# Multi-start Tabu Agents-Based Model for the Dual-Resource Constrained Flexible Job Shop Scheduling Problem

Farah Farjallah[1,4(✉)], Houssem Eddine Nouri[2,4], and Olfa Belkahla Driss[3,4]

[1] Ecole Nationale des Sciences de l'Informatique ENSI, Université de la Manouba, Manouba, Tunisia
farah.farjallah1@gmail.com
[2] Institut Supérieur de Gestion de Gabes, Université de Gabes, Gabes, Tunisia
[3] Ecole Supérieure de Commerce de Tunis, Université de la Manouba, Manouba, Tunisia
[4] LARIA La Recherche en Intelligence Artificielle, ENSI, Université de la Manouba, Manouba, Tunisia

**Abstract.** Typically, processing jobs on a production floor require both machines and human resources. However, most classical scheduling problems ignore possible constraints caused by the availability of workers and treat machines only as limited resource. This paper presents a Multi-Start Tabu Agents-based Model (MuSTAM) for Dual-Resource Constrained Flexible Job shop Scheduling Problem (DRCFJSP). It considers a set of initial solutions running in parallel using the intensification technique. It has a single objective which is to minimize the maximum completion time (makespan) due to its importance in research workshops. The proposed model consists of two classes of agents: MainAgent and TabuAgents. The MainAgent receives inputs, generates the initial population, creates TabuAgents based on the number of solutions in the initial population PopSize, launches the system and finally displays the best solution. Each TabuAgent takes a solution from the created initial population and applies Tabu Search using the technique of concentrated intensification to neighborhood search. TabuAgents cooperate and communicate between them in order to improve the search quality. In experimental phase, numerical tests are performed to evaluate our MuSTAM model compared to ITS based on FJSPW benchmark instances of Gong. The obtained results show the efficiency of the Multi-Start Tabu Agents-based Model in terms of makespan and CPU time.

**Keywords:** Flexible job shop · Single objective · Tabu search · Multi-agent · Workers flexibility · Dual-resource · Makespan

## 1 Introduction

Scheduling problems are present in all sectors of the economy, from manufacturing to IT. One of the most well-known production scheduling problems is the Job shop Scheduling Problem (JSP). It has several application fields such as airport scheduling, port scheduling, railway train scheduling. An important objective of this problem is how to find a sequence of operations which minimizes the maximum completion time


N. T. Nguyen et al. (Eds.): ICCCI 2022, LNAI 13501, pp. 674–686, 2022.
https://doi.org/10.1007/978-3-031-16014-1_53

(makespan) of the last one. In addition to the classical Job shop Scheduling Problem there are several generalizations [1] such as the Flexible Job shop Scheduling Problem (FJSP) proposed by Brucker and Schlie [2], which considers the flexibility of machines. The Flexible Job shop Scheduling Problem with Worker flexibility (FJSPW) according to Zheng and Wang [3] is an extension of the classical Job shop Scheduling, due to the importance of human factors in real world manufacturing systems which is commonly named as Dual-Resource Constrained Job shop Scheduling Problem (DRCJSP).

DRCFJSP was introduced by Nelson [4] and has three sub-problems: (i) machine selection, (ii) worker assignment, (iii) operation sequencing on the machines with workers constraint in order to minimize the end date of the last operation on all jobs which is known as makespan.

Figure 1 shows an example of a Dual-Resource Constrained Flexible Job shop Scheduling Problem. There are three jobs, six machines and five workers. Job1 is composed of three operations $O_{1,1}$, $O_{1,2}$ and $O_{1,3}$, job2 contains of two operations $O_{2,1}$ and $O_{2,2}$, Job3 is composed of two operations $O_{3,1}$ and $O_{3,2}$. $O_{1,1}$ can be executed by $M_1$ or $M_2$. $O_{1,2}$ can be processed on $M_2$ or $M_5$. $O_{1,3}$ can be operated on $M_4$ or $M_6$. $O_{2,1}$ may be executed by $M_3$ or $M_6$. $O_{2,2}$ can be processed on $M_4$ or $M_5$. $O_{3,1}$ can be operated on $M_1$ or $M_3$. $O_{3,2}$ can be executed by $M_2$ or $M_6$. The worker $W_1$ can operate on $M_2$ or $M_3$. $W_2$ can operate on $M_1$ or $M_2$. $W_3$ can operate on $M_1$ or $M_5$ or $M_6$. $W_4$ can operate on $M_3$ or $M_5$. $W_5$ can operate one of these machines: $M_4$, $M_5$ or $M_6$.

