Contents lists available at ScienceDirect

Applied Soft Computing

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

Memetic algorithms for the job shop scheduling problem with operators

Raúl Mencía[a], María R. Sierra[a], Carlos Mencía[b], Ramiro Varela[a,*]

[a] *Department of Computing, University of Oviedo, Campus of Gijón, 33204 Gijón, Spain*
[b] *CASL, University College Dublin, Ireland*

ARTICLE INFO

Article history:
Received 9 May 2014
Received in revised form 4 March 2015
Accepted 2 May 2015
Available online 12 May 2015

Keywords:
Metaheuristics
Scheduling with operators
Local search
Memetic algorithm

ABSTRACT

The job-shop scheduling problem with operators (JSO) is an extension of the classic job-shop problem in which an operation must be assisted by one of a limited set of human operators, so it models many real life situations. In this paper we tackle the JSO by means of memetic algorithms with the objective of minimizing the makespan. We define and analyze a neighborhood structure which is then exploited in local search and tabu search algorithms. These algorithms are combined with a conventional genetic algorithm to improve a fraction of the chromosomes in each generation. We also consider two different schedule builders for chromosome decoding. All these elements are combined to obtain memetic algorithms which are evaluated over an extensive set of instances. The results of the experimental study show that they reach high quality solutions in very short time, comparing favorably with the state-of-the-art methods.

1. Introduction

In this paper we propose some memetic algorithms (MAs) to solve the job-shop scheduling problem with operators (JSO). This problem was proposed in [1] and is motivated by manufacturing processes in which part of the work is done by human operators sharing the same set of tools. The problem is formalized as a classical job shop scheduling problem (JSP) in which the processing of an operation on a given machine requires the assistance of one operator. The number of operators is limited and so some tasks will need to be delayed until an operator is available. In [1], the authors propose a number of exact and heuristics algorithms to solve the JSO. These algorithms are outperformed by the genetic algorithm (GA) proposed in [2] and by the exact best-first search algorithm proposed in [3].

Starting from the GA proposed in [2] for the JSO, we introduce herein new versions of the decoding operator, derived from the schedule generation scheme termed $OG\&T$ in [3], which allow the GA to search in different spaces. We then turn the GA into a MA by adding a local search procedure. To this aim, we propose and formalize a neighborhood structure for the JSO which is inspired in a structure for the classic JSP. The essential virtue of this structure is that it avoids moves that will not improve the current solution without computing the makespan of the candidate neighbors, thus saving a lot of computation time. This structure is exploited in two local search strategies: hill climbing and tabu search. To assess the performance of the proposed MA, we have conducted a thorough experimental study across instances of different sizes and characteristics. The results of this study show that our approach is among the best performing methods for the JSO.

Summarizing, the main contributions of this paper are the following: (1) we define a number of new decoding algorithms for a previous genetic algorithm designed to solve the JSO, (2) we define and formally analyze a new neighborhood structure for the JSO, (3) we design a memetic algorithm that combines the previous genetic algorithm with a tabu search algorithm which incorporates the neighborhood structure in its core. This algorithm outperforms the current state of the art for the JSO, and (4) from the formal and experimental studies carried out on the JSO, we obtained insights that may be useful to devise new algorithms for this problem and some of its variants.

The remainder of the paper is organized as follows. In Section 2 we review the literature on memetic algorithms and metaheuristics applied to JSP and some of its extensions, such as the JSO. In Section 3 we formally define the JSO and give a disjunctive model to represent schedules. In Section 4 we describe the $OG\&T$ schedule generation scheme. Section 5 is devoted to explain and study the proposed neighborhood structure. Section 6 describes the local search algorithms that exploit this neighborhood structure. Section 7 presents the main components of the GA and discuss how to

* Corresponding author. Tel.: +34 985182508.
E-mail address: ramiro@uniovi.es (R. Varela).

http://dx.doi.org/10.1016/j.asoc.2015.05.004
1568-4946/© 2015 Elsevier B.V. All rights reserved.

combine the local search algorithms with the GA and the decoding algorithms. Section 8 reports the design and results of the experimental study. Finally, we summarize the main conclusions of the paper and propose some ideas for future research in Section 9.

2. Literature review

In this section we review the main notions regarding memetic algorithms and some of the most relevant metaheuristics that have been applied to solve the JSP and some related problems.

2.1. Memetic algorithms

The term memetic algorithm (MA) was coined by Moscato in [4] to refer to the combination of evolutionary algorithms with local search. These methods are inspired by natural systems that combine the evolutionary adaptation of a population with individual learning over the lifetime of its members. The term memetic comes from the concept of *meme*, defined [5] as a unit of cultural evolution that can exhibit local refinement. In the context of evolutionary optimization, a meme is a local refinement strategy that may improve individuals. Other terms as hybrid genetic algorithms or genetic local searchers are often used in the literature to name MAs.

