Applied Intelligence (2019) 49:1903–1924
https://doi.org/10.1007/s10489-018-1343-7

A novel dynamic assignment rule for the distributed job shop scheduling problem using a hybrid ant-based algorithm

Imen Chaouch[1,2] · Olfa Belkahla Driss[1,3] · Khaled Ghedira[1,4]

Published online: 15 December 2018

Abstract
Distributed scheduling problems are among the most investigated research topics in the fields of Operational Research, and represents one of the greatest challenges faced by industrialists and researchers today. The Distributed Job shop Scheduling Problem (DJSP) deals with the assignment of jobs to factories and with determining the sequence of operations on each machine in distributed manufacturing environments. The objective is to minimize the global makespan over all the factories. Since the problem is NP-hard to solve, one option to cope with this intractability is to use an approximation algorithm that guarantees near-optimal solutions quickly. Ant based algorithm has proved to be very effective and efficient in numerous scheduling problems, such as permutation flow shop scheduling, flexible job shop scheduling problems and network scheduling, etc. This paper proposes a hybrid ant colony algorithm combined with local search to solve the Distributed Job shop Scheduling Problem. A novel dynamic assignment rule of jobs to factories is also proposed. Furthermore, the Taguchi method for robust design is adopted for finding the optimum combination of parameters of the ant-based algorithm. To validate the performance of the proposed algorithm, intensive experiments are carried out on 480 large instances derived from well-known classical job-shop scheduling benchmarks. Also, we show that our algorithm can process up to 10 factories. The results prove the efficiency of the proposed algorithm in comparison with others.

Keywords Ant Colony · Job shop · Multi-factory · Scheduling · Swarm intelligence · Taguchi

1 Introduction

Following [5], scheduling problems can be broadly defined as ”the problems of the allocation of resources over time to perform a set of tasks”. The scheduling literature is full of very diverse scheduling problems [7, 25]. The Job shop Scheduling Problem (JSP) is one of the most important and complex problems in machine scheduling.

The classical JSP is NP-hard in the strong sense [27] and it represents probably one of the most computationally intractable combinatorial problems considered so far. A practical proof of this intractability comes from the fact that a small example with 10 jobs and 10 machines proposed by Muth and Thompson [43] remained open for over 15 years. It was solved in 1980 by Carlier and Pinson [9] as the achievement of a considerable amount of research. A feasible schedule is obtained by permuting the processing order of operations on the machines (operations sequence) without violating the technological constraints. Since each operation sequence can be permuted independently of the operation sequences of other machines, we have a maximum of $(n!)^m$ different solutions to a problem instance, where n denotes the number of jobs and m denotes the number of machines involved [31]. The explosive exponential growth in the number of alternative schedules with the size of the problem increases the difficulty of identifying one of these as the solution of the JSP problem. In the JSP, there are

✉ Imen Chaouch
imen.chaouch@ensi.rnu.tn

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

Khaled Ghedira
khaled.ghedira@isg.rnu.tn

1 COSMOS Laboratory, Université de la Manouba, Manouba, Tunisia

2 Ecole Nationale des Sciences de l'Informatique, Université de la Manouba, Manouba, Tunisia

3 Ecole Supérieure de Commerce, Université de la Manouba, Manouba, Tunisia

4 Institut Supérieur de Gestion de Tunis, Université de Tunis, Tunis, Tunisia

n jobs and *m* machines to be scheduled; each job consists of a predetermined sequence of operations, which have to be processed on a set of m machines, and each operation is characterized by the required machine and the fixed processing time.

The JSP, as a field of research, has displayed exponential growth during the last six decades and a vast amount of literature that has been published in this area. Roy and Sussmann [50] were the first to propose the disjunctive graph representation and Balas [3] was the first to apply an enumerative approach based on the disjunctive graph. Since, various strategies over the years have been applied trying to solve the problem: exact methods, heuristic and meta-heuristic algorithms. Exact methods, such as branch and bound [62], guarantee to find an optimal solution for the problem, but have an exponential computational complexity, so that the time to solve the problem may grow exponentially with its size. Therefore, the use of approximate methods and heuristics is inevitable for solving NP-hard problems. Unlike heuristics, which usually find reasonably ”good” solutions in a reasonable time, approximation algorithms provide provable solution quality and provable run-time bounds. For instance, dispatching rules were adapted in Grabot’s work [14, 60], bottleneck-based heuristics [11, 67], neural networks [64], expert systems and constraint satisfaction [57]. Within the local search category, many methods have been applied to the JSP since its appearance. They include algorithms such as Simulated Annealing (SA) [54], Genetic Algorithms (GA) [2, 28, 48, 63], Tabu Search (TS) [12, 24, 49], ant optimisation [68], scatter search, path relinking (SS& PR) [32] and Invasive Weed Optimization algorithm (IWO) [69]. Swarm intelligence and bio-inspired algorithms were widely used to solve the JSP. Heinonen and Pettersson [30] developed a hybrid approach based on an Ant Colony Optimization algorithm (ACO) and a post-processing algorithm to enhance the ACO performance for solving the JSP. Tasgetiren et al. [59] presented a hybrid method (PSO-VNS) based on the Particle Swarm Optimization (PSO) and the Variable Neighborhood Search (VNS) to improve the solution quality in JSP. To further improve efficiency of PSO, a new hybrid swarm intelligence algorithm (MPSO) consisting of particle swarm optimization, Simulated Annealing (SA) and a multi-type individual enhancement scheme was developed by Lin et al. [38]. Inspired by the decision making capability of bee swarms, Chong et al. [15] explored an evolutionary computation based on Bee Colony Optimization (BCO) to solve the JSP. Yao et al. [65] presented an Improved Artificial Bee Colony algorithm (IABC) to enhance the search performance of the original ABC for solving the JSP.

As can be seen, the JSP problem has been extensively discussed in the literature and has had several extensions [46, 53]. In the majority of studies, we have noticed a common assumption is that all jobs are processed in the same factory which is not always the case when it comes to model real-life problems. In fact, in today’s globalization economy, many companies have turned from traditional single-factory production to multi-factory production to remain competitive and for closer proximity to the market. This will allow companies to save time and reduce costs, hence, respond effectively to customer demands. Consequently, researchers have been increasingly attracted by the distributed scheduling problems which deal with the assignment of jobs to various factories geographically distributed and their scheduling over machines. In this context, the JSP has evolved from the classical version to the Distributed Job shop Scheduling Problem (DJSP) and becomes, increasingly, one of the most important issues to raise.

The DJSP is much more complicated than the JSP since other decisions have to be taken. The difficulty is two fold: first, there is the problem of allocating jobs to suitable factories and second, sequencing the operations on machines with the objective of minimizing one or more predefined performance criteria. Garey et al. [24] proved that the JSP is strongly NP-hard. Hence, the DJSP is ordinarily NP-hard and the case of the simple JSP can be obtained when the number of factories is equal to 1.

In this paper, we propose to solve the DJSP with makespan criterion using a novel Dynamic Assignment method of jobs to factories with a Hybrid Ant Colony Optimization algorithm (DAHACO). Our intention is to increase the diversity of the population and the portion of the search space that is explored. In fact, in the field of DJSP, researches have neglected communication between factories, and all works in the literature have considered the case of static assignment of jobs to factories. In other words, once the job is assigned to a factory, it is frozen and cannot be moved to another one. This limits the search for a better solution and reduces the problem to a classical one that comes to solving a scheduling problem in each factory separately and neglects the overall aspect of the problem. This is why we propose to develop a novel dynamic assignment method which apply a first allocation of job to factories using the workload method introduced in [44], then improve the results generated by applying numerous jobs-permutations between factories in order to have several combinations of assignments. These combinations will serve to broaden the initial population for the algorithm and offer a large research space to explore. Our proposed algorithm is combined with a local search in order to reduce the idle time of machines and improve the solution.

So, this study is the first attempt in the literature to solve the DJSP with a dynamic assignment of jobs to the factories.

A comparative study between static assignment rule and dynamic one is presented. Furthermore, we investigate on the influence of the different parameters by using the Taguchi method, and the best parameter setting is suggested. 480 benchmark instances have been used to evaluate the performance of the proposed DAHACO. The experimental results prove that the proposed DAHACO algorithm is both effective and efficient. In essence, the contributions of this paper are threefold as follows:

- It proposes a novel assignment rule of jobs to factories in distributed environment;
- It proposes a hybrid ant-based algorithm for solving the DJSP;
- It adopts the Taguchi method for parameter design in determining the optimum level of parameters used in the algorithm.

The rest of the paper is organized as follows. In Section 2, the distributed job shop scheduling problem is formally defined. Then, a comprehensive literature review of the problem is provided in Section 3. In Section 4, the original framework of the ant colony algorithm is presented in detail. Section 5 introduces the proposed DAHACO to solve the DJSP with makespan criterion. The Taguchi Method is introduced in Section 6. Parameter setting and numerical testing results are provided in Section 7. Conclusions are finally drawn in Section 8, along with recommendations for future researches.

2 The distributed job shop scheduling problem

A typical Distributed Job shop Scheduling Problem (DJSP) can be stated as a set of n jobs, which have to be processed on a set of m machines geographically distributed on f identical factories. The main objective of the DJSP is to find an optimal scheduling minimizing a specified criterion which is generally time related such as makespan, maximum tardiness or total tardiness. The most widely adopted criteria among literature is minimizing the makespan which can be set as the maximum of completion times needed for processing all jobs. Distributed Scheduling problems in multi-factory production are much more complicated than classical scheduling problems since two decisions have to be taken: assign jobs to suitable factories and generating a feasible scheduling while minimizing a predefined performance criterion.

A job consists of a set of operations which have to be processed on machines in a predetermined order. These operations are subject to various assumptions in the DJSP. All factories should be able to process all jobs and once a job is assigned to a factory, it cannot be moved to another one until all its operations are proceeded in the same plant. At the starting time, all jobs and machines are available. Jobs are independent and there are no precedence constraints among the operations of different jobs. A job can be processed by at most one machine at a time and a machine can process at most one job at a time. It is assumed that every job has to be processed on every machine exactly once and neither the release times nor due dates are specified. Each operation is characterized by the required machine and the fixed processing time. Setup times of machines and transit times between operations are negligible in our problem.

As discussed in the introduction, the DJSP is strongly NP-hard since we have more decisions to be taken and the global objective becomes the minimization of the makespan among several factories. For clarity's sake, we give an illustrative example of DJSP representation. Given an instance with six jobs, two machines and two factories, the processing times of the jobs on each machine is given in Table 1 (let us remind that factories are identical). A Distributed Job shop Problem instance can be visualized by a directed graph $D = (N, A, E)$, where N represents the set of nodes corresponding to all operations O processed in the same factory, A the set of conjunctive directed arcs, based on the precedence rules and E the set of disjunctive directed edges that connect two operations from two different jobs executed on the same machine and factory. Figure 1 shows the disjunctive graph of the example and a feasible Gantt chart of this problem is shown in Fig. 2. Makespan in factory 1 is 8 and makespan in factory 2 is 10, leading to the conclusion that the makespan of this DJSP is equal to the maximum makespan between the two factories, which is 10.

