Metaheuristics based on Clustering in a Holonic Multiagent Model for the Flexible Job Shop Problem

Houssem Eddine Nouri
Stratégies d'Optimisation et Informatique intelligentE (SOIE)
Higher Institute of Management of Tunis, Bardo, Tunis, Tunisia
houssemeddine.nouri@gmail.com

Olfa Belkahla Driss
Stratégies d'Optimisation et Informatique intelligentE (SOIE)
Higher Institute of Management of Tunis, Bardo, Tunis, Tunisia
olfa.belkahla@isg.rnu.tn

Khaled Ghédira
Stratégies d'Optimisation et Informatique intelligentE (SOIE)
Higher Institute of Management of Tunis, Bardo, Tunis, Tunisia
khaled.ghedira@isg.rnu.tn

ABSTRACT
The Flexible Job Shop scheduling Problem (FJSP) is a generalization of the classical Job Shop scheduling Problem (JSP) allowing to process operations on one machine out of a set of alternative machines. The FJSP is an NP-hard problem consisting of two sub-problems, which are the machine assignment and the operation scheduling problems. In this paper, we propose how to solve the FJSP by metaheuristics based on clustering in a holonic multiagent model. Firstly, a Neighborhood-based Genetic Algorithm (NGA) is applied by a scheduler agent for a global exploration of the search space. Secondly, a local search technique is used by a set of cluster agents to guide the research in promising regions of the search space and to improve the quality of the NGA final population. To evaluate our approach, numerical tests are made based on three sets of well known benchmark instances from the literature of the FJSP, which are Kacem, Brandimarte, Hurink. The experimental results show the efficiency of our approach in comparison to other approaches.

Categories and Subject Descriptors
I.2.8 [Problem Solving, Control Methods, and Search]: Scheduling; **I.2.11 [Distributed Artificial Intelligence]:** Multiagent systems

General Terms
Algorithms, Management

Keywords
Scheduling, Flexible job shop, Genetic algorithm, Local search, Clustering, Holonic multiagent

1. INTRODUCTION
The Flexible Job Shop scheduling Problem (FJSP) is a generalization of the classical Job Shop scheduling Problem (JSP) that allows to process operations on one machine out of a set of alternative machines. Hence, the FJSP is more computationally difficult than the JSP. Furthermore the operation scheduling problem, the FJSP presents an additional difficulty caused by the operation assignment problem to a set of available machines. This problem is known to be strongly NP-Hard even if each job has at most three operations and there are two machines [9].

To solve this problem, standard metaheuristic methods are used for an approximate resolution and to find near-optimal solutions for the FJSP with acceptable computational time. Brandimarte [4] proposed a hierarchical algorithm based on Tabu Search metaheuristic for routing and scheduling with some known dispatching rules to solve the FJSP. In [14] a Tabu Search procedure is developed for the job shop problem with multi-purpose machines. For [21], they used Tabu Search techniques and presented two neighborhood functions allowing an approximate resolution for the FJSP. Also, a Tabu Search approach based on a new golf neighborhood for the FJSP is presented by [2], and in the same year, in [3] another new model of a distributed Tabu Search algorithm is proposed for the FJSP, using a cluster architecture consisting of nodes equipped with the GPU units (multi-GPU) with distributed memory. For the Genetic Algorithm, it was adopted by [16] with an approach of localization to solve jointly the assignment and job shop scheduling problems with partial and total flexibility, and a second hybridization of this evolutionary algorithm with the fuzzy logic was presented in [17]. In [13] a new architecture is developed named LEarnable Genetic Architecture (LEGA) for learning and evolving solutions for the FJSP, allowing to provide an integration between evolution and learning in an efficient manner within a random search process. In addition, [8] adapted a hybrid Genetic Algorithm (G.A) and a Variable Neighborhood Descent (V.N.D) for FJSP. The G.A used two vectors to represent a solution and the disjunctive graph to calculate it. Then, a V.N.D was applied to improve the G.A final individuals. Recently, [22] presented a model of low-carbon scheduling in the FJSP considering three factors, the makespan, the machine workload for production and the carbon emission for the environmental influence. A metaheuristic hybridization algorithm was proposed combining the original Non-dominated Sorting Genetic Algorithm II (NSGA-II) with a Local Search algorithm based on a neighborhood search technique. Moreover, a new heuristic was developed by [23] for the FJSP. This heuristic is based on a constructive procedure considering simultaneously many factors having a great effect on the solution quality. Furthermore, distributed artificial intelligence techniques were used for this problem, such as the multiagent model proposed by [6] composed by three classes of agents, job agents, resource agents and an interface agent. This model is based on a local search method which is the tabu search to solve the FJSP. Also, [12] proposed a

GECCO'15 Companion, July 11 - 15, 2015, Madrid, Spain.
 ISBN 978-1-4503-3488-4/15/07…$15.00.
DOI: http://dx.doi.org/10.1145/2739482.2768449

multiagent model based on a hybridization of two metaheuristics, a local optimization process using the tabu search to get a good exploitation of the good areas and a global optimization process integrating the Particle Swarm Optimization (PSO) to diversify the search towards unexplored areas.

