Engineering Applications of Artificial Intelligence 26 (2013) 150–161

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research

R.F. Tavares Neto*, M. Godinho Filho

Industrial Engineering Department, Federal University of Sao Carlos, Rod. Washington Luis, Km 235, Sao Carlos, Sao Paulo 13565-905, Brazil

ARTICLE INFO

Article history:
Received 16 June 2011
Received in revised form
10 February 2012
Accepted 20 March 2012
Available online 3 May 2012

Keywords:
Ant Colony Optimization
Scheduling problems
Literature review

ABSTRACT

Ant Colony Optimization is a swarm intelligence approach that has proved to be useful in solving several classes of discrete and continuous optimization problems. One set, called scheduling problems, is extremely important both to academics and to practitioners. This paper describes how the current literature uses the ACO approach to solve scheduling problems. An analysis of the literature allows one to conclude that ACO is a hugely viable approach to solve scheduling problems. On the basis of the literature review, we were not only able to derive certain guidelines for the implementation of ACO algorithms but also to determine possible directions for future research.

1. Introduction

Ant Colony Optimization (ACO) is a meta-heuristic approach proposed by Colorni et al. (1991) and improved in later research (e.g., see Dorigo et al., 1996 for the Ant Colony System and Stützle and Hoos, 2000 for the Max–Min Ant System). The behavior common to all approaches involving ant-based algorithms lies in the mimicry of the behavior used by "real" ants to find the optimal path between their nest and a food source.

The earlier application of ACO was to solve the well-known NP-Hard Traveling Salesman Problem (Colorni et al., 1991; Dorigo et al., 1996; Dorigo and Gambardella, 1997). In this problem, there is a graph in which each node corresponds to a city, and the arcs correspond to the distances between cities. The problem consists of obtaining a minimal tour (sequence of cities) length that contains all the nodes.

Several studies have applied ACO to solve different discrete and continuous optimization problems, such as vehicle routing, quadratic assignment problems and graph coloring. Dorigo and Stützle (2004) reported more than 30 problems where ACO-based algorithms have been used successfully.

One of these applications involves scheduling problems. With the significance of these problems recognized because of their impact on real environments and their academic relevance (e.g., Pinedo, 2009), "scheduling problems" are a huge set of problems, and are mostly NP-Hard, that try to deal with a simple question: given a set of jobs, a set of resources, a set of constraints, and an objective function, how should the jobs be allocated to the resources?

Answering this question, however, usually requires complex and/or time-costly procedures. The great advantage in using a meta-heuristic such as ACO to obtain near-optimal solutions is that the time required to solve the problem is usually acceptable even though the 100% optimal solution may not be achieved.

This paper aims at reviewing and classifying published studies that use ACO to solve scheduling problems, and it focuses on the four classical manufacturing environments (single machine, parallel machine, flowshop and jobshop). Different scheduling problems, such as service scheduling (e.g., see Gutjahr and Rauner, 2007 for an ACO approach to the nurse-scheduling problem) are not included in the revision. In addition, this paper concentrates only on uses of the ACO meta-heuristic on its own: any "hybrid approach" is disregarded.

This paper contribution is twofold. In the first instance, it aims to help researchers apply this technique in production scheduling, demonstrating how research is being carried out in the literature. Therefore, some guidelines relating to the characteristics of the ACO algorithm applied to scheduling problems are derived. In the second instance, certain directions for future research in the field are highlighted.

To present the results, this paper is divided into six sections: Section 2 deals with the basics of scheduling problems; Section 3 is concerned with the ACO; Section 4 has to do with the classification method proposed and the papers reviewed; Section 5 presents an overview of the ACO application to scheduling problems; Section 6 shows a quantitative analysis of the literature; and Section 7 presents the final remarks.

* Corresponding author.
E-mail addresses: tavares@dep.ufscar.br (R.F. Tavares Neto), moacir@dep.ufscar.br (M. Godinho Filho).

0952-1976/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2012.03.011

2. Scheduling problems

Scheduling problems are devoted to allocating tasks to resources (Baker, 1943). Scheduling theory contains an almost unlimited set of problems (Brucker, 2007). In this paper, according to the current literature, scheduling problems are characterized by three main attributes: (i) the manufacturing environment, (ii) the constraints, and (iii) the objective function. By understanding these three attributes, it is possible to use the scheduling problem classification scheme proposed by Graham et al. (1979). The authors state that it is possible to represent a scheduling problem using the notation $\alpha/\beta/\gamma$, where α represents the manufacturing environment, β represents the constraints and γ represents the objective function. This section is devoted to presenting certain values of these three components that will be used in the following sections.

2.1. Manufacturing environments

The first question that arises in describing a scheduling problem has to do with production flow. Graham et al. (1979) described four main categories of scheduling problems as follows:

(i) A single machine environment ($\alpha=1$), where all the jobs must be processed by a single machine;
(ii) A parallel machine environment, where all the jobs must be processed by just one machine. The machines in this environment can be identical ($\alpha=P_m$), uniform ($\alpha=Q_m$) or unrelated ($\alpha=R_m$). Here m represents the number of machines;
(iii) A flowshop environment ($\alpha=F_m$), where each task contains a set of operations that must be performed on specific machines; each task uses the same sequence of resources, and m represents the number of stages in the production flow;
(iv) A job shop environment ($\alpha=J_m$), where each task contains a set of operations that must be performed on specific machines. In this case, each task contains its own production flow and, as with the flowshop, m represents the number of stages in the production flow.

This paper uses the notation $\alpha=M_m$. Following Li et al. (2009), this notation is used to indicate a manufacturing environment with parallel machines that allow the execution of batches.

2.2. Constraints

Practitioners usually find themselves bound by specific characteristics of the target scheduling problem. These constraints can be, for example, related to the sequence (e.g., when $\beta=$ prmu, the job assignment sequence of a flowshop environment is the same throughout the production flow), or to a maximum budget that can be used to outsource some jobs (when $\beta=$ Budget). Table 1 shows the constraints adopted by this paper.

2.3. Common objective functions

There are several performance measures used to describe scheduling problems. Although a description of all of these performance measures is beyond the scope of this paper, some must be defined. Hence, let us use a set of n jobs J_i, $i\in n$ released at r_i with processing time p_i and due date d_i. For any sequence, it is straightforward to calculate certain indicators, as follows:

(i) The *completion time* of a job J_i ($C_i=t_0+p_i$), t_0 is the start time of J_i;
(ii) The *lateness* of a job J_i ($L_i=d_i-C_i$);
(iii) The *earliness* of a job J_i ($E_i=\max\{d_i-C_i,0\}$);
(iv) The *tardiness* of a job J_i ($T_i=\max\{C_i-d_i,0\}$);
(v) The *makespan* of the sequenced jobs ($M=\max\{C_i\}$).

Table 1
Symbols used to describe scheduling problems presented in this paper.

Symbol (**position**)	Meaning
$1(\alpha)$	Single machine manufacturing environment
$P_m(\alpha)$	Identical parallel machine manufacturing environment
$Q_m(\alpha)$	Uniform parallel machine manufacturing environment
$R_m(\alpha)$	Unrelated parallel machine manufacturing environment
$F_m(\alpha)$	Flowshop manufacturing environment
$J_{n,m}(\alpha)$	Jobshop manufacturing environment
$r_i(\beta)$	Indicate a set of tasks with different release dates
batch(β)	Indicate the possibility of generate production batches
incompatible(β)	Indicate that exists tasks that cannot belong to the same processing batch
$s_{ij}(\beta)$	Sequence dependent setup
rush(β)	Rush orders
prmu(β)	Used to describe a permutational flowshop environment
$Q_i(\gamma)$	Qual-run-time
window(β)	Indicate the jobs contains lower and upper bounds of C_i
$A_{ij}(\beta)$	Indicate that some jobs are known only after the sequence is processing
nwt(β)	Indicates a no-wait constraint
$\sum C_i(\gamma)$	Sum of the completion time of all tasks
$\sum T_i(\gamma)$	Sum of tardiness of all tasks
$\sum w_i\cdot T_i(\gamma)$	Sum of weighted tardiness of all tasks
$\sum F_i(\gamma)$	Sum of the flowtimes of all tasks
CTV(γ)	Completion time variance
M or $C_{\max(\gamma)}$	Makespan of a sequence
idle(γ)	Indicates the idle time of all machines
$T_{\max}(\gamma)$	Maximum tardiness
$OC(\gamma)$	Total outsource cost
Budget(β)	Maximum value of OC
$\alpha,\delta(\gamma)$	Constants used to balance the fitness function

Table 1 shows all the terms used to describe the objective functions of the scheduling problems presented in this paper.

2.4. A simple example

This section shows a simple example of a scheduling problem. Let S_0 be a set of four jobs to be sequenced in a single machine environment. These four jobs are described in Table 2. In this table, column i indicates the job index; column p_i indicates the processing time of the job; and d_i indicates the due dates. These tasks can be sequenced according a large number of rules, generating different values for the performance measures. For example, Table 3 presents the completion time C_i and the tardiness T_i for each job, using the First In First Out (FIFO) rule. Table 4 presents the same performance measure ordering the jobs by their due dates (rule EDD—Earliest Due Date).

