J Intell Manuf (2015) 26:527–538
DOI 10.1007/s10845-013-0811-5

An ant colony optimization approach for the parallel machine scheduling problem with outsourcing allowed

Roberto Fernandes Tavares Neto · Moacir Godinho Filho · Fabio Molina da Silva

Received: 27 April 2013 / Accepted: 1 July 2013 / Published online: 27 July 2013

Abstract Several manufacturing environments can be represented as a set of identical parallel machines. Moreover, some industries uses third-part manufacturing to increase the production capacity for short periods. This paper proposes, implements and evaluates an ACO algorithm to solve the parallel machine scheduling problem with outsourcing allowed. The goal is to minimize the sum of outsource and delay costs (since, in many practical situations, the delay generates fine). To the best of our knowledge, this work is the first to address this problem. In order to evaluate the algorithm proposed, a mathematical programming model of the problem is also presented and implemented. The ACO algorithm proposed is composed of three sequential transition rules, each one responsible for one different decision: the first one decides the next job to be scheduled; the second decides the machine to schedule a job and the third decides if the job must be outsourced or not. Computational results show that this algorithm, for instances of size larger or equal to 20 jobs, could reach better solutions than the ones found using the mathematical programming method when the commercial solver used has its running time limited by 1 h. Moreover, the times required to reach a solution were significantly smaller when the ACO is executed.

Keywords Ant colony optimization · Parallel machine scheduling problem · Outsourcing

R. F. Tavares Neto · M. Godinho Filho (✉) · F. M. da Silva
Department of Production Engineering, Federal University of São Carlos, Via Washington Luiz, Km 235, São Carlos, SP CEP 13565-905, Brazil
e-mail: moacir@dep.ufscar.br

R. F. Tavares Neto
e-mail: tavares@dep.ufscar.br

F. M. da Silva
e-mail: fabio@dep.ufscar.br

Introduction

Scheduling problems are a set of combinatorial problems that deal with allocating resources over time to perform a set of tasks (Baker and Bertrand 1982). These tasks compete among themselves for resources, such as machine time, energy, and tools. One or more goals to be accomplished can be set: for example, the minimizing of tardiness, flow time or total number of late tasks. These tasks can also be carried out in different manufacturing environments. For instance, a set of tasks' flow time on one single machine, or in an n parallel machine environment, a flow-shop, job-shop, or open-shop environment can be minimized. These characteristics indicates why Brucker (2007) claims that "the theory of scheduling is characterized by a virtually unlimited number of problem types". Pinedo (2012) shows examples of relevant scheduling problems in real-world environments, such as semiconductor manufacturing, gate assignment in airports, and process scheduling on a computer CPU. Some recent books dealing with scheduling theme are: Leung and Anderson (2004), Brucker (2007), and Pinedo (2009).

According to Ruiz-Torres et al. (2006), the outsourcing of manufacturing operations continues to gain popularity, forcing companies to deal with the issue of scheduling orders in increasingly complex supply chains. Despite its importance in modern supply chain (Boulaksil and Fransoo 2009), outsourcing is a topic relatively low studied in the scheduling literature. Outsourcing can be defined as assigning to a third party some operations of an entire process according to that party's specialty, instead of managing all processes directly (Lee and Choi 2011). According to Antelo and Bru (2010), in the last few decades, many industries have followed a trend to successively outsource additional segments of the supply chain. To Yadav and Gupta (2008) and Gonzalez et al. (2006), the importance of outsourcing has increased in the last years.

Since the paper published by Colorni et al. (1991), many researchers use "artificial ants" to solve difficult combinatorial problems. This class of algorithms, named ACO (Ant Colony Optimization), based on the behavior of real ants, shows good results with regard to several well-known hard problems. For example, one can refer to Dorigo et al. (1996) and Gambardella and Dorigo (2000) for the solution of both symmetric and asymetric travelling salesman problem (TSP), Gambardella and Dorigo (2000) for the Sequential Ordering Problem (SOP), Bullnheimer et al. (1999) for the Capacitaded Vehicle Routing Problem. Other examples of manufacturing problems solved using ACO are: manufacturing scheduling (for example, Arnaout et al. 2010, 2012; Prakash et al. 2008), renovation scheduling in construction projects (for example, Lee 2012), location/allocation problem (for example, Arnaout 2013), maintenance optimization (for example, Samrout et al. 2007), product platform formation under mass customization approach (for example, Kumar and Allada 2007), process planning optimization (for example, Liu and Yi 2013). Additional problems and studies can be found in Dorigo and Stutzle (2004), which presents more than 60 publications on the theme. This algorithm - initially called "Ant System" and then extended to "Ant Colony System"(ACS, Dorigo et al. 1996), "Beam-ACO" (Dorigo and Blum 2005), "Max-Min Ant System"(MMAS, Stutzle and Hoos 2000), and others, has been demonstrated capable of showing encouraging results. The focus of this paper is the application of ACO to scheduling problems with outsourcing allowed.

Several studies use ACO to solve a large variety of scheduling problems. For example, one can observe single-machine-related researches like Zapfel and Bogl (2008), Holthaus and Rajendran (2005), and Xu et al. (2012); parallel machines (for example, Arnaout et al. 2008); flow shop (for example, Marimuthu et al. 2009) and job-shop (for example, Zhou et al. 2007a, b, 2008; Huang et al. 2013). For a review about the application of ACO algorithm to the scheduling problem, see Tavares Neto and Godinho Filho (2013).

Within this context, the present paper aims at proposing, implementing and evaluating an ACO algorithm to solve the parallel machine scheduling problem with outsourcing allowed. Similar problems have been studied in few papers, like Bartal et al. (2000) and Ruiz-Torres et al. (2006): the first one aims to minimize the sum of sequence makespan and outsource costs; the second minimizes the sum of number of late jobs and total outsource machine time. Based on this literature, the problem approached in the present paper is stated as follows: let J be a set of jobs, each one with a processing time p_i and a due date d_i. The cost of the delay of each job per time unit (that can be described as the time-unit fine for each job) is measured by a parameter w_i. The additional cost of outsource (difference between production cost and total outsourcing cost) is measured by a job parameter o_i. This problem is based on the premises: (i) that any delay job will generate a cost if delayed, and this cost increases according the tardiness value of the job, and (ii) the outsourcing cost for each job is always greater that the production costs. We assume that the fines related to each job can be different. The goal is to minimize the weighted sum of outsource costs and tardiness costs. This objective function is designed to allow the minimization of two major costs involved in such situations: the increase of the manufacturing costs due the outsourcing of some jobs and the fines related to tardy deliveries to each client. Since those two indicators are measured using the same unit (monetary units), the authors considered that is safe to allow the sum of them on the objective function, and it can be easily applied in practical situations, especially when the total cost is the main concern of the enterprise. To the best of our knowledge, the present work is the first to address this problem.

