Nouri, Houssem Eddine; Driss, Olfa Belkahla; Ghédira, Khaled

Article

Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model

Journal of Industrial Engineering International

Provided in Cooperation with:
Islamic Azad University (IAU), Tehran

Suggested Citation: Nouri, Houssem Eddine; Driss, Olfa Belkahla; Ghédira, Khaled (2018) : Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model, Journal of Industrial Engineering International, ISSN 2251-712X, Springer, Heidelberg, Vol. 14, Iss. 1, pp. 1-14, https://doi.org/10.1007/s40092-017-0204-z

This Version is available at:
https://hdl.handle.net/10419/195587

J Ind Eng Int (2018) 14:1–14
https://doi.org/10.1007/s40092-017-0204-z

ORIGINAL RESEARCH

Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model

Houssem Eddine Nouri[1,2] · **Olfa Belkahla Driss**[1,3] · **Khaled Ghédira**[1,2]

Received: 29 February 2016 / Accepted: 4 May 2017 / Published online: 16 May 2017

Abstract The flexible job shop scheduling problem (FJSP) is a generalization of the classical job shop scheduling problem that allows 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 assignment and the scheduling problems. In this paper, we propose how to solve the FJSP by hybrid metaheuristics-based clustered holonic multiagent model. First, a neighborhood-based genetic algorithm (NGA) is applied by a scheduler agent for a global exploration of the search space. Second, 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. The efficiency of our approach is explained by the flexible selection of the promising parts of the search space by the clustering operator after the genetic algorithm process, and by applying the intensification technique of the tabu search allowing to restart the search from a set of elite solutions to attain new dominant scheduling solutions. Computational results are presented using four sets of well-known benchmark literature instances. New upper bounds are found, showing the effectiveness of the presented approach.

Keywords Scheduling · Flexible job shop · Genetic algorithm · Local search · Holonic multiagent · Hybrid metaheuristics

✉ Houssem Eddine Nouri
houssemeddine.nouri@gmail.com

Olfa Belkahla Driss
Olfa.Belkahla@isg.rnu.tn

Khaled Ghédira
khaled.ghedira@anpr.tn

[1] SOIE-COSMOS, École Nationale des Sciences de l'Informatique, Université de la Manouba, Tunis, Tunisia

[2] Institut Supérieur de Gestion de Tunis, Université de Tunis, Bardo, Tunis, Tunisia

[3] École Supérieure de Commerce de Tunis, Université de la Manouba, Tunis, Tunisia

Introduction

Scheduling is a field of investigation which has known a significant growth these last years. The scheduling problems appear in all the economic areas, from computer engineering to industrial production and manufacturing. The job shop scheduling problem (JSP), which is among the hardest combinatorial optimization problems (Sonmez and Baykasoglu 1998), is a branch of the industrial production scheduling problems.

The flexible job shop scheduling problem (FJSP) is a generalization of the classical 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 (Garey et al. 1976).

To solve this problem, Pinedo (2002) developed a set of exact algorithms limited for instances with 20 jobs and 10 machines. Birgin et al. (2014) presented a mixed integer linear programming (MILP) model, but it took a very large time to generate a scheduling solution. Shafigh et al. (2015) developed a mathematical model integrating layout configuration and production planning in the design of dynamic

distributed layouts. Their model incorporated different manufacturing attributes such as demand fluctuation, system reconfiguration, lot splitting, work load balancing, alternative routings, machine capability, and tooling requirements. On the other hand, a community of researchers used the metaheuristics to find near-optimal solutions for the FJSP with acceptable computational time. Brandimarte (1993) proposed a hierarchical algorithm based on tabu search metaheuristic for routing and scheduling with some known dispatching rules to solve the FJSP. Hurink et al. (1994) developed a Tabu Search procedure for the job shop problem with multi-purpose machines. Dauzère-Pérès and Paulli (1997) presented a new neighborhood structure for the problem, and a list of Tabu moves was used to prevent the local search from cycling. Mastrolilli and Gambardella (2000) used tabu search techniques and presented two neighborhood functions allowing an approximate resolution for the FJSP. Bozejko et al. (2010a) presented a tabu search approach based on a new golf neighborhood for the FJSP, and in the same year, Bozejko et al. (2010b) proposed another new model of a distributed tabu search algorithm for the FJSP, using a cluster architecture consisting of nodes equipped with the GPU units (multi-GPU) with distributed memory. A novel hybrid tabu search algorithm with a fast Public Critical Block neighborhood structure (TSPCB) was proposed by Li et al. (2011) to solve the FJSP. For the genetic algorithm, it was adopted by Chen et al. (1999), where their chromosome representation of solutions for the problem was divided into two parts. The first part defined the routing policy and the second part took the sequence of operations on each machine. Kacem et al. (2002a) used a genetic algorithm 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 Kacem et al. (2002b). Jia et al. (2003) proposed a modified genetic algorithm for the FJSP, where various scheduling objectives can be achieved such as minimizing makespan, cost, and weighted multiple criteria. Ho et al. (2007) developed a new architecture 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. Gao et al. (2008) 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. Zhang et al. (2014) 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. Kar et al. (2015) presented a production-inventory model for deteriorating items with stock-dependent demand under inflation in a random planning horizon. This model is formulated as profit maximization problem with respect to the retailer and solved by two metaheuristics, which are the genetic algorithm and the particle-swarm optimization. Kia et al. (2017) treated the dynamic flexible flow line problem with sequence-dependent setup times. A set of composite dispatching rule-based genetic programming are proposed to solve this problem by minimizing the mean flow time and the mean tardiness objectives. Moreover, the particle-swarm optimization was implemented by Xia and Wu (2005) in a metaheuristic hybridization approach with the simulated annealing for the multi-objective FJSP. A combined particle-swarm optimization and a tabu search algorithm were proposed by Zhang et al. (2009) to solve the multi-objective FJSP. Moslehi and Mahnam (2011) presented a metaheuristic approach based on a hybridization of the particle-swarm optimization and local search algorithm to solve the multi-objective FJSP. In addition, other types of metaheuristics were developed in this last few years, such as (Yazdani et al. 2010) implementing a parallel variable neighborhood search (PVNS) algorithm to solve the FJSP using various neighborhood structures. A new biogeography-based optimization (BBO) technique is developed by Rahmati and Zandieh (2012) allowing to search a solution area for the FJSP and to find the optimum or near-optimum scheduling to this problem. Shahriari et al. (2016) studied the just in time single machine scheduling problem with a periodic preventive maintenance. A multi-objective version of the particle-swarm optimization algorithm is implemented to minimize the total earliness–tardiness and the makespan simultaneously. In addition, it is noted that metaheuristics based on constraint programming (CP) techniques have been used for the FJSP. Hmida et al. (2010) proposed a variant of the climbing discrepancy search approach (C.D.S) for solving the FJSP, where they presented various neighborhood structures related to assignment and sequencing problems. Pacino and Hentenryck (2011) considered a constraint-based scheduling approach to the flexible job shop problem. They studied both the large neighborhood search (LNS) and the adaptive randomized decomposition (ARD) schemes, using random, temporal, and machine decompositions. Oddi et al. (2011) adapted an iterative flattening search (IFS) algorithm for solving the flexible job shop scheduling problem (FJSSP). This algorithm applied two steps, a first relaxation step, in which a sub-set of scheduling decisions was randomly retracted from the current solution, and a second solving step, in which a new solution was incrementally recomputed from this partial schedule. Moreover, a new

