An Ant System with steps counter for the Job Shop Scheduling Problem

Emanuel Téllez-Enríquez, Efrén Mezura-Montes and Carlos A. Coello Coello

Abstract— **In this paper, we present an ant system algorithm variant designed to solve the job shop scheduling problem. The proposed approach is based on a recent biological study which showed that natural ants can count their steps when they build the path between the nest and their food source. Experiments using a set of well-known job shop scheduling problems and a comparison against state-of-the-art techniques show that the proposed approach can reduce the number of evaluations performed without a degradation of performance. Additionally, our proposed approach reduces the number of parameters that need to be tuned by the user (specifically the parameters that balance the importance between the pheromone trail and heuristic values), with respect to the original ant system algorithm.**

I. Introduction

Ant Colony Optimization (ACO) is a metaheuristic inspired by the foraging behavior of ants, which has been used to solve combinatorial optimization problems and the *Ant System* (AS) was the first algorithm within this class, which was developed by Dorigo [1]. In this paper, a variant of the AS algorithm is presented to solve the *Job Shop Scheduling Problem* (JSSP).

In the classical JSSP, a finite number of jobs are to be processed by a finite number of machines. Each job consists of a predetermined sequence of operations, which will be processed without interruptions by a period of time in each machine. The operations that correspond to the same job will be processed according to their technological sequence and none of them will be able to begin its processing before the precedent operation has finished. A feasible schedule is an assignment of operations in time on a machine without violation of the job shop constraints. A makespan is defined as the maximum completion time of all jobs. The objective of JSSP is to find a schedule that minimizes the makespan. In Complexity Theory, the JSSP is classified as an NP-hard combinatorial optimization problem [2].

ACO is modelled after the communication principles and cooperative work of real ants, which was inspired by the study of Argentinean ants done by Goss et al. [3]. Basically, an ACO algorithm has three procedures [4]:

1) *ConstructAntsSolutions*: It manages the colony of ants, building paths in a construction graph (graph that represents the problem) and moving ants from one node to another by applying local stochastic decisions considering the pheromone trail and heuristic information available.
2) *UpdatePheromones*: It is the process by which the pheromone trails are modified.
3) *DeamonActions*: This procedure is used to implement centralized actions which cannot be performed by single ants; for example, the activation of a local search optimization procedure.

The ACO metaheuristic has been applied to diverse hard combinatorial optimization problems. In the scheduling field, ACO has been applied to flow shop problems [5], to the travelling salesperson problem [6] and to the vehicle routing problem [7]. However, very few papers report work on ACO implementations for the JSSP. To the authors' best knowledge, Colorni et al. [8] were the first to apply an AS to the JSSP. Although the results reported in this early paper were not satisfactory, some other works in the same field were developed later on. Sjoerd van der Zwaan et al. [9] developed an ACO for JSSP in which a genetic algorithm was adopted for fine-tuning the parameters of an ACO approach. More recently, Blum et al. [10] have investigated the application of ACO to shop scheduling problems including the JSSP.

This paper shows a variant of the AS algorithm to solve the JSSP. Our method is based on a new study of natural ants that showed that ants can count their steps when walking on the path from the nest to the food source. Knowing that the measure of time in the JSSP is discrete, we can map it to the number of steps of an ant when walking. In this way, the starting time at which an ant has to begin walking can be computed sooner, introducing a form of heuristic knowledge to our approach.

The remainder of this paper is organized as follows. Section II introduces a formal description of the JSSP and Section III provides a description of the basic Ant System algorithm. Our AS variant is described in detail in Section IV. In Section V, we present our experimental results using benchmark problems and we compare them with respect to other state-of-the-art algorithms. Finally, in Section VI our conclusions and future work are established.

II. Job Shop Scheduling Problem (JSSP)

In our study, we adopted the classic JSSP, which is composed of n-jobs and m-machines and it is denoted by $n/m/T/C_{max}$, where the parameter n represents the number of jobs, m is the number of machines, T is the technological sequence of the jobs in each machine, and C_{max} indicates the performance measure which should be minimized (i.e.,

Emanuel Téllez-Enríquez and Carlos A. Coello Coello are with the Computer Science Department, CINVESTAV-IPN, Av. IPN No. 2508, Col. San Pedro Zacatenco, Mexico, D.F., 07300, México (phone: +52 55 50613800 x. 6564; email: ccoello@cs.cinvestav.mx). Efrén Mezura-Montes is with the National Laboratory on Advanced Informatics (LANIA A.C.), Rébsamen 80, Centro, Xalapa, Veracruz, 91000, México (phone: +52 228 8416100 x. 5006; email: emezura@lania.mx).

maximum time taken to complete all jobs). An instance of the JSSP can be represented by a matrix as it is shown in Table I.

TABLE I
AN INSTANCE 3×3 OF THE JSSP.

job	machine (time)		
1	1(3)	2(3)	3(3)
2	1(2)	3(3)	2(4)
3	2(3)	1(2)	3(1)

In the example of Table I, we have 3 jobs, 3 machines and a technological sequence represented in each row of the jobs. In the case of job 1 in Table I, we can see that it should be processed in machine 1 first with a processing time of 3 (in the matrix, this time is represented between parentheses). After that, this job 1 is processed in machine 2 with processing time of 3 and finishes in machine 3 with a processing time of 3. This description is called technological sequence of job 1. When a job i is processed in a machine j, it is called as *"operation (i,j)"*.