The goal of scheduling algorithms is to enhance (by maximizing or minimizing) the value of a certain performance measure. One notes in the presented example that the maximum tardiness of the jobs ordered by EDD is smaller than the maximum tardiness obtained by the FIFO rule.

To achieve this goal with small-size problems, the solution is straightforward: simply test all possibilities relating to each possible job sequence. However, when the problem size increases, different strategies must be used, due to the prohibitive computational time involved.

In the following section, the ACO algorithm will be presented. This algorithm has been used to solve scheduling problems, generating satisfactory results in a affordable computational time.

Table 2
An example set of four jobs.

i	p_i	d_i
1	20	30
2	15	20
3	30	40
4	5	35

Table 3
The jobs presented in Table 2 ordered by FIFO rule.

i	p_i	d_i	C_i	T_i
1	20	30	20	0
2	15	20	35	15
3	30	40	65	25
4	5	35	70	35

Table 4
The jobs presented in Table 2 ordered by EDD rule.

i	p_i	d_i	C_i	T_i
2	15	20	15	0
1	20	30	35	5
4	5	35	40	5
3	30	40	70	30

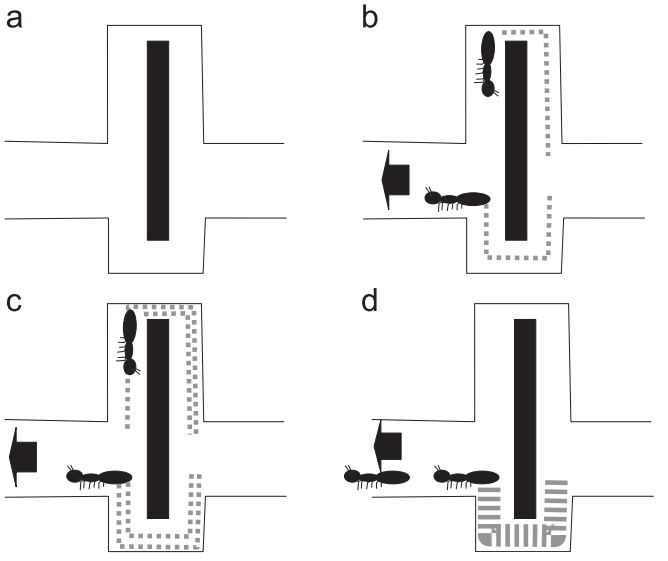

Fig. 1. Basic Ant Colony Optimization behavior at different time stamps. The gray areas represent the amount of pheromones on each trail.

3. Ant Colony Optimization

Ant Colony Optimization (ACO) is a class of metaheuristics proposed by Marco Dorigo in the early 1990s (e.g., Colorni et al., 1991). It has been used in dealing with several classes of problems, and has considerable practical appeal, such as in vehicle routing (e.g., Bullnheimer et al., 1997), examination scheduling problem (e.g., Dowsland and Thompson, 2005) and scheduling (e.g., Zhou et al., 2007). The concept behind ACO is based on the "natural" algorithm used by real ants to generate a near-optimal trail between their nest and the food source, as shown in Fig. 1. One of the main concepts with regard to this mechanism is the effect of pheromones on ants. In nature, it was found that real ants, when seeking food, are highly influenced by pheromones. This influence is mathematically modeled as a weighted, ponderated, random function, where the weight is calculated using existing pheromones.

As shown in Fig. 1, the behavior displayed by real ants to generate a near optimal trail can be explained in four steps: (a) At an initial moment, the environment is "clean", and both ants can choose any of the paths with the same probability; (b) Both ants choose a path, and in this case, one ant chooses a shorter path and the other a longer one. When they move, they deposit chemical substances called pheromones. (c) When the cycle repeats, the shorter path will have a stronger pheromone trail more quickly (since the corresponding ant arrives sooner). To choose the next move, an ant uses a probability function weighted according to the amount of pheromones on the trail. Thus, more ants will choose the shorter trail. (d) After a certain time, the first pheromones that were dropped evaporate, and the pheromone trail on the shorter path becomes dominant. In this case, all the ants will choose the shorter trail.

The idea of the ACO algorithm is to mimic this behavior. This mimicry is performed by generating a pheromone matrix $n \times m$, used by two main operations: the pheromone level adjustment (also known as *pheromone deposit* and *pheromone evaporation* rules) and a probabilistic rule that chooses a destination based on the pheromone level (the *state transition rule*).

The full pseudo-code of the ACO algorithm is shown in Algorithm 1. In this algorithm, m ants are used, in each cycle, to build a full solution. To perform this task, the solution is divided into *steps* (e.g., in scheduling problems a step is usually the assignment of one job into one position of the final sequence). To determine each step, two rules are used, as shown in Eqs. (1) and (2).

$$s = \begin{cases} \arg\max_{u \in N^*}\{[\tau(i,j)] \cdot [\eta(i,j)]^{\beta}\} & \text{if } q \le q_0 \text{ (exploration)} \\ S & \text{otherwise (exploitation)} \end{cases} \quad (1)$$

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}(t)^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N^*} \tau_{il}(t)^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } j \in N^* \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where

- k represents the index of the ant,
- i and j are two nodes on the graph that represents the search space,
- p_{ij}^k represents the probability of ant k choose node j after node i,
- s is an arc that connects i and j,
- q is a random value between 0 and 1,
- $0 \le \alpha \le 1$, $0 \le \beta \le 1$ $0 \le q_0 \le 1$ are parameters chosen during the implementation of the algorithm,
- S is a trail chosen according to the probability given by (2),
- N^* is the set of nodes that can be visited by the ant,
- $\tau(i,j)$ is the pheromone intensity between nodes i and j,
- $\eta(i,j)$ is a problem-specific value, referred to by some authors as *visibility* from node i to node j.

The update rules consists of two terms: the first, is the evaporation of the existing pheromone; the second, is the amount of added pheromone on the trail. This rule is shown in Eqs. (3) and (4)

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta_{ij}^k(t) \quad (3)$$

Table 5
Some of the most used ACO algorithms.

Algorithm	Source
Ant System (AS)	Colorni et al. (1991)
ASElite	Dorigo et al. (1996)
ASRank	Bullnheimer et al. (1997)
Ant Colony System (ACS)	Dorigo and Gambardella (1997)
Max–Min Ant System (MMAS)	Stützle and Hoos (2000)
Best–Worst Ant System (BWAS)	Cordon et al. (2000)

$$\Delta_{ij}^{k}(t) = \begin{cases} 1/L_k & \text{if ant } k \text{ goes from node } i \text{ to node } k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where

- $0 \le \rho \le 1$ is an algorithm parameter, usually named *evaporation parameter*,
- L_k is some value that indicates how much the pheromone trail should increase. Sometimes, the $1/L_k$ is replaced by a constant Q.

Algorithm 1. The ACO pseudo-code.

```
1   Initialize
2   repeat (At this level, each execution is called iteraction)
3       Each ant is positioned on the initial node
4       repeat(At this level, each execution is called step)
5           Each ant applies a state transition rule to
6           increment the solution
7           Apply the pheromone local update rule
8       until all the ants have built a complete solution
9       Apply a local search procedure
        Apply the pheromone global update rule
10  until the stop criteria is satisfied
```

Using this initial approach, one can produce several variations of ACO. The first ant-based algorithm found in our research is the Ant System (AS). This algorithm is modified, generating the ASElite (that only allows the pheromone of the best solution to be updated) and ASRank (where the pheromone update is performed only to a subset of the solutions found, according a rank). The Ant Colony System (ACS) incorporates an exploration–exploitation mechanism. The Max–Min Ant System (MMAS) fixes the pheromone bounds. The Best–Worst Ant System (BWAS) allows the pheromone update to the best and the worst solutions found. Table 5 presents the aforementioned ACO algorithms and the first implementation of each one in the literature. For this paper, three main algorithms are important: (i) the ACO algorithm; (ii) the ACS algorithm, based on ACO, using an exploitation/exploration mechanism with an elitist strategy; and (iii) the MMAS algorithm, which is an evolution of the ACS with minimal and maximal bounds of pheromone levels.

4. Review and classification of the application of the ACO algorithm to scheduling problems

A full search of the databases Compendex (http://www.engineeringvillage.com) and Scopus (http://www.scopus.com) was made, and 47 papers were found. On the basis of this literature review, a classification scheme was proposed, and is shown in Section 4.1. Section 4.2 presents the papers classified according to the scheme.