This paper is organized as follows: The following section presents a literature review about using ACO to solve the parallel scheduling problem and also about scheduling problems with outsourcing allowed. "Problem definition" section shows the definition of the problem focused in this paper. The next section presents a mathematical programming approach to solve the problem. "The proposed algorithm" is devoted to describing basics features of ACO and also the most common characteristics used to implement such algorithms in scheduling problems. Within this section, the ACO algorithm proposed is also shown. "Computational Results" presents the computational results followed by "Conclusions" section.

Literature review

A literature review about ACO applied to parallel scheduling problem with outsourcing allowed was performed. Actually, any paper applying ACO to the parallel machine scheduling problem with outsourcing allowed was found. Therefore, our paper is the first to consider all of these characteristics. In this section, we show basic features of the ACO technique, as well as the most common characteristics used to implement such algorithms, focused on how the current literature apply ACO to scheduling in parallel environments. Moreover, our literature review also presents how scheduling problems are approached considering the possibility of outsource a subset of jobs.

An ant colony optimization approach

The Ant Colony Optimization (ACO) heuristic presented by Colorni et al. (1991), and later extended by several researches (e.g. Dorigo et al. 1996's ACS), uses a swarm intelligence approach to solve the Traveling Salesman Problem (TSP). In this well-known NP-Hard problem, a set of n cities must be

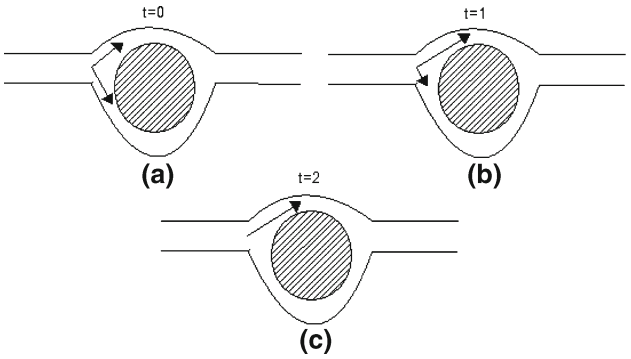

Fig. 1 The evolution of the pheromone impact on the path decision. The *arrow's* size are proportional to the probability of choose a path

visited by one single travelling salesman. The goal is to minimize the total distance length. Colorni et al. (1991), Dorigo et al. (1996), Gambardella and Dorigo (2000) and others create computational agents (called "artificial ants") and mimic the behavior of real ants going from their nests to a food source. To do so, the "artificial ants" (hereafter called just "ants") are placed on a graph and forced to move. Each single movement of the ant on the graph generates another piece of the solution. The set of moves generated the full problem solution. ACO has been used in several classes of problems with great practical appeal, such as vehicle routing (e.g. Bullnheimer et al. 1999), examination scheduling problem (e.g. Dowsland and Thompson 2005) and scheduling (e.g. Zhou et al. 2007a, b).

The meaning of an ant's move is highly dependent of the graph representation of the problem. For example, when solving the TSP problem, Dorigo et al. (1996) represents each node of the graph as a city, and each arc has the weight corresponding to the distance between the cities. When the ant moves from one node to another, it means the travel between the two corresponding cities. For permutational scheduling problems, the order of the visited nodes means the schedule of jobs (e.g. Merkle and Middendorf 2000 and Gajpal and Rajendran 2006).

The behavior of this algorithm can be exemplified as illustrated in Fig. 1 (Colorni et al. 1991). As one can note, at t=0, an ant randomly chooses one path or the other. When it reaches its goal, a fitness function is calculated (e.g., the TSP problem uses the total travelled distance), and an artificial pheromone is deposited according. This pheromone deposit is a*positive reinforcement* strategy of the algorithm, and it tends to privilege the best solutions. To avoid saturation, Colorni et al. (1991) also developed a *negative reinforcement* strategy, called *pheromone evaporation*, formulated as a multiplication of all pheromone values by a heuristic coefficient $0 < \rho < 1$. According Colorni et al. (1991), after some iterations, the best solution will tend to dominate the poor ones (Fig. 1c), allowing all the ants will choose one single route.

The algorithmic steps of the ACO design by Colorni et al. (1991) are shown in Algorithm 1.

Algorithm 1: The ACO pseudo-code

```
1  Initialize
2  Repeat (at this level, each execution is named iteraction)
3      Each ant is positioned on the initial node
4      Repeat (at this level, each execution is named step)
5          Each ant uses a transition rule to increment the solution
             set
6          Update the pheromones according the local pheromone
             update rule
7      Until all ants have built a complete solution
8      Apply a local search procedure
9      Apply a pheromone global update rule
10 Until a stop criteria is satisfied
```

This algorithm can be explained as follows:

1) Each iteraction begins with the positioning of the ant on an initial node following a specific rule
2) All ants are then moved according to a *transition rule*, until they create a complete solution (a solution is a set of moves)
3) To choose the next node, the *transition rule* uses a problem-specific *visibility function*

The transition rule is chosen based on the probability of an ant choose a track. This probability is normally given by Eq. (1).

$$r_{ij}^{a} = \begin{cases} \frac{(\tau_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{h \in AllowedH} (\tau_{ih})^{\alpha} \cdot (\eta_{ih})^{\beta}} & if\ j \in AllowedH \\ 0 & otherwise \end{cases} \quad (1)$$

Where:

- r_{ij}^{a} is the probability of an ant a moves from i to j
- τ_{ij} is the pheromone level between i and j
- $AllowedH$ is a set of all allowed destinations (e.g., unscheduled jobs)
- η_{ij} is the visibility of the path i to j
- $0 < \alpha < 1, \quad 0 < \beta < 1$ are parameters

The visibility function η_{ij} is a problem-specific function, and its definition changes according the problem that the ACO is applied. For example, Bauer et al. (1999), define a visibility rule for scheduling problems based on the Earliest Due Date (EDD) rule, as shown in Eq. 2.

$$\eta_{ij} = 1/d_i, \quad (2)$$

4) After each move (also referred in the literature as *step*), the pheromones are updated according to a *local update rule*

A very common update rule found in literature is shown in Eq. 3 (e.g. Dorigo et al. 1996; Bauer et al. 1999):

$$\tau_{ij}^{t+1} = \rho \cdot \tau_{ij}^{t} + \sum_{a \in ants} \Delta_{ij}^{a}, \tag{3}$$

Where:

- τ_{ij}^{t+1} is the pheromone between sites i and j on the next time frame
- $0 < \rho < 1$ is the evaporation constant
- $ants$ is the set of the ants
- Δ_{ij}^{a} is how much pheromone ant a deposits in the path ij (usually a positive constant if a moves from i to j, 0 otherwise)

5) After all the ants have built a solution, two procedures are executed: the first, optional, is a local search procedure that tries to improve the solution created by the ants; the second is the application of a pheromone *global update rule*

6) This cycle occurs until a stop criteria is satisfied. Two usually founded stop criteria in the literature are number of interactions (e.g. Shyu et al. 2004) and algorithm stabilization (e.g. Lin et al. 2008)

The literature about ACO applied to the parallel scheduling problem

In parallel machine scheduling problems, it's assumed the availability of a set of m machines. Those machines can be identical, unrelated or uniform, but all capable to process any of the given jobs (Graham et al. 1979). The environment is named identical when all jobs have the same processing time in any machine. When the processing time varies on the machines, the environment is identified as unrelated parallel machines. The uniform parallel machine environment occurs when there is a proportional factor of the processing time in each machine. In our literature review, we found ACO applications related to the first two environments.

Concerning identical parallel machine environments, Sankar et al. (2005) use an ACO algorithm, based on an n- dimensional TSP approach aiming at minimizing makespan for the identical parallel machines scheduling problem with sequence-dependent setups. The same problem is addressed by Behnamian and Zandieh (2009). These authors use the ACO to generate an initial solution, which is then improved by Simulated Annealing and Variable Neighborhood Search techniques. Behnamian and Zandieh (2009) also present three local search algorithms: (i) the first consists of swapping the positions of two jobs in the sequence of the same machine; (ii) the second consists of swapping the positions of two jobs in the sequence of different machines; and (iii) the last transfers jobs from the machine with higher makespan to the machine with a lower makespan. Raghavan and Venkataramana (2006) seek the minimization of 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. Chang et al. (2008) develop an ant colony optimization system to address a multi-stage job-shop parallel-machine-scheduling problem. This paper also deals with the decision about the numbers of parallel machines in workstations dynamically scheduled. Raghavan and Venkataramana (2009) develop an ACO algorithm to solve the scheduling problem in a system of parallel processors with the objective of minimizing total weighted tardiness. Ali Berrichi and Yalaoui (2013) propose a bi-objective model to deal with parallel machine scheduling and maintenance planning problems simultaneously. The solution of the integrated model is based on multi-objective ant colony optimization approach.

Regarding applications in unrelated parallel machine environment, Zhou et al. (2007a, b) extends the algorithm proposed by Liao and Juan (2007) aiming at minimizing weighted tardiness. In their approach, three different aspects were implemented in the ACO algorithm: (i) The authors uses two types of pheromones: the first indicates the desirability of processing one specific job at one specific machine, and the second is related to the choice to process a job Ji after a job Jj; (ii) An heurist rule is used to obtain the initial pheromone levels; (iii) A new visibility rule is defined. The same problem is also addressed by Monch (2008). The author uses several approaches, including ACO. In this case, the ATC rule is used as a visibility function. Still addressing unrelated parallel machine scheduling problems, Arnaout et al. (2008) and (2009) propose and implement an ACO algorithm for minimizing makespan considering dependent setup times. To solve this problem, a two-stage algorithm is proposed and implemented: in the first stage, the jobs are allocated to the machines. In the second stage, the jobs are sequenced on each machine. Keskinturk et al. (2012) focus on minimizing average relative percentage of imbalance (ARPI) with sequence-dependent setup times in a parallel-machine environment. He compares an ACO algorithm to some heuristics, a Genetic algorithm and also mathematical programming. These authors conclude that the ACO outperformed all the other methods. Lin et al. (2008) develop an ant colony optimization (ACO) algorithm to solve the problem of scheduling unrelated parallel machines to minimize total weighted tardiness.

The literature about scheduling problems with outsourcing allowed

The study of outsourcing under machine scheduling models starts recently and the most studies focuses on single-stage problems (Qi 2011). Bertrand and Sridharan (2001) work with the single-machine scheduling problem with outsourcing aiming at minimizing the total tardiness. Engels et al. (2003) considers a single-machine scheduling problem to minimize the weighted sum of the total weighted completion times and the total outsourcing cost. Lee and Sung (2008) considers a set of n jobs that must be allocated to a single machine or be done by a third party (outsourced). The objective of such authors was minimizing the weighted sum of the outsourcing cost and the scheduling measure represented by the sum of completion time, subject to outsourcing budget. The same authors also studied the single-machine scheduling problem with outsourcing allowed aiming at minimizing maximum lateness and outsourcing costs and also minimizing total tardiness and outsourcing costs (Lee and Sung 2008). Tavares Neto and Godinho Filho (2012) proposes an ACO algorithm to solve the problem considered by Lee and Sung (2008). This algorithm proved to be most efficient than Lee and Sung (2008). All of these studies assume that the subcontractor has a large enough capacity so that the main decision is to determine which jobs will be outsourced, without the need of explicitly scheduling the outsourced jobs. Other paper within this category is Chen and Li (2008). Qi (2008) relax this assumption, considering the scheduling of the outsourced jobs on the subcontractor/s machine. This author deals with the single machine scheduling problem, modeling the problem for four different objective functions and solving them with dynamic programming.

Regarding parallel machines considering outsourcing, only three papers were found. Bartal et al. (2000) considered a parallel machine scheduling problem in which the objective is to minimize the sum of makespan and the total outsourcing costs. Ruiz-Torres et al. (2006) develop several heuristics for solving the parallel-machine scheduling problem with outsourcing aiming at minimizing the number of late orders and the total outsource machine time. Mokhtari and Abadi (2013) addresses a single-stage scheduling problem with outsourcing allowed. The manufacturer has an unrelated parallel machine system, and each subcontractor has its own single machine. The objective is to minimize sum of the total weighted completion time and total outsourcing cost. An integer programming formulation is presented and also a heuristic algorithm. The present research, on the other hand, focuses on the minimization of the sum of the total outsource cost and the sum of weighted tardiness of all tasks (delay costs) on a parallel machine manufacturing environment.

Research on scheduling with outsourcing allowed in complex shop environments is very limited (Qi 2011). Lee et al. (2002) develops an Advanced Planning and Scheduling model with outsourcing option with the objective of minimizing late orders. This model is solved by means of a genetic algorithm. Chung et al. (2005) study a job shop scheduling problem with outsourcing option. Qi (2009) deals with two-stage flow shop scheduling problem with outsourcing option for stage one. Qi (2011) extends this work to include other forms of outsourcing. Other papers which deal with the two-stage flow shop scheduling problem with outsourcing option are Lee and Choi (2011) and Chung and Choi (2012). Tavares Neto and Godinho Filho (2011) extended Lee and Sung (2008) problem, proposing an ACO algorithm to solve the scheduling problem of minimizing makespan in a permutational flow shop environment with the possibility of outsourcing certain jobs.