heuristic was developed by Ziaee (2014) 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 Ennigrou and Ghédira (2004) 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. In addition, this model was improved in Ennigrou and Ghédira (2008), where the optimization role of the interface agent was distributed among the resource agents. Henchiri and Ennigrou (2013) proposed a 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. Rezki et al. (2016) proposed a multiagent system combining many intelligent techniques such as: multivariate control charts, neural networks, bayesian networks, and expert systems, for complex process monitoring tasks that are: detection, diagnosis, identification, and reconfiguration.

In this paper, we present how to solve the flexible job shop scheduling problem by a hybridization of two metaheuristics within 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 four data sets of Kacem et al. (2002b), Brandimarte (1993), Hurink et al. (1994), and Barnes and Chambers (1996) for the FJSP, where the experimental results show its efficiency in comparison with other approaches.

The rest of the paper is organized as follows. In the next section, we define the formulation of the FJSP with its objective function and a simple problem instance followed by which we detail the proposed hybrid metaheuristic algorithm with its clustered holonic multiagent levels. The experimental and comparison results are provided in the subsequent section. The final section rounds up the paper with a conclusion.

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 $C_{\max}$ in Eq. 1, where C_i is the completion time of a job J_i:

$$C_{\max} = \max_{1 \le i \le n}(C_i). \quad (1)$$

The FJSP scheduling problem is divided into two sub-problems:

- The operations assignment sub-problem assigns each operation to an appropriate machine.
- The operations sequencing sub-problem determines a sequence of operations on all the machines.

Furthermore, the adopted hypotheses in this problem are:

- All the machines are available at time zero.
- All jobs are ready for processing at time zero.
- The order of operations for each job is predefined and cannot be modified.
- There are no precedence constraints among operations of different jobs.
- The processing time of operations on each machine is defined in advance.
- Each machine can process only one operation at a time.
- Operations belonging to different jobs can be processed in parallel.
- Each job could be processed more than once on the same machine.
- The interruption during the process of an operation on a machine is negligible.

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.

A metaheuristic hybridization within a holonic multiagent model

Glover et al. (1995) 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.

Table 1 Simple instance of the FJSP

Job	Operation	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	9	4	5	1
	$O_{1,2}$	–	6	–	4	–
J_2	$O_{2,1}$	1	–	5	–	6
	$O_{2,2}$	3	8	6	–	–
	$O_{2,3}$	–	5	9	3	9
J_3	$O_{3,1}$	–	6	6	–	–
	$O_{3,2}$	3	–	–	5	4

Ferber (1999) 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 (Botti and Giret 2008).

Koestler (1967) 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 (Calabrese 2011). 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 behavior of the super-agent. Holons are agents able to show an architectural recursiveness (Giret and Botti 2004).

In this work, we propose a hybrid metaheuristic approach 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 genetic algorithm combined with tabu search in a holonic multiagent model (GATS+HM), see Fig. 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.

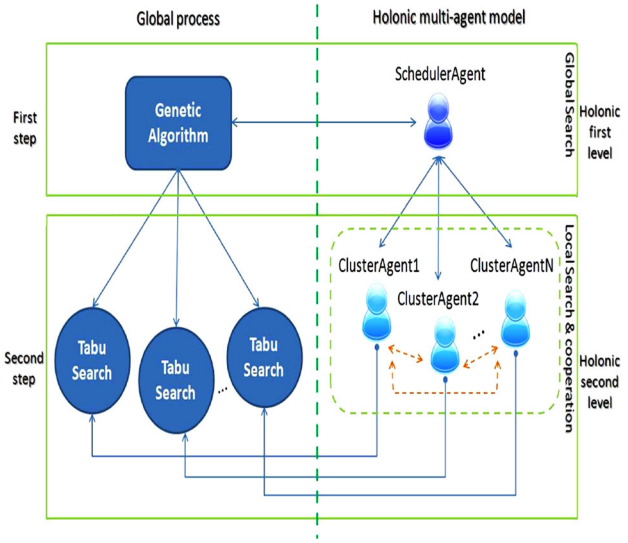

Fig. 1 Metaheuristic hybridization within a holonic multiagent model

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 using local search techniques to generate a more improved set of neighbourhood solutions from an initial solution. However, 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. Therefore, 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.