To apply the AS algorithm for JSSP we will use the graph representation $G = (V, C \bigcup D)$ described in [11] where:

- V is a set of nodes representing operations of the jobs together with two special nodes: a start (0) node and an end (*) node, representing the beginning and the end of the schedule, respectively.
- C is a set of conjunctive arcs representing technological sequences of the operations.
- D is a set of disjunctive arcs representing pairs of operations which must be processed on the same machine.

Figure 1 shows the corresponding graph for the instance of the JSSP described in Table I, whose nodes represent each operation (i, j) where i is the current job and j its corresponding machine (except for the nodes marked with (0) and (*) because they indicate the start and end of the graph). The processing time of each operation is denoted by t_{ij} on each node. The conjunctive arcs give the technological sequence connecting all operations of the same job and disjunctive arcs indicate pairs of operations in the same machine.

III. ANT SYSTEM (AS)

In this section, we describe the operation of the classical AS for the JSSP proposed in [8], in which a population of m artificial ants builds solutions by iteratively applying n times a probabilistic decision policy until obtaining a solution for the problem. In order to communicate the individual search experience to the colony, the ants mark the corresponding paths with some amount of pheromone according to the type of solutions found. This amount is inversely proportional to the cost of the path generated (i.e., if the path found is long, the amount of pheromone deposited is low; otherwise, the amount of pheromone deposited is high). Therefore, in the following iterations more ants will be attracted to the most promising paths. Besides the pheromone, the ants are guided

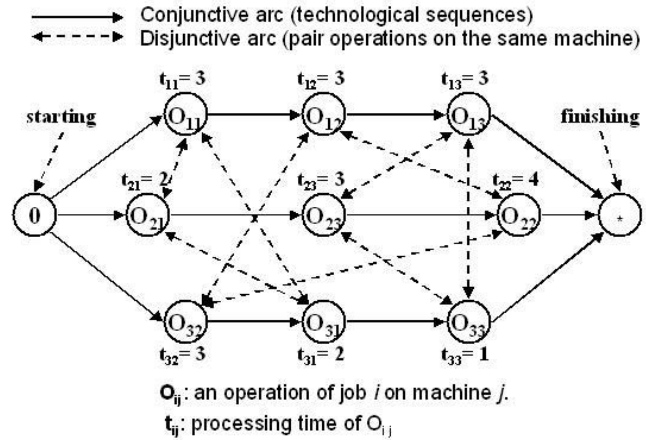


Fig. 1. A graph of a 3×3 JSSP.

by a heuristic value in order to help them in the construction process. All the decisions taken by the ant (the path found or solution), are stored in a tabu list (TL).

As it was indicated above, to apply the AS algorithm, the instance of the problem must be first constructed in a graphical representation G. The AS starts with a small amount of pheromone c along each edge on G. Each ant is then assigned a starting position, which is added to its tabu list. The initial ant position is usually chosen at random. Once the initialization phase is completed, each ant will independently construct a solution by using equation (1) at each decision point until a complete solution has been found. After every ant's tabu list is full, the cost C_{max} of the obtained solution is calculated. The pheromone amount along each edge (i,j) is calculated according to equation (2). Finally, all tabu lists are emptied. If the stopping criterion has not been reached, the algorithm will continue with a new iteration.

The decision of each ant is based, not only the amount of pheromone τ_{ij}, located along edge (i,j), but also on the heuristic value η_{ij} along this edge. The transition probability to move from node i to node j for the k^{th} ant at iteration t is defined as:

$$P_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{j \notin TL_k} [\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}, if\ j \notin TL_k \\ 0\ if\ \ j \in TL_k \end{cases} \tag{1}$$

where α and β are parameters which allow the user to balance the importance given to the heuristic (parameter β) with respect to the pheromone trails (parameter α). Setting $\beta = 0$ will result in only considering the pheromone information in the ant's decision, whereas if $\alpha = 0$, only the heuristic information will be used for the ant.

The pheromone trail levels to be used in the next iteration of the algorithm are given by the formula:

$$\tau_{ij}(t + 1) = \rho \times \tau_{ij}(t) + \Delta\tau_{ij} \tag{2}$$

where ρ is a coefficient, such that $(1 - \rho)$ can be interpreted as a trail evaporation coefficient; that is, $(1 - \rho) \times \tau_{ij}(t)$

represents the amount of trail which evaporates on each edge (i,j) in the period between iteration t and $t+1$.

The total amount of pheromone laid by the m ants $\Delta\tau_{ij}$, is calculated by:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

where $\Delta\tau_{ij}^{k}$ is calculated as:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{C_{max}^{k}} & \text{if the } k \text{ ant travels along edge } (i,j) \\ 0 & \end{cases} \tag{4}$$

where Q is a positive real valued constant and C_{max} is the cost of the solution of the k^{th} ant, while Q/C_{max}^{k} gives the quantity of pheromone per unit of time.

It is important to note that pheromone evaporation causes the amount of pheromone on each edge of G to decrease over time. The evaporation process is important because it prevents AS from prematurely converging to a sub-optimal solution. In this way, the AS has the capability of forgetting bad (or even partially good) solutions, which favors a more in-depth exploration of the search space.