Problem definition

The problem studied in this paper can be explained as follows:

The following elements are inputs to the problem:

- A set J containing the jobs to be processed
- A constant n representing the number of elements of J
- A constant m representing the number of machines
- A constant *Budget* representing the maximum value of the total outsource cost

Each job J_i is composed by:

- A processing time p_i
- A due-date d_i
- An outsourcing cost o_i
- An outsourcing lead time l_i
- A cost of each the tardiness of each time period w_i
- (m+1) subsets of J are used to compose the final solution:
  - An unordered set O_π containing the jobs that will be executed
  - m ordered sets S_π^k, $k \in m$, containing the jobs sequenced on the k parallel machines

Finally, the following performance measures can be calculated:

- C_i is the completion time if the job i
- M_k is the makespan of a sequence of the machine k, defined as $\max_{i \in S_\pi^k}\{C_i\}$
- T_i is the tardiness of the job, defined as $\max\{0, C_i - d_i\}$

The goal is to minimize the value of the Eq. 4:

$$\sum_{i \in O_\pi} o_i + \sum_{i \in S_\pi^k} w_i \cdot T_i \qquad k = 1 \ldots m \tag{4}$$

There is one constrain, represented in Eq. 5:

$$\sum_{i \in O_\pi} o_i \leq Budget \tag{5}$$

This problem can be stated as $P_m/Budget/oc + \sum w_i T_i$, using a notation based on Graham et al. (1979). To solve this problem, two approaches are stated: In "A mathematical programming approach for the problem" section, a mathematical programming approach is proposed. In the section "The proposed algorithm," a technique based on the Ant Colony Optimization meta-heuristic is proposed. In "Computational Results," the results of the application of both optimization strategies are presented.

A mathematical programming approach for the problem

Based on the mathematical model for the problem $P_m // \sum (E_i + T_i)$ well known in the literature (e.g. Arenales et al. 2007), this section presents a mathematical model to solve the problem $P_m/Budget/oc + \sum w_i T_i$ problem. This model is presented on Eqs. 6–12:

$$\min \sum_{i=1}^{n} w_i \cdot T_i + \sum_{i \in O_\pi} o_i \tag{6}$$

$$\sum_{k=1}^{m+1} \sum_{i=0}^{n} x_{ijk} = 1 \qquad j = 1 \ldots n \tag{7}$$

$$\sum_{j=0}^{n} x_{0jk} \leq 1 \qquad k = 1 \ldots m+1 \tag{8}$$

$$\sum_{\substack{i=0,\\ i \neq h}}^{n} x_{ihk} - \sum_{\substack{j=0,\\ j \neq h}}^{n} x_{hjk} = 0 \qquad \begin{array}{l} h = 1 \ldots n, \\ k = 1 \ldots m+1 \end{array} \tag{9}$$

$$C_{jk} \geq C_{ik} - M + (p_{jk} + M) * x_{ijk} \qquad \begin{array}{l} i = 1 \ldots n \\ j = 1 \ldots n \\ k = 1 \ldots m \end{array} \tag{10}$$

$$C_{j(m+1)} \geq l_j \cdot x_{ij(m+1)} \qquad \begin{array}{l} i = 0 \ldots n, \\ j = 1 \ldots n \end{array} \tag{11}$$

$$T_i \geq C_{ik} - d_i \qquad \begin{array}{l} i = 0 \ldots n, \\ k = 1 \ldots m+1 \end{array} \tag{12}$$

Where:

- C_{ik} is the completion time of the job i on machine k
- x_{ijk}
$$= \begin{cases} 1 \text{ if job } j \text{ is executed immediately after job} \\ \quad i \; at \; machine \, k \\ 0 \text{ otherwise} \end{cases}$$
- M is a large number
- A "ghost" machine m+1, representing the outsourcing option, is created.

The model represented by Eqs. (3)–(10) can be explained as follows:

- Equation (6) is the objective function;
- Equation (7) imposes that each job j contains a single job i executed immediately before;
- Equation (8) guarantees that each machine k contains a single processing sequence;
- Equation (9) assure that each job i is executed just after a single job j, excluding the job 0, that establishes the beginning and the end of the sequences;
- Equations (10, 11) determines the completion time of the jobs;
- Equation (12) determines the tardiness of each job;

Based in this model, the following modifications were performed to solve the $P_m/Budget/oc + \sum w_i T_i$ problem:

On the next section, an ACO approach for this problem will be presented.

The proposed algorithm

The same concept presented previously is now applied to solve the $P_m/Budget/oc + \sum w_i T_i$ problem using the ACO technique. This algorithm is based on the idea of merge three decisions: (i) what is the sequence of the jobs; (ii) in which machines each job must be processed; (iii) if each job must be outsourced or not. To do so, the proposed algorithm uses two sets of pheromones, instead of the single one presented on the original algorithm:

- τ^1_{ijk}, related to the desirability of move job j to position i on machine k. This set is named here as sequence-pheromone matrix.
- $\tau^2_{is}, \quad s = \{0, 1\}$,related to the desirability of outsource job i. τ^2_{i0} is related to the desirability of process the job into the parallel manufacturing environment; τ^2_{i1} is related to the desirability of outsource the job. This set is named here as outsource-pheromone matrix.

To embrace the three decisions, the transition rule used in the proposed algorithm is composed of three sub-rules:

- Transition rule #1, used to determine the next job to be scheduled;
- Transition rule #2, used to determine in which machine the next job must be allocated (if it's not outsourced);
- Transition rule #3, used to determine if a job must be outsourced or not;

The full pseudo-code of this algorithm is presented in Algorithm 2.

Algorithm 2: The proposed ACO algorithm

```
1   Initialize pheromones
2   Repeat (at this level, each execution is named iteraction)
3       Each ant is positioned on the initial node
4       Repeat (at this level, each execution is named step)
5           Use a transition rule #1 to choose the next job to be
            scheduled
6           Use a transition rule #2 to choose the machine where
            the job must be scheduled
7           Use a transition rule #3 to choose if the job must be
            outsourced or not
7       Until all ants have built a complete solution
8       Apply the local search procedure at each intervalLocalSearch
            iteractions
9       Apply a pheromone global update rule
10  Until a stop criteria is satisfied
```

As presented in Algorithm 2 and in the Transition Rules 1, 2 and 3, the main goal of this algorithm is to allow the ant, in a single moment, take the three specified decisions: the job to be processed, the position of this job in a sequence and the outsource decision. This is a difference between this algorithm and the ACO algorithms presented in the classic literature (such as Dorigo et al. 1996 and Stutzle and Hoos 2000), that usually use this phase to just one decision (e.g. in the case of the ACO applied to TSP, the only decision is regarding to choose the next city to be visited).