In this paper, we present how to solve the the flexible job shop scheduling problem by metaheuristics based on clustering in a holonic multiagent model. This new approach follows two principal steps. In the first step, a genetic algorithm is applied by a scheduler agent for a global exploration of the search space. Then, in the second step, a local search technique is used by a set of cluster agents to improve the quality of the final population. Numerical tests were made to evaluate the performance of our approach based on three data sets of [17], [4] and [14] for the FJSP, where the experimental results show its efficiency in comparison to other approaches.

The rest of the paper is organized as follows. In section 2, we define the formulation of the FJSP with its objective function and a simple problem instance. Then, in section 3, we detail the proposed metaheuristics based on clustering with their holonic multiagent levels. The experimental and comparison results are provided in section 4. Finally, section 5 rounds up the paper with a conclusion.

2. PROBLEM FORMULATION

The flexible job shop scheduling problem (FJSP) could be formulated as follows. There is a set of n jobs $J = \{J_1, \ldots, J_n\}$ to be processed on a set of m machines $M = \{M_1, \ldots, M_m\}$. Each job J_i is formed by a sequence of n_i operations $\{O_{i,1}, O_{i,2}, \ldots, O_{i,ni}\}$ to be performed successively according to the given sequence. For each operation $O_{i,j}$, there is a set of alternative machines $M(O_{i,j})$ capable of performing it. The main objective of this problem is to find a schedule minimizing the end date of the last operation of the jobs set which is the makespan. The makespan is defined by $Cmax$ in Equation 1, where C_i is the completion time of a job J_i.

$$Cmax = max_{1 \le i \le n} (C_i) \quad (1)$$

To explain the FJSP, a sample problem of three jobs and five machines is shown in Table 1, where the numbers present the processing times and the tags "–" mean that the operation cannot be executed on the corresponding machine.

Table 1: A simple instance of the FJSP

Job	Operation	M1	M2	M3	M4	M5
J1	O_{11}	2	9	4	5	1
	O_{12}	-	6	-	4	-
J2	O_{21}	1	-	5	-	6
	O_{22}	3	8	6	-	-
	O_{23}	-	5	9	3	9
J3	O_{31}	-	6	6	-	-
	O_{32}	3	-	-	5	4

3. A METAHEURISTIC HYBRIDIZATION IN A HOLONIC MULTIAGENT MODEL

Glover [11] elaborated a study about the nature of connections between the genetic algorithm and tabu search metaheuristics, searching to show the existing opportunities for creating a hybrid approach with these two standard methods to take advantage of their complementary features and to solve difficult optimization problems. After this pertinent study, the combination of these two metaheuristics has become more well-known in the literature, which has motivated many researchers to try the hybridization of these two methods for the resolution of different complex problems in several areas.

Ferber [7] defined a multiagent system as an artificial system composed of a population of autonomous agents, which cooperate with each other to reach common objectives, while simultaneously each agent pursues individual objectives. Furthermore, a multiagent system is a computational system where two or more agents interact (cooperate or compete, or a combination of them) to achieve some individual or collective goals. The achievement of these goals is beyond the individual capabilities and individual knowledge of each agent [1].

Koestler [18] gave the first definition of the term "holon" in the literature, by combining the two Greek words "hol" meaning whole and "on" meaning particle or part. He said that almost everything is both a whole and a part at the same time. In fact, a holon is recursively decomposed at a lower granularity level into a community of other holons to produce a holarchy [5]. Moreover, a holon may be viewed as a sort of recursive agent, which is a super-agent composed by a sub-agents set, where each sub-agent has its own behavior as a complementary part of the whole behaviour of the super-agent. Holons are agents able to show an architectural recursiveness [10].

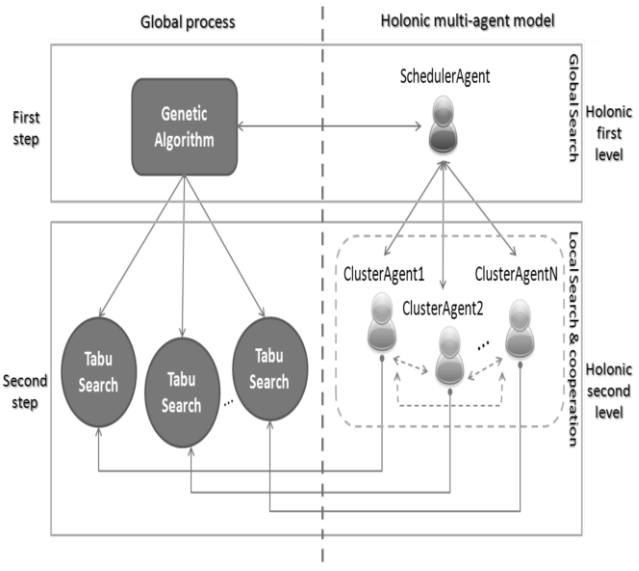

Figure 1. A metaheuristic hybridization in a holonic multiagent model

In this work, we propose a hybrid metaheuristic approach based on clustering processing two general steps: a first step of global exploration using a genetic algorithm to find promising areas in the search space and a clustering operator allowing to regroup them in a set of clusters. In the second step, a tabu search algorithm is applied to find the best individual solution for each