IV. Our Proposed Approach

In this section, we present the recent scientific findings that served as our inspiration to modify the AS algorithm for solving the JSSP, together with the description of our proposed approach.

A. Recent findings about real ants

For many years, the "*intelligent behavior*" of some insects has attracted the attention of researchers. In the case of ants, scientists have questioned how can these insects find the shortest path between the food source and the nest. If we observe desert ants on foraging expeditions where the ground is changing all the time, then, one wonders, how can they remember the right path to follow.

A variety of theories have been proposed to explain this ant behavior. For example, that ants use celestial cues to remain oriented in their path to the nest. Other theory states that ants can remember visual cues such as honeybeeps. Additionaly, some experiments have shown that ants can navigate in the dark despite the fact of being almost blind [12]. Other studies have shown that once ants find a good source of food, they teach other ants how to find it [12].

A recent study published in *Science* [13] reveals that counting their steps is a crucial part of the scheme nest-food-nest adopted by ants. Scientists trained desert ants (*cataglyphis fortis*), to walk along a straight path from their nest entrance to a feeder 30 feet away. If the nest or feeder was moved, the ants would break from their straight path after reaching the anticipated spot and search for their goal (i.e., they could reach either the nest or the feeder without problems, even if they were moved).

After that, the scientists glued stilt-like extensions to the legs of some ants to lengthen stride. On the other hand, the researchers shortened other ants' stride length by cutting off their feet and lower legs. As a result, the ants had reduced legs. By manipulating the ants' stride lengths, the researchers could determine whether the insects were using an odometer-like mechanism to measure the distance, or counting off steps with an internal pedometer. The ants on stilts took the right number of steps but the objective was left behind of them. Meanwhile, the ants with short legs never reached it. After getting used to their new legs, the ants were able to adjust their pedometer and succeeded at reaching their goal. Thus, the study concluded that ants count their steps using an internal pedometer.

B. Ant System with steps counter

Based on the observations previously described and knowing that time measurement in JSSP is discrete (such as the number of steps made by an ant), we propose here an AS algorithm with a step counter (*pedometer*) where η in equation (1) will be called *feasibility* and it represents the readiness of an ant to continue its journey without losing time standing by. Additionally, two mechanisms were included to the method in order to extend the search.

The procedure consists of three phases that we will explain next:

Phase 1) Assign priority rule: For driving the search, the ants must select an operation when building a solution. To do so, they use a heuristic value that in case of JSSP can be a priority rule. This rule indicates the relative importance of each operation. Many rules have been proposed in the specialized literature, but none of them is considered the best overall winner, and authors tend to adopt several rules in the AS implementations [8]. In our aproach we incorporated two rules:

LPT: Select the operation with the longest processing time.

SPT: Select the operation with the shortest processing time.

Before starting to build the solution, each ant randomly chooses a priority rule using a probability of 50% at each iteration. The priority rule of each ant is used for calculating the *feasibility*. This calculation is explained and used in phase three, since in this phase only the priority rule is assigned.

Phase 2) Assign α and β values: As we explained in Section III, α and β are parameters used to balance the importance given to the heuristic and the pheromone trail, respectively. These parameters need to be adjusted by the user in the classical AS. In our case, these parameters are assigned at each iteration using the following expressions:

$$\alpha = rndreal\,(0.01, 0.99)$$
$$\beta = 1.0 - \alpha$$

We adopted values within the range from 0.01 to 0.99 in order to guarantee that α and β never reach either zero percent or one hundred percent. This provides a more fair balance, and avoids losing completely the influence of any of these two parameters.

Phase 3) Steps Counter (SC): For explaining this phase, we will use the instance of the problem described in Table I. As typically done in the classical AS, the first operation is allocated at random for ant k. We will assume that the ant k starts moving towards operation $(3, 2)$. In its new position, the ant k determines the set S of possible operations that can be processed later. In our case, the operations are $S = \{(1, 1), (2, 1), (3, 1)\}$ (see Figure 2). All these operations from S need to be processed in machine 1.

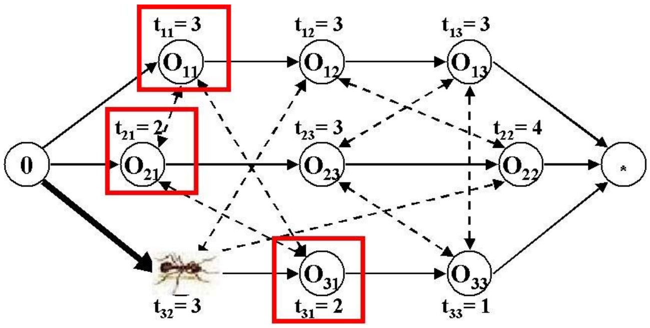


Fig. 2. Set S of possible operations for ant k.