4.1. Classification scheme for understanding ACO approaches to scheduling problems

To better understand results found in the literature, a classification scheme is proposed and applied to the review performed in this paper. This classification scheme consists of seven attributes, as follows:

Attribute 1 The scheduling problem addressed by the paper.
Attribute 2 Base algorithm used, according to Table 5.
Attribute 3 Indicates whether the algorithm presented in the paper uses a *job-to-position* approach to determine the sequence, namely, whether the algorithm looks for a job to be positioned at an absolute position in the list (e.g., it will determine that job J_2 is in the 12th position in the final sequence). This attribute can assume the values "yes" or "no".
Attribute 4 Indicates whether the algorithm presented uses a *job-to-job* approach to determine the sequence: that is, whether the position of a job is dependent on the previous job sequenced (e.g., it will determine that job J_3 will be inserted after job J_9). This attribute can assume the values "yes" or "no".
Attribute 5 Indicates whether the algorithm presented uses some dominance criteria. This attribute can assume the values "yes" or "no".
Attribute 6 Indicates the pheromone initialization of the trails, which can use a constant value (*cte*) or a heuristic rule (*heu*).
Attribute 7 Indicates how the algorithm builds the visibility function. This attribute can assume the following values: *cte* if the visibility function is calculated during the initialization phase of the algorithm and kept constant; *f(t)* if the value changes during the construction of the solution; and 1 if the visibility rule is not used in the transition rule (in this case, the problem characteristics are usually incorporated into the algorithm by a specific pheromone evaporation rule and/or by a local search mechanism).

4.2. Classification of the literature survey

Table 6 presents the papers found in the literature review. These papers are classified according to the seven attributes of the classification scheme proposed.

5. An overview of the literature

An overview of the literature is presented in this section, which is divided as follows: Sections 5.1–5.4 present papers related to the four main scheduling environments (single machine, parallel machines, flowshop, and job shop). Within each section, the papers are presented according to the objective function addressed.

5.1. ACO applied to single machine scheduling problems

Scheduling problems in manufacturing environments containing a single machine consist of finding a sequence where a set of n tasks must be accomplished to improve the value of an objective function. To solve this problem, a common approach is to generate a fully interconnected graph (e.g., Merkle and Middendorf, 2000). Fig. 2 shows an example of a graph generated for a 4 jobs scheduling problem.

Table 6
Results of the application of the classification method proposed.

	Source	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7
1	Bauer et al. (1999)	$1\|\|\sum T_i$ e $1\|\|\sum w_i \cdot T_i$	ACO	No	Yes	No	*cte*	$f(t)$
2	den Besten et al. (2000)	$1\|\|\sum w_i \cdot T_i$	ACO	No	Yes	No	*cte*	*cte*
3	Merkle and Middendorf (2000)	$1\|\|\sum w_i \cdot T_i$	ACO	No	Yes	Yes	*cte*	$f(t)$
4	Merkle and Middendorf (2003)	$1\|\|\sum w_i \cdot T_i$	ACO	No	Yes	Yes	*cte*	$f(t)$
5	Holthaus and Rajendran (2005)	$1\|\|\sum w_i \cdot T_i$	ACO	Yes	No	No	*heu*	1
6	Cheng et al. (2009)	$1\|\|\sum T_i$	ACO	No	Yes	Yes	*cte*	$f(t)$
7	Gagn et al. (2002)	$1/s_{ij}/\sum T_i$	ACO	No	Yes	Yes	*cte*	*cte*
8	Liao and Juan (2007)	$1/s_{ij}/\sum w_i \cdot T_i$	ACO	Yes	Yes	Yes	*heu*	$f(t)$
9	Anghinolfi et al. (2008)	$1/s_{ij}/\sum w_i \cdot T_i$	ACS	No	Yes	No	*cte*	$f(t)$
10	Kashan and Karimi (2008)	1/batch, incompatible, $s_{ij}/\sum w_i \cdot T_i$	ACO	No	Yes	No	*cte*	*cte*
11	Thiruvady et al. (2009)	$1/s_{ij}/M$	MMAS	No	Yes	Yes	*cte*	*cte*
11	Xu et al. (2012)	$1/r_j,s_j$, batch/$C_{\max}$	ACO	No	Yes	No	*cte*	*cte*
12	Cheng et al. (2008)	1/batch/$C_{\max}$	ACO	No	Yes	No	*heu*	*cte*
13	Sankar et al. (2005)	$P_m/s_{ij}/C_{\max}$	ACO	No	Yes	No	*cte*	*cte*
14	Behnamian et al. (2009)	$P_m/s_{ij}/C_{\max}$	MMAS	No	Yes	No	*cte*	*cte*
15	Gatica et al. (2010)	$P_m\|\|T_{\max}$	ACS	Yes	No	No	*cte*	*cte*
16	Raghavan and Venkataramana (2006)	P_m/batch, incompatible/$\sum w_i \cdot T_i$	ACO	No	Yes	No	*cte*	*cte*
17	Li et al. (2008)	P_m/A_{ij},batch, incompatible/$\sum w_i \cdot T_i$	ACO	No	Yes	No	*cte*	$f(t)$
18	Li et al. (2009)	$P_m/A_{ij},Q_i$,batch, incompatible/$\sum w_i \cdot T_i$	ACO	No	Yes	No	*cte*	$f(t)$
19	Zhou et al. (2007)	$R_m\|\|\sum w_i \cdot T_i$	ACS	No	Yes	No	*heu*	$f(t)$
20	Monch (2008)	$R_m\|\|\sum w_i \cdot T_i$	ACO	Yes	No	No	*heu*	$f(t)$
21	Arnaout et al. (2008)	$R_m/s_{ijk}/C_{\max}$	ACO	No	Yes	No	*cte*	*cte*
22	Arnaout et al. (2009)	$R_m/s_{ijk}/C_{\max}$	ACO	No	Yes	No	*cte*	*cte*
23	Ying and Liao (2003)	F_m/prmu/M	ACS	No	Yes	No	*heu*	*cte*
24	Rajendran and Ziegler (2004)	F_m/prmu/M	MMAS	Yes	No	No	*heu*	1
25	Ahmadizar et al. (2007)	F_m/prmu/M	ACO	No	Yes	No	*heu*	$f(t)$
26	Chen et al. (2008)	F_m/prmu/M	ACS	Yes	No	No	*cte*	1
27	Zhou and Qingshan (2009)	F_m/prmu/M	MMAS	Yes	No	No	*cte*	1
28	Ying and Lin (2007)	$F_m\|\|M$	ACS	No	Yes	No	*heu*	*cte*
29	Shyu et al. (2004)	F_2/nwt,setup/M	ACO	No	Yes	No	*heu*	*cte*
30	Yan-hai et al. (2005)	$F_m\|\|$prmu,rush,M	ACO	No	Yes	No	*heu*	*cte*
31	T'kindt et al. (2002)	$F_2\|\|\sum C_i$ e $F_2\|\|$Lex($C_{\max}$, $\sum Ci$)	MMAS	No	Yes	No	*cte*	*cte*
32	Li et al. (2011)	$F_m/s_{ij}/\sum C_i$	MMAS	No	Yes	No	*cte*	*cte*
33	Tavares Neto and Godinho Filho (2011)	F_m/prmu, budget/$(1-\delta)\cdot M+\delta\cdot OC$	MMAS	No	Yes	Yes	*cte*	*cte*
34	Rajendran and Ziegler (2004)	F_m/prmu/M e F_m/prmu/$\sum F_i$	MMAS	Yes	No	No	*cte*	1
35	Rajendran and Ziegler (2005)	F_m/prmu/$\sum F_i$	MMAS	Yes	No	No	*heu*	1
36	Gajpal and Rajendran (2006)	F_m/prmu/CTV	ACO	Yes	No	No	*cte*	1
37	Li and Zhang (2006)	$F_2\|\|\alpha\cdot\sum C_i+(1-\alpha)\cdot M$	ACO	Yes	Yes	Yes	*heu*	$f(t)$
38	Pasia et al. (2006)	$F_m\|\|\sum T_i,M$	ACO	Yes	No	No	*cte*	$f(t)$
39	Yagmahan and Yenisey (2008)	F_m/prmu/$\sum C_i,M,\sum F_i$,idle	ACS	No	Yes	Yes	*cte*	*cte*
40	Al-Anzi and Allahverdi (2009)	F_2/prmu/$u\cdot M+v\cdot\sum Ci$	ACO	No	Yes	No	*cte*	$f(t)$
41	Lin et al. (2008)	F_m/prmu/$u\cdot M+v\cdot\sum C_i$	ACO	Yes	Yes	Yes	*heu*	$f(t)$
42	Marimuthu et al. (2009)	F_m/batch/M e F_m/batch/$\sum F$	ACO	Yes	No	No	*cte*	*cte*
43	Huang and Yang (2009)	$F_m/r_j,S_{ij}/\alpha\sum_{i=1}^{m}\mathrm{MIT}_i+\beta\sum_{i=1}^{m}\sum_{j=1}^{B_j}\sum_{t=1}^{n}\mathrm{JWT}_{ijt}+\gamma\sum_{i=1}^{m}T_i$	ACS	Yes	No	No	*cte*	*heu*
44	Colorni et al. (1994)	$J_{n,m}\|\|M$	ACO	No	No	No	*cte*	*cte*
45	Zhang et al. (2010)	$J_{n,m}\|\|M$	ACO	Yes	No	No	*cte*	*cte*
46	Shih-Pang et al. (2011)	$J_{n,m}\|\|M$	ACO[a]	No	No	No	*cte*	*cte*
47	Yoshikawa and Terai (2006)	$J_{n,m}\|\|M$	ACO	No	Yes	No	*cte*	*cte*
48	Udomsakdigool and Kachitvichyanukul (2008)	$J_{n,m}\|\|M$	ACO	No	Yes	No	*cte*	$f(t)$
49	Heinonen and Pettersson (2007)	$J_{10,10}\|\|M$	ACO	No	Yes	No	*cte*	$f(t)$
50	Zhuo et al. (2007)	$J_{10,10}\|\|M$	ACS	No	Yes	No	*heu*	$f(t)$
51	Huang and Liao (2008)	$J_{10,10}\|\|M$	ACO	No	Yes	No	*heu*	$f(t)$
52	Eswaramurthy and Tamilarasi (2009)	$J_{n,m}\|\|M$	ACS	No	Yes	No	*cte*	*cte*
53	Mirabi (2011)	$J_{n,m}/s_{ij}/M$	ACO	No	No	No	*cte*	*cte*
54	Huang and Yang (2008)	$J_{n,m}$/window/$\sum(u\cdot E_i+v\cdot T_i)$	ACO	Yes	No	No	*cte*	*cte*