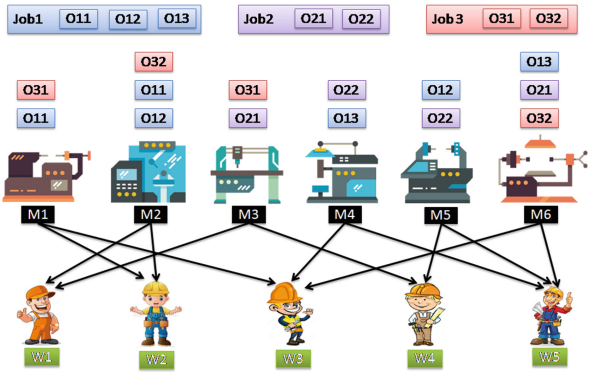


**Fig. 1.** Example of a practical dual-resource constrained flexible job shop scheduling problem

In this paper, we present a new multi-start tabu search algorithm based on a multi-agent model named MuSTAM for solving the Dual-Resource Constrained Flexible Job shop Scheduling Problem (DRCFJSP), where we detail the global process of the multi-agent system and then, each agent is described by its different operators.

The rest of the paper is organized as follows:

In Sect. 2, we present a state-of-the-art for solving the Dual-Resource Constrained Flexible Job shop Scheduling Problem (DRCFJSP), where we detail the different studies proposed for this problem and we propose a classification according to three criteria. In Sect. 3, we define the mathematical formulation of the DRCFJSP with its objective

function. Then, in Sect. 4, we detail the Multi-Start Tabu Agents-based Model. The experimental and comparison results are provided in Sect. 5. Finally, Sect. 6 ends the paper with a conclusion and future works.

## 2 State-of-the-Art

In this section, we present a study of the literature for solving DRCFJSP. Then, we classify this study according to three criteria: (1) Used method type (exact, metaheuristic), (2) Implemented method (tabu search, genetic algorithm, mixed integer programming,...) and (3) Optimization criteria (makespan, cost...) as shown in Table 1.

**Table 1.** Classification of the most recent papers studying the DRCFJSP

| Used method type | Implemented approaches | Optimization criteria | Authors |
|---|---|---|---|
| Exact | Mixed-integer linear programming | Profit, late orders and workforce | Da Silva et al. 2006 |
| | Integer programming model | Total cost | Wirojanagud et al. 2007 |
| | Variable neighborhood search | Makespan | Lei and Guo 2014 |
| Metaheuristic | Hybrid artificial bee colony algorithm | Makespan | Gong et al. 2020 |
| | Multiple populations for multiple objectives framework based genetic algorithm | Makespan, tardiness, total advance time, production cost | Liu et al. 2021 |
| | NSGA-II | Makespan, worker workload, tardiness | Vital-soto et al.2022 |

Da Silva et al. [5] created a mixed integer linear programming (MILP) model with multi objectives function consisting of maximize profit, minimize late orders, and minimize work force level changes. Wiriojangud et al. [6] developed a mixed integer programming model to determine the number of hires, firings, and cross-training at each General Cognitive Ability (GCA) level to minimize total costs, including training costs, salary costs, firing costs, and missed production costs over multiple time periods.