cluster. The global process of the proposed approach is implemented in two hierarchical holonic levels adopted by a recursive multiagent model, named a hybrid Genetic Algorithm with Tabu Search based on clustering in a Holonic Multiagent model (GATS+HM), see Figure 1. The first holonic level is composed by a Scheduler Agent which is the Master/Super-agent, preparing the best promising regions of the search space, and the second holonic level containing a set of Cluster Agents which are the Workers/Sub-agents, guiding the search to the global optimum solution of the problem. Each holonic level of this model is responsible to process a step of the hybrid metaheuristic algorithm and to cooperate between them to attain the global solution of the problem.

In fact, the choice of this new metaheuristic hybridization is justified by that the standard metaheuristic methods use generally the diversification techniques to generate and to improve many different solutions distributed in the search space, or by using local search techniques to generate a more improved set of neighbourhood solutions from an initial solution. But they did not guarantee to attain promising areas with good fitness converging to the global optimum despite the repetition of many iterations, that is why they need to be more optimized. So, the novelty of our approach is to launch a genetic algorithm based on a diversification technique to only explore the search space and to select the best promising regions by the clustering operator. Then, applying the intensification technique of the tabu search allowing to relaunch the search from an elite solution of each cluster autonomously to attain more dominant solutions of the search space.

The use of a multiagent system gives the opportunity for distributed and parallel treatments which are very complimentary for the second step of the proposed approach. Indeed, our combined metaheuristic approach follows the paradigm of "Master" and "Workers" which are two recursive hierarchical levels adaptable for a holonic multiagent model, where the Scheduler Agent is the Master/Super-agent of its society and the Cluster Agents are its Workers/Sub-agents.

3.1 Scheduler Agent

The Scheduler Agent (SA) is responsible to process the first step of the hybrid algorithm by using a genetic algorithm called NGA (Neighborhood-based Genetic Algorithm) to identify areas with high average fitness in the search space during a fixed number of iterations *MaxIter*. In fact, the goal of using the NGA is only to explore the search space, but not to find the global solution of the problem. Then, a clustering operator is integrated to divide the best identified areas by the NGA in the search space to different parts where each part is a cluster $CL_i \in CL$ the set of clusters, where $CL = \{CL_1, CL_2, \ldots, CL_N\}$. In addition, this agent plays the role of an interface between the user and the system (initial parameter inputs and final result outputs). According to the number of clusters N obtained after the integration of the clustering operator, the SA creates N Cluster Agents (CAs) preparing the passage to the next step of the global algorithm. After that, the SA remains in a waiting state until the reception of the best solutions found by the CA for each cluster. Finally, it finishes the process by displaying the final solution of the problem.

3.1.1 Individual's solution presentation

The flexible job shop problem is composed by two sub-problems: the machine assignment problem and the operation scheduling problem, that is why the chromosome representation is encoded in two parts: Machine Assignment part (MA) and Operation Sequence part (OS). The first part MA is a vector V_1 with a length L equal to the total number of operation and where each index represents the selected machine to process an operation indicated at position p, see Figure 2 (a). For example $p = 2$, $V_1(2)$ is the selected machine M_4 for the operation $O_{1,2}$. The second part OS is a vector V_2 having the same length of V_1 and where each index represents an operation $O_{i,j}$ according to the predefined operations of the job set, see Figure 2 (b). For example the operation sequence 1−2−1−3−2−3−2 can be translated to: $(O_{1,1},M_5) \rightarrow (O_{2,1},M_1) \rightarrow (O_{1,2},M_4) \rightarrow (O_{3,1},M_3) \rightarrow (O_{2,2},M_3) \rightarrow (O_{3,2},M_1) \rightarrow (O_{2,3},M_2)$.