[a] The paper compares AS, EAS, AS-Rank.

5.1.1. *Tardiness-related scheduling problems*

We found 10 papers dealing with single machine environment scheduling seeking to minimize the total tardiness or total weighted tardiness ($\sum T_i$ and $\sum w_i \cdot T_i$, respectively).

Bauer et al. (1999) designed an algorithm for both approaches ($1\|\cdot T_i$ and $1\|\sum w_i \cdot T_i$), using well-known dispatch rules such as Earliest Due Date (EDD) to determine the visibility values. The authors used the Adjacent Pairwise Interchange (API) and a 2-opt heuristic as a local search.

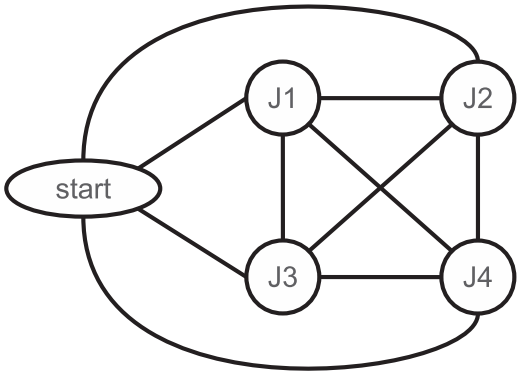

Fig. 2. A simple 4-job fully interconnected graph.

den Besten et al. (2000) approached the problem $1\|\sum w_i \cdot T_i$. Their results showed that the ACO algorithm was able to obtain the best solution for all test instances. The local search algorithms used were an insert–interchange and an interchange–insert.

The papers by Merkle and Middendorf (2000) and Merkle and Middendorf (2003) proposed improvements to the work of Bauer et al. (1999) relating to the problem $1\|\sum w_i \cdot T_i$. The authors developed a new visibility rule that constrained a deterministic criterion to restrict the search space and avoid generating poor solutions.

Also regarding the problem $1\|\sum w_i \cdot T_i$, Holthaus and Rajendran (2005) used the Fast Ant Colony Optimization (FACO), which initializes the pheromone levels using a constructive heuristic. In this case, the problem characteristics were embedded in the pheromone matrix, allowing the visibility rule to have a constant value of $\eta = 1$. In this case, because the visibility rule did not need be calculated, the algorithm is faster to execute.

Cheng et al. (2009) implemented a hybrid algorithm for the $1\|\sum T_i$ problem, based on results presented by Bauer et al. (1999). According to Cheng et al. (2009), the algorithm proposed contained a set of elimination rules that, inserted into the algorithm proposed by Bauer et al. (1999), simplified the search space and improved the quality of the final solution.

The algorithm presented by Gagn et al. (2002) addressed a different problem, $1/s_{ij}/\sum T_i$ (minimize total tardiness in a single machine environment with setup-dependent times). This algorithm reduced the search space using a candidate list. The visibility rule consisted of information related to (i) the setup time of the job and (ii) the slack of the job. The transition rule, which the authors named as *look-ahead information*, used the visibility values and added information regarding the others non-processed tasks.

A similar problem, $1/s_{ij}/\sum w_i \cdot T_i$, was addressed by Liao and Juan (2007). In this approach, which used the Apparent Tardiness Cost with Setup rule to calculate the value of the visibility, the authors used the *job-to-position* approach. Thus, in this case, the ant does not use the graph shown in Fig. 2; instead, it uses a graph that makes it possible to verify the likelihood of a job belonging to a position in the final sequence.

Kashan and Karimi (2008) approached the problem 1/batch, incompatible, $s_{ij}/\sum w_i \cdot T_i$ (minimize total weighted tardiness on a single machine environment with formation of processing batches where some tasks cannot be processed in the same batch as others with sequence-dependent setup times). In this paper, the authors proposed two ACO algorithms, each one having different visibility functions.

5.1.2. Applications related to job completion time

Although the makespan of some single machine scheduling problems can easily be obtained by calculating the sum of all processing times (for example, the $1\|\sum T_i$ problem has a constant makespan for any built sequence), for certain problems the determination of the makespan of the optimal sequence is less straightforward. One example is the problem $1/s_{ij}/M$, addressed by Thiruvady et al. (2009). To solve this problem, the authors used a hybrid algorithm composed of two heuristics: a MMAS and a Beam Search. In this case, the Beam Search was combined with a third heuristic and then used to reduce the search space for the MMAS.

Chen et al. (2008) presented an algorithm to solve the single machine scheduling problem using batches ($1/\text{batch}/C_{\max}$). In this paper, the Chaotic Ant Colony Optimization (CACO), a variation on the original ACO with a chaos local search algorithm embedded, was used. To group the jobs, the authors used the heuristic Batch First Fit (BFF).

The problem $1/r_j,s_j$, batch/$C_{\max}$ – minimize makespan of a set of jobs executed by a single machine with setup-dependent times and different release times – is approached by Xu et al. (2012). In this paper, the authors embed concepts found in the FFLPT-ERT heuristic to allow the ACO algorithm to form batches and compare it with a mixed-integer programming approach, two heuristics and a GA-based approach. The paper shows that ACO returns better results than the other strategies, when comparing the final value of the objective function and required computational time.

5.2. ACO approaches to parallel machine scheduling

The definition of a typical parallel machine scheduling problem is similar to a single machine problem: a set of jobs, each with one operation, must be allocated to a resource to accomplish a goal. In the parallel machine environment, the main difference is that a set of machines process the job operations.

As shown in Section 2.1, this set of machines can be identical, uniform, or unrelated. In the following sections, two sets of approaches are presented: those that use ACO algorithms to solve scheduling problems with identical machines and those that deal with an unrelated machine environment.

5.2.1. Applications in identical parallel machine environments

Sankar et al. (2005) used an *n*-dimensional TSP approach to solve the problem $P_m/s_{ij}/C_{\max}$ (a scheduling problem involving identical parallel machines with sequence-dependent setups to minimize makespan). The same problem was addressed by Behnamian et al. (2009). The authors used the ACO to generate an initial solution, which was then improved by Simulated Annealing and Variable Neighborhood Search techniques. Behnamian et al. (2009) also presented three local search algorithms: (i) the first consisted of swapping the positions of two jobs in the sequence of the same machine; (ii) the second consisted of swapping the positions of two jobs in the sequence of different machines; and (iii) the third transferred jobs from the machine with a higher makespan to the machine with a lower makespan.

Raghavan and Venkataramana (2006) approached the problem P_m/Batch, incompatible/$\sum w_i \cdot T_i$. This problem has to do with minimizing the sum of weighted tardiness in a parallel machine environment that allows the formation of batches. The proposed algorithm consists of two phases: (i) generation of allowed batches using the dispatch rule Apparent Tardiness Cost (ATC) and (ii) an ACO based algorithm to schedule those batches on different machines.

A similar scheduling problem, P_m/A_{ij}, Batch, incompatible/$\sum w_i \cdot T_i$, was studied by Li et al. (2008). The authors claimed that this problem was important in semiconductor manufacturing environments, and they showed that implementation of the ACO algorithm yielded better results than the sequence based only on the ATC dispatch rule.

The problem P_m/A_{ij}, Q_i, Batch, incompatible/$\sum w_i \cdot T_i$ is an extension of the problem P_m/A_{ij}, Batch, incompatible/$\sum w_i \cdot T_i$, now considering different release dates. The presented algorithm also produced better results than the ATC dispatch rule.