Lei and Guo [7] presented an effective variable neighbourhood search (VNS), where the solution is a quadruple string of the ordered operations and their resources. Two neighbourhood search procedures are executed to generate new solutions for two subproblems of the problem. Gong et al. [8] presented a Hybrid Artificial Bee Colony Algorithm (HABCA), were the goal was to minimize the makespan. A comparative study showed that the proposed algorithm was more effective than hybrid GA [9] and improved ABCA [10]. Vital-soto et al. [11] developed an elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) with an innovative operator to solve DRCFJSP where the multiobjective functions are (1) makespan, (2) worker workload, (3) tardiness. A novel Multiple Populations for Multiple Objectives (MPMO) framework-based Genetic Algorithm

(GA) approach (MPMOGA) was proposed by Liu et al. [12] to optimize five objectives simultaneously. Firstly, MPMOGA used five populations to optimize the five goals. Then, Secondly, an archive sharing technique was proposed to avoid that each population can be concentrated only on its corresponding mono objective. Therefore the populations can get optimization informations about the other goals according to the archive.

## 3 Mathematical Model of the DRCFJSP

### 3.1 Problem Description

The proposed DRCFJSP is illustrated by Gong et al. [8]: There are a set of $n$ jobs, a set of m machines, and a set of $l$ workers. Each job has a sequence of $r_i$ operations to be processed one after another according to the precedence constraint. Each operation must be processed by one selected worker from the worker set on one selected machine from the machine set. The processing time of each operation, which is operated by one worker on a selected machine, is fixed. Table 2 shows an example of DRCFJSP as a demonstration case. There are two jobs, three machines, and three workers. Job 1 and Job 2 have two operations. The jobs are listed in column 1. Each set of job's operations is listed in column 2. The machines that process the appropriate operations are listed in column 3. Column 4 shows the appropriate worker for each machine operations. Finally, the processing time of the operations performed by the corresponding workers is shown in column 5.

**Table 2.** Sample instance of the DRCFJSP

| Jobs | Operations | Machines | Workers | Time |
|---|---|---|---|---|
| J1 | $O_{1,1}$ | M1 | W1 | 10 |
| | | | W3 | 20 |
| | | M2 | W2 | 10 |
| | $O_{1,2}$ | M2 | W1 | 10 |
| | | | W2 | 15 |
| J2 | $O_{2,1}$ | M2 | W1 | 20 |
| | | | W2 | 10 |
| | | M3 | W3 | 15 |
| | $O_{2,2}$ | M1 | W1 | 15 |
| | | M2 | W3 | 20 |
| | | | W1 | 10 |

The objective of this paper is to minimize makespan. Some assumptions are considered:

- Each machine can process only one selected operation from its corresponding operation set at any given time.
- Each worker can operate on only one machine at a time, selected from the appropriate machine set.

- Each operation can only be processed once by one machine from the corresponding machine set, and must satisfy the operation constraints of all jobs.
- Each operation can be operated only once by a worker chosen from the corresponding worker set and operation constraints of all jobs should be satisfied.
- There are no precedence constraints among the operations of different jobs.
- preemption is not permitted.
- An operation of any job cannot be processed until its preceding operations are completed.
- The processing time is known in advance.

### 3.2 Problem Formulation

The concepts used in this article are as follow:

| Indices | |
|---|---|
| $i, h$ | Index of jobs, $i$,$h$= 1, 2, . . .,$n$ |
| $j, g$ | Index of operations, $j$,$g$=1,2,... $r_i$ |
| $k$ | Index of machines, $k$ = 1, 2,. . . , $m$ |
| $s$ | Index of worker, $s$ = 1, 2,. . . , $l$ |

| Parameters | |
|---|---|
| $n$ | Total number of job |
| $m$ | Total number of machine |
| $l$ | Total number of workers |
| $r_i$ | Total number of operations in job i |
| $O_{i,j}$ | The *jth* operation of job i |
| $T_{i,j,k,s}$ | Processing time of $O_{ij}$ by worker $s$ on machine $k$ |
| $CO_{i,j}$ | Completion time of operation $O_{i,j}$ |
| $C_i$ | Completion time of job $i$ |

Decision variables

$$X_{ijks} = \begin{cases} 1 \text{ if worker s is selected to operate on machine k for operation } \quad O_{i,j} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