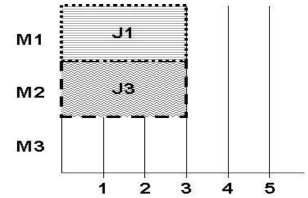
In order to continue with the solution to the problem, it is now necessary to know the starting time for each of the operations in S. The starting time is the number of steps that the ant k needs to give for initiating the operation; thus, it is our "steps counter", which we call SC. This value is computed based on the following criteria:

- The resource to be used as well as the time at which the operation can start, are taken into account, according to the current path generated by the ant.
- The constraint to start imposed by the previous operation.
- If there exists at least one operation in the set S with starting time equal to zero, then, a value of 1 must be added to all the starting times.

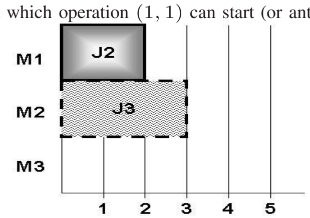
We can observe that the starting time for the operations $(1, 1)$ and $(2, 1)$ is zero, that is $SC_{1,1} = 0$ and $SC_{2,1} = 0$ (see Figures 3a and 3b). For the operation $(3, 1)$ the closest starting time is 3, due to the fact that its previous operation will end until time 3, that is $SC_{3,1} = 3$ (see Figure 3c). Now, using the SC, if ant k selects operations $(1, 1)$ or $(2, 1)$, then, it will save three steps if it selects operation $(3, 1)$ afterwards.

If there is at least one operation with an steps counter value equal to 0 (zero), the value of 1 (one) will be added to the steps counter of all operations; otherwise, the steps counter remains without change. Table II shows a summary of the calculation of the SC, where OP is the operation (i, j), PT_{ij} is the processing time of operation (i, j), SC_{ij} is the steps counter of operation (i, j) and $SC_{ij} + 1$ is an extra value that is computed if there is at least an operation with steps counter equal to zero.

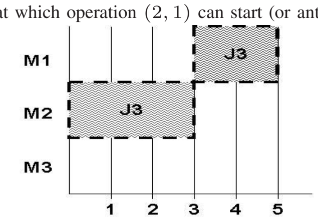
To conclude phase 3, we need to compute the feasibility η. For that sake, we need to apply one equation according to the priority rule selected in phase 1. The equations for the calculation of η are the following:



a) The earliest time at which operation $(1, 1)$ can start (or ant k can start to walk).



b) The earliest time at which operation $(2, 1)$ can start (or ant k can start to walk).



c) The earliest time at which operation $(3, 1)$ can start (or ant k can start to walk).

Fig. 3. Gantt diagrams of the earliest starting times for the operations: a) $(1, 1)$, b) $(2, 1)$ and c) $(3, 1)$, corresponding to the instance described in Table I.

TABLE II

STEPS COUNTER.

OP	PT_{ij}	SC_{ij}	$SC_{ij}+1$
(1,1)	3	0	1
(2,1)	2	0	1
(3,1)	2	3	4

if the priority rule is LPT:

$$\eta_{ij} = \frac{Q}{SC_{ij}} \times PT_{ij}$$

otherwise:

$$\eta_{ij} = \frac{Q}{SC_{ij}} \times \frac{1}{PT_{ij}}$$

where Q is a positive constant which can take any positive value.

Continuing with the example, we assume that the rule is LPT and $Q = 1$ for ant k. Thus, the calculation can be seen in Table III.

TABLE III

CALCULATION OF THE FEASIBILITY η.

OP	PT_{ij}	SC_{ij}	$SC_{ij}+1$	η_{ij}
(1,1)	3	0	1	3
(2,1)	2	0	1	2
(3,1)	2	3	4	0.5

Using the aforementioned procedure, we can know the readiness with which ant k can start *walking* and it is also possible to know the number of steps the ant k is saving as a result of choosing a particular operation.

Now, we use η to complete the calculation in equation (1). For example, assuming $\alpha = 0.5$, $\beta = 0.5$ and $\tau_{ij} = 1$, we can observe that those operations with a high value of readiness will also have a higher probability of being processed (see Table IV).

TABLE IV

CALCULATION OF p_{ij}

OP	PT_{ij}	SC_{ij}	$SC_{ij}+1$	η_{ij}	τ_{ij}	$\tau_{ij}^{\alpha}\eta_{ij}^{\beta}$	p_{ij}
(1,1)	3	0	1	3	1	1.73	0.45
(2,1)	2	0	1	2	1	1.41	0.37
(3,1)	2	3	4	0.5	1	0.70	0.18

The pheromone updating process is similar to the one used in the classical AS, as previously described in Section III (this process is executed at each iteration of the algorithm). This way, phases 1 and 2 help to extend the search generating more diversity while the phase 3 (using the *steps counter*) allows an ant to select the most suitable operation according to the solutions produced so far. We called this proposed approach "*Ant System with steps counter*", which we abbreviate as "AS_{sc}". Figure 4 shows the pseudocode of our proposed approach.

```
PROCEDURE AS_sc
  WHILE termination_criteria
    FOR k = 0 TO ANTS
      Phase 1:
        Select a priority rule for ant k
      Phase 2:
        Assign values α and β for ant k
      Phase 3:
        Set first operation in tabu of ant k
        REPEAT UP to filling tabu of ant k
          Determine set S and SC_ij
          Determine η
          Select next operation using equation (1)
          Move ant k to the next operation
    END FOR
    Update pheromone
  END WHILE
END PROCEDURE
```

Fig. 4. AS with steps counter AS_{sc}.

