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.