The mathematical model used in this article is as follows:
The objective function:

$$minf = max_{1 \le i \le n} \left(C_i\right) \tag{1}$$

**Subject to :**

$$CO_{ij} - CO_{ij-1} \geq T_{ijks} X_{ijks} \quad \forall i = 1,...,n; j = 2,...,r_i; k = 1,...,m; s = 1,...l \tag{2}$$

$$(CO_{hg} - CO_{ij} - T_{hgks}) X_{ijks} X_{hgks} \geq 0 \vee (CO_{ij} - CO_{hg} - T_{ijks}) X_{ijks} X_{hgks} \geq 0$$
$$\forall i = 1,...,n; j = 1,...,r_i; k = 1,...,m; s = 1,...,l \tag{3}$$

$$\sum_{j=1}^{r_i} X_{ijks} = 1 \quad \forall i = 1,...,n; k = 1,...,m; s = 1,...,l \tag{4}$$

$$\sum_{k=1}^{m} X_{ijks} = 1 \quad \forall i = 1,...,n; j = 1,...,ri; s = 1,...,l \tag{5}$$

$$\sum_{s=1}^{l} X_{ijks} = 1 \quad \forall i = 1,...,n; j = 1,...,ri; k = 1,...,m \tag{6}$$

Equation (1) for the objective function, consists on minimizing the makespan. Constraint (2) to ensure precedence constraints, constraint (3) to ensure that each machine can handle only one operation at any time, constraint (4) to ensure that an operation chosen from the operation set can only be handled once, the constraint (5) to ensure that only one machine is chosen from a corresponding machine set to process each operation. Constraint (6) to ensure that each operation can be executed by one and only one worker.

## 4 Multi-start Tabu Agents-Based Model (MuSTAM)

### 4.1 Tabu Search

Tabu Search is a metaheuristic introduced by Glover [13], it is a local search method combined with a set of techniques to avoid being trapped in a local minimum or recycling. Tabu Search (TS) has shown great efficiency for solving NP-hard problems. Recently this method is still used in the literature which explains its importance and performance in the field of research. Hajibabaei and Benhnamain [14] introduced Tabu Search (TS) algorithm for solving Flexible Job shop Scheduling Problem (FJSP) with unrelated parallel machines and sequence-dependent setup time. The results show that it performs better than Genetic Algorithm (GA) where the goal is to minimizing the costs of makespan, total weighted tardiness, delivery time and inventory.

### 4.2 The Basic Principle of MuSTAM

A multi-agent system composed of multiple interacting agents that cooperate, communicate and coordinate with each other to achieve a common goal [15]. We propose a Multi-Start Tabu Agents-based Model for solving Dual-Resource Constrained Flexible Job shop Scheduling Problems. It consists of two classes of agents: MainAgent and a set of PopSize TabuAgents, where PopSize is the number of solutions initially generated based on a neighborhood parameter, see Fig. 2.

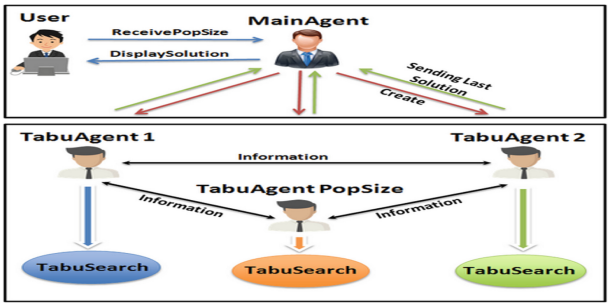


**Fig. 2.** Multi-start tabu agents-based model

### 4.3 MainAgent Process

It is the one that reacts with the user to receive the maximum number of iterations for the search operation. It is responsible for creating TabuAgents based on the size of the initial population PopSize and then provides for each TabuAgent its necessary information: agent identification (as an autonomous agent and also as a system member), its initial solution and the maximum number of instructions for the search. The MainAgent is satisfied when the stopping criterion is reached. In this case, it sorts the found solutions in terms of the makespan values and displays the best obtained solution to the user.