Pheromone initialization

This phase uses two approaches, both already found in the ACO literature applied to another problems: the first one, used in pheromone matrix 1, already initializes the pheromone matrix using some problem information. This approach, used in the Fast Ant Colony Optimization proposed by Holthaus and Rajendran (2005), allows a simplification of the transition rule, as will be presented in the section "The transition rule #2". In the case of the pheromone matrix 2, the problem information is incorporated only at the transition rule, and the pheromone levels are initialized at the maximum level. This decision was made to allow the incorporation of a dynamic makespan value, that changes according the construction of the solution.

The pheromone sets are initialized as shown in Eqs. 13 and 14:

$$\tau^2_{is} = \tau_{max} \qquad \begin{array}{l} i = 1 \ldots n, \\ s = 0, 1 \end{array} \tag{13}$$

$$\tau^1_{ijk} = \begin{cases} \tau_{max}, \text{ if the sequence } ij \text{ exists in the MDD sequence} & k = 1 \ldots m \\ \tau_{min}, \quad otherwise \end{cases} \tag{14}$$

Where:

- τ_{max} and τ_{min} are algorithm parameters
- The MDD sequence is a sequence of jobs obtained based on the well-known MDD dispatch rule. To perform it, the jobs are ordered following the value of $(\max\{T + p_i, d_i\} - T)/w_i$. In this case, T is the sum of the processing times of the jobs previously selected.

The transition rule #1

The first transition rule is used to determine which job will be the next one scheduled. To perform this task, it is applied a transition rule with no visibility rule associated. The transition rule, very similar to the one proposed by Holthaus and Rajendran (2005), is presented in Eq. 15:

$$r^A_{ijk} = \frac{\tau^1_{ijk}}{\sum_{h \in Allowed H} \tau^1_{ihk}} \tag{15}$$

The transition rule #2

Once the next job to be scheduled is known, the transition rule #2 is applied. This rule determines in which machine the job must be scheduled in the case of this jib not to be outsourced. The transition rule #2 is presented in Eq. 16:

$$r^B_{ijk} = \frac{\tau^2_{ijk} \cdot \eta^\beta_{ik}}{\sum_{h=1}^{m} \left(\tau^2_{ijh} \cdot \eta^\beta_{ih}\right)} \tag{16}$$

Where the visibility rule η^β_{ik} is defined in Eq. 17:

$$\eta^\beta_{ik} = \frac{1}{(M_k \cdot p_i) \cdot w_i} \tag{17}$$

The transition rule #3

The last choice is to determine if a job must be outsourced or not. To perform this task, the transition rule #3 is applied. This rule is composed by: (i) an equation to calculate the probability of the job be performed in the parallel machine environment (Eq. 18) and (ii) an equation to calculate the probability of outsourcing the job (Eq. 19). The terms a and b of Eqs. 18 and 19 are shown in Eqs. 20 and (21), respectively. Note that this step is just a weighted random rule, and no memory (pheromone) effect was used.

$$r^C_{i0} = \frac{a}{a+b} \tag{18}$$

$$r^C_{i1} = \frac{b}{a+b} \tag{19}$$

$$a = \left(\frac{o_i}{w_i} + \frac{\max\{0, l_i - d_i\}}{p_i}\right)^\beta \tag{20}$$

$$b = \left(\frac{p_i}{\left(\frac{o_i}{w_i}\right) + \max\{0, l_i - d_i\}}\right)^\beta \tag{21}$$

The local search procedure

The local search used refines the solution, moving jobs from the parallel machine sets to the outsource set if this improves the objective function. The behavior of this algorithm is presented in Algorithm 3.

Algorithm 3: The proposed local search algorithm

1 At each $nIteractionsLocalSearch$ iteractions
2 Find the jobs $J_{ik} \in S_\pi^k$, that, when moved from S_π^k to O_π, promote the higher increase on the objective function on each machine $k = 1, \ldots, m$.
3 Outsource the jobs found previously

The pheromone update rule

The strategy used to update the pheromone rules of the sequence-pheromone matrix, used by a large number of the ACO algorithms presented in the literature (e.g., see Dorigo et al. 1996) is presented Eq. 22:

$$\tau_{ijk}^1 (t+1) = \rho \cdot \tau_{ijk}^1 (t) + \Delta\tau_{ijk}^1 \quad (22)$$

Where $\Delta\tau_{ijk}^1$ is defined in Eq. 23.

$$\Delta\tau_{ijk}^1 = \begin{cases} \tau_{add} & \text{If job i is at position j and in machine k} \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

Where τ_{add} is a constant value added to the pheromone value.

The strategy used to update the outsource-pheromone matrix is presented in Eq. 24:

$$\tau_{is}^2 (t+1) = \rho \cdot \tau_{is}^2 (t) + \Delta\tau_{is}^2 \quad (24)$$

Where $\Delta\tau_{is}^2$ is defined in Eqs. 25 and 26:

$$\Delta\tau_{i0}^2 = \begin{cases} \tau_{add} & \text{If job i is processed in the machines} \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

$$\Delta\tau_{i1}^2 = \begin{cases} \tau_{add} & \text{If job i is outsourced} \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

Computational results

Parameters used

Both solution strategies (mathematical model and ACO) were implemented and their results compared. The relevant technical details of the implementations are:

- For each n number of jobs, n={10, 20, 30, 40, 50, 60 and 70} (resulting in 7 problem classes), 20 problem instances were generated. The problems are specified as follows: the processing time of the jobs are sampled from an interval [1,10], the due dates from [0.3*sump, 0.7*sump], where *sump* is the sum of processing times of all tasks in the set; the outsourcing costs from [1,40], the outsource lead times from [1,30] and the available budget from [50,150]. The number of available machines was set to m=2. This problem generation setting is very similar to the one used in a lot of previous works (for example Tavares Neto and Godinho Filho 2011)
- The mathematical model was implemented in GAMS 2.0.33.5, with solver CPLEX 10, and run in a Core 2 Duo P8700@2.53GHz processor, with 4GB RAM and executing Windows 7. The execution of each instance was limitated by 1 h (3,600 s), regardless if the solver could prove the optimal solution or not. For each instance i, the best value of the objective function $f_{GAMS}(i)$ and the time required to obtain this value $t_{GAMS}(i)$ was stored.
- The ACO algorithm was implemented in JAVA, and run in the same computer using Ubuntu Linux. For each instance, 10 runs were executed. The algorithm parameters were established as: $\tau_{max} = 20$, $\rho = 0.9$, $\tau_{add} = 1$,$\beta = 1$,$nIteractionsLocalSearch = 50$, number of ants=100, number of iterations = 300. This parameters were determined using the following procedure. For a problem with $n = 10$, random sampled from the test instances set, the following values were tested: $\tau_{max} = [10, 20, 30]$, $\rho = [0.8, 0.9, 0.95, 0.99]$, $\tau_{add} = (1-\rho) \cdot \tau_{max}$, $\beta = [1, 2, 5]$. The values of number of ants and number of iterations were set to a larger value (300 and 600, respectively) and then decreased until the solution start to deteriorate. Finally, the value of $nIteractionsLocalSearch$ were set aiming to allow the local search be executed when the pheromone levels are stabilized. For each execution e of instance i, the best value of the objective function $f_{ACO}(i, e)$ and the time required to obtain this value $t_{ACO}(i, e)$ was stored.

For each execution of each iteration obtained by the ACO, two measurements were calculated: $gap\,(i, e) = (f_{ACO}\,(i, e) - f_{GAMS}\,(i))/f_{GAMS}\,(i)$ and the average time spent to solve the instance.

Results

Table 1 presents the average value of $gap\,(i, e)$ and its standard deviation, summarized by problem class. A graphical analysis of such gaps is found on Fig. 2.

Table 2 shows the average time required to process each instance class. During our experiments, it was found that the difficulty on solve some instances was higher than in others. Table 2 shows the average time required to process each instance class, as well as the standard deviation. As one can

Table 1 Average and standard deviation of the values of $gap\ (i, e)$

Number of jobs (n)	Gap (Average) (%)	Gap (SD) (%)
10	3.9	4.5
20	9.2	14.8
30	8.2	27.0
40	−57.3	14.2
50	−75.4	8.4
60	−81.6	2.3
70	−84.8	2.2

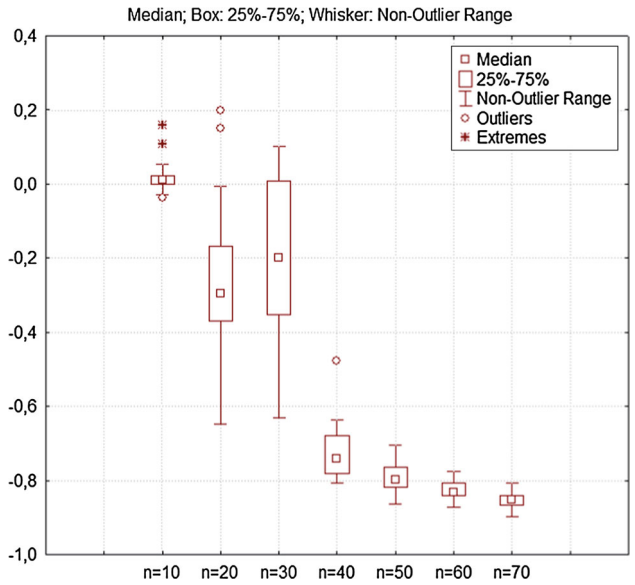

Fig. 2 Graphical analysis of the values of $gap\ (i, e)$

notice in this table, when n=10, the time required to solve using the mathematical programming approach contains a significantly higher standard deviation than any result from the ACO. Moreover, the values for the standard deviation for n>10 on the mathematical programming is zero, and the average is 3,600. This means that the upper limit for processing time of the exact method used is reached in all the instances with n>10.

From the presented analysis, the following remarks can be derived:

- For jobs with n=10, 20 and 30, seeing Table 1 (average gaps), one can notice that the mathematical programming reaches better solution than the ACO algorithm optimal solution. These average gaps are small, ranging from 3.9 to 9.2. Still regarding these instance sets, Fig. 2 shows that the dispersion of the gaps for n=10 are significantly shorter than the ones found for n=20 and n=30. In addition, the data presented in Fig. 2, allows one to conclude that the mathematical programming approach shows better results than ACO for mostly of the instances sets when n=10 (positive gaps). On the other hand, when n = 20 and 30, the ACO approach shows better results for mostly of the instances sets.
- For larger instances (n=40, 50, 60 and 70) the mathematical programming strategy could not find the optimal solution in the time specified (3,600 s). For these problems, the ACO algorithm presented better results than the mathematical programming approach, with low standard deviation.
- For medium-sized instances (n= 20 and 30), the ACO results contain a larger standard deviation. This occurs mainly because of the high proportional impact of small changes on the solution (e.g., the relative impact of switch the position of two tasks in a 20-size problem used to be larger than the same operation in a 70-size problem). For instances with n=40, 50, 60 and 70, the standard deviation decreases.
- As expected, the computational times required for the ACO approach increases accordingly to the size of the problem. However, this time is significantly lower than the time required for the mathematical programming approach (only for n=10, the optimal value was found within 3,600 s)

Table 2 Time required for processing each problem class (seconds)

Number of jobs (n)	ACO approach		Mathematical programming approach	
	Average time (s)	SD	Average time (s)	SD
10	0.2	0.01578711	1,815	1, 499
20	0.6	0.02555255	3,600	0
30	1.2	0.06094313	3,600	0
40	2.0	0.09435019	3,600	0
50	3.0	0.07265782	3,600	0
60	4.1	0.12696584	3,600	0
70	5.4	0.22329482	3,600	0

Conclusions

This paper aims at proposing, implementing and evaluating an ACO algorithm for the parallel machine scheduling problem with outsourcing allowed. The goal is to minimize the sum of outsource and delay costs. The problem can be stated as $P_m/Budget/oc+\sum w_i T_i$, using a notation based on Graham et al. (1979). To the best of our knowledge, this is the first paper to deal with this problem. Such problem can be found in practice in operations management field. Such examples are:

– an environment with parallel machines being bottlenecks of the production system: in these cases, some jobs can be

outsourced in order to relieve some load from the parallel machines
- a distribution center which performs some activities in a parallel machine environment and that these activities can also be outsourced in order to reduce lead time and prevent delays.

Firstly, a mathematical programming approach was developed. After this, an ACO algorithm was proposed. This algorithm is composed of three sequential transition rules, each one responsible for one different decision: the first one decides the next job to be scheduled; the second decides the machine to schedule a job (if the job is performed in-house); the third decides if the job must be outsourced or not.