Scheduler agent

The scheduler agent (SA) is responsible to process the first step of the hybrid algorithm using a genetic algorithm called neighborhood-based genetic algorithm (NGA) to identify areas with high average fitness in the search space during a fixed number of iterations *MaxIter*, see Fig. 2. In fact, the

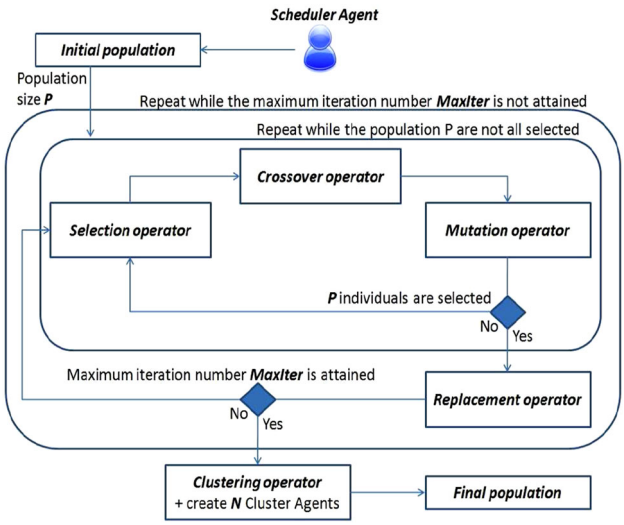

Fig. 2 First step of the global process by the scheduler agent

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 $\mathrm{CL}_i \in \mathrm{CL}$ the set of clusters, where $\mathrm{CL} = \{\mathrm{CL}_1, \mathrm{CL}_2, \ldots, \mathrm{CL}_N\}$. 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.

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, where each index represents the selected machine to process an operation indicated at position p, see Fig. 3a. 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, where each index represents an operation $O_{i,j}$ according to the predefined operations of the job set, see Fig. 3b. For example, the operation sequence 1–2–1–3–2–3–2 can be translated to: $(O_{1,1}, M_5) \rightarrow (O_{2,1}, M_1) \rightarrow (O_{1,2}, M_4) \rightarrow (O_{3,1}, M_3) \rightarrow (O_{2,2}, M_3) \rightarrow (O_{3,2}, M_1) \rightarrow (O_{2,3}, M_2)$.

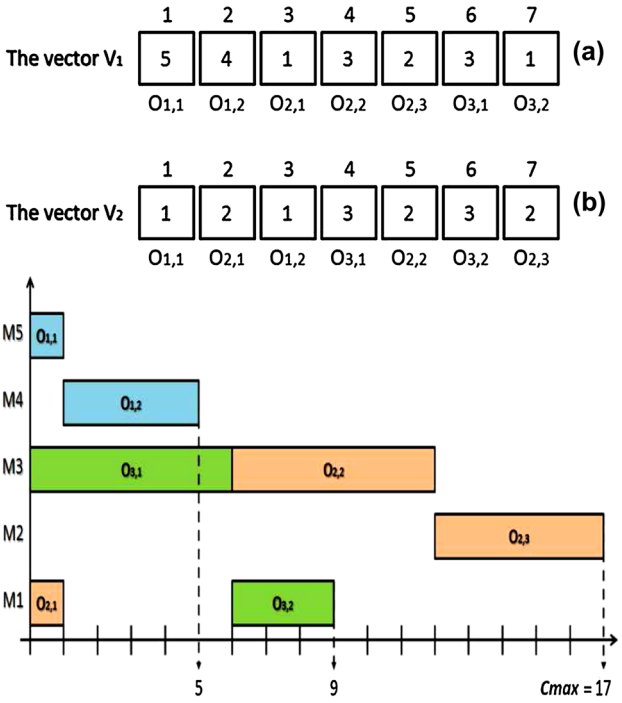

Fig. 3 Chromosome representation of a scheduling solution

To convert the chromosome values to an active schedule, we used the priority-based decoding of Gao et al. (2008). 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 $O_{i,j}$. Therefore, 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 Eq. 2:

$$\begin{aligned} &\text{if } j = 1,\ t^S{}_m + p_{ijm} \leq t^E{}_m \\ &\text{if } j \geq 2,\ \max\{t^S{}_m, C_{i,(j-1)}\} + p_{ijm} \leq t^E{}_m. \end{aligned} \tag{2}$$

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. Therefore, 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. Therefore, 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 $C_{\max}(i)$ is its makespan value, where $i \in \{1,\ldots,P\}$ and P is the total population size, see Eq. 3:

$$\text{Fitness}(i) = \frac{1}{C_{\max}(i)}. \tag{3}$$

Population initialization

The initial population is generated randomly following a uniform law 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 Bozejko et al. (2010a), 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. Therefore, 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 $\text{Chrom1}(\text{MA}_1, \text{OS}_1)$ and $\text{Chrom2}(\text{MA}_2, \text{OS}_2)$ 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 first 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 how to proceed in Algorithm 1.

Algorithm 1 How to calculate the dissimilarity distance between two solutions

```
function
    Dist ← 0, k ← 1
    for k from 1 to L do
        if Chrom1(MA1(k)) ≠ Chrom2(MA2(k)) then
            Dist ← Dist + M(Oi,j)
        end if
        if Chrom1(OS1(k)) ≠ Chrom2(OS2(k)) then
            Dist ← Dist + 1
        end if
    end for
    return Dist
end function
```

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

$$\text{Distmax} = \left[\sum_{i=1}^{n}\sum_{i,1}^{i,ni} M(O_{i,j})\right] + L. \tag{4}$$

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. However, 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 $j \in \{1,\ldots,P\}\backslash\{i\}$ in the P population and where $j \neq i$. In fact, to use this random method, we should calculate the fitness-neighborhood total FN for the population, see Eq. 5, the selection probability sp_k for each individual I_k, see Eq. 6, and the cumulative probability cp_k, see Eq. 7. After that, a random number r will be generated from the uniform range [0,1]. If $r \leq \text{cp}_1$, then the second parent takes the first individual I_1, else it gets the kth individual $I_k \in \{I_2,\ldots,I_P\}\backslash\{I_i\}$ and where $\text{cp}_{k-1} < r \leq \text{cp}_k$.

- The fitness-neighborhood total for the population:

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

- The selection probability sp_k for each individual I_k:

$$\text{sp}_k = \frac{1/(C_{\max}[k] \times \text{Neighborhood}[i][k])}{\text{FN}}. \tag{6}$$

- The cumulative probability cp_k for each individual I_k:

$$\text{cp}_k = \sum_{h=1}^{k} \text{sp}_h. \tag{7}$$

$\Longrightarrow$ For Eqs. 5, 6, and 7, $k = \{1,2,\ldots,P\}\backslash\{i\}$

Crossover operator

The crossover operator has an important role in the global process, allowing to combine in each case the chromosomes of two parents 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, see Algorithm 2.