### 4.4 Initial Population

To make the individual solutions more diversified and distributed in the search space, the starting population is created at random using a uniform law based on a neighborhood parameter. In fact, to be considered as a new member of the initial solutions, each new solution should have a predetermined distance from all existing solutions. The used method based on neighborhood parameter was inspired from [16]. Equation 7 presents the formula for total distance of dissimilarity between two solutions.

$$Dist_{max} = [\sum_{i=1}^{n} \sum_{i,1}^{i,ni} M(O_{i,j})] + L \tag{7}$$

where the number of alternative machines for each operation is M($O_{i,j}$). The total number of operations for all tasks is $L$. The distance between them is $Dist$ and, $n_i$ is a total number of operations to be performed successively according to the given sequence for each sequence of job $J_i$. In-addition, for each case, we select a solution (i) and we verify consecutively for each next solution (j) from the remaining population, if the dissimilarity distance between solution (i) and solution (j) is less than or equal to a fixed threshold $Dist_{fix}$ (representing a percentage of difference X% relatively to $Dist_{max}$, look Eq. (8)) which is illustrated from [17]. In our case we choose the fair percentage 50%.

$$Dist_{fix} = Dist_{max} \times X\% \tag{8}$$

### 4.5 Representation of Solutions

The solutions are represented in three vectors, the first vector $O$ for the operations, the second $M$ for the machines and the third $W$ for the workers and final vector that combines the three vectors together. Figure 3 shows an example of solution representation.

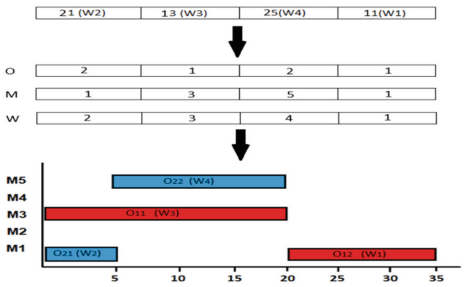


**Fig. 3.** Solution representation

The final vector explanation is as follows: in the first $r$ vector, all operations are numbered as a series of consecutive integers. In this example, job 1 has operations numbered 1 and 2, and job 2 has operations numbered 1 and 2. Therefore, the indices of the vector ([2, 4, 1, 3]) represent two jobs [($O_{11}$, $O_{12}$, $O_{21}$, $O_{22}$)]. In the second vector, each machine selected to process the corresponding operation is numbered as an integer according to its order in the predefined set of machines. In this example, $O_{11}$, $O_{12}$, $O_{21}$ and $O_{22}$ are handled by machines $M_3$, $M_1$, $M_1$ and $M_5$. In the third vector, each worker selected to operate the corresponding operation is numbered as an integer according to its sequence in a predefined worker set. In this example, $O_{11}$, $O_{12}$, $O_{21}$, and $O_{22}$ are processed by workers $W_3$, $W_1$, $W_2$, and $W_4$.

### 4.6 TabuAgents Optimization Process

The TabuAgents are created by the MainAgent each one represents a solution. All TabuAgents simultaneously start their process by performing a Tabu Search algorithm using the technique of concentrated intensification to neighborhood search to guarantee the multi-start mechanism of our model. The cooperation and communication of TabuAgents manage the sending and receiving of messages between them containing their necessary information in order to improve the search quality. The TabuAgents complete their process by sending theirs last solutions to the MainAgent, which consider the best of them as the global solution for the DRCFJSP.

### 4.7 Neighborhood Parameter

This parameter starts from an initial feasible solution s of the initial solutions S, to build a set of neighboring solutions of this solution, at each iteration it completely examines

the neighborhood V(s) of the current solution s. Then it chooses the best solution of V(s)-s, even if it does not improve the value of the objective function. There are three possible moves on permutations as follows:

1. **Inversion:** the position of two neighboring is reversed. It gives us *n-1* possible moves. Figure 4 shows an example of permutation.


**Fig. 4.** Example of inversion

2. **Transposition:** the position of two non-neighboring is changed. This movement is more efficient than displacement for some problem like the quadratic assignment problem because it only changes the position of the transposed objects. It gives us *n(n-1)/2* possible moves as shown in Fig. 5.


