European Journal of Operational Research 245 (2015) 35–45

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Scatter search with path relinking for the flexible job shop scheduling problem

Miguel A. González, Camino R. Vela, Ramiro Varela*

Department of Computing, University of Oviedo. Campus of Gijón, 33204, Gijón, Spain

ARTICLE INFO

Article history:
Received 10 May 2014
Accepted 26 February 2015
Available online 3 March 2015

Keywords:
Scheduling
Flexible job shop
Scatter search
Path relinking
Neighborhood structures

ABSTRACT

The flexible job shop scheduling is a challenging problem due to its high complexity and the huge number of applications it has in real production environments. In this paper, we propose effective neighborhood structures for this problem, including feasibility and non improving conditions, as well as procedures for fast estimation of the neighbors quality. These neighborhoods are embedded into a scatter search algorithm which uses tabu search and path relinking in its core. To develop these metaheuristics we define a novel dissimilarity measure, which deals with flexibility. We conducted an experimental study to analyze the proposed algorithm and to compare it with the state of the art on standard benchmarks. In this study, our algorithm compared favorably to other methods and established new upper bounds for a number of instances.

1. Introduction

The job shop scheduling problem (JSP) is a simple model of many real production processes. It is one of the most classical and difficult scheduling problems and it has been studied for decades (Meeran & Morshed, 2014). However, in many environments the production model has to consider additional characteristics or complex constraints. In this work we consider the possibility of selecting alternative routes among the machines, which is useful in production environments where multiple machines are able to perform the same operation (possibly with different processing times), as it allows the system to absorb changes in the demand of work or in the performance of the machines. This problem is known as the flexible job shop scheduling problem (FJSP). It was first addressed by Brucker and Schlie (1990) and it has been object of intensive research since then.

Initially, researchers proposed hierarchical approaches, in which the machine assignment and the scheduling of operations were studied separately (Brandimarte, 1993). However, most of the works in the literature consider both subproblems at the same time. Mastrolilli and Gambardella (2000) developed two neighborhood structures to improve the TS algorithm proposed by Dauzère-Pérès and Paulli (1997). More recently, Ho et al. proposed a learnable genetic architecture (LEGA) (Ho, Tay, & Lai, 2007). It is also remarkable the hybrid genetic algorithm combined with a variable neighborhood descent search (hGA) developed by Gao, Sun, and Gen (2008). Other approaches as the climbing depth-bounded discrepancy search (CDDS) algorithm proposed by Hmida, Haouari, Huguet, and Lopez (2010), the hybrid harmony search and large neighborhood search (HHS/LNS) by Yuan and Xu (2013) or the hybrid genetic algorithm combined with tabu search (GA + TS) proposed by González, Vela, and Varela (2013) also obtain good results on standard instances. Bozejko, Uchronski, and Wodecki (2010) present a parallel approach with two double-level parallel metaheuristic algorithms based on neighborhood determination (TSBM^{2}h), with good results on one standard benchmark. Gutierrez and Garcia-Magario (2011) combine a modular genetic algorithm with repairing heuristics (MGARH), and obtain new upper bounds in one standard benchmark as well. Just to have a general picture of the state of the art in FJSP, we could said that TS, hGA, CDDS, HHS/LNS, TSBM^{2}h, MGARH and GA + TS show the best performance among the aforementioned methods. However, none of them dominates the others in the sense of obtaining the best solutions or taking the lowest time for all benchmarks. Also, the performance of some of these methods strongly varies with the benchmark and some of them have not been evaluated on all the common benchmarks. For example, MGARH is among the best for one particular benchmark while it has not been evaluated on others; and hGA is also the best method for one benchmark, while it is clearly not so good in others.

In spite of that metaheuristics have been widely applied to scheduling problems, a powerful method as scatter search with path relinking has been rarely used in flexible environments. Maybe this is due to the difficulty of defining a tight distance between schedules. As far as we known, only in Jia and Hu (2014), where the authors combine path relinking with tabu search and consider multiobjective optimization, a distance is defined for the FJSP.