Algorithm 2 How to generate new children by the MA crossover

```
procedure
    Generate a random number r in [0, 1]
    if r < 0.5 then
        Child1-MA'1[i] = Parent1-MA1
        Child2-MA'2[i] = Parent2-MA2
    else
        Child1-MA'1[i] = Parent2-MA2
        Child2-MA'2[i] = Parent1-MA1
    end if
end procedure
```

Operation vector crossover An improved precedence preserving order based on crossover (iPOX), inspired from Lee et al. (1998), 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, see the example in Fig. 4.

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. In addition, 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, see Fig. 5.

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, see Fig. 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.

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 Johnson (1967) to divide the final population into N clusters, see Fig. 7, 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 Eq. 8) and where Cluster(Indiv(i)) $\neq$ Cluster(Indiv(j)). If it is the case, Merge(Cluster

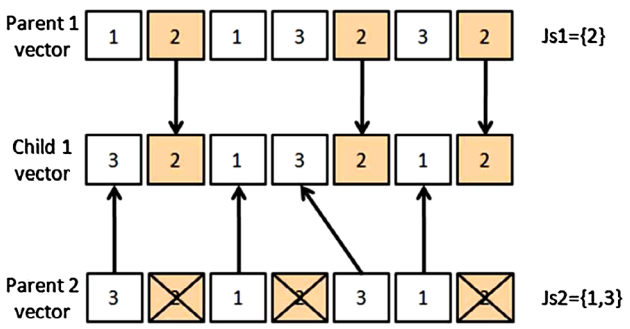

Fig. 4 iPOX crossover example for the OS vector

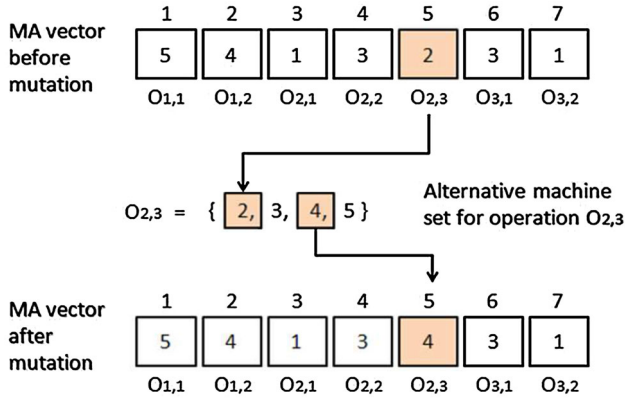

Fig. 5 Mutation operator example for the MA vector

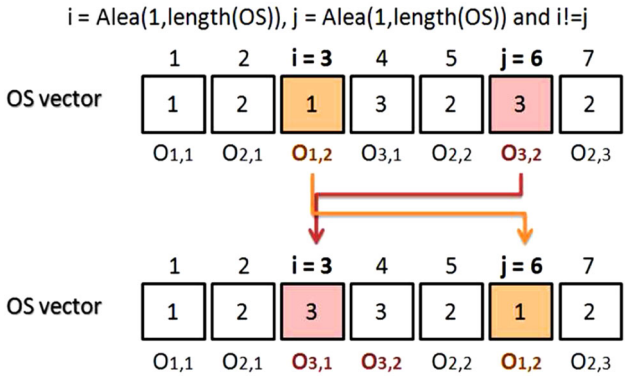

Fig. 6 Mutation operator example for the OS vector

$(\text{Indiv}(i)), \text{Cluster}(\text{Indiv}(j)))$, 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:

$$\text{Distfix} = \text{Distmax} \times X\%. \quad (8)$$

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 Fig. 8. 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) 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

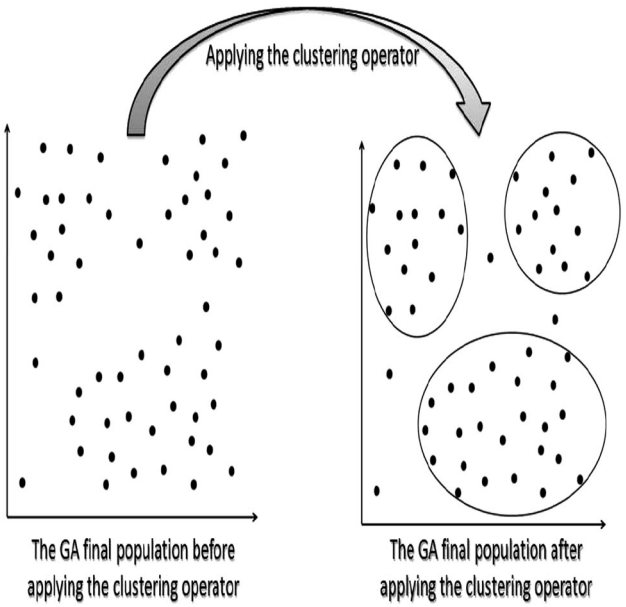

Fig. 7 Final population transformation by applying the clustering operator

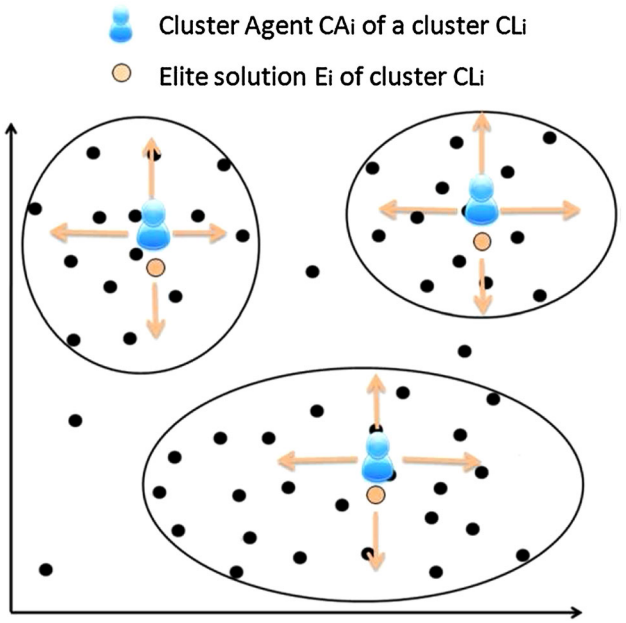

Fig. 8 Distribution of the cluster agents in the different clusters of the search space

found elite solutions by the remaining CAs, each CL_i plays the role of the tabu list with a dynamic length, and $C_{\max}$ is the makespan of the obtained solution. Therefore, the search process of this local search starts from an elite solution E using the *move and insert* method of Mastrolilli and Gambardella (2000), 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 $C_{\max}(E')$ dominates $C_{\max}(E)$ ($C_{\max}(E') < C_{\max}(E)$), 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 how to proceed in Algorithm 3.