**Fig. 5.** Example of transposition

3. **Shifting:** we change the position of a single box. This movement is better for sequencing problems because it is the sequencing that is important, not the individual position of the tasks. It gives us (n-1)2 possible moves. Figure 6 presents this function


**Fig. 6.** Example of shifting

Since the permutation movement is on three levels i.e. if there is a change in the O vector there are also changes in the M and W vectors the ITS takes the transposition for the permutation movement for decrease the number of changes.

### 4.8 Neighborhood Evaluation

After the generation of all neighbor solutions, the neighbor evaluation step is started. This stage takes place over two stages. The first step consists in choosing the best non-tabu neighbors among the set of neighboring solutions, this neighbor will be the current solution. The second step compares the value of makespan for the current solution with the makespan of the best solution, if it is lower, it will be the best solution.

### 4.9 Tabu List

An essential element of Tabu Search is the use of flexible short-term memory, which can keep track of some recent past actions. The size of the tabu list used is static "TL". If the size of the list exceeds the maximum allowed size, we remove the oldest item from this list (FIFO strategy: first in, first out: the element that arrives first is the first to be removed).

### 4.10 Stopping Criterion

In our model, we adopt the maximum number of iterations as the stopping criterion. The Tabu Search process tries to improve the solution after a maximum number of iterations (maxiter = 10000). Then, the best schedule found during the search and its makespan time are returned.

## 5 Experimental Results

To evaluate our model and to show the importance of using the multi-agent systems, we have developed an enhanced Tabu Search method named Improved Tabu Serach (ITS), which is based on an initial population of solutions. The Improved Tabu Search (ITS) and Multi-Start Tabu Agents-based Model (MuSTAM) are implemented in Java on PC with 2,50 GHz intel (core i5 vPro) and 12 GB of RAM memory, using Eclipse IDE to code these two approaches and the Jade platform to create the MuSTAM multi-agent system.

To compare and evaluate the efficiency of these approaches, numerical tests are provided in Table 3 showing results of different problem instances of Gong data [8], from the literature of the FJSPW. It consists of 13 problems (FGW01-FGW13) with a number of operations ranging from 2 to 15 for all jobs which will be processed on a number of machines ranging from 5 to 15 by a number of workers ranging from 5 to 15. FGW01-03 are smaller, while FGW04-13 are large instances.

DRCFJSP can be divided into total worker flexibility and partial worker flexibility. In this work, we consider the total worker flexibility case, which means that each machine can be operated by all workers.

The comparison experimental results are shown in Table 3, in which PopSize means the population size of initial solutions, $C_{max}$ is the best makespan among 5 runs, Avg is the average value from the 5 runs, and CPU time is the best computation time (in seconds) for each best makespan. Best makespan and CPU values for each case of the benchmark instances are marked in bold.

The obtained results show that MuSTAM provides good results in terms of $C_{max}$ and CPU time. For all used instances, we find the best CPU time values. Figure 7 ensures this best results for PopSize=100. While for the comparisons between the two approaches in terms of Avg makespan ($C_{max}$), we find for population size 25 that MuSTAM is 69% of the instances, better than ITS. In addition, for population size 50 MuSTAM is 53,8% of the instances, better than ITS and for population size 100 MuSTAM is 92,3% of the instances, better than ITS.

