, 66, 7, 67, 68, 70, 85, 73, 75, 72, 77, 74, 43, 83, 79, 20, 81, 84, 82, 71, 87, 88, 89, 86, 26, 92, 58, 94, 95, 93, 96, 98, 99, 97, 100, 23, 102, 4, 105, 104, 106, 107, 108, 109, 111, 110, 50, 112, 80, 117, 116, 115, 114, 113, 118, 120]. The Gantt chart is shown in **Figure 3**.

TABLE 16 | The GATS factors for the real problem.

Factors	Values
Population size P	25
Selection method	Selection probability
Crossover probability Pc	0.8
Crossover method	POX
Mutation probability Pm	0.2
Mutation method	SWAP
Replacement method	Acceptance threshold
Threshold coefficient K	2
Neighborhood operator	Permutation
Iteration factor I	10
Number of run R	5

The total weighted tardiness time T and the number of late delivery orders N of the real size problem is as follows:

$$T = 277.50(h),$$
$$N = 5.$$

The total weighted tardiness time T has been decreased by 21% from 354.95 (h) to 277.50 (h). The number of late delivery orders N has been decreased from 31 to 5.

Conclusion

The FSS problem to be solved is modeled with the objective to minimize the total weighted tardiness of orders, and constraints on the sequence of orders, on the sequence of operations in the orders, and on machine changeover time.

The GATS algorithm is developed. In the algorithm, GA is used as the platform for global search, and TS is used to support GA in local search. GA diversifies the search by exploring the search space, and TS intensifies the search by exploiting the best solutions found. GA generates the elite population and generates genetic populations from the elite population by using GA operators. Then TS generates the neighborhood population from the genetic population by using neighborhood operators. Finally, GA generates the next population from the current population, the elite population, and the neighborhood population by using replacement operators.

Based on the model of the problem, the GATS algorithm is used to solve the pilot problem of small size, with 10 orders, scheduling on four machines. The total weighted tardiness time according to the algorithm (123.07 h) is better than the total weighted tardiness time according to the EDD heuristic currently used (215.95 h).

The algorithm is also used to solve the real FSS problem with 120 jobs on four machines. The results show that the algorithm gives better results than the current heuristic

EDD method. The total weighted tardiness time T has been decreased by 21% from 354.95 (h) to 277.50 (h). The number of late delivery orders N has been decreased from 31 to 5.

However, the factors of the model, including the population size P, the crossover probability P_c, the mutation probability P_m, the method of crossover, of mutation, the method of finding the neighborhood chromosomes, the method and parameter of the termination rule, etc., are only selected empirically, so the results are not very good. Future research is to use experimental design DOE to optimize the model factors to get suboptimal results.

Funding

The authors declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Etiler O, Toklu B, Atak M, Wilson J. A genetic algorithm for flow shop scheduling problems. *J Oper Res Soc.* (2004) 55:830–5. doi: 10.1057/palgrave.jors.2601766
2. Engin O, Ceran G, Yilmaz MK. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Appl Soft Comp.* (2011) 11:3056–65. doi: 10.1016/j.asoc.2010.12.006
3. Phong NN, Ngan K, Thi N, Huyen T, Thi TV. Application of genetic algorithm, GA to solve a flow shop scheduling problem with changeover times in operations: a case study. *BOHR Int J Oper Manag Res Pract.* (2024) 3(1):19–26. doi: 10.54646/bijomrp.2024.24
4. Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Comp Oper. Res.* (2007) 35:3202–12.
5. Gupta J, Palanimuthu N, Chen C-L. Designing a tabu search algorithm for the two-stage flow shop problem with secondary criterion. *Prod Plann Cont Manage Oper.* (1999) 10:251–65. doi: 10.1080/095372899233217
6. Phong NN, Nhi T, Thi N. Application of Tabu Search, TS to solve a flow shop scheduling problem with changeover times in operations. A case study. *BOHR Int J Oper Manag Res Pract.* (2024) 3(1):1–7. doi: 10.54646/bijomrp.2024.22
7. Burduk A, Musiał K, Kochanska J, Górnicka D, Stetsenko A. Tabu search and genetic algorithm for production process scheduling problem. *Sci J Logist.* (2019):181–9. doi: 10.17270/J.LOG.2019.315
8. Umam MS, Mustafid M, Suryono S. A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem. *J King Saud Univ. Comp Inform Sci.* (2022) 34:7459–67. doi: 10.1016/j.jksuci.2021.08.025
9. Ben-Daya M, Al-Fawzan M. A tabu search approach for the flow shop scheduling problem. *Eur J Oper Res.* (1997) 09:88–95.
10. Phong NN, Ngan K, Thi N. Application of GATS, a hybrid mega heuristic model, to solve flow shop scheduling problems FFS with changeover times in operations. A case study. *The 4th International Conference on Applied Convergence Engineering (ICACE 2023).* Ho Chi Minh City, Vietnam: Ho Chi Minh City University of Technology, VNU-HCM (2023).
11. Phong NN, Nhi T, Thi N. Application of TSGA, a hybrid mega heuristic model, to solve flow shop scheduling problems FFS with changeover times in operations. A case study. *The 4th International Conference on Applied Convergence Engineering (ICACE 2023).* Ho Chi Minh City, Vietnam: Ho Chi Minh City University of Technology, VNU-HCM (2023).
12. Phong NN, Trang T, Le N. Application of GATS, a hybrid mega heuristic model, and DOE, Design of Experiment, to solve flexible flow shop scheduling problems - A case study. *BOHR Int J Oper Manag Res Pract.* (2023) 2(1):28–35. doi: 10.54646/bijomrp.2023.14