The results showed that this algorithm, for instances of size larger or equal to 20 jobs, could reach better solutions than the ones found using the mathematical programming method when this was limited by 1 h execution time. Moreover, the times required to reach a solution were significantly smaller when the ACO is executed. This result, where the output of natural inspired algorithms reaches better results than exact mathematical methods (when those are bounded by time and/or computational resources), is also presented in the literature. Some examples are: ACO (Tavares Neto and Godinho Filho 2011), Genetic Algorithms (Chen and Ji 2007), among others (Jin et al. 2013).

Besides the proposition of a NP-Hard scheduling problem and the proposal of two implementation techniques (a MIP model and an ACO algorithm), this paper brings an industrial relevance on the field of operational level decisions when discuss the problem of allowing partial outsourcing on a scheduling problem. This issue is a rising trend in the literature, that is demanding for studies that aims on the formal definition of the problem (discussed here in the MIP formulation) and optimization techniques that allows one to obtain results relevant to practical applications.

Some future work that can be developed in the algorithm itself, especially regarding future advances on the algorithm, specially aiming to find a pheromone initialization procedure of stages two and three to allow their transition rules be simplified to Eq. 15.

From the point of view of the problem itself, there are several future trends that can be pursued: following the scheduling literature, maybe the logical next step is to add sequence-dependent setups for the tasks, as well as different characterizations of the outsourcing problem (e.g., adding a capacity limit to the supplier, or logistic boundaries).

References

Ali Berrichi, A., & Yalaoui, F. (2013). Efficient bi-objective ant colony approach to minimize total tardiness and system unavailability for a parallel machine scheduling problem. *International Journal of Advanced Manufacturing Technology*,. doi:10.1007/s00170-013-4841-0.

Antelo, M., & Bru, L. (2010). Outsourcing or restructuring: The dynamic choice. *International Journal of Production Economics*, *123*, 1–7.

Arenales, M., Armentano, V., Morabito, R., & Yanasse, H. (2007). *Pesquisa operacional. [S.l.]*. Amsterdam: Elsevier.

Arnaout, J. (2013). Ant colony optimization algorithm for the Euclidean location-allocation problem with unknown number of facilities. *Journal of Intelligent Manufacturing*, *24*(1), 45–54.

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., Musa, R., & Rabadi, G. (2012). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines-part II: Enhancements and experiments. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-012-0672-3.

Arnaout, J., Rabadi, G., & Musa, R. (2009). A two-stage ant colony optimization algorithm to minimize the makespan on unrelatedparallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*.

Arnaout, J., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *21*(6), 693–701.

Baker, K. R., & Bertrand, J. W. M. (1982). A dynamic priority rule for scheduling against due-dates. *Journal of Operations Management*, *3*, 37–42.

Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., & Stougie, L. (2000). Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics*, *13*, 64–78.

Bauer, A., Bullnheimer, B., Hartl, R. F., & Strauß, C. (1999). Applying ant colony optimization to solve the single machine total tardiness problem. Report Series SFB "Adaptive Information Systems and Modelling in Economics and Management Science", 42. SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, Vienna.

Behnamian, J., & Zandieh, M. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an aco, sa and vns hybrid algorithm. *Expert Systems with Applications*, *36*(6), 9637–9644.

Bertrand, J. W. M., & Sridharan, V. (2001). A study of simple rules for subcontracting in make-to-order manufacturing. *European Journal of Operational Research*, *128*(3):509–531.

Boulaksil, Y., & Fransoo, J. C. (2009). Order release strategies to control outsourced operations in a supply chain. *International Journal of Production Economics*, *119*, 149–160.

Brucker, P. (2007). *Scheduling algorithms. [S.l.]* (5th ed.). Berlin, Heidelberg: Springer.

Bullnheimer, B., Hartl, R., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, *89*, 319–318.

Chang, P.-T., Lin, K.-P., Pai, P.-F., Zhong, C.-Z., Lin, C.-H., & Hung, L.-T. (2008). Ant colony optimization system for a multi-quantitative and qualitative objective job-shop parallel-machine-scheduling problem. *International Journal of Production Research*, *46*(20), 5719–5759.

Chen, K. J., & Ji, P. (2007). Development of a genetic algorithm for scheduling products with a multi-level structure. *International Journal of Advanced Manufacturing Technology*, *33*, 1229–1236.

Chen, Z.-L., & Li, C.-L. (2008). Scheduling with subcontracting options. *IIE Transactions*, *40*, 1171–1184.

Chung, D., Lee, K., Shin, K., & Park, J. (2005). A new approach to job shop scheduling problems with due date constraints considering