3 Related works

In recent years, considerable attention has been given to the DJSP, but the literature is still relatively limited since the issue is recent. So far, the methods that have dealt with the DJSP can be divided into two categories: exact

Table 1 Processing time matrix

Job	Machine		Processing route
	1	2	
1	2	3	{1,2}
2	1	4	{2,1}
3	4	2	{2,1}
4	2	2	{1,2}
5	3	5	{1,2}
6	3	2	{2,1}

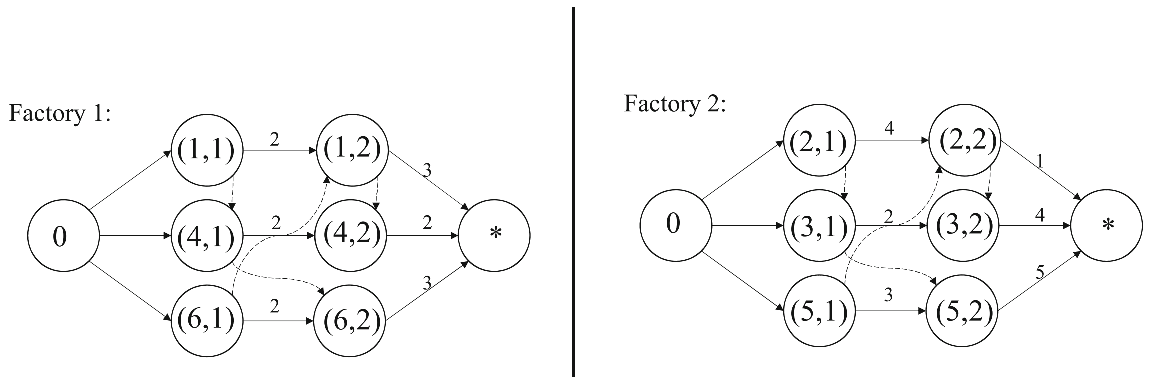


Fig. 1 Directed graph representation for the DJSP with $f = 2; n = 6$ and $m = 2$

methods and metaheuristics. For the exact methods, we can find [44] that have mathematically formulated the DJSP with two different Mixed Integer Linear Programming models (MILP). The first model dealt with the problem as a sequencing decision while the second one dealt it as a positional one. The authors have proposed another MILP model in their next paper [45]. In order to evaluate the developed MILP models, the authors used two performance measures: size and computational complexities. All other studies applied metaheuristics to solve the DJSP. Indeed, Jia et al. [34] presented the first attempt to solve the problem. In their study, they proposed a web-based system to enable production scheduling with the utilization of the World Wide Web technology, in order to facilitate collaboration between geographically distributed plants. A Genetic Algorithm (GA) approach was adopted to deal with the distributed scheduling problems. In their next paper, authors in [36] presented a Modified Genetic Algorithm (MGA) in which a two-step encoding method was used. The first one to encode the factory candidates and the second one, to affect jobs and operations. To evaluate the performance of their MGA, the authors used benchmark instances (10 jobs/10 machines and 20 jobs/5 machines) proposed by [43]. In fact, MT1963 are instances for simple job shop problems which are adapted to the DJS by distributing different jobs on the factories. Later, Jia et al. [35] refined their previous approach and proposed a GA integrated with Gantt Chart (GC) to derive the factory combination and schedule. The experimental results showed that the application of the GC is able to facilitate the chromosome evaluation procedures and thus improve the computational performance algorithm. Recently, in [44], three well-known heuristics were applied; these are Shortest Processing Time first (SPT), Longest Processing Time first (LPT) and Longest Remaining Processing Time (LRPT). Finally, three Greedy Heuristics have been developed and adopted to solve the problem (GH1, GH2 and GH3). The performance of the two proposed mathematical models and six heuristics (SPT, LPT, LRPT, GH1, GH2 and GH3) are evaluated and tested. With regards to the obtained optimal solutions, it is concluded that the developed GH3 performs better than the other algorithms. In their next paper [45], the authors applied three different versions of Simulated Annealing (SA) and designed two different local search

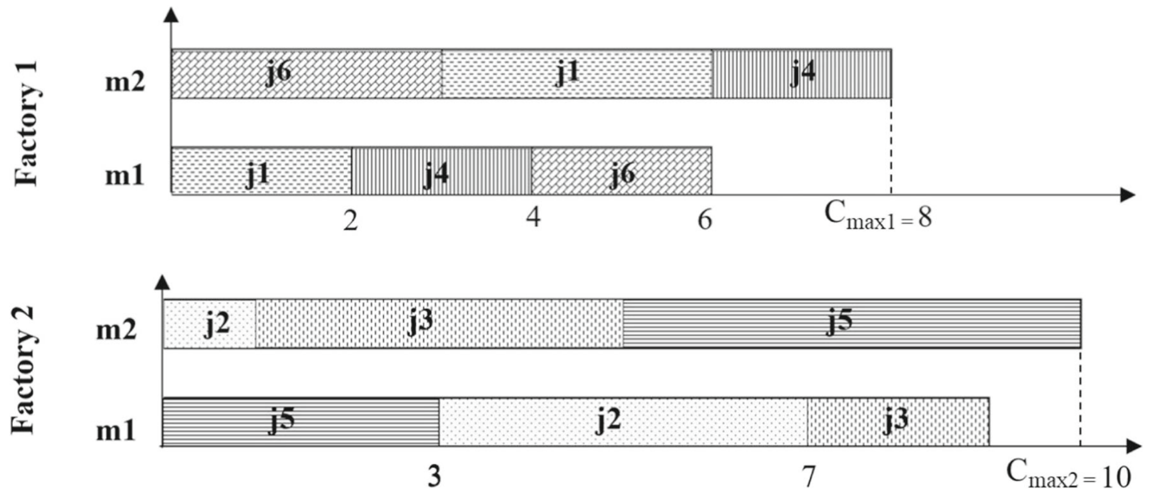


Fig. 2 Gantt Chart of the DJSP with $f = 2; n = 6$ and $m = 2$

engines to improve operation sequence for the factories. The first local search aims to decrease the makespan and the second one aims to increase the makespan. The first version of the proposed algorithms, called SA, is implemented without any local search. The second one, called hybridized simulated annealing (HSA), is hybridized with local search type 1 which assumes that the job number is first inserted into m random positions. Then, the position of one randomly selected job number is shifted to a random position. Finally, the third version, called greedy simulated annealing (GSA), employs the greedy local search 2 which assumes that job number is added into permutation one by one. To evaluate the proposed algorithms, three sets of instances were generated. The first set is for parameter tuning, the second one is for the experiment with small instances, and finally, the last one is for the experiment with larger instances. The solutions proposed in this article obtained promising results and outperformed the other tested algorithms. Chaouch et al. [10] surveyed the literature related to the DJSP. They presented papers dealing with the problem and a classification of the employed techniques is well established. As we can see, researchers have in recent years attempted to resolve the DJSP using various methods and producing different results compared to each others. Hence, trying to apply new and leading edge approaches is a growing challenge in order to find better results. Table 2 summarizes all papers dealing with the DJSP and provides a clear classification of them in term of year of publication, objective function, employed techniques for resolution and scheduling type (dynamic or static). In the next section, we present an overview of ant colony optimization algorithm as well as the main motivation to employ it as a solution for the DJSP with makespan criterion.

4 Ant colony optimization

Imitating natural evolutionary processes of living beings to solve hard optimization problems have received a great interest which gave rise to Evolutionary Algorithms (EAs) in the late 1960s. Since that, we witnessed the emergence of many metaheuristic algorithms belonging to EAs that can often outperform classical optimization methods when applied to difficult real world problems. Swarm Intelligence is a collection of nature-inspired algorithms under the big umbrella of evolutionary computation [47], among them, Ant Colony Optimization (ACO) [19] is perhaps the most widely known type of swarm intelligence algorithms used. Marco Dorigo and colleagues introduced the first ACO algorithms in the early 1990s [19, 21, 22]. Then, several studies have applied the ACO to solve different problems, such as Traveling Salesman Problem [20], vehicle routing [26], quadratic assignment problems [42], graph coloring [23] and scheduling problems [6]. Ant colony-based algorithms have been shaped by a number of people who have made valuable contributions to the development of the field. Table 3 summarizes some of the proposed ACO algorithms in the literature.

Table 3 Some of the most used ACO algorithms

Algorithm	Source
Ant System (AS)	[16]
ASElite (EAS)	[22]
ASRank	[8]
Ant Colony System (ACS)	[20]
Max–Min Ant System (MMAS)	[52]
Best–Worst Ant System (BWAS)	[18]

The inspiring source of ACO algorithms was the observation of the collective behaviour of ants and more specifically, the ants' foraging behaviour. When searching for food, ants are able to find the shortest path between a food source and their nest according to their indirect communication by means of chemical pheromone trails. Initially, ants follow different paths surrounding their nest in a random way. As soon as an ant finds a food source, and while walking from food sources to the nest, ant lays down a chemical pheromone on the ground, forming a pheromone trail which guides other ants towards best path. When

Table 2 Classification of distributed job shop scheduling problem papers

Paper	Proposed method	Objective function	Scheduling type		Method	
	•	•	Static	Dynamic	Exact	Approx.
[34]	GA	Multiple criteria	✓	–	–	✓
[36]	GA	Multiple criteria	✓	–	–	✓
[35]	GA	Multiple criteria	✓	–	–	✓
[44]	MILP + Heuristics	Minimize Makespan	✓	–	✓	✓
[45]	MILP + SA	Minimize Makespan	✓	–	✓	✓
[10]	Survey	–	–	–	–	–

choosing their way, ants tend to choose, probabilistically, paths marked by strong pheromone concentrations which may depend on the quantity and quality of the food. In the literature, several studies have considered the ACO algorithm in the resolution of the Job shop Scheduling Problem and led to good results.

In a paper by Colorni et al. [17] Ant System (AS) was applied to job shop scheduling problem and proved to be a noteworthy candidate when faced with the task of choosing a suitable algorithm for scheduling problems. The paper concluded that the AS is one of the most easily adaptable population-based heuristics so far proposed and that its computational paradigm is indeed effective under very different conditions. Lu and Romanowski [39] have also achieved good results when applying the ACO to the dynamic job shop problem. As an example of ACO robustness, Jayaraman et al. [33] used an ACO algorithm in solving a combinatorial optimization problem of multiproduct batch scheduling as well as the continuous function optimization problem for the design of multiproduct plant with single product campaigns and horizon constraints. Further real world applications with regard to ACO algorithms would be using ACO to solve an established set of vehicle routing problems as done by Bell and McMullen [4] and a dynamic regional nurse-scheduling problem in Austria by Gutjahr and Rauner [29]. The former paper concluded that the results were competitive and in the latter paper ACO was compared to a greedy assignment algorithm and achieved highly significant improvements. Ying et al. [66] applied the ant colony system to permutation flow-shop sequencing and effectively solved the problem *n/m/P/Cmax*, and commented that the ant colony system metaheuristic is well worth exploring in the context of solving different scheduling problems.

Motivated by the effectiveness of the ant algorithms in solving different kinds of optimization problems, our approach, presents in the first part, a novel dynamic assignment method of jobs to factories, then, the ant colony optimization algorithm combined with a local search is applied to solve the Distributed Job shop Scheduling Problem.

5 The proposed DAHACO algorithm for the DJSP

In the field of DJSP, researches have overlooked the communication between factories which is a fundamental aspect to consider when solving distributed scheduling problems. Allocating jobs to suitable factories have a significant influence over the quality of the scheduling and so on the value of the makespan. This is why we propose to develop, in the assignment phase, a novel dynamic assignment method which apply the workload method introduced in [44], then improve the results generated by applying numerous jobs-permutations between factories in order to have several combinations of assignments. These combinations will serve to broaden the initial population for DAHACO and offer a large research space to explore.

Our choice to develop an ant-based algorithm is highly motivated by the distributed aspect of the ACO algorithm as we are dealing with a distributed scheduling problem. The coordinated behaviour of real ant is exploited to organize populations of artificial agents, which are artificial ants, that communicate with each other to solve computational problems. If we take a closer look, we can find several similarities between the foraging process of ant colonies and the DJSP. Consider the operations as ants, they must find the best order of processing on the machines minimizing the makespan. By analogy, ants want to search the best path for food while minimizing the length of the path. The start and dummy operation are like an ants' nest and a food source, respectively. And finally, the link of any two operations can be seen as an alternative route for ant when considering an operation as an ant's path for foraging food. Our proposed DAHACO is combined with a local search in order to reduce the idle time of machines. In fact, despite the performance of ant colony algorithms, they can easily be trapped in a local optimum due to their stochastic aspects (for example, random solution construction). This is what leads us to think that implementing a local search procedure to improve the solution is strongly recommended, and this will be proven in the experimental part.