Algorithm 3 The local search process

```
function
    E ← Elite(CL_i)
    while N(E) ≠ ∅ do
        E' ← Move-and-insert(E) | E' ∈ N(E) | E' ∉ CL_i
        CL_i ← E'
        if Cmax(E') < Cmax(E) and E' ∉ GL_i then
            E ← E'
            Send-to-all(E', CA_i)
        end if
    end while
    return E
end function
```

By finishing this local search step, the CA 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, see Fig. 9.

Experimental results

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 multi-agent platform *Jade* (Bellifemine et al. 1999) to create the different agents of our holonic model. To evaluate its efficiency, numerical tests are made based on four sets of well-known benchmark instances in the literature of the FJSP:

- *Kacem data* (Kacem et al. 2002b): The data set consists of five problems considering a number of jobs ranging from 4 to 15 with a number of operations for each job ranging from 2 to 4, which will be processed on a number of machines ranging from 5 to 10.
- *Brandimarte data* (Brandimarte 1993): The data set consists of ten problems considering a number of jobs ranging from 10 to 20 with a number of operations for each job ranging from 5 to 15, which will be processed on a number of machines ranging from 4 to 15.
- *Hurink edata* (Hurink et al. 1994): The data set consists of 40 problems (la01–la40) inspired from the classical job shop instances of Lawrence (1984), where three test problems are generated: rdata, vdata, and edata which are used in this paper.
- *Barnes data* (Barnes and Chambers 1996): The data set consists of 21 problems considering a number of jobs ranging from 10 to 15 with a number of operations for each job ranging from 10 to 15, which will be processed on a number of machines ranging from 11 to 18.

Due to the non-deterministic nature of the proposed algorithm, we run it five independent times for each one of the four instances Kacem et al. (2002b), Brandimarte (1993), Hurink et al. (1994), and Barnes and Chambers (1996) 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 Eq. 9. $\mathrm{Mk_o}$ is the makespan obtained by **o**ur algorithm and $\mathrm{Mk_c}$ is the makespan of an algorithm that we chose to **c**ompare to

$$\mathrm{Dev} = [(\mathrm{Mk_c} - \mathrm{Mk_o})/\mathrm{Mk_c}] \times 100\%. \quad (9)$$

The used parameter settings for our algorithm are adjusted experimentally and presented as follows:

- Crossover probability 1.0.
- Mutation probability 0.5.
- Maximum number of iterations 1000.

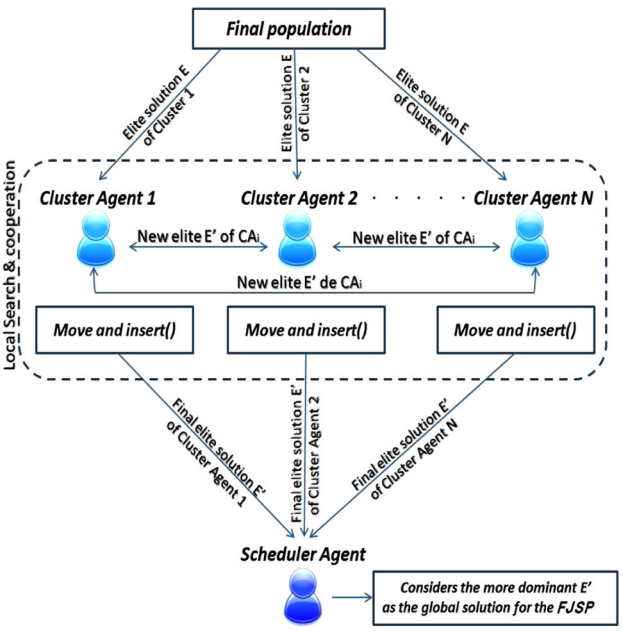

Fig. 9 Second step of the global process by the cluster agents

- The population size ranged from 15 to 400 depending on the complexity of the problem.
- The fixed threshold Distfix represents 50% of the maximal dissimilarity distance Distmax.

Experimental comparisons

To show the efficiency of our GATS+HM algorithm, we compare its obtained results from the four previously cited data sets with other well-known algorithms in the literature of the FJSP.

The chosen algorithms are:

- The TS of Brandimarte (1993), N1-1000 of Hurink et al. (1994) (with its literature lower bound LB), and the AL+CGA of Kacem et al. (2002b) obtained the first results in the literature for their proposed instances.
- The LEGA of Ho et al. (2007), the BBO of Rahmati and Zandieh (2012), and the Heuristic of Ziaee (2014) are standard heuristic and metaheuristic methods.
- The TS3 of Bozejko et al. (2010a) is the paper from which we inspired the computation method of the dissimilarity distance.
- The MOPSO+LS of Moslehi and Mahnam (2011) and the Hybrid NSGA-II of Zhang et al. (2014) are two recent hybrid metaheuristic algorithms.
- The MATSLO+ of Ennigrou and Ghédira (2008) and the MATSPSO of Henchiri and Ennigrou (2013) are two new hybrid metaheuristic algorithms distributed in a multiagent model.