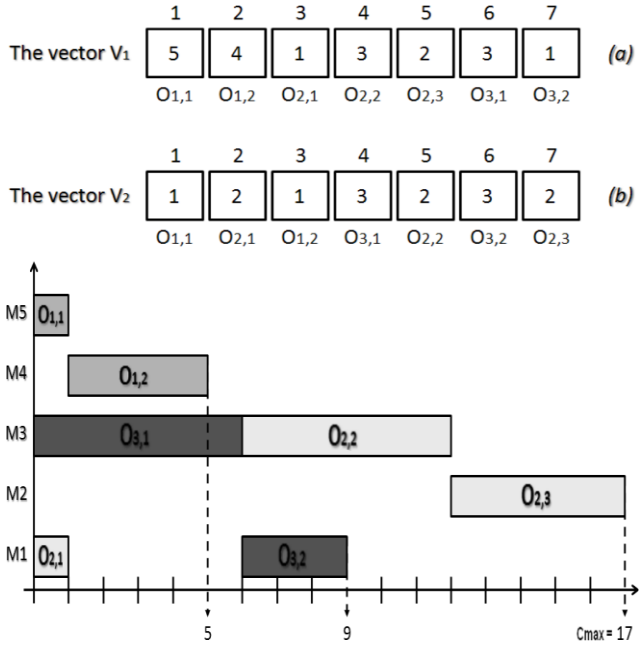

Figure 2. The chromosome representation of a scheduling solution

To convert the chromosome values to an active schedule, we used the priority-based decoding of [8]. This method considers the idle time which may exist between operations on a machine m, and which is caused by the precedence constraints of operations belonging to the same job i. Let $S_{i,j}$ is the starting time of an operation $O_{i,j}$ (which can only be started after processing its precedent operation $O_{i,(j-1)}$) with its completion time $C_{i,j}$. In addition, we have an execution time interval $[t^S m, t^E m]$ starts form $t^S m$ and ends at $t^E m$ on a machine m to allocate an operation $O_{i,j}$. So, if $j = 1$, $S_{i,j}$ takes $t^S m$, else if $j \geq 2$, it takes $max\{t^S m, C_{i,(j-1)}\}$. In fact, the availability of the time interval $[t^S m, t^E m]$ for an operation $O_{i,j}$ is validated by verifying if there is a sufficient time period to complete the execution time p_{ijm} of this operation, see Equation 2:

$$if\ j=1,\ t^S_m + p_{ijm} \leq t^E_m \qquad (2)$$

$$if\ j \geq 2,\ max\{t^S_m,\ C_{i,(j-1)}\} + p_{ijm} \leq t^E_m$$

The used priority-based decoding method allows in each case to assign each operation to its reserved machine following the presented execution order of the operation sequence vector V_2. Also, to schedule an operation $O_{i,j}$ on a machine m, the fixed idle time intervals of the selected machine are verified to find an

allowed available period to its execution. So, if a period is found, the operation $O_{i,j}$ is executed there, else it is moved to be executed at the end of the machine m.

Noting that the chromosome fitness is calculated by $Fitness(i)$ which is the fitness function of each chromosome i and $Cmax(i)$ is its makespan value, where $i \in \{1, \ldots, P\}$ and P is the total population size, see Equation 3.

$$Fitness(i) = \frac{1}{Cmax\,(i)} \tag{3}$$

3.1.2 Population initialization

The initial population is generated randomly following a uniform law and based on a neighborhood parameter to make the individual solutions more diversified and distributed in the search space. In fact, each new solution should have a predefined distance with all the other solutions to be considered as a new member of the initial solution. The used method to determinate the neighborhood parameter is inspired from [2], which is based on the permutation level of operations to obtain the distance between two solutions. In fact, the dissimilarity distance is calculated by verifying the difference between two chromosomes in terms of the placement of each operation $O_{i,j}$ on its alternative machine set in the machine assignment vector V_1 and its execution order in the operation sequence vector V_2. So, if there is a difference in the vector V_1, the distance is incremented by $M(O_{i,j})$ (is the number of possible n placement for each operation on its machine set, which is the alternative machine number of each operation $O_{i,j}$) because it is in the order of $O(n)$. Then, if there is a difference in the vector V_2, the distance is incremented by 1 because it is in the order of $O(1)$. Let $Chrom1(MA_1,OS_1)$ and $Chrom2(MA_2,OS_2)$ two chromosomes of two different scheduling solutions, $M(O_{i,j})$ the alternative number of machines of each operation $O_{i,j}$, L is the total number of operations of all jobs and $Dist$ is the dissimilarity distance. The distance is calculated firstly by measuring the difference between the machine assignment vectors MA_1 and MA_2 which is in order of $O(n)$, then by verifying the execution order difference of the operation sequence vectors OS_1 and OS_2 which is in order of $O(1)$, we give here how to proceed:

```
Begin
 Dist=0, k=1
 For k from 1 to L
   If Chrom1(MA1(k)) ≠ Chrom2(MA2(k))
     Dist = Dist + M(O1,j)
   End if
   If Chrom1(OS1(k)) ≠ Chrom2(OS2(k))
     Dist = Dist + 1
   End if
 End for
 Return Dist
End.
```

Noting that $Distmax$ is the maximal dissimilarity distance and it is calculated by Equation 4, representing 100% of difference between two chromosomes.

$$Distmax = \sum_{i,1}^{i,ni} [M(Oi,j)] + L \tag{4}$$

3.1.3 Selection operator

The selection operator is used to select the best parent individuals to prepare them to the crossover step. This operator is based on a fitness parameter allowing to analyze the quality of each selected solution. But progressively the fitness values will be similar for the most individuals. That is why, we integrate the neighborhood parameter, where we propose a new combined parent selection operator named Fitness-Neighborhood Selection Operator (FNSO) allowing to add the dissimilarity distance criteria to the fitness parameter to select the best parents for the crossover step. The FNSO chooses in each iteration two parent individuals until engaging all the population to create the next generation. The first parent takes successively in each case a solution i, where $i \in \{1, \ldots, P\}$ and P is the total population size. The second parent obtains its solution j randomly by the roulette wheel selection method based on the two Fitness and Neighborhood parameters relative to the selected first parent, where $j \in \{1, \ldots, P\} \setminus \{i\}$ in the P population and where j ≠ i. In fact, to use this random method, we should calculate the Fitness-Neighborhood total FN for the population, see Equation 5, the selection probability sp_k for each individual I_k, see Equation 6, and the cumulative probability cp_k, see Equation 7. After that, a random number r will be generated from the uniform range [0,1]. If $r \leq cp_1$ then the second parent takes the first individual I_1, else it gets the k^{th} individual $I_k \in \{I_2, \ldots, I_P\} \setminus \{I_i\}$ and where $cp_{k-1} < r \leq cp_k$.