Finally, the problem $P_m \| T_{\max}$ was approached by Gatica et al. (2010). The authors compared usage of four dispatch rules as a visibility rule (Earliest Due Date, EDD; Shortest Processing Time, SPT; Largest Processing Time, LPT; Least Slack, SLACK). These visibility rules were compared with each other and then with a genetic algorithm. To perform the comparison, four performance variables were used: (i) best percentage error of the best found solution; (ii) mean percentage error of the best found solution; (iii) mean objective value; and (iv) percentage of runs where the ACO found the best known value (named "hit ratio").

5.2.2. Applications in an unrelated parallel machine environment

Zhuo et al. (2007) extended the algorithm proposed by Liao and Juan (2007) and created an algorithm to solve the problem $R_m \| \sum w_i \cdot T_i$. In their approach, three main modifications were implemented.

(i) The authors used two types of pheromones: the first indicated the desirability of processing one specific job on one specific machine, while the second related to the decision to process a job J_i after a job J_j.
(ii) A heuristic rule was used to obtain the initial pheromone levels.
(iii) Visibility was defined using the dispatch rule VMDD.

Monch (2008) also addressed this problem using several approaches, including ACO. In this case, the ATC rule was used as a visibility function.

Again addressing unrelated parallel machine scheduling problems, Arnaout et al. (2008) and Arnaout et al. (2009) proposed and implemented an ACO algorithm for the $R_m/s_{ijk}/C_{\max}$ problem (allocate jobs in an unrelated parallel machine manufacturing environment considering dependent setup). To solve this problem, a two-stage algorithm was proposed and implemented: in the first stage, jobs are allocated to the machines, while in the second stage, jobs were sequenced on each machine.

5.3. ACO approaches to flowshop scheduling problems

The scheduling problem using a flowshop manufacturing environment consists of jobs, each containing a set of operations o_{ij}, where i is the job index, and j is the index of the resource that must be used to perform this operation. In a flowshop environment, all resources must be allocated in the same sequence in all jobs. If the job sequence needs to be maintained during all production stages, the problem is referred to as a permutational flowshop scheduling problem. If the job sequence can vary during each operation, the problem is referred to as non-permutational flowshop scheduling problem (Baker, 1943; Pinedo, 2008).

5.3.1. Applications in minimizing makespan

A well-known permutational flowshop problem is to minimize the makespan (F_m/prmu/M). This problem was addressed by ACO in eight studies: Ying and Liao (2003), Rajendran and Ziegler (2004), Ahmadizar et al. (2007), Chen et al. (2008), Zhou and Qingshan (2009), Ying and Lin (2007), Shyu et al. (2004) and Yan-hai et al. (2005).

Ying and Liao (2003) solved this problem using a graph similar to that shown in Fig. 2, using a specific visibility function. The two algorithms proposed by Rajendran and Ziegler (2004) also proposed specific visibility functions.

The approach by Ahmadizar et al. (2007) to this problem generated the visibility value based on two priority rules: the first involved an extension of Johnson's rule (Johnson, 1953) and the second involved an extension of the SPT dispatch rule.

The visibility function was discarded in the algorithm proposed by Chen et al. (2008). In their approach, characteristics of the problem influenced the solution only by the application of the pheromone update rule and by the local search algorithm used.

A solution to large problems (100–200 jobs being processed on 10–20 machines) of the F_m/prmu/M was proposed by Zhou and Qingshan (2009). Their solution consisted of a hybrid algorithm of ACO and a tabu search. Similar to the algorithm proposed by Chen et al. (2008) and Zhou and Qingshan (2009) this paper set $\eta = 1$.

Ying and Lin (2007) proposed an approach to the non-permutational flowshop problem $F_m \| M$. The choice of the value of the visibility η was based on a random function chosen from a set of 20 constructive heuristics (such as SPT, LPT, and others).

Shyu et al. (2004) used the ACS algorithm to solve the F_2/nwt,setup/M problem (minimize makespan in a 2-machine non-permutational flowshop with sequence-dependent setups, in an environment that does not allow a wait between operations). In this algorithm, the visibility was defined using the job's setup and processing times.

To solve the problem of minimizing makespan in a non-permutational flowshop with rush orders, Yan-hai et al. (2005) used an ACO to generate an initial solution. When rush orders arise, the algorithm continues its execution, adjusting the solution to the new problem specification.

5.3.2. Other applications with single criteria objective functions

Discarding the makespan criterion, we found four papers using ACO to solve flowshop scheduling problems with a single criterion present in the objective function.

The first, by T'kindt et al. (2002), presented an MMAS algorithm to solve the problem of minimizing the total completion time of n tasks in a two-machine flowshop environment ($F_2 \| \sum C_i$). In this approach, the transition rule was simplified, with $\eta = 1$. The local search algorithm was based on the Adjacent Pairwise Interchange algorithm.

Rajendran and Ziegler (2004, 2005) approached the F_n/prmu/$\sum F_i$ problem using a similar algorithm. The major change that allowed the author's results to be improved had to do with research using the NEH algorithm, proposed by Nawaz et al. (1983), to initialize the pheromone levels.

The problem of minimizing the completion time variance in a permutational flowshop environment was addressed by Gajpal and Rajendran (2006), who used a modified MMAS algorithm. In their approach, the NEH algorithm was used to build an initial solution. The local search used is called random-job-insertion, based on the random change of a random job.

The work of Li et al. (2011) dealt with the minimization of the sum of completion time in a permutational flowshop environment with sequence-dependent setups. In this paper, the authors used a time-limited dynamic programming algorithm to perform a post-optimization strategy. The authors undertook a comparison between two different approaches: ACO and mathematical programming. According to the authors, mathematical programming is a viable strategy for small-size problems. When the size of the problems increases, the computational time becomes unacceptable, making usage of the ACO a preferable strategy.

Similar comparisons between mathematical programming and ACO approaches were also performed by Tavares Neto and Godinho Filho (2011). This paper deals with the problem of minimizing the weighted sum of makespan and outsourcing cost in a permutational flowshop environment with a budget

restriction, namely F_m/prmu,budget/$(1-\delta)\cdot M+\delta\cdot OC$. The paper presents a two-stage ACO algorithm that demonstrates a good performance, even for large-size problems.

5.3.3. Applications in problems with objective functions having more than one performance measure

As well as proposing an approach to the $F_2\|\sum C_i$ problem, T'kindt et al. (2002) also applied ACO to the $F_2\|\text{Lex}(C_{\max},\sum C_i)$ problem. In one, it was necessary to schedule a set of jobs in a 2-machine permutational flowshop environment to minimize the total tardiness problem with minimum makespan.

Li and Zhang (2006) approached the problem of minimizing the weighted sum of total completion time and makespan in a two-machine flowshop environment ($F_2\|\alpha\cdot C_i+(1-\alpha)\cdot M$). The authors used the following modifications: (i) the rules API and NAP were used to set the pheromone values; (ii) the generation of pheromone trails reinforced the desirability of a job being positioned after a job (job-to-job position), in a fixed position (job-to-position); and (iii) some dominance criteria were applied to avoid the construction of non-promising solutions.

The problem $F_m\|\sum T_i,M$ (minimize total completion time and makespan in a m-machine flowshop environment) was addressed by Pasia et al. (2006). The authors used the Pareto Ant Colony Optimization (PACO), establishing a set of pheromones for each problem objective.

Yagmahan and Yenisey (2008) used ACO to solve a permutational flowshop problem with four objectives: minimize total tardiness, makespan, total flowtime and idle time (F_m/prmu/$\sum C_i,M,\sum F_i$,idle). To accomplish this task, the following modifications were made to the original algorithm: (i) the visibility values were defined in the initialization phase, based on a heuristic executed at the beginning of the algorithm; (ii) a candidate list was used to restrict the search space; and (iii) a local search algorithm was composed by removing a random job and inserting it into another random position.

To approach the problem of minimizing the weighted sum of makespan and mean completion time in a 2-machine permutational flowshop environment (F_2/prmu/$u\cdot M+\nu\cdot\sum C_i n$), Al-Anzi and Allahverdi (2009) use, as a parameter to update the pheromone levels, the number of interactions already executed, allowing a higher influence of the later executions of the algorithm (this behavior is named *algorithm maturity*).

Lin et al. (2008) addressed the problem of minimizing the weighted sum of makespan and total completion time in a permutational flowshop manufacturing environment (F_m/prmu/$u\cdot M+\nu\cdot\sum C_i$). In their approach, the authors used a new transition rule, based on: (i) a set of pheromones representing the relative positioning between each pair of jobs; (ii) a set of pheromones that represented the occupation of a position in the final sequence by a job; and (iii) a visibility function. The authors also proposed a further modification composed of two dominance criteria, shown as efficient strategies to improve the final result of the implemented algorithm.

In all papers examined, only the one by Marimuthu et al. (2009) considered batches in a flowshop environment. The authors proposed solutions for two problems: the first to minimize makespan and the second to minimize total flowtime (F_m,lot$\|M$ and F_m,lot$\|\sum F_i$). For both problems, the authors developed an ACO algorithm that used only the information generated by the pheromone matrix and a local search to determine the best solution. This local search consisted in relocating each job on the sequence in order to improve the value of the objective function.