* Corresponding author. Tel.: +34 985182508.
E-mail addresses: mig@uniovi.es (M. A. González), crvela@uniovi.es (C. R. Vela), ramiro@uniovi.es (R. Varela).

http://dx.doi.org/10.1016/j.ejor.2015.02.052
0377-2217/

In this paper we propose new neighborhood structures for the FJSP with makespan minimization. We define feasibility and non improving conditions as well as algorithms for fast estimation of neighbors' quality. We also define a dissimilarity measure between two solutions which takes into account the flexible nature of the problem. These new structures and the dissimilarity measure are incorporated into a hybrid metaheuristic which uses scatter search with path relinking and tabu search as improvement method. We conducted an experimental study to analyze our proposal and to compare it with the state of the art.

The remainder of the paper is organized as follows. In Section 2 we formulate the problem and describe the solution graph model. In Section 3 we define the proposed neighborhood structures. Section 4 details the new dissimilarity measure and the metaheuristics used. In Section 5 we report the results of the experimental study, and finally Section 6 summarizes the main conclusions of this paper.

2. Problem formulation

In the job shop scheduling problem (JSP), there are a set of jobs $J = \{J_1, \ldots, J_n\}$ that must be processed on a set $M = \{M_1, \ldots, M_m\}$ of physical resources or machines, subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i = 1, \ldots, n$, consists of n_i operations $O_i = \{o_{i1}, \ldots, o_{in_i}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each operation o_{ij} requires the uninterrupted and exclusive use of one of the machines for its whole processing time.

In the flexible JSP (FJSP), an operation o_{ij} is allowed to be executed in any machine of a given set $M(o_{ij}) \subseteq M$. The processing time of operation o_{ij} on machine $M_k \in M(o_{ij})$ is $p_{o_{ij}k} \in \mathbb{N}$. Notice that the processing time of an operation may be different in each machine and that a machine may process several operations of the same job. The goal is to build up a feasible schedule which consists in assigning both a machine and a starting time to each operation in the set $O = \cup_{1 \le i \le n} O_i$, in such a way that all constraints hold. The objective function is the makespan, which should be minimized. The FJSP is NP-hard as it is a generalization of the JSP which has proven to be NP-hard (Garey, Johnson, & Sethi, 1976).

A solution can be alternatively viewed as a pair (α, π) where α represents a feasible assignment of each operation $o_{ij} \in O$ to a machine $M_k \in M(o_{ij})$, denoted $\alpha(o_{ij}) = k$, and π is a processing order of the operations on all the machines in M compatible with the job sequences.

Let $PJ_{o_{ij}}$ and $SJ_{o_{ij}}$ denote the operations just before and after o_{ij} in the job sequence and $PM_{o_{ij}}$ and $SM_{o_{ij}}$ the operations right before and after o_{ij} in the machine sequence in a solution (α, π), if they exist. The starting and completion times of o_{ij}, denoted $St_{o_{ij}}$ and $C_{o_{ij}}$ respectively, can be calculated as $St_{o_{ij}} = \max(C_{PJ_{o_{ij}}}, C_{PM_{o_{ij}}})$ (if an operation is the first in its job or in its machine sequence, the corresponding $C_{PJ_{o_{ij}}}$ or $C_{PM_{o_{ij}}}$ is taken to be 0) and $C_{o_{ij}} = St_{o_{ij}} + p_{o_{ij}k}$ being $k = \alpha(o_{ij})$. The objective is to find a solution (α, π) that minimizes the makespan, denoted as $C_{max}(\alpha, \pi) = \max_{o_{ij} \in O} C_{o_{ij}}$.

2.1. Solution graph and criticality

We define the following solution graph model for the FJSP. In accordance with this model, a machine assignment α and a feasible operation processing order π can be represented by an acyclic directed graph $G(\alpha, \pi) = (V, A \cup R(\alpha, \pi))$, where each node v in V represents either an operation of the problem, labeled with the assigned machine M_k, or one of the dummy nodes *start* and *end*, which are fictitious operations with processing time 0.