**Table 3.** Results of the Gong instances

| | ITS | | | | | | | | | MuSTAM | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cmax | | | | | | CPU(s) | | | Cmax | | | | | | CPU(s) | | |
| | Best | | | Avg | | | Best | | | Best | | | Avg | | | Best | | |
| Instances | PopSize | | | | | | | | | | | | | | | | | |
| | 25 | 50 | 100 | 25 | 50 | 100 | 25 | 50 | 100 | 25 | 50 | 100 | 25 | 50 | 100 | 25 | 50 | 100 |
| FGW01 | 15 | **13** | 13 | 15,2 | 14 | 14,2 | 96 | 183 | 362 | **12** | **13** | **12** | 14,4 | 14,4 | 13 | **7** | **9** | **12** |
| FGW02 | 47 | 46 | 41 | 49,2 | 47,2 | 45,25 | 66 | 211 | 463 | **45** | **44** | **39** | 48,8 | 47,3 | 44 | **13** | **18** | **18** |
| FGW03 | **36** | 36 | 36 | 28,2 | 38,25 | 35,75 | 123 | 265 | 1993 | **36** | **34** | **31** | 38,6 | 37,6 | 36,2 | **8** | **10** | **451** |
| FGW04 | **77** | **74** | 74 | 80,8 | 78,14 | 75,83 | 253 | 290 | 607 | **77** | 76 | **72** | 79,8 | 77,8 | 77,2 | **14** | **23** | **39** |
| FGW05 | 81 | 79 | 74 | 82 | 80,5 | 76,25 | 245 | 346 | 625 | **73** | **70** | **72** | 75,4 | 74 | 74,6 | **13** | **19** | **35** |
| FGW06 | 499 | 485 | 485 | 508 | 492 | 491,5 | 680 | 1260 | 2521 | **485** | **478** | **478** | 497,2 | 495,2 | 496,6 | **65** | **113** | **214** |
| FGW07 | 135 | 134 | 134 | 138,5 | 135,25 | 137,25 | 332 | 627 | 1311 | **128** | **129** | **128** | 138,75 | 132,66 | 134,75 | **23** | **38** | **35** |
| FGW08 | **326** | **319** | 319 | 335,5 | 324 | 323,25 | 540 | 830 | 1580 | 327 | 321 | **302** | 331 | 329 | 316,25 | **26** | **51** | **100** |
| FGW09 | **224** | 224 | 211 | 236 | 234,25 | 226,5 | 718 | 1093 | 3916 | 234 | **216** | **210** | 240 | 226,25 | 225,5 | **48** | **105** | **212** |
| FGW10 | 389 | **388** | 388 | 398 | 397 | 393 | 415 | 742 | 1499 | **379** | 399 | **376** | 390,33 | 405,75 | 380,5 | **23** | **45** | **84** |
| FGW11 | **873** | **852** | 852 | 896 | 876 | 859,3 | 1051 | 2279 | 4885 | 890 | 857 | **835** | 906 | 871,25 | 852 | **104** | **214** | **425** |
| FGW12 | 809 | **799** | **781** | 829,6 | 818,4 | 811,8 | 923 | 1829 | 3700 | **771** | 818 | 796 | 800,5 | 837 | 810,25 | **118** | **277** | **554** |
| FGW13 | **614** | **614** | 614 | 656,5 | 642 | 628 | 945 | 2049 | 4103 | 655 | 652 | **612** | 680 | 657,5 | 633 | **124** | **310** | **485** |
| | | | AVG | 4253,5 | 4176,99 | 4117,88 | | | | | | AVG | 4240,78 | 4205,71 | 4093,85 | | | |

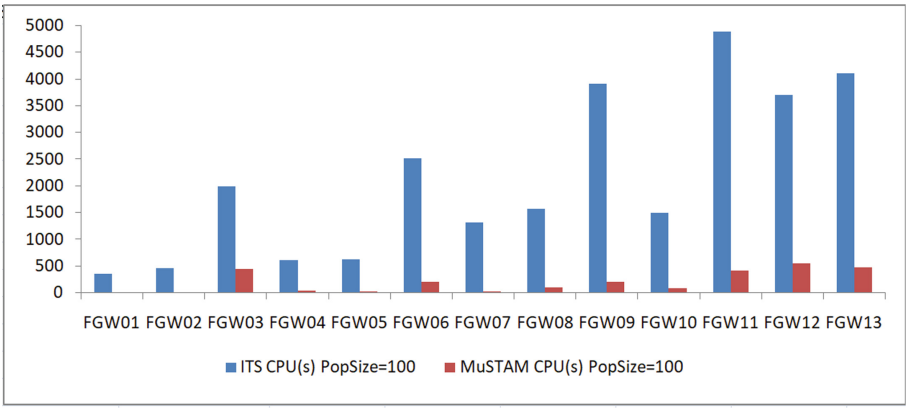


**Fig. 7.** Comparison results in terms of CPU time

These best results guarantee the usage of a multi-agent system, which comprises first and foremost of reducing execution time and enhancing the quality of neighborhood search, due to the cooperation and communication between TabuAgents.