In order to adapt the ant colony optimization algorithm to solve the DJSP, a number of elements have to be defined:

- The way in which jobs should be assigned to factories
- Sequencing of jobs assigned to each factory

5.1 Job factory assignment phase

An important step in solving the Distributed Job shop Scheduling Problem is the allocation of jobs to factories. The objective is to partition jobs on factories so as to equilibrate the workloads. In other words, the aim is to have near values of workloads between factories. The job factory assignment phase will be divided in two steps. In the first one, we will use the job-facility assignment rule introduced in [44] with the static assignment of jobs: once jobs are allocated, they cannot be moved from one factory to another. Then, we propose a novel dynamic assignment rule allowing movement of jobs which has proved its effectiveness in most cases.

5.1.1 Workload rule

Let us consider n jobs that will be assigned to f factories. The workload on each machine is separately calculated using the following rule [44], which is defined for each job j on each machine i as follows:

$$workload(j,i) = \left(\sum_{k \in R_{j,i}} p_{j,k} \right) + p_{j,i} \quad , \forall_{i,j} \tag{1}$$

Where $R_{j,i}$ is the set of all machines preceding machine i in the processing of job j and $P_{j,i}$ is the processing time of job j on machine i. The workload of each operation is calculated and regarding the total workloads, the jobs are ranked in descending order, from highest workload to the lowest ones. The f first jobs are assigned to factories $1 \ldots f$, respectively. The workload of machines in different factories becomes equal to those of the assigned jobs and the maximum workload in the f factories is determined. To assign the next job, the maximum workload is recalculated if the job is assigned to a factory. All the possibilities should be enumerated and the workload is calculated at each time. Then, the job is assigned to the factory with a minimum of the maximum workload. The procedure repeats for subsequent jobs until all jobs are assigned.

5.1.2 The proposed assignment procedure

The Workload Rule method [44] proved to be efficient to well equilibrate workloads in different factories. But the main disadvantage of this method is that jobs are allocated in a static way, once they are assigned to factories, they can't be moved from one factory to another and only one assignment combination is explored.

This is what prompted us to think about making the assignment phase dynamic by allowing jobs to move from one factory to another based on the workload method. Our proposed approach takes the result of the workload assignment and performs a set of jobs permutations between factories while maintaining a certain balance in terms of the number of jobs in each factory. Our approach performs better for large instances with large number of jobs and machines. It has allowed to generate new combinations, thus exploiting new possibilities. To illustrate this concept, it is applied to Tai15x15 instance from well-known benchmarks for the Job shop Scheduling proposed by Taillard [55]. The two methods are implemented and ran 100 times, on Tai15x15 with 5 factories, 15 machines and 15 jobs. Figure 3 shows the obtained results and as we can see, in almost all cases, our proposed dynamic assignment method gives better values of makespan.

5.2 Sequencing phase

Once all jobs are assigned to their corresponding factory, they need to be sequenced. To do this, a hybrid ant-based algorithm is proposed aiming to explore more search space and potentially finding better possible solutions for the problem. In nature, ants are able to find the shortest path between a food source and their nest according to their collective behaviour. In DJSP, the aim is to find the best path giving the minimum makespan among all possible paths.

In this section, the main steps of our proposed algorithm DAHACO will be detailed. The basic principle of the DAHACO is to improve the solution generated by the Ant Colony System (ACS) by applying a local search procedure in order to explore more search space and found better results. The main steps of proposed DAHACO are listed below:

Step 1: At the beginning of the algorithm, all parameters are initialized including the initial pheromone value of edges $\tau_0 > 0$. Initially, ants choose their paths randomly and therefore, search the solution space more effectively.

Step 2: Ants are placed on the first operation of each job.

Step 3: This procedure consists of a probabilistic construction of solutions by all the ants according to the State Transition Rule as follows:

$$P(i,s)(t) = \begin{cases} \dfrac{[\tau_{i,s}(t)]^{\alpha} \times \left[\frac{1}{d_{i,s}}\right]^{\beta}}{\sum_{j \in AllowedNodes} [\tau_{i,j}(t)]^{\alpha} \times \left[\frac{1}{d_{i,j}}\right]^{\beta}} \\ 0, \; otherwise \end{cases} \tag{2}$$

With:

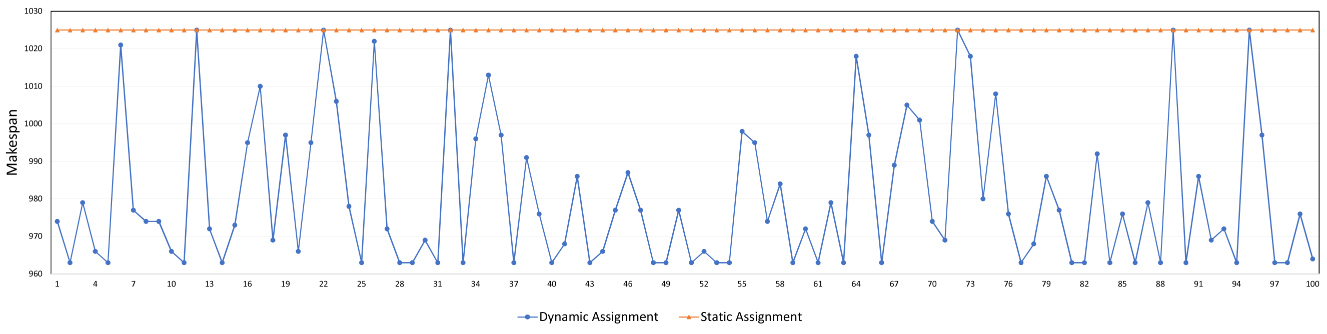


Fig. 3 Comparison between static and dynamic assignment

- $\tau_{i,j}$ quantity on pheromone between the $node_i$ and $node_j$
- $d_{i,j}$ heuristic distance between $node_i$ and $node_j$. In our case, $d_{i,j}$ is the processing time of the operation.
- $P(i,s)$ probability to branch from $node_i$ to $node_s$
- The parameters α and β tune the relative importance in probability of the amount of pheromone versus the heuristic distance.

The probability for an ant to choose the next operation is directed by both the amount of pheromone on the route and heuristic distance from its current position to the next one. Artificial ants can be considered as stochastic greedy procedures that construct a solution in a probabilistic manner by adding solution components to partial ones until a complete solution is derived [56]. In the general ACS, the set of next operations for an ant in $node_i$ is all not visited nodes. Which is not the case in the DJSP, choosing the next operation should respect the operation precedence constraints. Therefore, for each transition from a $node_i$ to $node_j$, the ant has to build its allowed list containing the operation that can visit.

Step 4: While constructing its solution, an ant will modify the amount of pheromone on the visited edges by applying the local updating pheromone rule:

$$\tau_{i,s}(t+n) = (1-\rho) \times \tau_{i,s}(t) + \rho \times \tau_0(t+n) \quad (3)$$

Where ρ is the coefficient representing pheromone evaporation ($0 < \rho < 1$). The purpose of the local pheromone update rule is to make the visited edges less and less attractive as they are visited by ants, indirectly favouring the exploration of not yet visited edges. As a consequence, ants tend not to converge to a common path.

Step 5: Once all ants have generated a solution, the global updating rule (7) is applied in two phases:

- An evaporation phase where a fraction of the pheromone evaporates and decreases automatically, so as to diversify the search procedure into larger solution spaces.
- A reinforcement phase where each ant deposits an amount of pheromone which is proportional to the generated solutions

$$\tau_{i,s}(t+n) = (1-\xi) \times \tau_{i,s}(t) + \xi \times \Delta\tau_{i,s}(t+n) \quad (4)$$

$$\Delta\tau_{i,s}(t+n) = \frac{Q}{BestC_{max_{ant}}} \quad (5)$$

Here: $0 < \xi < 1$ is the pheromone decay parameter and Q is a constant. The global updating pheromone is only applied on the best solution found by the algorithm. The process is repeated until a termination condition has been reached which is a fixed number of iterations in our case.

Step 6 : local search procedure Once all ants have generated a solution, a local search procedure is applied in order to explore more search space and found better result if it is possible. First of all, the last scheduled job over all machines is determined and selected to make the first improvement moves. The method consists of finding all inactive time intervals of machines and tries to insert the operations of the selected job in those intervals while respecting DJSP constraints. To place the operation on the inactive interval, the following constraints should be verified:

- The end time of the selected operation > End time of the inactive interval
- Processing time of the selected operation ≤ The length of the interval
- Precedence constraints of the DJSP = True

The main steps of the proposed method are represented in the following diagram, cf. Figure 4 and the two pseudo-codes below:

Algorithm 1 DAHACO algorithm

```
1: for each distribution do
2:     Workload Procedure
3:     Dynamic Assignment Rule of jobs to factories
4:     for each generated combination do
5:         Set parameters, initialize pheromone trails
6:         for each iteration do
7:             Position each ant in a starting node
8:             repeat
9:                 for each ant do
10:                    Choose next node by applying the state transition rule
11:                    Apply step by step pheromone update
12:                end for
13:            until every ant has built a solution
14:            Update best solution
15:            Apply Global updating of pheromone
16:        end for
17:        Local Search Procedure
18:    end for
19: end for
```

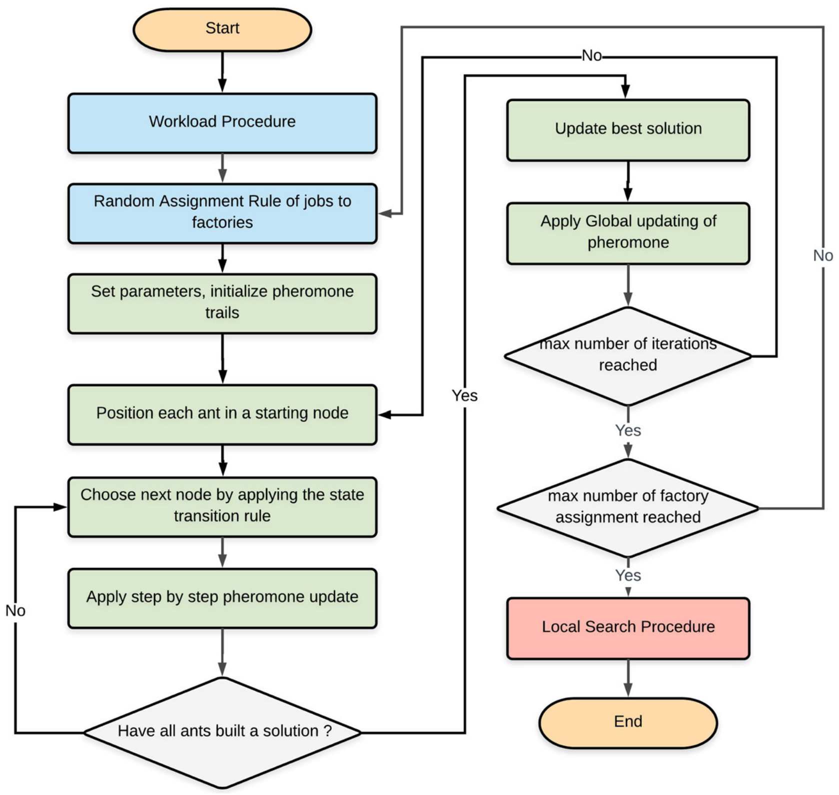

Start
Workload Procedure
Random Assignment Rule of jobs to factories
Set parameters, initialize pheromone trails
Position each ant in a starting node
Choose next node by applying the state transition rule
Apply step by step pheromone update
Have all ants built a solution ?
No
Yes
Update best solution
Apply Global updating of pheromone
max number of iterations reached
No
Yes
max number of factory assignment reached
No
Yes
Local Search Procedure
End

Fig. 4 The flowchart of the proposed DAHACO

Algorithm 2 Local search procedure

```
    Determine inactive time interval of machines
2:  if ((EndTime of the selected operation > EndTime of the inactive interval) and (Processing time of the selected operation ≤ The interval))
    or ((Processing time of the selected operation > The interval) and (BeginTime of the inactive interval = EndTime of the selected operation)) and (Precedence constraints = True) then
4:      Place the operation
    end if
```

In order to show the relevance and the influence of each implemented part of the algorithm, namely the dynamic assignment rule and the local search procedure, three versions are implemented, cf. Figure 5. In the next section, we conduct extensive computational experiments to analyse and validate the efficiency of our model.