V. EXPERIMENTS AND COMPARISON OF RESULTS

We present in this section the results obtained by our proposed AS_{sc}, which was tested using a set of well-known JSSP instances found in the OR-Library [14]. The OR-Library contains different types of problems with different sizes and degrees of difficulty. They are grouped in classes. For our experiments, we adopted the LA class because it is composed of 40 different instances (with different sizes and degrees of difficulty). All our tests were executed on a PC with an Intel Pentium III processor running at 2.00 GHz with 512 MB of RAM and using Microsoft Windows XP OS Professional Edition. Our algorithm was implemented in the C programming language and was compiled using Dev C.

The parameters used in our experiments are the following: $\rho = 0.7$ and 1000 iterations. These values were empirically derived. It is worth remarking that only 2 parameters must be tuned in our AS_{sc}, compared with the typical AS. We performed 10 independent runs for each test problem and the number of ants in each test was defined as follows:

$$ANTS = \frac{\sum_{i=0}^{J-1} j_i}{2}$$

where J is the total number of jobs for each instance.

We compared the results obtained by AS_{sc} with respect to those provided by other algorithms found in the specialized literature for which enough information is provided as to allow a more quantitative comparison (i.e., both the best solution found and the number of evaluations required to reach it). For our comparison of results we used two measures: the quality of the best solution found and the number of evaluations of the objective function. The algorithms selected for the comparative study are the following:

- **Artificial Immune System (AIS):** This is an algorithm based on the operation of our immune system, whose results, reported in [15] were very competitive with respect to those provided by a Parallel Genetic Algorithm (PGA) [16], the Hybrid Genetic Algorithm (HGA) [17] and the Greedy Randomized Adaptative Search Procedure (GRASP) [18].
- **Cultural Algorithm (CULT):** This is an algorithm based on social and archaeological theories which try to model cultural evolution. In [19], a cultural algorithm was successfully applied to the JSSP. This approach was selected for comparing results because it requires a low number of evaluations of the objective function with respect to the number used by a Parallel Genetic Algorithm (PGA) [16] and the Greedy Randomized Adaptative Search Procedure (GRASP) [18].
- **Tabu Search (TS):** This is a metaheuristic which adopts memory in order to avoid recycling and getting trapped in local optima. TS is mainly used to solve combinatorial optimization problems. To the best of our knowledge, TS has obtained the best results reported so far for the JSSP [20]. However, it is important to note that TS requires the use of another heuristic (called INSA) to generate the initial solution to be improved by TS. The computational cost consumed by INSA is not reported by its authors in [20].

It is worth noting that, unlike the other approaches with respect to which we compared results, we did not apply any repair mechanism to improve the solutions obtained. Evidently, the use of such repair mechanisms add a computational overhead to the search process (e.g., when using *permissible left shifts* [15]).

Also, it is important to indicate that we did not compare results with respect to other ACO algorithms that have been proposed for the JSSP, due to the lack of results that can be directly comparable. For example, in [8], only five instances are used, and the best known value is not reached in any of them. In [10], the authors only present results with respect to a single instance. In [21], a real-world JSSP is adopted, but no results are reported for a benchmark such as the one adopted here.

1) Quality of solutions: Table V shows the results obtained by our AS_{sc}. The first column indicates the name of the instance. The second column indicates the size of each instance and the third column shows the best known solution (BKS). The last three columns show the best, medium and worst solutions obtained by our approach, respectively. We shown in boldface the best known solution per instance and also the solutions in which our algorithm reached such value. As we can observe in Table V, for problems with size 10 × 5, our AS_{sc} does not have difficulties to find the best known result, with the exception of problem LA04, where our best result is very close to the corresponding best known value (i.e., the difference is of 5 units). Our approach exhibits a similar behavior for most of the problems, despite the fact that the problem sizes grow. For example, in problems of sizes 15 × 5 and 20 × 5, our AS_{sc} reaches the best known solution in 3 out of 5 problems.

The comparison of results of our AS_{sc} with respect to state-of-the-art approaches is presented in Table VI, where C_{max} is the makespan obtained at each instance and #Eval is the number of evaluations of the objective function performed (except for #Eval in the case of INSA, since this value is not reported by their authors [20]).

From these results, we noted that, for problems with 5 machines (LA01, LA02, LA03, LA04 and LA05 of size 10 × 5, LA06, LA07, LA08, LA09, LA10 of size 15 × 5 and LA11, LA12, LA13, LA14 and LA15 of size 20 × 5), the performance of AS_{sc} is quite similar to those provided by the other three compared approaches (AIS, CULT and TS). For problems with 10 machines (LA16, LA17, LA18, LA19 and LA20 of size 10 × 10, LA21, LA22, LA23, LA24, and LA25 of size 15 × 10, LA26, LA27, LA28, LA29, and LA30 of size 20 × 10 and LA31, LA32, LA33, LA34, and LA35 of size 30 × 10) AS_{sc} provided results close to those obtained by AIS, CULT and TS. However, for problems LA36, LA37, LA38, LA39 and LA40 of size 15 × 15 our AS_{sc} could obtain competitive results in 3 of 5 problems, like TS, outperforming AIS and CULT. Summarizing, TS reaches the best known values 85% of the time, CULT reaches the best known values 63% of the time, AIS reaches the best known values 68% of the time, and our AS_{sc} reaches the best known values 55% of the time. However, it is worth noting, that the maximum deviation from the best known value is of 3% for our AS_{sc}, and it only occurs in one instance. For all the others, the maximum deviation from the best known value is of less than 1%. However, as we will discuss next, our approach requires a much lower number of evaluations than the others to reach these results.