Finally, Huang and Yang (2009) addressed the problem $F_m/r_j,S_{ij}/\alpha\sum\text{MIT}_i+\beta\sum\text{JWT}_{ijt}+\gamma\sum T_i$. In this problem, a flowshop manufacturing environment with jobs having different release times and sequence-dependent setup time is considered. The objective function minimized the weighted sum of the following: (i) total machine idle time; (ii) total waiting time of all jobs on all machines; and (iii) total tardiness. In their ACO approach, the authors developed a heuristic to determine the visibility function and used no local search.

5.4. ACO approaches to jobshop scheduling problems

As with the previous one, this section initially presents studies that proposed ACO approaches to minimize the makespan in a job shop manufacturing environment (Section 5.4.1) and then, in Section 5.4.2, present jobshop problems with more than one performance indicator.

5.4.1. Applications in minimizing makespan

Most of the papers found in this bibliographic survey that deal with the use of ACO in a jobshop environment sought minimization of the makespan ($J_{n,m}\|M$). The first study to address this problem was by Colorni et al. (1994). In addition to being the first application of ACO to the job shop problem, this paper also presents an interesting analysis: according to the authors, the parameter sensitivity was very low when the problem size was changed.

Yoshikawa and Terai (2006) also worked with the same problem. In this case, the authors presented an ACO algorithm, in which each graph node represented the execution of one job operation on one machine. A constructive heuristic indicated a set of operations to be executed immediately. The ACO is then used to sequence those operations.

Udomsakdigool and Kachitvichyanukul (2008) approached the problem $J_{n,m}\|M$ by developing a set of ACO algorithms with visibility rules defined according to a set of 10 different dispatch rules (EDD, SPT, LDT, and others). They also use a candidate list based on one of the following directives: (i) all pending jobs can be executed; (ii) all jobs that can be performed without adding waiting time are considered; and (iii) a random set of jobs is considered. The authors tested all 10 dispatch rules with the three candidate list generation directives, generating a set of 30 ACO algorithms. The author concluded that the best combination depends on characteristics such as the problem size.

The same problem was addressed by Zhang et al. (2010). The proposed algorithm presented a priority rule adapted to the job shop problem and a specific local search, allowing the alteration of unimportant pheromone information and the simplification of further ACO algorithms.

Heinonen and Pettersson (2007) also used the ACO algorithm to solve the problem $J_{10,10}\|M$. In their approach, an MMAS algorithm was implemented and hybridized with a tabu search local search. The same problem was solved by Zhou et al. (2007), using an ACO with a visibility function proportional to the time left to accomplish a job.

Mirabi (2011) add a sequence-dependent setup to the $J_{n,m}\|M$, approaching the problem $J_{n,m}/s_{ij}/M$. The proposed algorithm contained a new pheromone initialization procedure and a local search, both allowing execution of the algorithms on large-scale real problems.

5.4.2. Applications in problems with multiple objectives

Huang and Yang (2008) proposed an ACO approach to solve the problem of minimizing the sum of earliness and tardiness in a jobshop environment with time windows. In their research, the authors used only the relative positioning of a job.

6. Quantitative analysis of the literature

The bibliographic review performed in this paper made it possible to derive quantitative analyses.

The first analysis could be performed on the basis of the article publication date (Fig. 3). Examining this characteristic, we observed that in the first stage of studies (1994–2000), the majority of papers dealt with single-machine scheduling problems. In the second period (after 2002), researchers began to apply ACO in flowshop environments. Parallel-machine environments and ACO were studied in the third stage (after 2005). From 2006 to the present, we find most job shop scheduling uses ACO.

Another possible analysis could address the number of components of the objective function. As presented in Fig. 4, only one jobshop and eight flowshop scheduling problems studied dealt with more than one component in the objective function. Otherwise, objective functions with a single component were applied in all manufacturing environments, which did not happen with the multi-criteria objective function.

Fig. 5 summarizes the base algorithm used in the researched papers. One can observe that most of the papers (36) used the first version of ACO (or a specific variation of it). ACS is the second most used ant-based algorithm (used in 10 articles), followed by MMAS (used by eight articles). It is noteworthy that some of the articles involving ACO implement some features from ACS or MMAS (such as the pheromone bounds of MMAS). The three algorithms were used to solve scheduling problems in the parallel machine and flowshop environments. Job-shop and single machine scheduling problems were solved only by ACO and ACS. This observation indicates that the three algorithms have all proven to be useful in the solution of scheduling problems (Fig. 6).

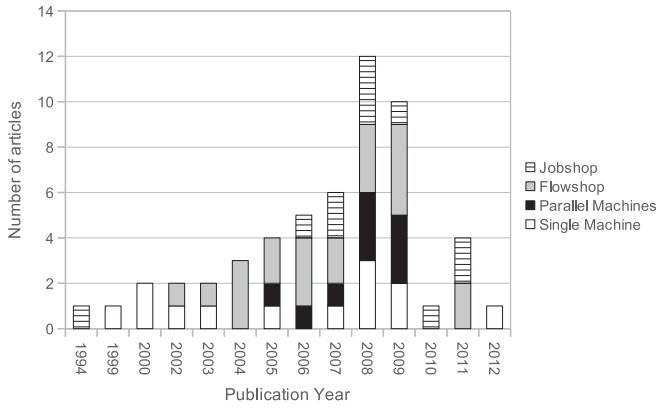

Fig. 3. Articles found according publication date.

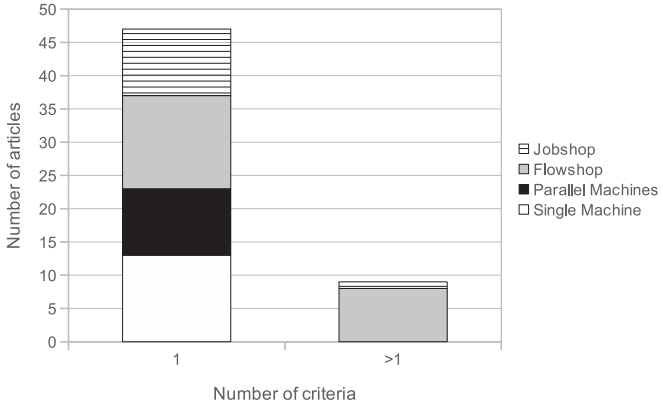

Fig. 4. Number of articles according to the number of components of the objective function.

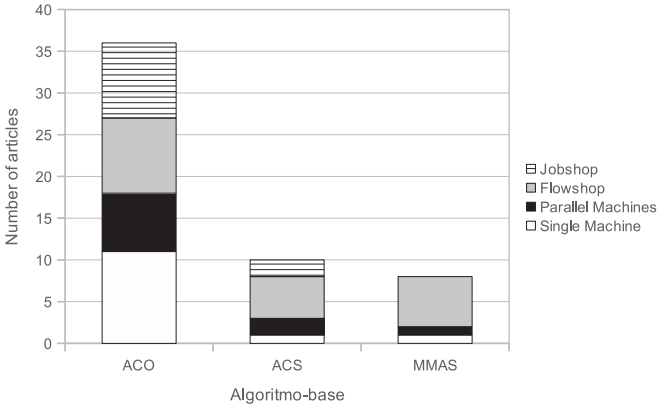

Fig. 5. Base algorithms used in the papers.

Fig. 6. Job positioning schemes used in the papers.

Another analysis relates to how the job sequence is represented by the algorithms. The job-to-position strategy has been used more often to solve flowshop scheduling problems (nine articles). This strategy is less used to solve single machine, parallel machines and jobshop scheduling problems (1, 2 and 5 articles). The job-to-job strategy is used more often, in all manufacturing environments. This finding may indicate that the job-to-job strategy can be an efficient approach to mostly scheduling problems.

One important observation relates to the algorithms presented by Liao and Juan (2007), Li and Zhang (2006) and Lin et al. (2008). In these algorithms, both strategies (job-to-job and job-to-position) were used. One may infer that the use of both strategies generates more useful information for the algorithm.

Regarding the adoption of dominance criteria, one may note that, as shown in Fig. 7, only problems dealing with single machine and flowshop problems adopt this strategy. In the case of single machine scheduling problems, the number of articles that adopt dominance criteria is the same as the number that do not use it.

The literature analysis also allows certain conclusions to be draw about the pheromone initialization scheme. Although, as shown in Fig. 8, most of the applications use a static initial value for the pheromone levels, some researchers chose to initialize it based on the results of some constructive heuristic. This strategy was used in most of the articles that addressed flowshop scheduling problems.