6 Robust parameter design - parameter tuning

The time required to complete an experiment is extremely long, especially, for evaluating large numbers and various

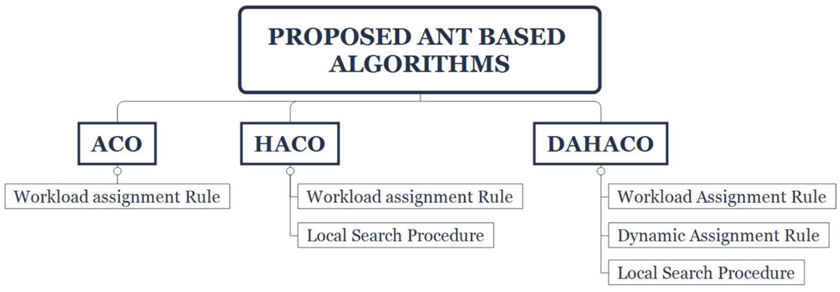


Fig. 5 A diagram outlining the 3 implemented versions

Table 4 Parameters and their levels

Parameter	α	β	ρ	Q
Level 1	0	0	0.3	1
Level 2	0.5	1.0	0.5	100
Level 3	1.0	2.0	0.7	1,000
Level 4	2.0	3.0	0.9	5,000
Level 5	5.0	5.0	0.999	10,000

factors (parameters). The difficulties are even more complicated when experiments have to be repeated for several times until accurate and validated result is obtained. Considering these difficulties, Dr. Genichi Taguchi has developed a new experimental strategy, Taguchi method, for an experiment in the late 1940s [37, 51]. The application of Taguchi method has attracted more attention in the literature for the past 20 years and nowadays the Taguchi method has been widely applied to various fields, such as manufacturing systems [41]. Taguchi method is a powerful design of experiments which provide effective and efficient approach to optimize designs for performance, quality and cost. The most important difference between classical experimental design and Taguchi method is that the former tends to focus solely on the mean of the quality characteristic while the latter considers the minimisation of the variance of the characteristic of interest. In the present research, Taguchi method is used for parameter design. The purpose is to determine the factors (parameters) and their settings that will provide the best result. The two major tools used in Taguchi methodology are:

– Orthogonal arrays which accommodate many design factors (parameters) simultaneously
– Signal-to-noise ratio (S/N ratio) which measures quality with emphasis on variation

Using Orthogonal Array technique significantly reduces the number of experiments and a more reliable estimation of the effect of parameters can be obtained. The signal to noise ratio provides a measure of the impact of noise factors on performance. The larger the S/N, the more robust

Table 5 L_{25} Orthogonal array

Experiment No.	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2	3	4	5	1	2
9	2	4	5	1	2	3
10	2	5	1	2	3	4
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3	3	5	2	4	1
14	3	4	1	3	5	2
15	3	5	2	4	1	3
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4	3	1	4	2	5
19	4	4	2	5	3	1
20	4	5	3	1	4	2
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5	3	2	1	5	4
24	5	4	3	2	1	5
25	5	5	4	3	2	1

the product is against noise. The S/N ratio is expressed in decibel (dB) units. Based on the type of performance characteristics, different categories of S/N ratios have been defined. Three main categories are: nominal-the-better, smaller-the-better, and larger-the-better. In this study, smaller-the-better (STB) is considered to minimize the objective function value. For this case, the corresponding loss function for the objective of smaller-the-better can be expressed as follows:

$$L_{STB} = RPD \tag{6}$$

where,

- L_{STB} is the loss function for smaller-the better
- RPD is the Relative Percentage Deviation used as performance measure in this study. It can be calculated as follows:

$$RPD = \frac{Alg - Min}{Min} \times 100 \tag{7}$$

where: Alg is the makespan obtained by a given algorithm alternative on a given instance and Min is the lowest makespan obtained for the same instance. The objective is to maximize the S/N ratio. For minimization objectives, the S/N ratio is:

$$S/Nratio = -10 \times log_{10}(RPD)^2 \tag{8}$$

An analysis is carried out to identify which levels of the parameters are used which brings us closer to the desired value (Orthogonal arrays) and which maximize the S/N ratio. Taguchi methodology for parameter design can be applied in other algorithms such as genetic algorithm [13], particle swarm optimisation [1], and artificial bee colony algorithm [61] for NP-hard problems. They provide a cost-effective way of studying many factors in one experiment, at the expense of ignoring some high-order interactions. This is considered to be low risk, as high order interactions are usually insignificant and difficult to interpret anyway [58].

7 Experimental evaluation

Experimental results are conducted over two phases. We first performed parameter tuning using the Taguchi method

Table 6 Modified L_{25} Orthogonal array

Experiment No.	α	β	ρ	Q
1	0	0	0.3	1
2	0	1	0.5	100
3	0	2	0.7	1,000
4	0	3	0.9	5,000
5	0	5	0.999	10,000
6	0.5	0	0.5	1,000
7	0.5	1	0.7	5,000
8	0.5	2	0.9	10,000
9	0.5	3	0.999	1
10	0.5	5	0.3	100
11	1	0	0.7	10,000
12	1	1	0.9	1
13	1	2	0.999	100
14	1	3	0.3	1,000
15	1	5	0.5	5,000
16	2	0	0.9	100
17	2	1	0.999	1,000
18	2	2	0.3	5,000
19	2	3	0.5	10,000
20	2	5	0.7	1
21	5	0	0.999	5,000
22	5	1	0.3	10,000
23	5	2	0.5	1
24	5	3	0.7	100
25	5	5	0.9	1,000

Table 7 Average of response and S/N ratio for different experiments

Experiment No.	Average RPD	S/N Ratio
1	2,59	−8,25
2	2,50	−7,97
3	2,86	−9,12
4	2,83	−9,02
5	1,28	−2,13
6	1,99	−5,99
7	1,42	−3,07
8	2,10	−6,43
9	1,62	−4,17
10	1,72	−4,71
11	1,15	−1,19
12	2,40	−7,62
13	4,28	−12,63
14	2,31	−7,26
15	2,33	−7,33
16	2,21	−6,89
17	1,99	−5,99
18	2,34	−7,38
19	2,77	−8,84
20	2,72	−8,71
21	2,05	−6,22
22	2,41	−7,64
23	2,22	−6,94
24	1,36	−2,64
25	2,04	−6,19
Average	2,22	−6,57

and then experiments were conducted on well-known benchmarks with different level of f, proposed by Taillard [55]. The results described in the following sections have been obtained on a personal computer with 3.4 GHz Intel Core i7 and 8 GB of RAM memory.

7.1 Parameter design for DAHACO using Taguchi method

The choice of parameter values for metaheuristics highly influences the performance. Thus, an experiment is first conducted to fine tune the parameters of the proposed DAHACO. The algorithm includes four parameters, namely, α, β, ρ, and Q. In this work, five levels or values are considered for each parameter applied to Tai01 (15 jobs, 15 machines) proposed by Taillard [55]. The levels tested in this study are fixed based on Dorigo et al. [22] and are shown in Table 4. The number of ants is taken as 100 and the number of iterations in a cycle is fixed at 500. When five levels for each of the four parameters are considered, then the total number of experiments to be carried out for finding the optimum level of parameters results in 4^5 experimental settings, i.e., 1,024. If each experiment is repeated five times, the total number of experiments required is 5,120. Obviously, this design becomes inefficient since it requires a large number of required trials. By applying the Taguchi method for robust design, the total number of experiments needed can be reduced to 125, i.e., 25 experiments, each with five replications as shown in Table 5.

This experimental design of Taguchi is based on the technique of matrix experiments [40]. In matrix experiments, there is a set of experiments where the user changes the settings of various parameters. Orthogonal arrays are used to study the effect of various factors efficiently. The columns of orthogonal array are pairwise orthogonal, i.e., for every pair of columns, all combinations of factor levels occur an equal number of times. The columns represent the parameters to be studied and the rows represent the individual experiments. The number of rows gives the total number of experiments and the number of columns gives the

Table 8 Results of level average response analysis

Experiment No.	α	β	ρ	Q
Level 1	− 3,65	− 2,88	− 3,52	− 3,57
Level 2	− 2,44	− 3,23	− 3,71	− 3,48
Level 3	− 3,63	− 4,25	− 2,50	− 2,88
Level 4	− 3,78	− 3,19	− 3,62	− 3,30
Level 5	− 2,96	− 2,91	− 3,14	− 2,65
Max S/N	− 2,44	− 2,88	− 2,50	− 2,65
Min S/N	− 3,78	− 4,24	− 3,71	− 3,56
Range	1,34	1,36	1,20	0,91
Rank	2	1	3	4

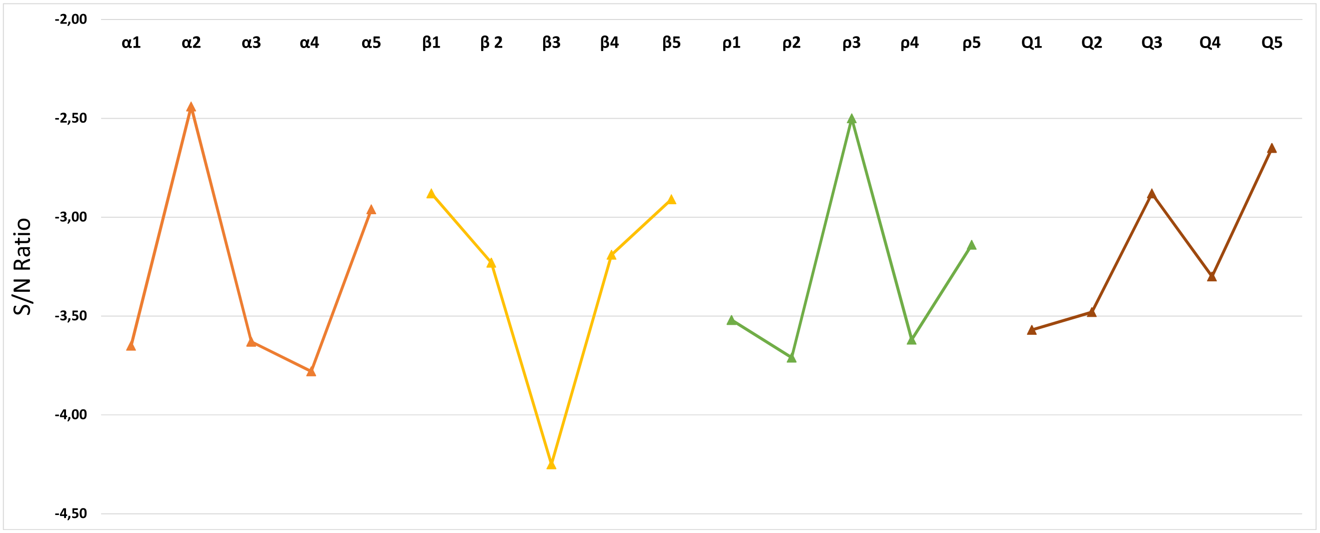


Fig. 6 Graphical representation of the effect of parameters

maximum number of parameters that can be studied using the orthogonal array. Based on the number of parameters to be analysed and their levels, orthogonal array L_{25} is selected in the present study. The assigned experimental array is shown in Table 6.