2) Number of evaluations of the objective function: In Table VI, we show with boldface in column #Eval the minimum number of evaluations of the objective function used by AS_{sc}, AIS, and CULT. TS only reports the evaluations made after INSA provided the initial solution. In fact, this initial solution may be very good, like in problems LA01 or LA09, where TS reports zero evaluations of the objective function, because the best solution was reached by the only use of INSA.

As was mentioned before, AIS, CULT and TS reported results slightly better than those provided by AS_{sc} in problems of size 10 × 10, 15 × 10, 20 × 10 and 30 × 10. However, the corresponding number of evaluations for the first three approaches is clearly higher. For problems of size 15 × 15, our proposed AS_{sc} provided very competitive results with a clearly lower computational cost, compared with AIS and CULT. TS, in these problems, required more evaluations than AS_{sc} simply to improve an initial solution provided by INSA so that it could reach similar results with respect to our AS_{sc}. If we look at Table VII, the significant savings achieved by our approach (in terms of the number of evaluations

TABLE V

RESULTS OBTAINED BY OUR PROPOSED AS_{sc}.

Instance	Size	BKS	Best	Median	Worst
LA01	10 × 5	**666**	**666**	666	666
LA02	10 × 5	**655**	**655**	655	655
LA03	10 × 5	**597**	**597**	597	597
LA04	10 × 5	**590**	595	595	597
LA05	10 × 5	**593**	**593**	593	593
LA06	15 × 5	**926**	928	928	943
LA07	15 × 5	**890**	**890**	890	890
LA08	15 × 5	**863**	**863**	870	872
LA09	15 × 5	**951**	**951**	951	978
LA10	15 × 5	**958**	965	970	970
LA11	20 × 5	**1222**	**1222**	1222	1235
LA12	20 × 5	**1039**	1040	1040	1045
LA13	20 × 5	**1150**	**1150**	1150	1150
LA14	20 × 5	**1292**	**1292**	1292	1301
LA15	20 × 5	**1207**	1210	1210	1212
LA16	10 × 10	**945**	946	970	971
LA17	10 × 10	**784**	**784**	784	784
LA18	10 × 10	**848**	855	855	872
LA19	10 × 10	**842**	**842**	842	856
LA20	10 × 10	**902**	908	908	908
LA21	15 × 10	**1046**	1055	1060	1060
LA22	15 × 10	**927**	**927**	927	930
LA23	15 × 10	**1032**	1047	1047	1060
LA24	15 × 10	**935**	941	941	943
LA25	15 × 10	**977**	**977**	977	980
LA26	20 × 10	**1218**	**1218**	1218	1220
LA27	20 × 10	**1235**	1240	1240	1260
LA28	20 × 10	**1216**	**1216**	1216	1218
LA29	20 × 10	**1157**	1164	1164	1170
LA30	20 × 10	**1355**	**1355**	1355	1360
LA31	30 × 10	**1784**	1791	1791	1844
LA32	30 × 10	**1850**	**1850**	1850	1912
LA33	30 × 10	**1719**	1754	1754	1772
LA34	30 × 10	**1721**	**1721**	1721	1841
LA35	30 × 10	**1888**	1941	1973	1984
LA36	15 × 15	**1268**	1270	1270	1355
LA37	15 × 15	**1397**	1406	1406	1450
LA38	15 × 15	**1196**	**1196**	1196	1198
LA39	15 × 15	**1233**	**1233**	1233	1236
LA40	15 × 15	**1222**	**1222**	1222	1222

performed) becomes evident. For example, the AIS approach performs, on average, 49 times more evaluations than our AS_{sc}. Analogously, CULT performs, on average, 127 times more evaluations than our AS_{sc}. Even TS performs, on average, 3 times more evaluations than our AS_{sc}, and this cost does not include the number of evaluations performed by INSA, since, as indicated before, that value is not reported in [20]. Thus, we argue that our proposed approach is a good alternative to obtain a good approximation of the optimum of JSSP, with a low number of evaluations of the objective function.

TABLE VII

THE AVERAGE NUMBER OF EVALUATIONS PERFORMED BY EACH ONE OF COMPARED ALGORITHMS.

Algorithm	Average of #Eval	Difference of #Eval with AS_{cp}
AS_{cp}	**3,564**	0
AIS	175,058	171,494
$CULT$	454,525	450,961
TS	11,108	7,544

VI. Conclusions and Future Work

We have presented a variant of the classical Ant System to solve the JSSP. The proposed approach adds a mechanism, which is based on recent scientific studies with real ants. The key feature that we adopted was the counting of steps that real ants seem to perform when constructing a path between their nest and their food source. Our proposed approach uses three phases to calculate the most convenient operation to be performed, according to the schedule encoded in each ant, and this information is used to calculate the readiness of each operation.