Finally, one can analyze how the visibility function is defined. As presented in Fig. 9, three different strategies were found in the related literature:

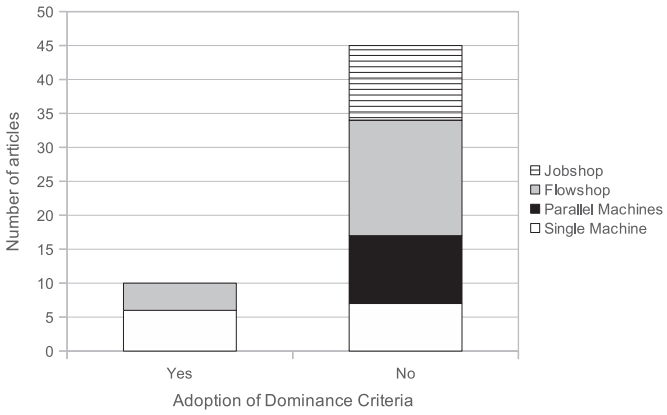

Fig. 7. Adoption of dominance criteria.

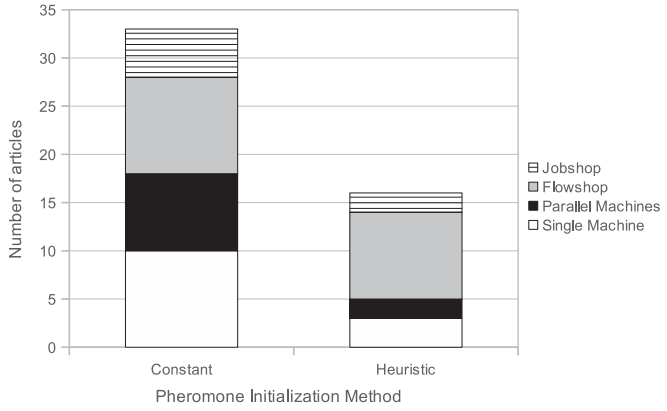

Fig. 8. Pheromone initialization method used in the papers.

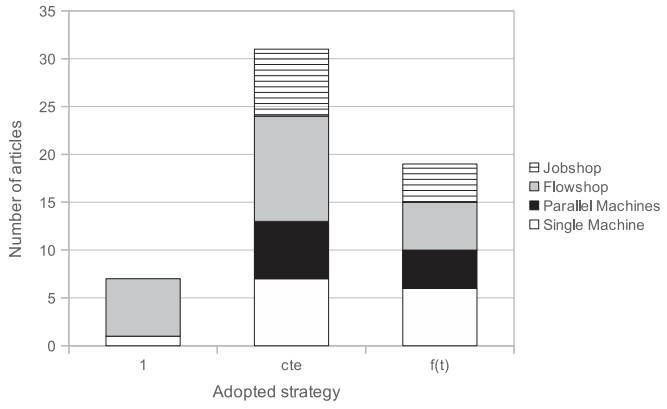

Fig. 9. Strategy used to define the visibility function.

1. Visibility was defined in the initialization phase of the algorithm and then used during its execution. This strategy was used in all four manufacturing environments studied.
2. Visibility was defined according to a function that used the partial sequence already defined by the ant. This strategy was used in 50% on the articles concerning single machine scheduling problems and in 50% of the articles that addressed the jobshop environment; it was also used with regard to the parallel machine and flowshop problems.
3. Visibility was not used. In this case, the specific characteristics of the problems were incorporated by the algorithm, either in the pheromone initialization phase or in the local search algorithm. This strategy was used only in one article that addressed a single machine scheduling problem and in six articles that solved flowshop scheduling problems.

7. Conclusion

This paper presents a literature review concerning the use of the ACO metaheuristic in scheduling problems. The scope of this survey covers four manufacturing environments: single machine, parallel machine, flowshop, and job shop. Through the analysis, it was possible to verify that ant-based algorithms (ACO, ACS, and MMAS) are valid strategies for solving scheduling problems. Although the ACO algorithm is straightforward, the possibility to easily incorporate scheduling-specific heuristics and other metaheuristics increases the complexity and the research opportunities involving the use of this metaheuristic for this category of problems.

The analysis presented in this paper indicates that this field of research is a relatively new one, and only the initial results have been published. One can easily note that the first studies had to do with single-machine scheduling problems. Only later were flowshop, parallel-machine, and job shop environments addressed. The evolution of knowledge in this research area can also be observed in terms of the new features added to address certain problems (e.g., dominance criteria involving single-machine and flowshop scheduling problems).

From the quantitative analysis performed, we can derive certain guidelines for the application of ACO algorithms to scheduling problems. The first concerns the job-position scheme. Our research shows that the job-to-job position scheme is used more often than the job-to-position. This may indicate that the job-to-job scheme is usually a straightforward strategy to begin new applications of ACO to scheduling problems. As regards the job-to-position scheme, an interesting strategy is a hybrid approach, which uses both job-to-job and job-to-position, indicating that, when necessary, it is possible to combine both strategies. Another conclusion to be drawn from this paper is that the definition of visibility as a function of the partial solution being built by a single ant is a promising strategy for scheduling applications, since it was used in almost 20 papers in our review. In addition, our literature review highlights the benefits of incorporating scheduling-specific information into ACO implementations. This can be observed from:

(i) The adoption of dominance criteria in some papers;
(ii) The use of problem-specific pheromone initialization methods, aimed at speeding up the algorithm.

Potential topics for future research also emerged. We observed that a single-criterion objective function was used in the four manufacturing environments studied in this paper. This was not the case for multi-criteria objective functions. Therefore, one can conclude that using ACO for scheduling problems aimed at optimizing multi-criteria objective functions is a relatively new research field (no paper with this characteristic was found for single and parallel environments, and only one paper with regard to job shop), and can be addressed. In addition, our review indicates that the general evolution of ACO applications relating to scheduling problems basically followed an expected path: single machine, parallel machine, flowshop, and job shop. Other promising areas for future research involve:

- *Setup-dependent times*. Although some of the approached problems address setup-dependent times, we note that there is no application of ACO to solve job shop problems with this constraint. A similar remark can be made regarding the possibility of generating production batches, which is still not considered in job shop problems.

- *Makespan*. Still observing the job shop problems addressed, it is possible to note that makespan is the most frequent criterion, showing that a research opportunity exists to use other objective functions in job shop scheduling. The other three studied environments present a wider range of objective functions.
- *Time-window constraint*. Of all the papers examined, only one deals with scheduling with a time-window constraint (Huang and Yang, 2008). This constraint is common in real scheduling problems, and can therefore represent an opportunity for research regarding application of the ACO metaheuristic.

Acknowledgements

The authors would like to acknowledge the support received by Fapesp Brazilian Agency in this research.