From Table 7, the overall mean for S/N ratio is found to be -6,57 dB. The level average response analysis is based on averaging the experimental results achieved at each level for each parameter. In the present study, each level of the parameter is contained in five experiments. The level-average response analysis is carried out by calculating the average of the results (i.e., S/N ratio) from the five experiments corresponding to each level of the parameter as shown in Table 8. Figure 6 shows graphically the effect of the four parameters on the objective function value. Since S/N ratio has the characteristic of the larger the better, the analysis of the results leads to the conclusion that the best combination of each parameter is α at level 2, β at level 1, ρ at level 3, and constant Q at level 5. The corresponding values are as follows: $\alpha_2 = 1$, $\beta_1 = 0$, $\rho_3 = 0.7$, $Q_5 = 10000$. From the responses for the experiments, the range of S/N ratio is calculated as follows:

$$Range = max\ S/N - min\ S/N \tag{9}$$

where *max S/N* is the maximum of S/N ratio and *min S/N* is the minimum of S/N ratio for the parameter considered. A ranking of parameters is made based on the range values. It can be seen that parameter β has the largest effect on the response of the experiment and parameter Q has the least effect.

7.2 Confirmation experiment

The purpose of the confirmation experiment is to validate the conclusions drawn from the experiments. Once the optimal level of the parameters is selected, the final step is to predict and verify the improvement of the performance characteristics using the optimal level of the parameters. The estimated value of the S/N ratio $\eta_{predicted}$ with the optimal levels of parameter combination can be calculated with the help of following prediction equation:

$$\eta_{predicted} = \eta_m + (\eta_{\alpha_2} - \eta_m) + (\eta_{\beta_1} - \eta_m) + (\eta_{\rho_3} - \eta_m) + (\eta_{Q_5} - \eta_m) \tag{10}$$

Table 9 Confirmation Experiment data

Parameter combination	S/N ratio
Predicted value $\eta_{predicted}$	– 1,08
Actual value η_{actual}	– 1,19

Table 10 The three implemented algorithms

	Workload assignment	Local search	Dynamic assignment
ACO	+	–	–
HACO	+	+	–
DAHACO	+	+	+

Table 11 Performance comparison of DAHACO, HACO and ACO

Problem	size	2F			3F			4F			5F			6F			7F		
		DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO
Tai01	15 × 15	**1381**	1433	1708	**1141**	1141	1369	**1047**	1053	1154	**977**	996	1025	**963**	963	963	**963**	963	963
Tai02	15 × 15	1213	**1195**	1570	**1138**	1138	1264	**1033**	1112	1051	**987**	1043	1012	**942**	999	986	**942**	942	999
Tai03	15 × 15	**1376**	1484	1616	**1102**	1102	1449	**1015**	1062	1104	**962**	1015	1081	**934**	1038	1046	**927**	933	933
Tai04	15 × 15	**1246**	1406	1740	**1090**	1179	1383	**980**	1011	1061	**924**	980	1038	**930**	930	1009	**911**	911	930
Tai05	15 × 15	**1380**	1505	2188	**1106**	1123	1392	**1014**	1137	1187	**940**	1076	1137	**940**	940	1004	**940**	940	940
Tai06	15 × 15	**1347**	1467	1627	**1063**	1141	1404	**1054**	1063	1333	**944**	969	1017	**913**	920	1017	**903**	909	991
Tai07	15 × 15	**1402**	1561	1811	**1179**	1230	1342	**1042**	1079	1246	**976**	1023	1066	**956**	994	995	**942**	961	961
Tai08	15 × 15	**1295**	1494	1713	**1167**	1186	1216	**1077**	1077	1159	**977**	976	997	**970**	985	985	**963**	963	963
Tai09	15 × 15	**1434**	1445	1947	**1199**	1207	1443	**1112**	1171	1248	**1021**	1076	1132	**982**	982	1001	**982**	982	982
Tai10	20 × 15	**1259**	1594	1886	**1149**	1175	1247	**1044**	1159	1133	**969**	988	1087	**926**	953	988	**901**	944	966
Tai11	20 × 15	**1444**	1641	1976	**1352**	1356	1528	**1165**	1195	1299	**1114**	1121	1284	**980**	1119	1546	**956**	956	956
Tai12	20 × 15	**1483**	1696	2117	**1275**	1531	1663	**1199**	1495	1279	**1142**	1143	1170	**1013**	1012	1077	**1012**	1012	1043
Tai13	20 × 15	**1511**	1647	2253	**1219**	1593	1548	**1135**	1141	1278	**1067**	1108	1119	**1028**	1083	1119	**975**	1047	1047
Tai14	20 × 15	**1450**	1477	2207	**1219**	1334	1522	**1134**	1202	1312	**1018**	1065	1217	**990**	990	1060	**990**	990	991
Tai15	20 × 15	**1571**	1852	2138	**1332**	1448	1785	**1147**	1200	1335	**1045**	1154	1192	**984**	984	1172	**948**	968	968
Tai16	20 × 15	**1564**	1535	2257	**1351**	1384	1663	**1228**	1308	1476	**1102**	1217	1235	**1083**	1114	1146	**982**	1073	1073
Tai17	20 × 15	**1618**	1681	2174	**1304**	1419	1532	**1161**	1254	1353	**1090**	1203	1189	**988**	1094	1144	**979**	979	1031
Tai18	20 × 15	**1505**	1664	2209	**1306**	1387	1527	**1216**	1207	1433	**1059**	1138	1239	**1049**	1001	1053	**973**	985	1049
Tai19	20 × 15	**1555**	1604	2157	**1294**	1325	1580	**1106**	1106	1268	**1009**	1022	1162	**1006**	1020	1134	**940**	1020	1066
Tai20	20 × 20	**1543**	1759	2108	**1286**	1530	1621	**1142**	1328	1366	**1036**	1177	1195	**1003**	1141	1076	**983**	1033	1063
Tai21	20 × 20	**1986**	2060	2509	**1626**	1883	2200	**1441**	1685	1796	**1319**	1405	1608	**1232**	1232	1676	**1232**	1217	1346
Tai22	20 × 20	**1876**	2012	2790	**1511**	1851	2163	**1434**	1609	1710	**1330**	1417	1827	**1310**	1387	1654	**1240**	1224	1331
Tai23	20 × 20	**1684**	1759	2749	**1528**	1598	1949	**1440**	1492	1744	**1369**	1411	1570	**1327**	1336	1560	**1253**	1287	1355
Tai24	20 × 20	**1832**	2309	2590	**1614**	1883	2174	**1396**	1659	1988	**1377**	1428	1722	**1291**	1322	1637	**1212**	1212	1395
Tai25	20 × 20	**1874**	2240	2690	**1582**	1754	2155	**1381**	1563	1866	**1286**	1465	1514	**1275**	1333	1496	**1215**	1228	1355
Tai26	20 × 20	**1895**	2112	3103	**1576**	1802	2187	**1426**	1558	1871	**1349**	1665	1855	**1294**	1386	1678	**1282**	1348	1424
Tai27	20 × 20	**1967**	2173	3079	**1717**	1849	2241	**1538**	1632	1945	**1422**	1543	1759	**1345**	1407	1636	**1331**	1387	1465
Tai28	20 × 20	**1964**	2686	2809	**1585**	1698	2105	**1516**	1486	1746	**1374**	1375	1840	**1322**	1387	1674	**1293**	1336	1432
Tai29	20 × 20	**1945**	1969	2931	**1761**	1885	1988	**1473**	1509	1702	**1470**	1512	1643	**1400**	1400	1521	**1300**	1317	1317
Tai30	20 × 20	**1753**	2091	2492	**1599**	1813	2264	**1434**	1434	1732	**1345**	1398	1792	**1283**	1440	1421	**1246**	1365	1420
Tai31	30 × 15	**2296**	2362	3012	**1669**	1660	2186	**1415**	1481	1796	**1223**	1418	1499	**1210**	1393	1488	**1132**	1114	1246
Tai32	30 × 15	**1983**	2179	3212	**1649**	2155	2415	**1410**	1430	1847	**1319**	1425	1820	**1266**	1407	1638	**1103**	1164	1310
Tai33	30 × 15	**2037**	2331	3421	**1537**	1731	2278	**1480**	1570	1924	**1267**	1490	1576	**1133**	1468	1545	**1151**	1151	1361

Table 11 (continued)