operating subcontracts. *International Journal of Production Economics*, *98*, 238–250.
Chung, D., & Choi, B. (2012). Outsourcing and scheduling for two-machine ordered flow shop scheduling problems. *European Journal of Operational Research*, *226*(1), 46–52.
Colorni, A., Dorigo, M., & Maniezzo, V. (1991). *Distributed optimization by ant colonies*. Paris: European Conference of Artificial Life.
Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, *26*, 29–41.
Dorigo, M., & Stutzle, T. (2004). *Ant colony optimization. A bradfort book*. Cambridge: MIT Press.
Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, *344*(2–3), 243–278.
Dowsland, K. A., & Thompson, J. M. (2005). Ant colony optimization for the examination scheduling problem. *Journal of the Operational Research Society*, *56*, 426–438.
Engels, D. W., Karger, D. R., Kolliopoulos, S. G., Sengupta, S., Uma, R. M., & Wein, J. (2003). Techniques for scheduling with rejection. *Journal of Algorithms*, *49*, 175–191.
Gajpal, Y., & Rajendran, C. (2006). An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. *International Journal of Production Economics*, *101*, 259–272.
Gambardella, L. M., & Dorigo, M. (2000). An ant colony system with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, *12*, 237–255.
Gonzalez, R., Gasco, J., & Llopis, J. (2006). Information systems outsourcing: A literature analysis. *Information and Management*, *43*, 821–834.
Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic machine scheduling: A survey. *Annals of Discrete Mathematics*, *5*, 287–326.
Holthaus, O., & Rajendran, C. (2005). A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. *Journal of the Operational Research Society*, *56*, 947–953.
Huang, R., Yang, C., & Cheng, W. (2013). Flexible job shop scheduling with due window—a two-pheromone ant colony approach. *International Journal of Production Economics*, *141*(2), 685–697.
Jin, X., Li, K., & Sivakumar, A. I. (2013). Scheduling and optimal delivery time quotation for customers with time sensitive demand. *International Journal of Production Economics*, Article in Press.
Keskinturk, T., Yildirim, M. B., & Barut, M. (2012). An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times. *Computers and Operations Research, 39*(6), 1225–1235.
Kumar, R., & Allada, V. (2007). Scalable platforms using ant colony optimization. *Journal of Intelligent Manufacturing*, *18*(1), 127–142.
Lee, Y. H., Jeong, C. S., & Moon, C. (2002). Advanced planning and scheduling with outsourcing in manufacturing supply chain. *Computers and Industrial Engineering*, *43*, 351–374.
Lee, I. S., & Sung, C. S. (2008a). Minimizing due date related measures for a single machine scheduling problem with outsourcing allowed. *European Journal of Operational Research*, *186*, 931–952.
Lee, I. S., & Sung, C. S. (2008b). Single machine scheduling with outsourcing allowed. *International Journal Production Economics*, *111*, 623–634.
Lee, K., & Choi, B. (2011). Two-stage production scheduling with an outsourcing option. *European Journal of Operational Research*, *213*, 489–497.
Lee, H.-Y. (2012). Renovation scheduling to minimize user impact of a building that remains in operation. *Automation in Construction*, *22*(1), 398–405.
Leung, J. Y.-T., & Anderson, J. H. (2004). *Handbook of scheduling: Algorithms, models, and performance analysis* (1224 p) Chapman & Hall/CRC.
Liao, C.-J., & Juan, H.-C. (2007). An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. *Computers & Operations Research*, *34*(7), 1899–1909.
Lin, C.-W., Lin, Y.-K., & Hsieh, H.-T. (2013a). Ant colony optimization for unrelated parallel machine scheduling. *International Journal of Advanced Manufacturing Technology*, doi:10.1007/s00170-013-4766-7.
Lin, B., Lu, C., Shyu, S., & Tsai, C. (2008). Development of new features of ant colony optimization for flowshop scheduling. *International Journal of Production Economics*, *112*(2), 742–755.
Liu, X.-J., & Yi, Hong. (2013b). Application of ant colony optimization algorithm in process planning optimization. *Journal of Intelligent Manufacturing*, *24*(1), 1–13.
Marimuthu, S., Ponnambalam, S. G., & Jawahar, N. (2009). Threshold accepting and ant colony optimization algorithms for scheduling m-machine flow shops with lot streaming. *Journal of Materials Processing Technology*, *209*, 1026–1041.
Merkle, D., & Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In S. Cagnoni, R. Poli, Y. Li, B. Paechter, & T. C. Fogarty (Eds.), *Real-world applications of evolutionary computing. EvoWorkshops 2000: EvoIASP, EvoSCONDI, EvoTel, EvoSTIM, EvoROB, and EvoFlight* (pp. 287–296). London, UK: Springer-Verlag.
Mokhtari, H., & Abadi, I. N. K. (2013). Scheduling with an outsourcing option on both manufacturer and subcontractors. *Computers & Operations Research*, *40*(5), 1234–1242.
Monch, L. (2008). Heuristics to minimize total weighted tardiness of jobs on unrelated parallel machines. In *4th IEEE international conference on automation science and engineering*, Washington DC, USA, August 23–26.
Pinedo, M. L. (2009). *Planning and scheduling in manufacturing and services. [S.l.], 2nd edition*. Berlin: Springer p. 555.
Pinedo, M. L. (2012). *Scheduling: Theory, algorithms and systems, 4th Edition* (693 p), Berlin: Springer.
Prakash, A., Tiwari, M. K., & Shankar, R. (2008). Optimal job sequence determination and operation machine allocation in flexible manufacturing systems: An approach using adaptive hierarchical ant colony algorithm. *Journal of Intelligent Manufacturing*, *19*(2), 161–173.
Qi, X. (2008). Coordinated logistics scheduling for in-house production and outsourcing. *IEEE Transactions on Automation Science and Engineering*, *5*(1), 188–192.
Qi, X. (2009). Two-stage production scheduling with an option of outsourcing from a remote supplier. *Journal of Systems Science and Systems Engineering*, *18*(1), 1–15.
Qi, X. (2011). Outsourcing and production scheduling for a two-stage flow shop. *International Journal of Production Economics*, *129*(1), 43–50.
Raghavan, N. R. S., & Venkataramana, M. (2006). Scheduling parallel batch processors with incompatible job families using ant colony optimization. In *Proceedings of the 2006 IEEE international conference on automation science and engineering*. Shangai, China, October 7–10.
Raghavan, N. R. S., & Venkataramana, M. (2009). Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization. *International Journal of Advanced Manufacturing Technology*, *41*, 986–996.
Ruiz-Torres, A. J., Ho, J. C., & López, F. J. (2006). Generating Pareto schedules with outsource and internal parallel machines. *International Journal of Production Economics*, *103*, 810–825.
Samrout, M., Kouta, R., Yalaoui, F., Chatelet, E., & Chebbo, N. (2007). Parameter/s setting of the ant colony algorithm applied in preventive maintenance optimization. *Journal of Intelligent Manufacturing*, *18*(6), 663–677.

Sankar, S. S., Ponnambalam, S. G., Rathinavel, V., & Visveshvaren, M. S. (2005). Scheduling in parallel machine shop: an ant colony optimization approach. *Proceedings of IEEE International Conference on Industrial Technology*, *2005*, 276–280.

Shyu, S., Lin, B., & Yin, P. (2004). Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. *Computers and Industrial Engineering*, *47*, 181–193.

Stutzle, T., & Hoos, H. H. (2000). Max-min ant system. *Future Generation Computer Systems*, *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. *Computers and Operations Research*, *38*, 1286–1293.

Tavares Neto, R. F., & Godinho Filho, M. (2012). Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*. doi:10.1016/j.engappai.2012.03.011.

Tavares Neto, R. F., & Godinho Filho, M. (2013). Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, *26*(1), 150–161.

Xu, R., Chen, H., & Li, X. (2012). Makespan minimization on single batch-processing machine via ant colony optimization. *Computers and Operations Research*, *39*(3), 582–593.

Yadav, V., & Gupta, R. K. (2008). A paradigmatic and methodological review of research in outsourcing. *Information Resources Management Journal*, *21*(1), 27–43.

Zapfel, G., & Bogl, M. (2008). Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, *113*, 980–996.

Zhou, H., Li, Z., & Wu, X. (2007a). Scheduling unrelated parallel machine to minimize total weighted tardiness using ant colony optimization. In *Proceedings of IEEE international conference on automation and logistics*, Jinan, China, August 18–21.

Zhou, R., Lee, H. P., & Nee, A. Y. C. (2008). Applying ant colony optimization algorithm to dynamic job shop scheduling problems. *International Journal of Manufacturing Research*, *3*(3), 301–320.

Zhuo, X., Zhang, J., & Chen, W. (2007b). A new pheromone design in acs for solving jsp. In *Proceedings of IEEE congress on evolutionary computation*, 25–28 September, Singapore.