References

Ahmadizar, F., Barzinpour, F., Arkat, J., 2007. Solving permutation flow shop sequencing using ant colony optimization. In: 2007 IEEE International Conference on Industrial Engineering and Engineering Management, pp. 753–757.
Al-Anzi, A.S., Allahverdi, A., 2009. Heuristics for a two-stage assembly flowshop with bicriteria of maximum lateness and makespan. Comput. Oper. Res. 36 (9), 2682–2689.
Anghinolfi, D., Boccalatte, A., Paolucci, M., Vecchiola, C., 2008. Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In: Simulated Evolution and Learning. Springer, Berlin/Heidelberg, pp. 411–420.
Arnaout, J., Musa, R., Rabadi, G., 2008. Ant colony optimization algorithm to parallel machine scheduling problem with setups. In: 4th IEEE Conference on Automation Science and Engineering.
Arnaout, J., Rabadi, G., Musa, R., 2009. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. J. Intell. Manuf.
Baker, K., 1943. Introduction to Sequencing and Scheduling. John Wiley & Sons.
Bauer, A., Bullnheimer, B., Hartl, R.F., Staruss, C., 1999. Applying Ant Colony Optimization to Solve the Single Machine Total Tardiness Problem. Technical Report. Vienna University of Economics and Business Administration.
Behnamian, J., Zandieh, M., Fatemi Ghomi, S.M.T., 2009. Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. Expert Syst. Appl. 36 (6), 9637–9644.
Brucker, P., 2007. Scheduling Algorithms. Springer, Berlin/Heidelberg.
Bullnheimer, Brend, Hartl, Richard F., Strauss, Christine, 1997. An improved ant system algorithm for the vehicle routing problem. Ann. Oper. Res. 89, 319–328.
Chen, R., Lo, S., Wu, C., Lin, T., 2008. An effective ant colony optimization-based algorithm for flow shop scheduling. In: IEEE Conference on Soft Computing in Industrial Applications.
Cheng, B.-Y., Chen, H.-P., Shao, H., Xu, R., Huang, G.Q., 2008. A chaotic ant colony optimization method for scheduling a single batch-processing machine with non-identical job sizes. In: IEEE Congress on Evolutionary Computation, pp. 40–43.
Cheng, T.C.E., Lazarev, A.A., Gafarov, E.R., 2009. A hybrid algorithm for the single-machine total tardiness problem. Comput. Oper. Res. 36 (2), 308–315.
Colorni, A., Dorigo, M., Maniezzo, V., 1991. Distributed optimization by ant colonies. In: European Conference of Artificial Life, pp. 134–142.
Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., 1994. Ant system for job-shop scheduling. Belg. J. Oper. Res.
Cordon, O., Vianna, I.F., Herrera, F., Moreno, L., 2000. A new ACO model integrating evolutionary computation concepts: the best–worst ant system. In: Ants2000.
den Besten, M., Stützle, T., Dorigo, M., 2000. Ant colony optimization for the total weighted tardiness problem. In: PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature. Springer-Verlag, London, UK, pp. 611–620.
Dorigo, M., Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1, 53–66.
Dorigo, M., Maniezzo, V., Colorni, A., 1996. The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B 26, 29–41.
Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press.
Dowsland, K.A., Thompson, J.M., 2005. Ant colony optimization for the examination scheduling problem. J. Oper. Res. Soc. 56, 426–438.
Eswaramurthy, V.P., Tamilarasi, A., 2009. Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems. Int. J. Adv. Manuf. Technol. 40, 1004–1015.
Gagn, C., Price, W.L., Gravel, M., 2002. Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. J. Oper. Res. Soc. 53, 895–906.
Gajpal, Y., Rajendran, C., 2006. An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. Int. J. Prod. Econ. 101, 259–272.
Gatica, C.R., Esquivel, S.C., Leguizamon, G.M., 2010. An ACO approach for the parallel machines scheduling problem. Intell. Artif. 46, 84–95.
Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., 1979. Optimization and approximation in deterministic machine scheduling: a survey. Ann. Discrete Math. 5, 287–326.
Gutjahr, W.J., Rauner, M.S., 2007. An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. Comput. Oper. Res. 34.
Heinonen, J., Pettersson, F., 2007. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. Appl. Math. Comput. 187, 989–998.
Holthaus, O., Rajendran, C., 2005. A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. J. Oper. Res. Soc. 56, 947–953.
Huang, K., Liao, C., 2008. Ant colony optimization combined with taboo search for the job shop scheduling problem. Comput. Oper. Res. 35, 1030–1046.
Huang, R., Yang, C., 2008. Ant colony system for job shop scheduling with time windows. Int. J. Adv. Manuf. Technol. 39, 151–157.
Huang, R., Yang, C., 2009. Solving a multi-objective overlapping flow-shop scheduling. Int. J. Adv. Manuf. Technol. 42, 955–962.
Johnson, S.M., 1953. Optimal two- and three-stage production schedules with setup times included. Naval Res. Log. Q. 1, 61–68.
Kashan, A.H., Karimi, B., 2008. Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework. J. Oper. Res. Soc. 59, 1269–1280.
Li, J., Zhang, W., 2006. Solution to multi-objective optimization of flow shop problem based on ACO algorithm. In: 2006 International Conference on Computational Intelligence and Security.
Li, L., Qiao, F., Wu, Q.D., 2009. ACO-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints. Int. J. Adv. Manuf. Technol. 44, 985–994.
Li, L., Quiao, F., Wu, Q., 2008. ACO-based scheduling of parallel batch processing machines with incompatible job families to minimize total weighted tardiness. In: Ant Colony Optimization and Swarm Intelligence. Springer, Berlin/Heidelberg, pp. 219–226.
Li, X., Baki, M.F., Aneja, Y.P., 2011. Flow shop scheduling to minimize the total completion time with a permanently present operator: models and ant colony optimization metaheuristic. Comput. Oper. Res. 38, 152–164.
Liao, C.-J., Juan, H.-C., 2007. An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. Comput. Oper. Res. 34 (7), 1899–1909.
Lin, B., Lu, C., Shyu, S., Tsai, C., 2008. Development of new features of ant colony optimization for flowshop scheduling. Int. J. Prod. Econ. 112 (2), 742–755. URL ⟨http://ideas.repec.org/a/eee/proeco/v112y2008i2p742-755.html⟩.
Marimuthu, S., Ponnambalam, S.G., Jawahar, N., 2009. Threshold accepting and ant-colony optimization algorithms for scheduling *m*-machine flow shops with lot streaming. J. Mater. Process. Technol. 209, 1026–1041.
Merkle, D., Middendorf, M., 2000. An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In: EvoWorkshops, pp. 287–296. URL ⟨citeseer.ist.psu.edu/article/merkle00ant.html⟩.
Merkle, D., Middendorf, M., 2003. Ant colony optimization with global pheromone evaluation for scheduling a single machine. Appl. Intell. 1, 105–111.
Mirabi, M., 2011. Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. Int. J. Adv. Manuf. Technol. 55, 317–326.
Monch, L., 2008. Heuristics to minimize total weighted tardiness of jobs on unrelated parallel machines. In: IEEE International Conference on Automation Science and Engineering, 2008 (CASE 2008).
Nawaz, M., Enscore, E., Ham, I., 1983. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. Omega 11, 91–95.
Pasia, J.M., Hartl, R.F., Doerner, K.F., 2006. Solving a bi-objective flowshop scheduling problem by pareto-ant colony optimization. Lect. Notes Comput. Sci. 4150, 294–305 (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).
Pinedo, M.L., 2008. Scheduling: Theory, Algorithms and Systems. Springer.
Pinedo, M.L., 2009. Planning and Scheduling in Manufacturing and Services. Springer.
Raghavan, N.R.S., Venkataramana, M., 2006. Scheduling parallel batch processors with incompatible job families using ant colony optimization. In: Proceeding of the 2006 IEEE International Conference on Automation Science and Engineering.
Rajendran, C., Ziegler, H., 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur. J. Oper. Res. 155 (2), 426–438. URL ⟨http://ideas.repec.org/a/eee/ejores/v155y2004i2p426-438.html⟩.
Rajendran, C., Ziegler, H., 2005. Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. Comput. Ind. Eng. 48 (4), 789–797.
Sankar, S.S., Ponnambalam, S.G., Rathinavel, V., Visveshvaren, M.S., 2005. Scheduling in parallel machine shop: an ant colony optimization approach. In: IEEE International Conference on Industrial Technology, 2005 (ICIT 2005), pp. 276–280. URL ⟨http://dx.doi.org/10.1109/ICIT.2005.1600649⟩.
Shih-Pang, T., Chun-Wei, T., Jui-Le, C., Ming-Chao, C., Chu-Sing, Y., 2011. Job shop scheduling based on ACO with a hybrid solution construction strategy. In: IEEE International Conference on Fuzzy Systems (FUZZ).
Shyu, S., Lin, B., Yin, P., 2004. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. Comput. Ind. Eng. 47, 181–193.

Stützle, T., Hoos, H.H., 2000. Max–min ant system. Future Gener. Comput. Syst. 16 (9), 889–914.
Tavares Neto, R.F., Godinho Filho, M., 2011. An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. Comput. Oper. Res. 38, 1286–1293.
Thiruvady, D., Blum, C., Meyer, B., Ernst, A., 2009. Hybridizing beam-ACO with constraint programming for single machine job scheduling. In: Lecture Notes in Computer Science. Springer Link, pp. 30–44.
T'kindt, V., Monmarche, N., Tercinet, F., Laugt, D., 2002. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. Eur. J. Oper. Res. 142 (2), 250–257.
Udomsakdigool, A., Kachitvichyanukul, V., 2008. Multiple colony ant algorithm for job-shop scheduling problem. Int. J. Prod. Res. 46, 4155–4175.
Xu, R., Chen, H., Li, X., 2012. Makespan minimization on single batch-processing machine via ant colony optimization. Comput. Oper. Res. 39, 582–593.
Yagmahan, B., Yenisey, M.M., 2008. Ant colony optimization for multi-objective flow shop scheduling problem. Comput. Ind. Eng. 54, 411–420.
Hu, Yan-hai, Yan, Jun-qi, Ye, Fei-fan, Yu, Jun-he, 2005. J. Zhejiang Univ. Sci. A 6 (10).
Ying, K., Lin, S., 2007. Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems. Int. J. Adv. Manuf. Technol. 33, 793–802.
Ying, K.-C., Liao, C.-J., 2003. An ant colony system approach for scheduling problems. Prod. Plann. Control 14. 68–75(8). URL 〈http://www.ingentaconnect.com/content/tandf/tppc/2003/00000014/00000001/art00006〉.
Yoshikawa, M., Terai, H., 2006. A hybrid ant colony optimization technique for job-shop scheduling problems. In: Proceedings of the 4th International Conference on Software Engineering Research, Management and Applications.
Zhang, Z., Zhang, J., Li, S., 2010. A modified ant colony algorithm for the job shop scheduling problem to minimize makespan. In: International Conference on Mechanic Automation and Control Engineering (MACE).
Zhou, H., Li, Z., Wu, X., 2007. Scheduling unrelated parallel machine to minimize total weighted tardiness using ant colony optimization. In: IEEE International Conference on Automation and Logistics.
Zhou, P., Qingshan, D., 2009. Hybridizing fast taboo search with ant colony optimization algorithm for solving large scale permutation flow shop scheduling problem. In: IEEE International Conference on Granular Computing.
Zhuo, X., Zhang, J., Chen, W., 2007. A new pheromone design in ACS for solving JSP. In: IEEE Congress on Evolutionary Computation.