- The Fitness-Neighborhood total for the population:

$$FN = \sum_{k=1}^{P} [1/(Cmax[k] \times Neighborhood[i][k])] \tag{5}$$

- The selection probability sp_k for each individual I_k:

$$sp_k = \frac{1/(Cmax[k] \times Neighborhood[i][k])}{FN} \tag{6}$$

- The cumulative probability cp_k for each individual I_k:

$$cp_k = \sum_{h=1}^{k} p_h \tag{7}$$

⇒ For Equations 5, 6 and 7, $k = \{1, 2, \ldots, P\} \setminus \{i\}$

3.1.4 Crossover operator

The crossover operator has an important role in the global process, allowing to combine in each case the chromosomes of two parents in order to obtain new individuals and to attain new better parts in the search space. In this work, this operator is applied with two different techniques successively for the parent's chromosome vectors MA and OS.

Machine vector crossover.

A uniform crossover is used to generate in each case a mixed vector between two machine vector parents, Parent1-MA1 and Parent2-MA2, allowing to obtain two new children, Child1-MA1′ and Child2-MA2′. This uniform crossover is based on two

assignment cases, if the generated number is less than 0.5, the first child gets the current machine value of parent1 and the second child takes the current machine value of parent2. Else, the two children change their assignment direction, first child to parent2 and the second child to parent1.

Operation vector crossover.

An improved precedence preserving order-based on crossover (iPOX), inspired from [20], is adapted for the parent operation vector OS. This iPOX operator is applied following four steps, a first step is selecting two parent operation vectors (OS_1 and OS_2) and generating randomly two job sub-sets Js_1/Js_2 from all jobs. A second step is allowing to copy any element in OS_1/OS_2 that belong to Js_1/Js_2 into child individual OS'_1/OS'_2 and retain them in the same position. Then the third step deletes the elements that are already in the sub-set Js_1/Js_2 from OS_1/OS_2. Finally, fill orderly the empty position in OS'_1/OS'_2 with the reminder elements of OS_2/OS_1 in the fourth step.

3.1.5 Mutation operator

The mutation operator is integrated to promote the children generation diversity. In fact, this operator is applied on the chromosome of the new children generated by the crossover operation. Also, each part of a child chromosome MA and OS has separately its own mutation technique.

Machine vector mutation.

This first operator uses a random selection of an index from the machine vector MA. Then, it replaces the machine number in the selected index by another belonging to the same alternative machine set.

Operation vector mutation.

This second operator selects randomly two indexes index1 and index2 from the operation vector OS. Next, it changes the position of the job number in the index1 to the second index2 and inversely.

3.1.6 Replacement operator

The replacement operator has an important role to prepare the remaining surviving population to be considered for the next iterations. This operator replaces in each case a parent by one of its children which has the best fitness in its current family.

3.1.7 Clustering operator

By finishing the maximum iteration number *MaxIter* of the genetic algorithm, the Scheduler Agent applies a clustering operator using the hierarchical clustering algorithm of [15] to divide the final population into N Clusters to be treated by the Cluster Agents in the second step of the global process. The clustering operator is based on the neighbourhood parameter which is the dissimilarity distance between individuals. The clustering operator starts by assigning each individual $Indiv(i)$ to a cluster CL_i, so if we have P individuals, we have now P clusters containing just one individual in each of them. For each case, we fixe an individual $Indiv(i)$ and we verify successively for each next individual $Indiv(j)$ from the remaining population (where i and $j \in \{1, \ldots, P\}$, $i \neq j$) if the dissimilarity distance $Dist$ between $Indiv(i)$ and $Indiv(j)$ is less than or equal to a fixed threshold $Distfix$ (representing a percentage of difference X% relatively to $Distmax$, see Equation 8) and where $Cluster(Indiv(i)) \neq Cluster(Indiv(j))$. If it is the case, $Merge(Cluster(Indiv(i)), Cluster(Indiv(j)))$, else continue the search for new combination with the remaining individuals. The stopping condition is by browsing all the population individuals, where we obtained at the end N Clusters.