We have shown in the paper that our approach was able to reach very competitive results (in terms of the quality of the solutions obtained), but requiring an average number of evaluations that is much lower than that required by the other approaches compared. Thus, we argue that our proposed approach is a viable alternative to obtain reasonably good approximations of the optimum in JSSP, when it is important to perform a low number of objective function evaluations. Additionally, our proposed approach reduces the number of parameters to be fine-tuned by the user with respect to those adopted in the classical AS. In our approach, we only require three parameters (ρ, number of ants and number of iterations), while the classical AS [8] requires five parameters (α, β, ρ, number of ants and number of iterations).

Although we are aware of the existence of more advanced ACO algorithms [4], we decided to adopt the original AS, mainly because of its simplicity. By using the original AS, we only had to perform relatively small changes to incorporate the mechanisms that we wanted to experiment with. Despite the relative simplicity of the search engine adopted, we believe that our results in the JSSP are very encouraging. Evidently, as part of our future work, we are interested in adopting state-of-the-art ACO algorithms as our search engine, to see if our results can improve, particularly in terms of the quality of the solutions obtained. We are also interested in exploring alternative mechanisms to manipulate the pheromone values and in studying mechanisms that allow the online adaptation of ρ and the number of iterations to be performed. We also plan to test our technique with other scheduling problems (e.g., flow shop problems) and we aim to include multiobjective problems as well [22], [23]. Additionally, the validation of our approach in more complex JSSP instances (including perhaps real-world instances) is also one of our future goals.

Acknowledgments

The authors thank Guillermo Leguizamón for his valuable comments which greatly helped them to improve the contents of this paper. The first author acknowledges support from the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through a scholarship to pursue graduate studies at CINVESTAV-IPN. The second author acknowledges support from CONACyT through project number 52048-Y. The third author acknowledges support from CONACyT through project number 42435-Y.

References

[1] M. Dorigo and G. D. Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, UK: McGraw-Hill, 1999, pp. 11–32.

[2] M. R. Garey and D. S., *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[3] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenschaften*, vol. 76, pp. 579–581, 1990.

[4] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, Massachusetts, USA: The MIT Press, 2004.

[5] T. Stützle, "An ant approach for the flow shop problem," in *6th European Congress on Intelligent Techniques and Soft Computing EUFIT'98*, vol. 3, Aachen, Germany, September 1998, pp. 1560–1564.

[6] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the travelling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[7] L. M. Gambardella, E. Taillard, and G. Agazzi, "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Window," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, UK: McGraw-Hill, 1997, pp. 63–76.

[8] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job-shop scheduling," *JORBEL-Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 34, no. 1, pp. 39–53, 1994.

[9] S. van der Swaan and C. Marques, "Ant colony optimization for job shop scheduling," in *Proceedings of Third workshop on Genetic Algorithms and Artificial Life*, 1999.

[10] C. Blum and M. Samples, "An ant colony optimization algorithm for FOP shop scheduling: A case study on different pheromone representations," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, vol. 2. IEEE Press, 2002, pp. 1558–1563.

[11] T. Yamada and R. Nakano, "Genetic algorithms for job-shop scheduling problems," in *Proceedings of Modern Heuristic for Decision Support*. London: ,UNICOM seminar, March 1997, pp. 67–81.

[12] B. Carey, "When ants go marching, they count their steps," 2006, http://www.livescience.com/animalworld/060629_ant_pedometers.html.

[13] M. Wittlinger, R. Wehner, and H. Wolf, "The Ant Odometer: Stepping on Stilts and Stumps," *Science*, vol. 312, no. 5782, pp. 1965–1967, June 2006.

[14] J. E. Beasley, "OR-Library: Distributing Test Problems by Electronic Mail," *Journal of the Operations Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

TABLE VI

COMPARISON OF RESULTS.