Problem	size	2F			3F			4F			5F			6F			7F		
		DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO
Tai34	30 × 15	**2161**	2867	2876	**1574**	1934	2265	**1457**	1437	1831	**1358**	1455	1818	**1230**	1348	1438	**1173**	1347	1393
Tai35	30 × 15	**1969**	3015	2933	**1598**	1823	2345	**1397**	1777	1876	**1323**	1328	1489	**1167**	1409	1408	**1185**	1185	1305
Tai36	30 × 15	**2046**	2354	3232	**1672**	1809	2344	**1436**	1638	1798	**1322**	1357	1634	**1239**	1386	1359	**1199**	1214	1309
Tai37	30 × 15	**2187**	2266	3194	**1619**	1759	2304	**1482**	1450	1875	**1290**	1495	1771	**1231**	1437	1576	**1225**	1225	1433
Tai38	30 × 15	**1915**	2164	2850	**1579**	1888	2048	**1409**	1321	1798	**1285**	1256	1578	**1207**	1293	1515	**1075**	1279	1406
Tai39	30 × 15	**1904**	1974	2717	**1518**	1616	2004	**1367**	1444	1783	**1192**	1463	1477	**1074**	1295	1404	**1163**	1205	1313
Tai40	30 × 15	**1992**	2404	2544	**1604**	1638	2014	**1430**	1474	1745	**1281**	1336	1592	**1159**	1288	1407	**1134**	1233	1358
Tai41	30 × 20	**2249**	2668	4388	**2027**	2155	2995	**1756**	1868	2382	**1589**	1676	2067	**1528**	1625	2105	**1394**	1407	1889
Tai42	30 × 20	**2328**	2400	3970	**1904**	2102	2561	**1779**	1729	2253	**1439**	1642	1993	**1407**	1471	1836	**1361**	1439	1812
Tai43	30 × 20	**2467**	2562	4099	**1896**	2042	2570	**1684**	1769	2276	**1547**	1675	2001	**1380**	1468	1778	**1361**	1361	1678
Tai44	30 × 20	**2228**	2439	4252	**1933**	2276	2981	**1713**	2056	2403	**1571**	1946	2055	**1515**	1603	1929	**1408**	1599	1952
Tai45	30 × 20	**2364**	2974	4307	**1962**	2149	2905	**1725**	1824	2457	**1543**	1751	2062	**1447**	1566	1928	**1408**	1469	1718
Tai46	30 × 20	**2215**	3371	4574	**1796**	2158	3104	**1847**	2057	2264	**1592**	1693	1943	**1562**	1552	1827	**1408**	1366	1749
Tai47	30 × 20	**2163**	2395	4195	**1932**	2067	2842	**1704**	1704	2269	**1546**	1580	2057	**1543**	1563	1900	**1472**	1543	1733
Tai48	30 × 20	**2250**	3005	3974	**1932**	2013	2696	**1754**	1875	2351	**1566**	1815	1987	**1421**	1671	1850	**1388**	1409	1634
Tai49	30 × 20	**2256**	3058	4291	**1963**	2197	2998	**1625**	1748	2390	**1509**	1745	2002	**1501**	1521	1890	**1361**	1513	1675
Tai50	30 × 20	**2239**	2451	3957	**1869**	1909	2930	**1685**	1867	2559	**1481**	1757	2046	**1481**	1561	1804	**1415**	1546	1746
Tai51	50 × 15	**2824**	3599	5804	**2095**	2166	4177	**1923**	2095	2929	**1624**	1743	2742	**1512**	1657	2140	**1415**	1627	1862
Tai52	50 × 15	**3047**	3714	5804	**2163**	2491	3917	**1846**	1881	2782	**1612**	1823	2513	**1535**	1575	2033	**1378**	1605	1707
Tai53	50 × 15	**2795**	4037	6269	**2231**	2237	4062	**1930**	2228	2977	**1656**	1893	2500	**1542**	1620	2174	**1424**	1648	1814
Tai54	50 × 15	**2745**	2745	5408	**2076**	2181	3724	**1892**	2030	2870	**1640**	1751	2274	**1504**	1555	2072	**1357**	1548	1810
Tai55	50 × 15	3077	**2969**	5674	**2127**	2451	4145	**1885**	2338	3360	**1688**	1936	2598	**1503**	1762	2187	**1398**	1568	1839
Tai56	50 × 15	**2699**	3074	6061	**2228**	2228	3933	**2041**	2178	3038	**1655**	1893	2565	**1628**	1565	2010	**1437**	1501	1813
Tai57	50 × 15	**2829**	3238	5933	**2241**	2241	4211	**2012**	2012	3277	**1721**	1868	2635	**1629**	1780	2417	**1587**	1746	1946
Tai58	50 × 15	**2690**	4604	5720	**2151**	2605	3936	**1932**	2033	3157	**1647**	1720	2643	**1579**	1577	2377	**1424**	1512	1889
Tai59	50 × 15	**2894**	3250	5427	**2173**	2742	3699	**2113**	2050	3144	**1596**	1779	2527	**1448**	1544	2132	**1408**	1779	1797
Tai60	50 × 15	**2717**	3386	5747	**2326**	2241	3918	**1932**	2226	3074	**1648**	1746	2654	**1599**	1588	2225	**1499**	1521	1862
Tai61	50 × 20	**3338**	3338	7403	**2470**	2475	4432	**2290**	2092	3956	2214	**1871**	2987	**1871**	2214	2799	**1737**	2214	2437
Tai62	50 × 20	**6884**	6522	7953	**3535**	4811	5500	**2471**	2569	4151	**2088**	2143	3542	**1796**	1920	2632	**1754**	1773	1777
Tai63	50 × 20	**5390**	5390	7864	**2644**	2644	5340	**2234**	2295	3896	**1902**	2007	3459	1839	**1797**	2927	**1670**	1674	1728
Tai64	50 × 20	**5929**	6193	7932	**2422**	2475	5172	**2186**	2186	4197	**1939**	1960	3121	**1750**	1765	2444	**1729**	1750	1751
Tai65	50 × 20	**6071**	6603	7583	**2828**	4811	5340	**2147**	2189	3877	**1906**	1906	3389	1752	**1696**	2916	**1643**	1712	1657
Tai66	50 × 20	**5234**	5234	7755	**2521**	2644	5397	**2321**	2268	4206	2076	**1937**	3402	**1682**	2017	2983	**1632**	1632	1971

Table 11 (continued)

Problem	size	2F			3F			4F			5F			6F			7F		
		DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO	DAHACO	HACO	ACO
Tai67	50 × 20	**4348**	4348	7695	**2407**	3588	4979	**2153**	2178	4122	**1970**	2178	3307	1852	**1824**	2965	1824	**1756**	2712
Tai68	50 × 20	**4429**	4429	8109	**2720**	3103	5433	2418	**2217**	4049	**1921**	2152	3418	1873	**1809**	2938	**1642**	1688	1831
Tai69	50 × 20	6817	**6139**	8287	**2383**	2383	5262	**2236**	2635	4297	2115	**2072**	3506	**2008**	2041	3191	**1690**	1803	1648
Tai70	50 × 20	**4829**	5840	8429	**2753**	2468	5185	**2285**	2349	3722	2129	**2036**	3199	1869	**1851**	2501	**1652**	1652	1595
Tai71	100 × 20	**12759**	12759	19216	**9834**	12299	11958	**6239**	7963	8590	4983	**4950**	7181	**4763**	4816	5566	**3774**	4188	4972
Tai72	100 × 20	**13886**	14197	18189	**9302**	9302	11578	7446	**7435**	8440	**6186**	6831	7801	5068	**4879**	5528	**3965**	3965	4628
Tai73	100 × 20	**13462**	14668	20098	**9329**	12549	13138	**7773**	8810	8615	**5942**	5942	7752	**5465**	6000	5964	**4748**	5009	4889
Tai74	100 × 20	**12492**	13771	18634	**9039**	11024	12039	**7482**	8435	8221	5548	**5460**	7047	**5097**	5134	5636	**3396**	3396	4685
Tai75	100 × 20	**12955**	15775	19174	**9542**	10935	11858	**6743**	7029	8812	5939	**5587**	7449	**4815**	5159	5490	**3715**	4272	4774
Tai76	100 × 20	**14999**	16734	19593	9718	**9325**	11800	8161	**7963**	8816	**6346**	6664	7196	**5074**	5182	5511	4889	**4697**	5009
Tai77	100 × 20	15981	**14861**	19207	**10135**	10135	11994	**7435**	7435	8629	**6442**	6546	7402	5531	**5381**	5639	4429	**4191**	4885
Tai78	100 × 20	**16029**	16763	18566	**10254**	10935	11172	7680	**7406**	8598	**5487**	5487	6825	**4149**	4560	5548	**3983**	3983	4603
Tai79	100 × 20	**14097**	17563	19306	**10061**	11004	11528	**8460**	8644	8802	**6303**	6341	6818	5416	**5100**	5382	**3722**	3722	4734
Tai80	100 × 20	**11574**	12180	17530	**7504**	7688	11316	**5461**	5461	8353	**5282**	5526	6589	**3876**	3910	5429	**2720**	3514	4560

The symbols in bold show the best results obtained

Where:
η_m: overall mean for S/N ratio
η_{α_2}: S/N ratio for parameter α at designated level 2
η_{β_1}: S/N ratio for parameter β at designated level 1
η_{ρ_3}: S/N ratio for parameter ρ at designated level 3
η_{Q_5}: S/N ratio for parameter Q at designated level 5

A confirmation experiment is executed five times with the optimal combination of parameters. The actual value of the S/N ratio (η_{actual}) is calculated based on this experiment. The predicted value which is calculated using Eq. (10) and the actual value of S/N ratio are provided in Table 9.

The predicted value of the S/N ratio with the optimal level of parameters is very close to the S/N ratio of the actual value. This validates the experiment for predicting the best levels of the parameters using the Taguchi method.

7.3 Evaluation of the DAHACO algorithm

The novelty of the proposed approach is twofold. The first contribution lies in the dynamic assignment of jobs to factories and the second offers a hybrid ant-based algorithm which integrates a local search in order to improve the solutions. To evaluate our proposed DAHACO and prove the effectiveness of introducing local search on the one hand and the dynamic assignment on the other hand, we have developed three different variants of ant-based algorithms to show the relevance and the influence of each improvement made on our DAHACO. Table 10 summarizes the 3 implemented versions:

- The first one, called ”ACO”, uses the ant colony system algorithm without any hybridization and a static assignment procedure (workload).
- The second one, called ”HACO” uses a hybrid ant colony system algorithm that integrates a local search procedure and a static assignment procedure (workload).
- The third one, called ”DAHACO” uses a dynamic assignment of jobs to factories and the hybrid ant colony system algorithm for the scheduling.

The three algorithms were implemented and ran a certain number of times experimentally determined using the best parameters set determined previously $\alpha_2 = 1$, $\beta_1 = 0$, $\rho_3 = 0.7$, $Q_5 = 10000$. We used instances from well-known benchmarks proposed by [55]. This benchmark includes 8 combinations for n and m, and 10 instances for each combination. Each instance is solved with different level of *f (f = 2, 3, 4, 5, 6, 7)* summing up 480 instances. The computational results are depicted from Tables 11, 12, 13, 14, and 15. Table 11 presents the value of makespan

Table 12 Relative Percentage Deviation (RPD) of the 3 algorithms grouped by *n* and *m*

Problem	(n x m)	DAHACO	HACO	ACO	Problem	(n x m)	DAHACO	HACO	ACO
Tai01	15 × 15	**0,00**	1,54	9,80	Tai41	30 × 20	**0,00**	7,35	46,98
Tai02	15 × 15	**0,25**	3,23	9,57	Tai42	30 × 20	**0,48**	6,31	39,58
Tai03	15 × 15	**0,00**	4,96	13,79	Tai43	30 × 20	**0,00**	5,21	36,39
Tai04	15 × 15	**0,00**	5,04	16,29	Tai44	30 × 20	**0,00**	15,08	47,02
Tai05	15 × 15	**0,00**	6,20	21,54	Tai45	30 × 20	**0,00**	11,18	43,60
Tai06	15 × 15	**0,00**	3,53	18,04	Tai46	30 × 20	**0,62**	15,01	44,95
Tai07	15 × 15	**0,00**	5,01	12,98	Tai47	30 × 20	**0,00**	4,34	41,35
Tai08	15 × 15	**0,02**	3,09	7,97	Tai48	30 × 20	**0,00**	13,28	37,50
Tai09	15 × 15	**0,00**	2,02	13,53	Tai49	30 × 20	**0,00**	13,86	45,28
Tai10	15 × 15	**0,00**	8,26	15,49	Tai50	30 × 20	**0,00**	9,29	44,79
Tai11	20 × 15	**0,00**	5,22	22,40	Tai51	50 × 15	**0,00**	11,95	66,53
Tai12	20 × 15	**0,02**	9,87	15,30	Tai52	50 × 15	**0,00**	11,85	55,75
Tai13	20 × 15	**0,00**	9,46	18,30	Tai53	50 × 15	**0,00**	15,88	63,33
Tai14	20 × 15	**0,00**	3,65	19,92	Tai54	50 × 15	**0,00**	6,10	56,32
Tai15	20 × 15	**0,00**	7,29	20,04	Tai55	50 × 15	**0,61**	13,89	65,87
Tai16	20 × 15	**0,32**	5,25	19,58	Tai56	50 × 15	**0,67**	6,57	59,92
Tai17	20 × 15	**0,00**	6,97	16,43	Tai57	50 × 15	**0,00**	7,05	64,10
Tai18	20 × 15	**0,93**	4,24	18,74	Tai58	50 × 15	**0,02**	18,02	67,15
Tai19	20 × 15	**0,00**	2,79	19,00	Tai59	50 × 15	**0,51**	13,82	57,39
Tai20	20 × 15	**0,00**	13,62	18,84	Tai60	50 × 15	**1,26**	8,42	62,66
Tai21	20 × 20	**0,21**	7,17	25,80	Tai61	50 × 20	**4,63**	3,09	73,31
Tai22	20 × 20	**0,22**	9,06	30,58	Tai62	50 × 20	**0,93**	8,45	43,84
Tai23	20 × 20	**0,00**	3,18	25,38	Tai63	50 × 20	**0,39**	1,42	61,75
Tai24	20 × 20	**0,00**	11,28	30,91	Tai64	50 × 20	**0,00**	1,63	56,87
Tai25	20 × 20	**0,00**	10,52	26,91	Tai65	50 × 20	**0,55**	14,17	57,49
Tai26	20 × 20	**0,00**	11,79	35,33	Tai66	50 × 20	**1,59**	4,13	70,24
Tai27	20 × 20	**0,00**	6,93	28,15	Tai67	50 × 20	**0,90**	9,94	76,69
Tai28	20 × 20	**0,34**	8,70	27,44	Tai68	50 × 20	**2,10**	4,82	69,55
Tai29	20 × 20	**0,00**	2,48	16,81	Tai69	50 × 20	**2,61**	4,82	62,68
Tai30	20 × 20	**0,00**	9,73	27,08	Tai70	50 × 20	**3,44**	4,55	56,63
Tai31	30 × 15	**0,36**	6,43	24,53	Tai71	100 × 20	**0,11**	10,80	33,93
Tai32	30 × 15	**0,00**	11,12	37,59	Tai72	100 × 20	**0,67**	2,11	20,85
Tai33	30 × 15	**0,00**	13,38	37,53	Tai73	100 × 20	**0,00**	12,02	23,92
Tai34	30 × 15	**0,23**	14,52	28,99	Tai74	100 × 20	**0,27**	7,61	28,31
Tai35	30 × 15	**0,00**	19,25	28,89	Tai75	100 × 20	**1,05**	10,46	29,80
Tai36	30 × 15	**0,00**	8,85	27,64	Tai76	100 × 20	**2,02**	2,76	16,32
Tai37	30 × 15	**0,37**	7,48	33,33	Tai77	100 × 20	**2,67**	0,27	16,65
Tai38	30 × 15	**1,50**	9,78	32,77	Tai78	100 × 20	**0,62**	3,52	19,25
Tai39	30 × 15	**0,00**	10,45	28,78	Tai79	100 × 20	**1,03**	6,12	16,08
Tai40	30 × 15	**0,00**	8,34	23,46	Tai80	100 × 20	**0,00**	7,06	47,95