$$Distfix = Distmax \times X\% \quad (8)$$

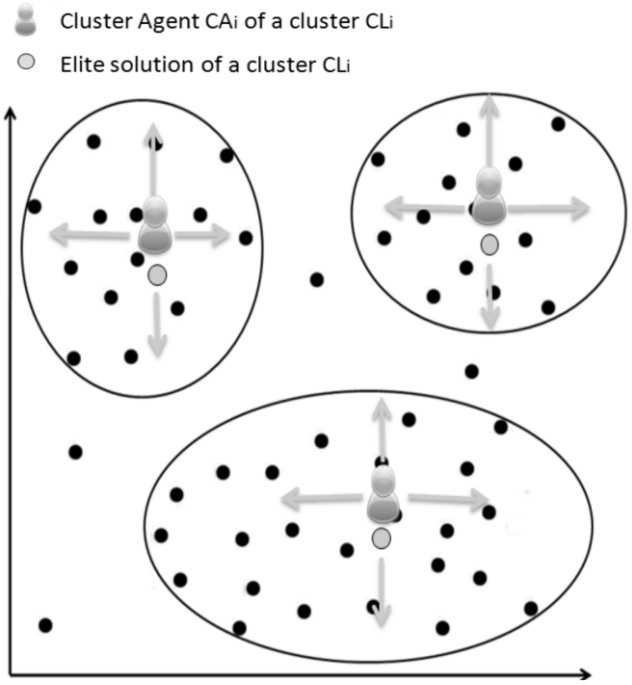

Figure 3. Distribution of the Cluster Agents in the different clusters of the search space

3.2 Cluster Agents

Each Cluster Agent CA_i is responsible to apply successively to each cluster CL_i a local search technique which is the Tabu Search algorithm to guide the research in promising regions of the search space and to improve the quality of the final population of the genetic algorithm. In fact, this local search is executed simultaneously by the set of the CAs agents, where each CA starts the research from an elite solution of its cluster searching to attain new more dominant individual solutions separately in its assigned cluster CL_i, see Figure 3. The used Tabu Search algorithm is based on an intensification technique allowing to start the research from an elite solution in a cluster CL_i (a promising part in the search space) in order to collect new scheduling sequence minimizing the makespan. Let E the elite solution of a cluster CL_i, $E' \in N(E)$ is a neighbor of the elite solution E, GL_i is the Global List of each CA_i to receive new found elite solutions by the remaining CAs, each CL_i plays the role of the tabu list with a dynamic length and $Cmax$ is the makespan of the obtained solution. So, the search process of this local search starts from an elite solution E using the *move and insert* method of [21], where each Cluster Agent CA_i changes the position of an operation $O_{i,j}$ from a machine m to another machine n belonging to the same alternative machine set of this selected operation $O_{i,j}$, searching to generate new scheduling combination $E' \in N(E)$. After that, verifying if the makespan value of this new generated solution $Cmax(E')$ dominates $Cmax(E)$ ($Cmax(E') < Cmax(E)$), and if it is the case CA_i saves E' in its tabu list (which is CL_i) and sends it to all the other CAs agents to be placed in their Global Lists

$GLs(E',CA_i)$, to ensure that it will not be used again by them as a search point. Else continues the neighborhood search from the current solution E. The stopping condition is by attaining the maximum allowed number of neighbors for a solution E without improvement. We give here how to proceed:

```
Begin
 E ← Elite(CL_i)
 While N(E) ≠ ∅
  E' ← {Move and insert(E) | E' ∈ N(E) | E' ∉ GL_i}
  If Cmax(E') < Cmax(E) and E' ∉ CL_i
    E ← E'
    CL_i ← E'
    Send_to_all(E',CA_i)
  End if
 End while
 Return E
End.
```

By finishing this local search step, the CAs agents terminate the process by sending their last best solutions to the SA agent, which considers the best one of them the global solution for the FJSP.

4. EXPERIMENTAL RESULTS

4.1 Experimental setup

The proposed GATS+HM is implemented in java language on a 2.10 GHz Intel Core 2 Duo processor and 3 Gb of RAM memory, where we use the Integrated Development Environment (IDE) *eclipse* to code the algorithm and the multiagent platform *Jade* to create the different agents of our holonic model. To evaluate its efficiency, numerical tests are made based on three sets of well known benchmark instances in the literature of the FJSP: *Kacem data* [17] consisting of 5 problems considering a number of jobs ranging from 4 to 15, which will be processed on a number of machines ranging from 5 to 10. *Brandimarte data* [4] consisting of 10 problems considering a number of jobs ranging from 10 to 20, which will be processed on a number of machines ranging from 4 to 15. *Hurink edata* [14] consisting of 40 problems (la01-la40) inspired from the classical job shop instances of [19], where three test problems are generated: rdata, vdata and edata which is used in this paper.

Due to the non-deterministic nature of the proposed algorithm, we run it five independent times for each one of the three instances [17], [4] and [14] in order to obtain significant results. The computational results are presented by five metrics such as the best makespan (*Best*), the average of makespan (*Avg Cmax*), the average of CPU time in seconds (*Avg CPU*), and the standard deviation of makespan (*Dev %*) which is calculated by Equation 9. The Mk**o** is the makespan obtained by **O**ur algorithm and Mk**c** is the makespan of an algorithm that we chose to **C**ompare to.

$$Dev = [(Mkc - Mko)/Mkc] \times 100\% \quad (9)$$

The used parameter settings for our algorithm are adjusted experimentally and presented as follow: the crossover probability 1.0, the mutation probability 1.0 and the maximum number of iterations 1000. The population size ranged from 15 to 400 depending on the complexity of the problem.