MAs are a class of hybrid metaheuristics [6] have been successfully used to solve complex optimization problems in discrete and continuous domains in areas such as operations research and artificial intelligence. In particular, they have a large track of success in constrained optimization problems such as scheduling [7]. MAs combine the capability of evolutionary algorithms, such as genetic algorithms, to explore solutions over the whole search space with the capability of local searchers to intensify the search in some promising areas. For this reason, designing a competent MA requires a proper combination of diversification and intensification search efforts. One of the most common approaches to combining them consists in applying local search to chromosomes just after they are generated, as it is pointed in [8], where the authors present an extensive taxonomy of MAs and discuss the main design issues. Some of these important issues are the chromosomes local search should be applied to, how often or for how long [7]. Another important decision is whether Lamarckian evolution is preferred or not. In Lamarckian evolution the characteristics learned during the local search are coded back into the chromosome. This way, these characteristics may better preserved for subsequent generations, but at the same time it may contribute to premature convergence. These and other issues are well discussed in [4,6–8].

2.2. Solving scheduling problems with metaheuristics

Hybrid metaheuristics have been applied to scheduling problems over the last decades. Arguably, one of the problems that has attracted most interest is the classic JSP with makespan minimization. The reason for this may that the JSP can model real life situations [9] and at the same time it is very hard to solve; its decision version is NP-complete in the strong sense and it is considered as one of the hardest problems in this class. As a result, a good number of approaches can be found in the literature, hybrid metaheuristics standing out among them. Building on seminal ideas proposed by Van Laarhoven et al. [10] and Dell'Amico and Trubian [11], sophisticated neighborhood structures have been designed which constitute the core of current state-of-the-art techniques as, for example, the tabu search algorithms proposed in [12,13], the hybrid genetic algorithms proposed in [14], or the recent approach in [15] which combines a differential algorithm with the well-known *i*-TSAB procedure proposed in [12].

In addition, several extensions to the JSP have been proposed in the literature, considering additional characteristics and constraints from real-world scenarios. In these settings, hybrid metaheuristics have often been shown to perform remarkably well, representing the state of the art in many cases. For example, the JSP with sequence dependent setup times was solved in [16] with a memetic algorithm; the JSP with maintenance considerations was considered in [17,18], and solved by hybrid evolutionary and chemical-reaction optimization algorithms respectively. The fuzzy flexible JSP in which the processing times of the tasks are uncertain and each task can be processed in different machines is considered in [19,20]; the processing times are modeled with fuzzy numbers and in both cases the problem is solved by means of hybrid metaheuristics. The JSO is also an extension of the JSP introduced in [1], where the authors propose an exact algorithm based on dynamic programming and some heuristic algorithms. These methods were outperformed by the genetic algorithms proposed in [2,21] and by the best-first search algorithm proposed in [3]. In this last paper, the authors propose the $OG\&T$ schedule generation schema which is used to generate the search space of their best-first search algorithm. The GA proposed in [21] exploits the $OG\&T$ as decoder. In [2], the authors introduce some improvements to this GA, such as the concept of weak Lamarckian evolution and a way to reduce the search space.

3. Description of the problem

In the JSO, we are given a set of n jobs $\{J_1, \ldots, J_n\}$, a set of m resources or machines $\{R_1, \ldots, R_m\}$ and a set of p operators $\{O_1, \ldots, O_p\}$. The job J_i consists of a sequence of v_i operations or tasks $(\theta_{i1}, \ldots, \theta_{iv_i})$. The task v has a single resource requirement R_v and a positive integer duration p_v. A feasible schedule is an assignment of starting times st_v and operators O_v for all operations v that satisfies the following constraints: (i) the operations of each job are sequentially scheduled, (ii) each machine can process at most one operation at any time, (iii) no preemption is allowed and (iv) each operation is assisted by one operator and one operator cannot assist more than one operation at a time. The objective is finding a feasible schedule that minimizes the completion time of all the operations, i.e. the makespan. The significant cases of this problem are those with $p < min(n, m)$, otherwise the problem is equivalent to the classic JSP.

A feasible schedule $\mathcal{S}$ can be represented by an acyclic graph of the form $G_{\mathcal{S}} = (V, K \cup H \cup I \cup Q)$, where the set V includes one node for each operation, nodes that represent the beginning of the sequence of operations assisted by the operator O_i, O_i^{start}, and the dummy nodes *start* and *end*. K represents precedence constraints among operations of the same job. I includes arcs of the form $(start, O_i^{start})$, $i = 1, \ldots, p$, and arcs $(start, \theta_{i1})$ and (θ_{ik_i}, end), $i = 1, \ldots, n$. H expresses the processing order of operations on the machines and Q expresses the sequences of operations assisted by each operator. The makespan is the cost of a *critical path* in $G_{\mathcal{S}}$. A critical path is a longest cost path from node *start* to node *end*.

Fig. 1 shows a solution graph for an instance with 3 jobs, 3 machines and 2 operators. Discontinuous arrows represent arcs in Q. So, the sequences of operations assisted by operators O_1 and O_2 are $(\theta_{21}, \theta_{11}, \theta_{32}, \theta_{12}, \theta_{13})$ and $(\theta_{31}, \theta_{22}, \theta_{23}, \theta_{33})$ respectively. In order to simplify the picture, if there are two arcs between the same pair of nodes, only the operator arc in Q is drawn. Continuous arrows represent arcs in K and doted arrows represent arcs in H. In this example, the critical path is given by the sequence $(\theta_{21}, \theta_{11}, \theta_{32}, \theta_{12}, \theta_{33})$, so the makespan is 14. Fig. 2 is a Gantt chart of the schedule represented by Fig. 1.

In order to simplify expressions, we define the following notation for a feasible schedule. The *head* r_v of an operation v is the cost