Instance	Size	BKS	Algorithms									
			AS_{sc}		AIS		$CULT$		$INSA$		TS	
			C_{max}	#Eval	C_{max}	#Eval	C_{max}	#Eval	C_{max}	#Eval	C_{max}	#Eval
LA01	10×5	666	**666**	**570**	**666**	2345	**666**	4000	**666**	-	**666**	0
LA02	10×5	655	**655**	**1285**	**655**	171095	**655**	20000	722	-	**655**	5353
LA03	10×5	597	**597**	**1487**	**597**	545876	603	8000	681	-	**597**	7546
LA04	10×5	590	595	**909**	**590**	4826	**590**	700000	659	-	**590**	21
LA05	10×5	593	**593**	386	**593**	**84**	**593**	2000	**593**	-	**593**	0
LA06	15×5	926	928	3533	**926**	**135**	**926**	2000	950	-	**926**	6
LA07	15×5	890	**890**	1671	**890**	**1664**	**890**	4000	976	-	**890**	23
LA08	15×5	863	**863**	1524	**863**	**1468**	**863**	4000	868	-	**863**	1
LA09	15×5	951	**951**	1003	**951**	**548**	**951**	2000	**951**	-	**951**	0
LA10	15×5	958	965	**1524**	**958**	2624	**958**	2000	**958**	-	**958**	0
LA11	20×5	1222	**1222**	1542	**1222**	**157**	**1222**	2000	1293	-	**1222**	11
LA12	20×5	1039	1040	2227	**1039**	10521	**1039**	**2000**	1044	-	**1039**	1
LA13	20×5	1150	**1150**	1783	**1150**	**880**	**1150**	2000	1154	-	**1150**	1
LA14	20×5	1292	**1292**	1997	**1292**	**27**	**1292**	2000	1328	-	**1292**	1
LA15	20×5	1207	1210	**3300**	**1207**	7838	**1207**	8000	1323	-	**1207**	90
LA16	10×10	945	946	**1924**	**945**	26451	946	25000	1077	-	**945**	19695
LA17	10×10	784	**784**	**2229**	**784**	41266	**784**	25000	821	-	**784**	1031
LA18	10×10	848	855	**2020**	**848**	18636	**848**	140000	926	-	**848**	18023
LA19	10×10	842	**842**	**1657**	**842**	33531	**842**	32000	971	-	**842**	15039
LA20	10×10	902	908	**2208**	907	29171	907	100000	1003	-	**902**	37959
LA21	15×10	1046	1055	**3987**	**1046**	328405	1089	1700000	1179	-	1047	3732
LA22	15×10	927	**927**	**3091**	**927**	458947	945	1700000	1032	-	**927**	2246
LA23	15×10	1032	1047	**3543**	**1032**	42285	**1032**	200000	1132	-	**1032**	208
LA24	15×10	935	941	**3304**	**935**	473731	964	1000000	1021	-	939	30001
LA25	15×10	977	**977**	**3092**	979	407427	993	1000000	1147	-	**977**	27677
LA26	20×10	1218	**1218**	**3576**	**1218**	478017	**1218**	1700000	1397	-	**1218**	2106
LA27	20×10	1235	1240	**4834**	1240	476269	1269	200000	1466	-	1236	8971
LA28	20×10	1216	**1216**	**3300**	**1216**	484701	1241	1800000	1485	-	**1216**	16630
LA29	20×10	1157	1164	**3703**	1170	600328	1189	1800000	1385	-	1160	63469
LA30	20×10	1355	**1355**	**2268**	**1355**	113932	**1355**	200000	1463	-	**1355**	282
LA31	30×10	1784	1791	9534	**1784**	**6880**	**1784**	15000	1966	-	**1784**	84
LA32	30×10	1850	**1850**	10302	**1850**	**8401**	**1850**	40000	1982	-	**1850**	135
LA33	30×10	1719	1754	11583	**1719**	**6039**	**1719**	20000	1767	-	**1719**	9
LA34	30×10	1721	**1721**	**10754**	**1721**	21603	**1721**	160000	1844	-	**1721**	302
LA35	30×10	1888	1941	10039	**1888**	**7849**	**1888**	160000	1967	-	**1888**	48
LA36	15×15	1268	1270	**3986**	1281	272736	1292	1000000	1445	-	**1268**	64473
LA37	15×15	1397	1406	**3635**	1408	497661	1451	1400000	1726	-	1407	41183
LA38	15×15	1196	**1196**	**4091**	1204	506151	1276	1800000	1307	-	**1196**	15579
LA39	15×15	1233	**1233**	**4389**	1249	460554	1266	200000	1393	-	**1233**	31991
LA40	15×15	1222	**1222**	**4785**	1228	451273	1265	1000000	1387	-	1229	30417

[15] C. A. Coello Coello, D. Cortés Rivera, and N. Cruz Cortés, "Job shop scheduling using the clonal selection principle," in *Adaptive Computing in Design and Manufacture VI*, I. Parmee, Ed. London: Springer, April 2004, pp. 113–124.

[16] J. F. Gonçalves and N. C. Beirão, "Um Algoritmo Genético Baseado em Chaves Aleatórias para Sequenciamento de Operações," *Revista Associação Portuguesa de Desenvolvimento e Investigação Operacional*, vol. 19, pp. 123–137, 1999.

[17] José Fernando Gonçalves and Jorge José Mendes and Mauricio G. C. Resende, "A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem," AT&T Labs Research, 180 Park Avenue, Florham Park, NJ 07932 USA, Tech. Rep. TD-5EAL6J, September 2002.

[18] R. M. Aiex, S. Binato, and M. G. C. Resende, "Parallel GRASP with path-relinking for job shop scheduling," *Parallel Computing*, vol. 29, no. 4, pp. 393–430, 2003.

[19] R. Landa Becerra and C. A. Coello Coello, "A Cultural Algorithm for Solving the Job Shop Scheduling Problem," in *Knowledge Incorporation in Evolutionary Computation*, Y. Jin, Ed. Springer-Verlag, 2005, pp. 37–55.

[20] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Management Science*, vol. 42, no. 6, pp. 797–813, 1996.

[21] J. Montgomery, C. Fayad, and S. Petrovic, "Solution Representation for Job Shop Scheduling Problems in Ant Colony Optimisation," in *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, Eds. Springer. Lecture Notes in Computer Science Vol. 4150, September 2006, pp. 484–491.

[22] V. T'kindt and J.-C. Billaut, *Multicriteria Scheduling. Theory, Models and Algorithms*. Berlin: Springer, 2002, iSBN 3-540-43617-0.

[23] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and Computers in Simulation*, vol. 60, no. 3-5, pp. 245–276, September 2002.