4.2 Experimental comparisons

To show the efficiency of our GATS+HM algorithm, we compare its obtained results from the three previously cited data sets with other well known algorithms in the literature of the FJSP. The chosen algorithms are : the TS of [4], N1-1000 of [14] (with their literature lower bound LB) and the AL+CGA of [17] which obtained the first results in the literature for their proposed instances. The Heuristic of [23] which is a standard heuristic method. The Hybrid NSGA-II of [22] is a recent hybrid metaheuristic algorithm. The MATSLO+ of [6] and the MATSPSO of [12] are two new hybrid metaheuristic algorithms distributed in multiagent models.

The different comparative results are displayed in the Tables 2, 3 and 4 where the first column takes the name of each instance, the second column gives the size each instance, with n the number of jobs and m the number of machines ($n \times m$), and the remaining columns detail the experimental results of the different chosen approaches in terms of the best *Cmax* (*Best*) and the standard deviation (*Dev %*). The bold values in the tables signify the best obtained results and the *N/A* means that the result is not available.

4.2.1 Analysis of the comparative results

By analyzing the Table 2, it can be seen that our approach GATS+HM is the best one which solves the fives instances of Kacem. In fact, the GATS+HM outperforms the AL+CGA in four out of five instances, the Hybrid NSGA-II in two out of five instances, and the Heuristic in three out of five instances. Also, our approach attains the same results obtained by the chosen approaches, such as in the case 1 (4×5) for the Hybrid NSGA-II and the Heuristic; in the case 4 (10×10) for all the three algorithms; in the case 5 (15×10) for the Hybrid NSGA-II.

From Table 3, the results show that the GATS+HM obtains nine out of ten best results for the Brandimarte instances. In fact, our approach outperforms the TS in nine out of ten instances. Moreover, for the comparison with MATSLO+, our GATS+HM outperforms it in eight out of ten instances. Furthermore, the MATSPSO attained the best result for the MK01 instance, but our approach obtains a set of solutions better than it for the remaining instances. By solving this second data set, our GATS+HM attains the same results obtained by some approaches such as the MK01 for MATSLO+, the MK02 for MATSPSO and the MK08 for all methods.

From the results in Table 4, we can see that the GATS+HM obtains seven out of ten best results for the Hurink edata instances (la01-la05) and (la16-la20). Indeed, our approach outperforms the N1-1000 in eight out of ten instances. Moreover, our GATS+HM outperforms the MATSLO+ in seven out of ten instances. For the comparison with the literature lower bound LB, the GATS+HM attains the same results for the la01, la02, la04, la05, la16, la17 and la20 instances, but it gets slightly worse result for the la03, la18 and la19 instances. Furthermore, by solving this third data set, our GATS+HM attains the same results obtained by the chosen approaches such as in the la01 for the MATSLO+; in the la02 for the N1-1000 and the MATSLO+; in the la05 for the N1-1000 and the MATSLO+.

By analyzing the computational time in seconds and the comparison results of our algorithm in term of makespan, we can distinguish the efficiency of the new proposed GATS+HM relatively to the literature of the FJSP. This efficiency is explained by the flexible selection of the promising parts of the search space by the clustering operator after the genetic algorithm process and

Table 2. Results of the Kacem instances

Instance	Problem	AL+CGA		Hybrid NSGA-II		Heuristic		GATS+HM		
	n×m	Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Avg Cmax	Avg C.P.U (in seconds)
case 1	4×5	16	31,250	**11**	0	**11**	0	**11**	11,00	0,05
case 2	8×8	15	6,666	15	6,666	15	6,666	**14**	14,20	0,36
case 3	10×7	15	26,666	N/A	--	13	15,384	**11**	11,40	0,72
case 4	10×10	**7**	0	**7**	0	**7**	0	**7**	7,60	1,51
case 5	15×10	23	52,173	**11**	0	12	8,333	**11**	11,60	29,71

Table 3. Results of the Brandimarte instances

Instance	Problem	TS		MATSLO+		MATSPSO		GATS+HM		
	n×m	Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Avg Cmax	Avg C.P.U (in seconds)
Mk01	10×6	42	4,761	40	0	**39**	-2,564	40	40,80	0,93
Mk02	10×6	32	15,625	32	15,625	**27**	0	**27**	27,80	1,18
Mk03	15×8	211	3,317	207	1,449	207	1,449	**204**	204,00	1,55
Mk04	15×8	81	20,987	67	4,477	65	1,538	**64**	65,60	4,36
Mk05	15×4	186	6,989	188	7,978	174	0,574	**173**	174,80	8,02
Mk06	10×15	86	24,418	85	23,529	72	9,722	**65**	67,00	110,01
Mk07	20×5	157	8,280	154	6,493	154	6,493	**144**	144,00	19,73
Mk08	20×10	**523**	0	**523**	0	**523**	0	**523**	523,00	11,50
Mk09	20×10	369	15,718	437	28,832	340	8,529	**311**	311,80	79,68
Mk10	20×15	296	25	380	41,578	299	25,752	**222**	224,80	185,64