## 6 Conclusion and Future Researches Directions

In this paper, we have proposed a new Multi-Start Tabu Agents-based Model (MuSTAM) for the DRCFJSP with makespan objective. It is composed of two types of agents, which was the MainAgent and a set of TabuAgents, which cooperate and communicate between to improve the quality of neighborhood search to find the best solution. To determine its performance, numerical tests was created using a well-known data set from literature of the DRCFJSP. The experimental results showed that our MuSTAM obtained the best set of solutions in comparison to ITS in terms of makespan and CPU time.

In the future works, we will be able to adapt our work to a multi-objective case of the solved problem. Thereby, we will compare the generated results by our work with other approaches from the literature. Also, it will be possible to add new constraints such as workers age which will present the real effect of the human energy on the production process.

## References

1. Dhiflaoui, M., Nouri, H.E., Driss, O.B.: Dual-resource constraints in classical and flexible job shop problems: a state-of-the-art review. Procedia Comput. Sci. **126**, 1507–1515 (2018)
2. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990)
3. Zheng, X.L., Wang, L.: A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem. Int. J. Product. Res. **54**(18), 5554–5566 (2016)

4. Nelson, R.T.: Labor and machine limited production systems. Manag. Sci. **13**(9), 648–671 (1967)
5. da Silva, C.G., Figueira, J., Lisboa, J., Barman, S.: An interactive decision support system for an aggregate production planning model based on multiple criteria mixed integer linear programming. Omega **34**(2), 167–177 (2006)
6. Wirojanagud, P., Gel, E.S., Fowler, J.W., Cardy, R.: Modelling inherent worker differences for workforce planning. Int. J. Product. Res. **45**(3), 525–553 (2007)
7. Lei, D., Guo, X.: Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. Int. J. Product. Res. **52**(9), 2519–2529 (2014)
8. Gong, G., Chiong, R., Deng, Q., Gong, X.: A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility. Int. J. Product. Res. **58**(14), 4406–4420 (2020)
9. Gao, J., Gen, M., Sun, L.Y.: Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. J. Intell. Manuf. **17**(4), 493–507 (2006)
10. Wang, L., Zhou, G., Xu, Y., Wang, S., Liu, M.: An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. Int. J. Adv. Manuf. Technol. **60**(1–4), 303–315 (2012)
11. Vital-Soto, A., Baki, M.F., Azab, A.: A multi-objective mathematical model and evolutionary algorithm for the dual-resource flexible job-shop scheduling problem with sequencing flexibility. Flex. Serv. Manuf. J., 1–43 (2022)
12. Liu, S.C., Chen, Z.G., Zhan, Z.H., Jeon, S.W., Kwong, S., Zhang, J.: Many-objective job-shop scheduling: a multiple populations for multiple objectives-based genetic algorithm approach. IEEE Trans. Cybern. (2021)
13. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
14. Hajibabaei, M., Behnamian, J.: Flexible job-shop scheduling problem with unrelated parallel machines and resources-dependent processing times: a tabu search algorithm. Int. J. Manag. Sci. Eng. Manag. **16**(4), 242–253 (2021)
15. Xiong, W., Fu, D.: A new immune multi-agent system for the flexible job shop scheduling problem. J. Intell. Manuf. **29**(4), 857–873 (2015). https://doi.org/10.1007/s10845-015-1137-2
16. Bożejko, W., Uchroński, M., Wodecki, M.: The new golf neighborhood for the flexible job shop problem. Procedia Comput. Sci. **1**(1), 289–296 (2010)
17. Nouri, H.E., Driss, O.B., Ghédira, K.: A holonic multiagent model based on a combined genetic algorithm–tabu search for the flexible job shop scheduling problem. In: Bajo, J., et al. (eds.) PAAMS 2015. CCIS, vol. 524, pp. 43–54. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19033-4_4