The symbols in bold show the best results obtained

Table 13 Average relative percentage deviation of the 3 algorithms grouped by *n* and *m*

	DAHACO	HACO	ACO
Average RPD	0,43	7,95	35,42

obtained by the 3 algorithms with the 480 instances. First, we notice that the total execution time of jobs on machines decreases when the number of plants increases. This is naturally explained by the fact that executing *n* jobs on *f* factories is faster than executing them on *f/2* factories. This brings us directly back to thinking about the usefulness of

Table 14 Average RPD of the three algorithms grouped by *f*

F	DAHACO	HACO	ACO
2	0,39	13,12	57,08
3	0,25	10,67	46,53
4	0,58	6,39	34,85
5	0,59	6,93	32,03
6	0,44	5,76	26,07
7	0,34	4,85	15,97
Average	0,43	7,95	35,42

factory distribution and its significant role in time saving. Table 12 summarizes the calculated Relative Percentage Deviation for the 480 instances, such as each value is the average value between different levels of f compared with the best value found by the three algorithms. As can be witnessed, the DAHACO outperforms the HACO and the ACO in almost all instances. Table 13 shows the Average Relative Percentage Deviation of the 3 algorithms grouped by *n* and *m*. As it can be seen, DAHACO beats the two other algorithms clearly, bringing down the average RPD to just 0.43 compared to a deviation of 7.95 for HACO algorithm and 35.42 for ACO. The smaller Average Relative Percentage Deviation shows that the proposed DAHACO is more robust and competitive than the HACO and ACO. Table 14 shows the average RPD of the three considered algorithms grouped by the number of factories *f*, i.e., each cell of the table contains the average RPD of 80 instances. It can be seen that DAHACO outperforms the other 2 algorithms in different factories levels and gets the best results for all combinations.

Figure 7 shows the average RPD of the tested algorithms vs. the number of factories. We can see that DAHACO algorithm outperforms the other algorithms regardless of the number of factories. As we can see, the DAHACO is stable with some factory as with several factories which is not the case for the other algorithms. But it is clear that when the number of factories increases, the performance of all algorithms is getting better. This observation is predictable because its logic that scheduling n jobs on two factories is costlier in terms of time than scheduling n jobs on three factories and so on. And this is what makes the distribution of factories a necessity in our days for the considerable gain of time.

The data in Table 15 show the influence of the number of factories on the computational time of the proposed DAHACO. We can notice that the computational time is interestingly reduced when we increase the number of factories. Our DAHACO is perfectly adapted to the distributed job shop scheduling problem and performs better in a distributed environment. It is important to mention that this is the first study that can deal with 10 factories in the DJSP. Figure 8 illustrates the curve of the computational time versus the number of factories. The efficiency compared to the 2 other algorithms is visible, especially, in the case of a small number of factories where job scheduling is more complicated because we have a larger number of jobs to execute in each factory. When the number of plants increases, the execution time is reduced considerably until reaching 0.07 sec for the 15x15 instance with 10 factories.

To sum up, we can confirm that the proposed DAHACO outperforms two other algorithms in solving the DJSP. In fact, there are two important reasons for the rapid convergence and the good scheduling solutions of the proposed algorithm. The first is the variety of the assignment combinations, which offers more diversification thus exploiting new possibilities. The second is due to the local search procedure which enhances the capability of the algorithm to escape from local minimum and quickly guide the search to different regions of the solution space.

8 Conclusions and perspectives

This study proposed a novel dynamic assignment of jobs to factories, applied to a hybrid ant colony optimization algorithm combined with local search to solve the distributed job shop scheduling problem with makespan criterion. Firstly, the effective workload rule for assignment

Table 15 The computational time of the DAHACO (sec)

	2f	3f	4f	5f	6f	7f	8f	9f	10f
15 × 15	7,21	1,31	0,53	0,23	0,16	0,11	0,08	0,07	0,07
20 × 15	18,93	4,73	1,75	0,72	0,47	0,27	0,23	0,16	0,12
20 × 20	44,46	10,96	3,86	1,77	0,91	0,50	0,38	0,26	0,16
30 × 15	208,39	35,52	11,95	5,71	2,87	1,98	1,11	0,78	0,49
30 × 20	351,52	85,14	25,57	10,81	5,89	3,60	2,16	1,55	0,79

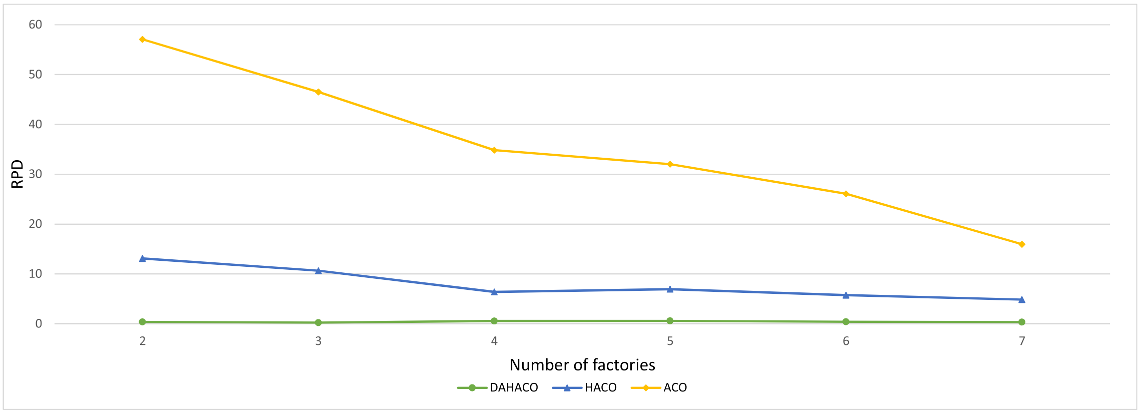


Fig. 7 The average RPD of the tested algorithms versus the number of factories

phase was combined with random permutations to create a dynamic assignment of jobs to factories. This step is very important to generate a diverse population for the ant-based algorithm. Moreover, a local search procedure is associated with ant-based algorithm in order to explore more search space and find better results. Furthermore, the Taguchi method for robust design is adopted for finding the optimum combination of parameters of the DAHACO algorithm. By applying the Taguchi method, the total number of experiments carried out for finding an optimum level of parameters has been reduced considerably. From the confirmation tests, a good agreement between the predicted S/N ratio and the actual S/N ratio is observed. This validates the proposed experiment based on the Taguchi method for parameter design.

The intensive experiments have been performed to evaluate our novel algorithm. Results proved that the effectiveness of the proposed DAHACO is able to reach best solutions. Besides, in order to verify the superiority of DAHACO, we have compared it with two ant-based algorithms with the use of the well-known instances proposed by Taillard [55]. Comparisons have proved that DAHACO can usually produce better solutions than other algorithms.

Even though this study proposed an effective DAHACO algorithm for the DJSP, there are some impediments that should be improved in future research work. In fact, this study has just considered the case when all factories are identical. This can be an idealization to the real problem, since it is rarely the case. On the basis of the satisfactory results obtained from this work and the limitations presented above, we plan to explore the following issues:

- Combining the proposed DAHACO algorithm with other methods in order to better explore the search space and improve the solution.
- Considering more complex real world constraints such as setup times and maintenance considerations.
- Investigating the applicability of the proposed approach to the multi-objective DJSP.
- Considering the case with non-identical factories.

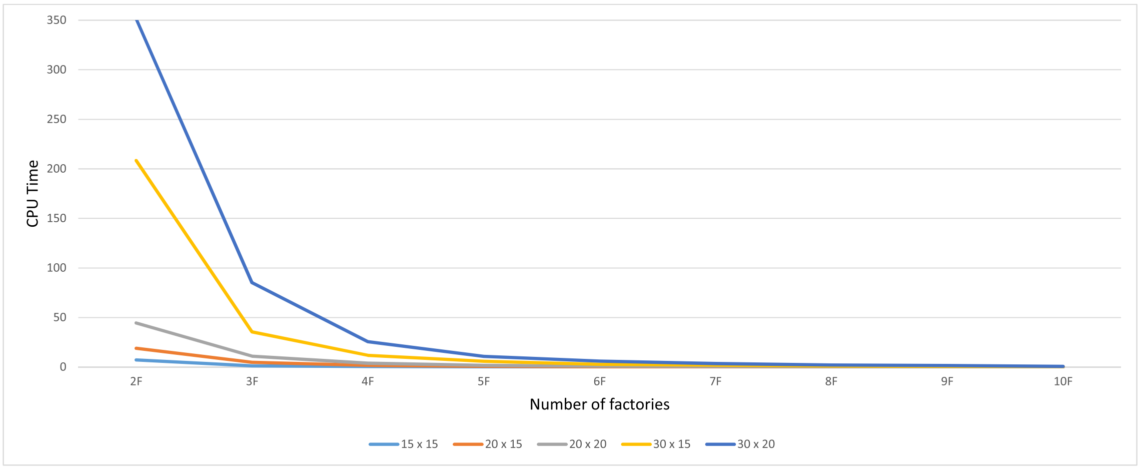


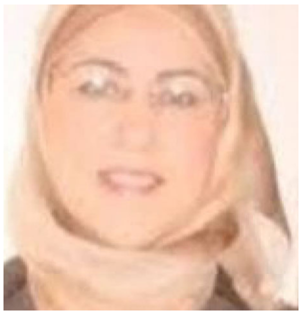
Fig. 8 The curve of the computational time versus the number of factories