Table 4. Results of the Hurink edata instances

Instance	Problem	LB		N1-1000		MATSLO+		GATS+HM		
	n×m	Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Avg Cmax	Avg C.P.U (in seconds)
la01	10×5	**609**	0	611	0,327	**609**	0	**609**	609,00	24,64
la02	10×5	**655**	0	**655**	0	**655**	0	**655**	655,00	4,65
la03	10×5	**550**	-3,091	573	1,047	575	1,391	567	567,40	10,67
la04	10×5	**568**	0	578	1,730	579	1,900	**568**	569,60	22,13
la05	10×5	**503**	0	**503**	0	**503**	0	**503**	503,00	10,22
la16	10×10	**892**	0	924	3,463	896	0,446	**892**	909,60	73,14
la17	10×10	**707**	0	757	6,605	708	0,141	**707**	709,60	116,58
la18	10×10	**842**	-0,119	864	2,431	845	0,237	843	848,60	34,98
la19	10×10	**796**	-1,005	850	5,412	813	1,107	804	813,40	36,88
la20	10×10	**857**	0	919	6,746	863	0,695	**857**	859,80	70,36

by applying the intensification technique of the tabu search allowing to start from an elite solution to attain new more dominant solutions.

5. CONCLUSION

In this paper, we present a new metaheuristic hybridization based on clustering in a holonic multiagent model, called GATS+HM, for the flexible job shop scheduling problem (FJSP). In this approach, a Neighborhood-based Genetic Algorithm is adapted by a Scheduler Agent (SA) for a global exploration of the search space. Then, a local search technique is applied by a set of Cluster Agents (CAs) to guide the research in promising regions of the search space and to improve the quality of the final population. To measure its performance, numerical tests are made using three well known data sets in the literature of the FJSP. The experimental results show that the proposed approach is efficient in comparison to others approaches. In the future work, we will search to treat other extensions of the FJSP, such as by integrating new transportation resources constraints in the shop process. So, we will make improvements to our approach to adapt it to this new transformation and study its effects on the makespan.

6. REFERENCES

[1] V. Botti and A. Giret. *ANEMONA: A Multi-agent Methodology for Holonic Manufacturing Systems*. Springer Series in Advanced Manufacturing. Springer-Verlag, 2008.

[2] W. Bozejko, M. Uchronski, and M. Wodecki. The new golf neighborhood for the flexible job shop problem. In *Proceedings of the International Conference on Computational Science*, pages 289–296, May 2010.

[3] W. Bozejko, M. Uchronski, and M. Wodecki. Parallel hybrid metaheuristics for the flexible job shop problem. *Computers and Industrial Engineering*, 59(2):323–333, September 2010.

[4] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, September 1993.

[5] M. Calabrese. *Hierarchical-granularity holonic modelling*. Doctoral thesis, Universita degli Studi di Milano, Milano, Italy, March 2011.

[6] M. Ennigrou and K. Ghédira. New local diversification techniques for the flexible job shop problem with a multi-agent approach. *Autonomous Agents and Multi-Agent Systems*, 17(2):270–287, October 2008.

[7] J. Ferber. Multi-Agent Systems: *An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1999.

[8] J. Gao, L. Sun, and M. Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers and Operations Research*, 35(9):2892–2907, September 2008.

[9] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research*, 1(2):117–129, May 1976.

[10] A. Giret and V. Botti. Holons and agents. *Journal of Intelligent Manufacturing*, 15(5):645–659, 2004.

[11] F. Glover, J. P. Kelly, and M. Laguna. Genetic algorithms and tabu search: Hybrids for optimization. *Computers and Operations Research*, 22(1):111–134, January 1995.

[12] A. Henchiri and M. Ennigrou. Particle swarm optimization combined with tabu search in a multi-agent model for flexible job shop problem. In *Proceedings of the 4th International Conference on Swarm Intelligence*, *Advances in Swarm Intelligence*, pages 385–394, June 2013.

[13] N. B. Ho, J. C. Tay, and E. M. K. Lai. An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, 179(2):316–333, June 2007.

[14] J. Hurink, B. Jurisch, and M. Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations Research Spektrum*, 15(4):205–215, December 1994.

[15] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, September 1967.

[16] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, 32(1):1–13, February 2002.

[17] I. Kacem, S. Hammadi, and P. Borne. Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 60(3-5):245–276, September 2002.

[18] A. Koestler. *The Ghost in the Machine*. Hutchinson, London, United Kingdom, 1st edition, 1967.

[19] S. Lawrence. Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. *Technical report*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[20] K. Lee, T. Yamakawa, and K. M. Lee. A genetic algorithm for general machine scheduling problems. In *Proceedings of the second IEEE international Conference on Knowledge-Based Intelligent Electronic Systems*, pages 60–66, April 1998.

[21] M. Mastrolilli and L. Gambardella. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(1):3–20, January 2000.

[22] C. Zhang, P. Gu, and P. Jiang. Low-carbon scheduling and estimating for a flexible job shop based on carbon footprint and carbon efficiency of multi-job processing. *Journal of Engineering Manufacture*, 39(32):1–15, April 2014.

[23] M. Ziaee. A heuristic algorithm for solving flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 71(1-4):519–528, March 2014.