The different comparative results are displayed in Tables 2, 3, 4, 5, 6, and 7, 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

Table 2 Results of the Kacem instances (part 1)

Instance	Problem $n \times m$	AL+CGA		LEGA		MOPSO+LS		BBO	
		Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Dev (%)
Case 1	4 × 5	16	13.250	**11**	0	16	31.25	**11**	0
Case 2	8 × 8	15	6.666	N/A	–	**14**	0	**14**	0
Case 3	10 × 7	15	26.666	**11**	0	15	26.666	N/A	–
Case 4	10 × 10	**7**	0	**7**	0	**7**	0	**7**	0
Case 5	15 × 10	23	52.173	12	8.333	**11**	0	12	8.333

Table 3 Results of the Kacem instances (part 2)

Hybrid NSGA-II		Heuristic		GATS+HM		
Best	Dev (%)	Best	Dev (%)	Best	Avg C_{max}	Avg CPU (in s)
11	0	**11**	0	**11**	11.00	0.05
15	6.666	15	6.666	**14**	14.20	0.36
N/A	–	13	15.384	**11**	11.40	0.72
7	0	**7**	0	**7**	7.60	1.51
11	0	12	8.333	**11**	11.60	29.71

Table 4 Results of the Brandimarte instances (part 1)

Instance	Problem $n \times m$	TS		LEGA		MATSLO+		TS3	
		Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Dev (%)
MK01	10 × 6	42	4.761	40	0	40	0	40	0
MK02	10 × 6	32	15.625	29	6.896	32	15.625	29	6.896
MK03	15 × 8	211	3.317	N/A	–	207	1.449	**204**	0
MK04	15 × 8	81	20.987	67	4.477	67	4.477	65	1.538
MK05	15 × 4	186	6.989	176	1.704	188	7.978	**173**	0
MK06	10 × 15	86	24.418	67	2.985	85	23.529	68	4.411
MK07	20 × 5	157	8.280	147	2.040	154	6.493	**144**	0
MK08	20 × 10	**523**	0	**523**	0	**523**	0	**523**	0
MK09	20 × 10	369	15.718	320	2.812	437	28.832	326	4.601
MK10	20 × 15	296	25	229	3.056	380	41.578	227	2.202

Table 5 Results of the Brandimarte instances (part 2)

BBO		MATSPSO		Heuristic		GATS+HM		
Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Avg C_{max}	Avg CPU (s)
40	0	**39**	−2.564	42	4.761	40	40.80	0.93
28	3.571	**27**	0	28	3.571	**27**	27.80	1.18
204	0	207	1.449	**204**	0	**204**	204.00	1.55
64	0	65	1.538	75	14.666	**64**	65.60	4.36
173	0	174	0.574	179	3.351	**173**	174.80	8.02
66	1.515	72	9.722	69	5.797	**65**	67.00	110.01
144	0	154	6.493	149	3.355	**144**	144.00	19.73
523	0	**523**	0	555	5.765	**523**	523.00	11.50
310	−0.322	340	8.529	342	9.064	311	311.80	79.68
230	3.478	299	25.752	242	8.264	**222**	224.80	185.64

Table 6 Results of the Hurink edata instances

Instance	Problem $n \times m$	LB		N1-1000		MATSLO+		GATS+HM		
		Best	Dev (%)	Best	Dev (%)	Best	Dev (%)	Best	Avg C_{max}	Avg CPU (in s)
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

Table 7 Results of the Barnes data instances

Instance	Problem $n \times m$	BBO					GATS+HM				
		Pop	Best	Avg C_{max}	Dev (%)	Avg CPU (in s)	Pop	Best	Avg C_{max}	Avg CPU (in s)	
mt10c1	10 × 11	350	946	947.00	2.008	401	300	**927**	930.00	84.26	
mt10cc	10 × 12	350	946	946.00	3.065	405	300	**917**	918.60	78.40	
mt10x	10 × 11	350	955	961.00	3.350	416	300	**923**	931.40	82.56	
mt10xx	10 × 12	350	939	945.00	2.236	480	300	**918**	924.40	81.73	
mt10xxx	10 × 13	350	954	954.50	3.773	497	300	**918**	921.00	95.22	
mt10xy	10 × 12	350	951	951.00	4.521	458	300	**908**	910.00	93.48	
mt10xyz	10 × 13	350	**858**	858.00	−1.165	495	300	868	871.80	72.03	
setb4c9	15 × 11	350	959	959.00	3.336	762	250	**927**	936.60	104.10	
setb4cc	15 × 12	350	944	950.00	0.635	770	300	**938**	946.80	150.34	
setb4x	15 × 11	350	**942**	951.00	−0.212	749	200	944	956.20	61.05	
setb4xx	15 × 12	350	967	967.00	2.585	761	300	**942**	953.60	145.74	
setb4xxx	15 × 13	350	991	991.00	4.238	797	300	**949**	958.60	133.19	
setb4xy	15 × 12	350	978	982.00	4.805	778	250	**931**	941.80	118.25	
setb4xyz	15 × 13	350	930	930.50	0.430	651	200	**926**	929.80	62.04	

machines ($n \times m$), and the remaining columns detail the experimental results of the different chosen approaches in terms of the best $C_{\max}$ *(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.

Analysis of the comparative results

By analyzing Tables 2 and 3, it can be seen that our algorithm 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 Heuristic in three out of five instances; and the LEGA, the MOPSO+LS, BBO, and the Hybrid NSGA-II in two out of five instances. In addition, by solving this first data set, our GATS+HM attains the same results obtained by the chosen approaches such as in the case 1 for LEGA, BBO, Hybrid NSGA-II, and Heuristic; in the case 2 for MOPSO+LS and BBO; in the case 3 for LEGA; in the case 4 for all the algorithms; and in the case 5 for MOPSO+LS and Hybrid NSGA-II.

From Tables 4 and 5, the comparison results show that the GATS+HM obtains eight out of ten best results for the Brandimarte instances. Indeed, our algorithm outperforms the TS in nine out of ten instances. Moreover, our GATS+HM outperforms the LEGA and the MATSLO+ in eight out of ten instances. In addition, our hybrid approach outperforms the TS3 in five out of ten instances. For the comparison with the BBO, the GATS+HM obtains the best solutions for the MK02, MK06, and MK10 instances, but it gets slightly worse result for the MK09 instance. Furthermore, the MATSPSO attained the best result for the MK01 instance, but our algorithm obtains a set of solutions better than it for the remaining instances. In addition, our algorithm outperforms the Heuristic in all the Brandimarte instances. By solving this second data set, our GATS+HM attains the same results obtained by some approaches such as in the MK01 for LEGA, MATSLO+ and TS3; in the MK02 for MATSPSO; in the MK03 for TS3, BBO and Heuristic; in the MK04 for BBO; in the MK05 for TS3 and BBO; in the MK07 for BBO and TS3; and in the MK08 for all the algorithms only it is not the case for the Heuristic.