The set A contains *conjunctive arcs* representing job processing orders and the set $R(\alpha, \pi)$ contains *disjunctive arcs* representing machine processing orders. The arc (v, w) is weighted with the processing time, p_{vk}, of the operation in v on the assigned machine M_k. If w is the first operation in the processing order of its job, $J(w)$, there is an arc $(start, w)$ in $G(\alpha, \pi)$ with weight 0 and if w is the last operation in the job processing order, there is an arc (w, end) with weight p_{wk}, $k = \alpha(w)$. Fig. 1 shows a solution graph for a problem with 3 jobs and 3 machines.

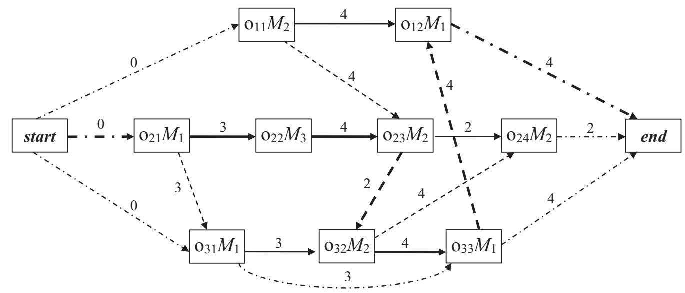

Fig. 1. A feasible schedule to a problem with 3 jobs and 3 machines represented by a solution graph. Bold-face arcs show a critical path whose length, i.e., the makespan, is 21.

The set $R(\alpha, \pi)$ is partitioned into subsets $R_k(\alpha, \pi)$, where $R_k(\alpha, \pi)$ is a minimal set of arcs defining a processing order for all operations requiring the machine M_k.

The makespan of the solution (α, π) is the cost of a critical path in $G(\alpha, \pi)$, i.e., a directed path from node *start* to node *end* having maximum cost. Bold-face arcs in Fig. 1 represent a critical path. Nodes and arcs in a critical path are also termed critical. We define a critical block as a maximal subsequence of consecutive operations in a critical path requiring the same machine. Notice that with this definition, a critical block may contain more than one operation of the same job. This makes a difference w.r.t. other definitions in the literature, as for example those given in Mastrolilli and Gambardella (2000) or in González et al. (2013), and has some influence in the critical block length and neighborhood size, as we will detail later.

The concept of critical block is important as most neighborhood structures proposed for job shop problems rely on exchanging the processing order of operations in critical blocks (Amico & Trubian, 1993; Mati, Dauzere-Peres, & Lahlou, 2011; Van Laarhoven, Aarts, & Lenstra, 1992). The structures proposed in Section 3.1 include moves of this type as well, but as we shall see, in the FJSP we have to introduce an additional type of move to deal with the machine assignment subproblem.

To formalize the description of the neighborhood structures, we introduce the concepts of head and tail of an operation v, denoted r_v and q_v respectively, which are calculated as follows:

$$r_{start} = q_{end} = 0$$
$$r_v = \max(r_{PJ_v} + p_{PJ_v k_1}, r_{PM_v} + p_{PM_v k})$$
$$k = \alpha(v), k_1 = \alpha(PJ_v)$$
$$r_{end} = \max_{v \in PJ_{end}, k = \alpha(v)} \{r_v + p_{vk}\}$$
$$q_v = \max(q_{SJ_v} + p_{SJ_v k_2}, q_{SM_v} + p_{SM_v k})$$
$$k = \alpha(v), k_2 = \alpha(SJ_v)$$
$$q_{start} = \max_{v \in SJ_{start}, k = \alpha(v)} \{q_v + p_{vk}\}$$

Abusing notation, SJ_{start} (PJ_{end}) denotes the set consisting of the first (last) operation processed in each of the n jobs. A node v is critical if and only if $C_{max} = r_v + p_{v\alpha(v)} + q_v$.

3. Neighborhood structures

In this section we propose several neighborhood structures for the FJSP, some of them focus on the sequencing subproblem and so they rely on changing the processing order of operations on a machine,