References

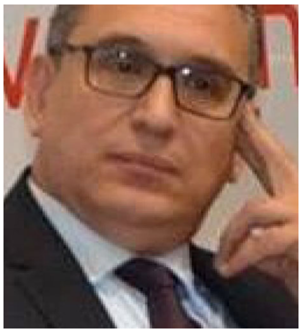
1. Akjiratikarl C, Yenradee P, Drake PR (2007) Pso-based algorithm for home care worker scheduling in the uk. Comput Ind Eng 53(4):559–583
2. Asadzadeh L, Zamanifar K (2010) An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. Math Comput Model 52(11):1957–1965
3. Balas E (1969) Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Oper Res 17(6):941–957
4. Bell JE, McMullen PR (2004) Ant colony optimization techniques for the vehicle routing problem. Adv Eng Inform 18(1):41–48
5. Blazewicz J, Ecker KH, Pesch E, Schmidt G, Weglarz J (1997) Scheduling computer and manufacturing processes. J Oper Res Soc 48(6):659–659
6. Blum C, Sampels M (2004) An ant colony optimization algorithm for shop scheduling problems. J Math Model Algorithm 3(3):285–308
7. Brucker P, Brucker P (2007) Scheduling algorithms, vol 3. Springer, Berlin
8. Bullnheimer B, Hartl RF, Strauss C (1997) An improved ant system algorithm for the vehicle routing problem
9. Carlier J, Pinson É (1989) An algorithm for solving the job-shop problem. Manag Sci 35(2):164–176
10. Chaouch I, Belkahla Driss O, Ghedira K (2017) A survey of optimization techniques for distributed job shop scheduling problems in multi-factories. In: Silhavy R, Senkerik R, Kominkova Oplatkova Z, Prokopova Z, Silhavy P (eds) Cybernetics and mathematics applications in intelligent systems. Springer International Publishing, Cham, pp 369-378
11. Chen CL, Chen CL (2009) Bottleneck-based heuristics to minimize total tardiness for the flexible flow line with unrelated parallel machines. Comput Ind Eng 56(4):1393–1401
12. Chen L, Bostel N, Dejax P, Cai J, Xi L (2007) A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. Eur J Oper Res 181(1):40–58
13. Cheng BW, Chang CL (2007) A study on flowshop scheduling problem combining taguchi experimental design and genetic algorithm. Expert Syst Appl 32(2):415–421
14. Chiang TC, Fu LC (2007) Using dispatching rules for job shop scheduling with due date-based objectives. Int J Prod Res 45(14):3245–3262
15. Chong CS, Low MYH, Sivakumar AI, Gay KL (2006) A bee colony optimization algorithm to job shop scheduling. In: Proceedings of the 2006 winter simulation conference, pp 1954–1961
16. Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies, actes de la première conférence européenne sur la vie artificielle (pp 134–142). Elsevier Publishing, France
17. Colorni A, Dorigo M, Maniezzo V, Trubian M (1994) Ant system for job-shop scheduling. Belg J Oper Res Stat Comput Sci 34(1):39–53
18. Cordon O, De Viana IF, Herrera F, Moreno L (2000) A new aco model integrating evolutionary computation concepts: The best-worst ant system
19. Dorigo M (1992) Optimization learning and natural algorithms. PhD Thesis, Politecnico di Milano
20. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66
21. Dorigo M, Maniezzo V, Colorni A, Maniezzo V (1991) Positive feedback as a search strategy
22. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41
23. Dowsland KA, Thompson JM (2008) An improved ant colony optimisation heuristic for graph colouring. Discret Appl Math 156(3):313–324
24. Eswaramurthy VP, Tamilarasi A (2009) Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems. Int J Adv Manuf Technol 40(9):1004–1015
25. French S (1982) Sequencing and scheduling, mathematics and its applications
26. Gambardella LM, Taillard É, Agazzi G (1999) Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: New ideas in optimization, Citeseer
27. Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1(2):117–129
28. Gonçalves JF, de Magalhães Mendes JJ, Resende MG (2005) A hybrid genetic algorithm for the job shop scheduling problem. Eur J Oper Res 167(1):77–95
29. Gutjahr WJ, Rauner MS (2007) An aco algorithm for a dynamic regional nurse-scheduling problem in austria. Comput Oper Res 34(3):642–666
30. Heinonen J, Pettersson F (2007) Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. Appl Math Comput 187(2):989–998
31. Hoitomt DJ, Luh PB, Pattipati KR (1993) A practical approach to job-shop scheduling problems. IEEE Trans Robot Autom 9(1):1–13. https://doi.org/10.1109/70.210791
32. Jain AS, Meeran S (2002) A multi-level hybrid framework applied to the general flow-shop scheduling problem. Comput Oper Res 29(13):1873–1901
33. Jayaraman V, Kulkarni B, Karale S, Shelokar P (2000) Ant colony framework for optimal design and scheduling of batch plants. Comput Chem Eng 24(8):1901–1912
34. Jia H, Fuh J, Nee A, Zhang Y (2002) Web-based multi-functional scheduling system for a distributed manufacturing environment. Concurr Eng 10(1):27–39
35. Jia H, Fuh J, Nee A, Zhang Y (2007) Integration of genetic algorithm and gantt chart for job shop scheduling in distributed manufacturing systems. Comput Ind Eng 53(2):313–320
36. Jia HZ, Nee AYC, Fuh JYH, Zhang YF (2003) A modified genetic algorithm for distributed scheduling problems. J Intell Manuf 14(3):351–362
37. Kamaruddin S, Khan ZA, Foong S (2010) Application of taguchi method in the optimization of injection moulding parameters for manufacturing products from plastic blend. Int J Eng Technol 2(6):574
38. Lin TL, Horng SJ, Kao TW, Chen YH, Run RS, Chen RJ, Lai JL, Kuo IH (2010) An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Syst Appl 37(3):2629–2636
39. Lu MS, Romanowski R (2012) Multi-contextual ant colony optimization of intermediate dynamic job shop problems. Int J Adv Manuf Technol 60(5):667–681
40. Madahav SP (1989) Quality engineering using robust design. New Jersey
41. Mahfouz A, Hassan SA, Arisha A (2010) Practical simulation application: Evaluation of process control parameters in twisted-pair cables manufacturing system. Simul Model Pract Theory 18(5):471–482
42. Maniezzo V, Colorni A (1999) The ant system applied to the quadratic assignment problem. IEEE Trans Knowl Data Eng 11(5):769–778

43. Muth JF, Thompson GL (1963) Industrial scheduling. Prentice-Hall
44. Naderi B, Azab A (2014) Modeling and heuristics for scheduling of distributed job shops. Expert Syst Appl 41(17):7754–7763
45. Naderi B, Azab A (2015) An improved model and novel simulated annealing for distributed job shop problems. Int J Adv Manuf Technol 81(1):693–703
46. Nouri HE, Belkahla Driss O, Ghedira K (2016) Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment. Appl Intell 45(3):808–828
47. Panigrahi BK, Shi Y, Lim MH (2011) Handbook of swarm intelligence: concepts, principles and applications, vol 8. Springer Science & Business Media, Berlin
48. Perez E, Posada M, Herrera F (2012) Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. J Intell Manuf 23(3):341–356
49. Pezzella F, Merelli E (2000) A tabu search method guided by shifting bottleneck for the job shop scheduling problem. Eur J Oper Res 120(2):297–310
50. Roy B, Sussmann B (1964) Problème d'ordonnancement avec contraintes disjonctives. Technical Report DS No 9
51. Singha H, Kumarb P (2005) Optimizing cutting force for turned parts by taguchi's parameter design approach. Indian J Eng Mater Sci 12:97–103
52. Stützle T, Hoos HH (2000) Max–min ant system. Futur Gener Comput Syst 16(8):889–914
53. Sundar S, Suganthan PN, Jin CT, Xiang CT, Soon CC (2017) A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. Soft Comput 21(5):1193–1202
54. Suresh R, Mohanasundaram K (2006) Pareto archived simulated annealing for job shop scheduling with multiple objectives. Int J Adv Manuf Technol 29(1):184–196
55. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285
56. Talbi EG (2009) Metaheuristics: from design to implementation, vol 74. Wiley, New York
57. Tan Y, Liu S, Wang D (2010) A constraint programming-based branch and bound algorithm for job shop problems. In: 2010 Chinese control and decision conference, pp 173–178
58. Tanco M, Viles E, Pozueta L (2009) Comparing different approaches for design of experiments (DoE). Springer, Dordrecht, pp 611–621
59. Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2006) Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. Int J Prod Res 44(22):4737–4754
60. Tay JC, Ho NB (2008) Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput Ind Eng 54(3):453–473
61. Wang L, Zhou G, Xu Y, Liu M (2012) An enhanced pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. Int J Adv Manuf Technol 60(9):1111–1123
62. Wang S, Liu M, Chu C (2015) A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. Int J Prod Res 53(4):1143–1167
63. Watanabe M, Ida K, Gen M (2005) A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. Comput Ind Eng 48(4):743–752
64. Weckman GR, Ganduri CV, Koonce DA (2008) A neural network job-shop scheduler. J Intell Manuf 19(2):191–201
65. Yao BZ, Yang CY, Hu JJ, Yin GD, Yu B (2010) An improved artificial bee colony algorithm for job shop problem. In: Applied mechanics and materials, trans tech publ, vol 26, pp 657-660
66. Ying KC, Liao CJ (2004) An ant colony system for permutation flow-shop sequencing. Comput Oper Res 31(5):791–801
67. Zhang R, Wu C (2010) A hybrid approach to large-scale job shop scheduling. Appl Intell 32(1):47–59
68. Zhou R, Nee A, Lee H (2009) Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. Int J Prod Res 47(11):2903–2920
69. Zhou Y, Chen H, Zhou G (2014) Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem. Neurocomputing 137:285–292

Imen Chaouch received her Engineer degree in Industrial Computing and Automatic from Institut National des Sciences Appliquées et de Technologie (INSAT), University of Carthage, Tunisia in 2014. She is currently a PhD candidate in computer science and a member of COSMOS Laboratory at Ecole Nationale des Sciences de l'Informatique, University of Manouba, Tunisia. She is also a teacher assistant in Faculté des Sciences de Tunis, Tunis - El Manar University, Tunisia. Her main research interest centres on Distributed manufacturing systems, optimization, metaheuristics and multiagent systems.



Olfa Belkahla Driss received the B.Sc., M.Sc., Ph.D. and HdR degrees in computer science from Institut Supérieur de Gestion de Tunis, University of Tunis, Tunisia, in 1997, 2000, 2006 and 2018 respectively. She is an Assistant Professor with the Department of Computer Science at Ecole Supérieure de Commerce de Tunis, University of Manouba from 2003. She is actually a supervisor of the Research Master in Computational Intelligence and Decision Making applied to Management from 2010. Her main research interests are in the field of industrial engineering, scheduling, transport and production logistics, smart cities, e-government, data analysis, artificial intelligence, multi-agent systems, and optimization. She has authored more than 60 research papers. She is member of technical committee of many international conference and reviewer for many ranked journals.



Khaled Ghedira has received the Engineer degree in hydraulic (ENSEEIHT-France); the specialized Engineer degree in computer science and applied mathematics (ENSIMAGFrance), both the M.Sc and the Ph.D degrees in Artificial Intelligence (ENSAE-France) and the HdR in computer science from Ecole Nationale des Sciences de l'Informatique (ENSI-Tunisia). He was Research Fellow at Institut d'Informatique et d'Intelligence Artificielle (IIIA-Switzerland 1992-96), expert consultant at British Telecom (England 1995), the head of ENSI (2002-2008), the general managing director of the Tunis Science City (2011-2014), and the general director of the Tunisian National Agency for scientific Research Promotion (2014-2017). He is also the president of the administration council of Institut de la Francophonie pour l'Ingénierie de la Connaissance et la formation á distance (AUF-IFIC). He is Professor at ISG (University of Tunis), the founding president of the Tunisian Association of Artificial Intelligence (ATIA-Tunisia), the founder and director of both the research Unit URIASIS (1999-2011) and the SOIE laboratory (2011-2013). He is member of several international scientific committees and is often invited as keynote speaker/visiting professor at national and international level. He is/was also member of the think national committee for higher education and member/president of several committees: evaluation of higher education institutions, research projects reviewing, teachers recruiting, LMD. His research areas include MAS, CSP, transport and production logistics, mono and multi-objective optimization, metaheuristics and security in M/Egovernment. He has led several national and international research projects. He has supervised more than fourty PhD thesis and fifty master thesis. He has co/authored about 360 journal/conference/book research papers. He has written four books in ICT at the international level and co-authored some others.