From Table 6, the obtained results show 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.

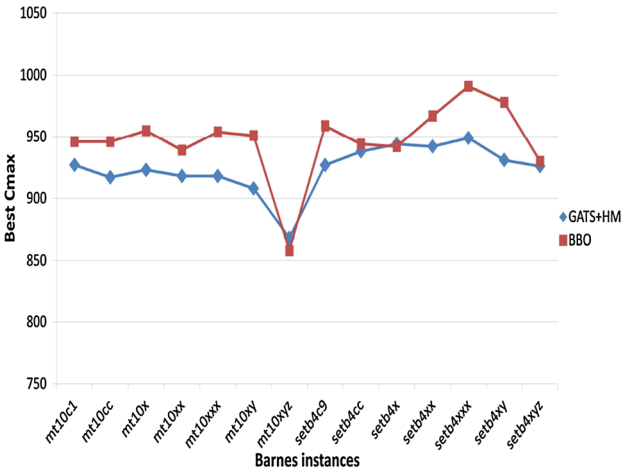

Fig. 10 $C_{\max}$ comparison of GATS+HM and BBO for the Barnes data (Barnes and Chambers 1996)

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+; and in the la05 for the N1-1000 and the MATSLO+.

From Table 7, the results for the Barnes instances demonstrate that our GATS+HM dominates the BBO algorithm in different criteria such as the $C_{\max}$, the Avg $C_{\max}$, the Avg CPU, the deviation, and the population size. In fact, for the $C_{\max}$ criterion, our GATS+HM outperforms the BBO in 12 out of 14 instances, see Fig. 10, with deviations varying from 0.430 to 4.805%. In addition, we attain average values for the $C_{\max}$ solutions dominating the BBO in 12 times. In addition, as shown in Fig. 11, the used population sizes for our algorithm are less than the BBO in all the 14 instances, which influenced on the CPU execution time for each solution, see Fig. 12.

By analyzing the computational time in seconds and the comparison results of our algorithm in terms 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 by applying the intensification technique of the tabu search allowing to start from an elite solution to attain new more dominant solutions.

Conclusion

In this paper, we present a new metaheuristic hybridization algorithm-based clustered holonic multiagent model, called GATS+HM, for the flexible job shop scheduling problem (FJSP). In this approach, a neighborhood-based genetic

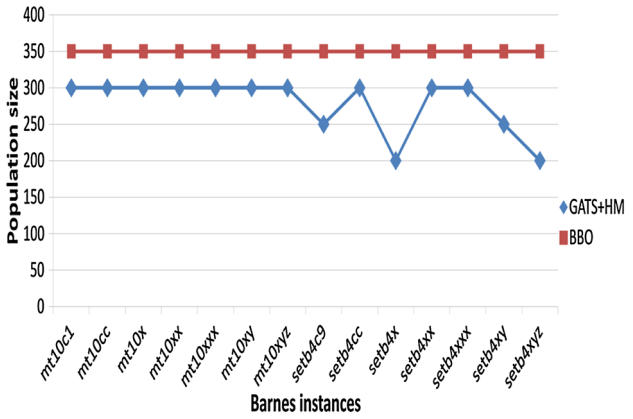

Fig. 11 Population size comparison of GATS+HM and BBO for the Barnes data (Barnes and Chambers 1996)

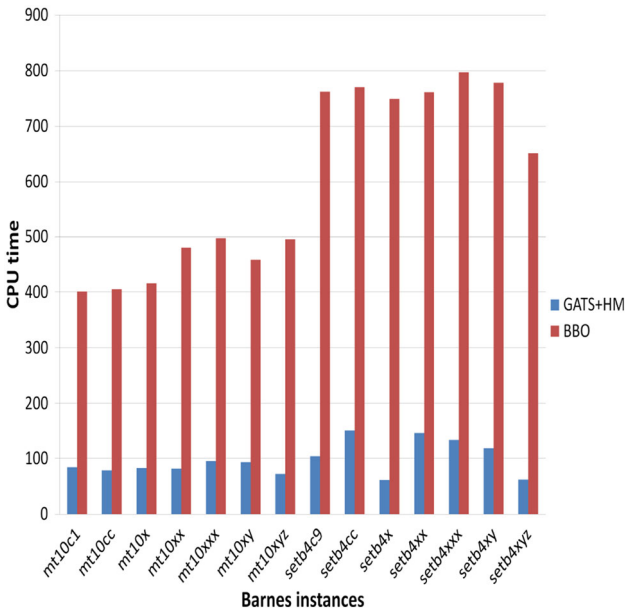

Fig. 12 CPU time comparison of GATS+HM and BBO for the Barnes data (Barnes and Chambers 1996)

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 four well-known data sets in the literature of the FJSP. The experimental results show that the proposed approach is efficient in comparison with others approaches. In the future works, we will search to treat other extensions of the FJSP, such as by integrating new transportation times in the shop process, where each operation must be transported by a moving robot to continue its treatment on its next machine. In addition, this problem can be improved by considering a non-unit transport capacity for the moving robots, where the problem becomes a flexible job shop scheduling problem with transportation times and non-unit transport capacity robots. Therefore, we will plan to make improvements to our approach to adapt it to this new transformation of the problem, and study its effects on the makespan.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This paper does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in this paper.

References

Barnes JW, Chambers JB (1996) Flexible job shop scheduling by tabu search. Tech. rep., Graduate program in operations research and industrial engineering, The University of Texas at Austin

Bellifemine F, Poggi A, Rimassa G (1999) Jade—a fipa-compliant agent framework. In: In Proceedings of the 4th international conference and exhibition on the practical application of intelligent agents and multi-agent technology, pp 97–108

Birgin EG, Feofiloff P, Fernandes CG, Melo ELD, Oshiro MTI, Ronconi DP (2014) A milp model for an extended version of the flexible job shop problem. Optim Lett 8(4):1417–1431

Botti V, Giret A (2008) ANEMONA: a multi-agent methodology for holonic manufacturing systems. Springer series in advanced manufacturing. Springer, Berlin

Bozejko W, Uchronski M, Wodecki M (2010a) The new golf neighborhood for the flexible job shop problem. In: In Proceedings of the international conference on computational science, pp 289–296

Bozejko W, Uchronski M, Wodecki M (2010b) Parallel hybrid metaheuristics for the flexible job shop problem. Comput Ind Eng 59(2):323–333

Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41(3):157–183

Calabrese M (2011) Hierarchical-granularity holonic modelling. Doctoral thesis, Universita degli Studi di Milano, Milano, Italy

Chen H, Ihlow J, Lehmann C (1999) A genetic algorithm for flexible job-shop scheduling. In: In Proceedings of the IEEE international conference on robotics and automation, pp 1120–1125

Dauzère-Pérès S, Paulli J (1997) An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann Oper Res 70:281–306

Ennigrou M, Ghédira K (2004) Approche multi-agents basée sur la recherche tabou pour le job shop flexible. In: 14ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle RFIA, pp 28–30

Ennigrou M, Ghédira K (2008) New local diversification techniques for the flexible job shop problem with a multi-agent approach. Auton Agents Multi-Agent Syst 17(2):270–287

Ferber J (1999) Multi-agent systems: an introduction to distributed artificial intelligence, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston

Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 35(9):2892–2907

Garey MR, Johnson DS, Sethi R (1976) The complexity of flow shop and job shop scheduling. Math Oper Res 1(2):117–129

Giret A, Botti V (2004) Holons and agents. J Intell Manuf 15(5):645–659

Glover F, Kelly JP, Laguna M (1995) Genetic algorithms and tabu search: hybrids for optimization. Comput Oper Res 22(1):111–134

Henchiri A, Ennigrou M (2013) Particle swarm optimization combined with tabu search in a multi-agent model for flexible job shop problem. In: In Proceedings of the 4th international conference on swarm intelligence. Advances in swarm intelligence, pp 385–394

Hmida AB, Haouari M, Huguet M, Lopez P (2010) Discrepancy search for the flexible job shop scheduling problem. Comput Oper Res 37(12):2192–2201

Ho NB, Tay JC, Lai EMK (2007) An effective architecture for learning and evolving flexible job-shop schedules. Eur J Oper Res 179(2):316–333

Hurink J, Jurisch B, Thole M (1994) Tabu search for the job-shop scheduling problem with multi-purpose machines. Oper Res Spektrum 15(4):205–215

Jia H, Nee A, Fuh J, Zhang Y (2003) A modified genetic algorithm for distributed scheduling problems. J Intell Manuf 14(3):351–362

Johnson SC (1967) Hierarchical clustering schemes. Psychometrika 32(3):241–254

Kacem I, Hammadi S, Borne P (2002a) Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Trans Syst Man Cybern 32(1):1–13

Kacem I, Hammadi S, Borne P (2002b) Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Math Comput Simul 60(3–5):245–276

Kar MB, Bera S, Das D, Kar S (2015) A production-inventory model with permissible delay incorporating learning effect in random planning horizon using genetic algorithm. J Ind Eng Int 11(4):555–574

Kia H, Ghodsypour SH, Davoudpour H (2017) New scheduling rules for a dynamic flexible flow line problem with sequence-dependent setup times. J Ind Eng Int 1(1):1–10

Koestler A (1967) The ghost in the machine, 1st edn. Hutchinson, London

Lawrence S (1984) Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Tech. rep., Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania

Lee K, Yamakawa T, Lee KM (1998) A genetic algorithm for general machine scheduling problems. In: In Proceedings of the 2nd IEEE international conference on knowledge-based intelligent electronic systems, pp 60–66

Li J, Pan Q, Suganthan P, Chua T (2011) A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. Int J Adv Manuf Technol 52(5):683–697

Mastrolilli M, Gambardella L (2000) Effective neighbourhood functions for the flexible job shop problem. J Sched 3(1):3–20

Moslehi G, Mahnam M (2011) A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. Int J Prod Econ 129(1):14–22

Oddi A, Rasconi R, Cesta A, 2011 SS (2011) Iterative flattening search for the flexible job shop scheduling problem. In: In Proceedings of the 22th international joint conference on artificial intelligence, pp 1991–1996

Pacino D, Hentenryck PV (2011) Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In: In Proceedings of the 22th international joint conference on artificial intelligence, pp 1997–2002

Pinedo ML (2002) Scheduling: theory, algorithms, and systems. Prentice-Hall, Englewood Cliffs

Rahmati SHA, Zandieh M (2012) A new biogeography-based optimization (bbo) algorithm for the flexible job shop scheduling problem. Int J Adv Manuf Technol 58(9–12):1115–1129

Rezki N, Kazar O, Mouss LH, Kahloul L, Rezki D (2016) On the use of multi-agent systems for the monitoring of industrial systems. J Ind Eng Int 12(1):111–118

Shafigh F, Defersha FM, Moussa SE (2015) A mathematical model for the design of distributed layout by considering production planning and system reconfiguration over multiple time periods. J Ind Eng Int 11(3):283–295

Shahriari M, Shoja N, Zade AE, Barak S, Sharifi M (2016) Jit single machine scheduling problem with periodic preventive maintenance. J Ind Eng Int 12(3):299–310

Sonmez AI, Baykasoglu A (1998) A new dynamic programming formulation of (nm) flow shop sequencing problems with due dates. Int J Prod Res 36(8):2269–2283

Xia W, Wu Z (2005) An effective hybrid optimization approach for multiobjective flexible job-shop scheduling problems. Comput Ind Eng 48(2):409–425

Yazdani M, Amiri M, Zandieh M (2010) Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Syst Appl 37(1):678–687

Zhang C, Gu P, Jiang P (2014) Low-carbon scheduling and estimating for a flexible job shop based on carbon footprint and carbon efficiency of multi-job processing. J Eng Manuf 39(32):1–15

Zhang G, Shao X, Li P, Gao L (2009) An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Comput Ind Eng 56(4):1309–1318

Ziaee M (2014) A heuristic algorithm for solving flexible job shop scheduling problem. Int J Adv Manuf Technol 71(1–4):519–528