Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematical Modelling 33 (2009) 1560–1572

APPLIED MATHEMATICAL MODELLING

www.elsevier.com/locate/apm

Permutation-induced acyclic networks for the job shop scheduling problem

Tamer F. Abdelmaguid [*]

Mechanical Design and Production Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt

Received 15 May 2007; received in revised form 2 February 2008; accepted 11 February 2008
Available online 20 February 2008

Abstract

In the literature of the combinatorial optimization problems, it is a commonplace to find more than one mathematical model for the same problem. The significance of a model may be measured in terms of the efficiency of the solution algorithms that can be built upon it. The purpose of this article is to present a new network model for the well known combinatorial optimization problem – the job shop scheduling problem. The new network model has similar structure as the disjunctive graph model except that it uses permutations of jobs as decision variables instead of the binary decision variables associated with the disjunctive arcs. To assess the significance of the new model, the performances of exact branch-and-bound algorithmic implementations that are based on both the new model and the disjunctive graph model are compared.

Keywords: Job shop scheduling; Disjunctive graph model; Branch-and-bound; Exact algorithms

1. Introduction

The job shop scheduling problem (JSSP) is concerned with sequencing a set of jobs, J, on a set of technologically different machines or work centers, M, each is capable of processing at most one job at a time. Jobs follow dissimilar processing routes among the machines, and a job cannot be processed on more than one machine simultaneously. Furthermore, preemption is not allowed, and a job is permitted to have multiple visits to any machine. This article addresses the static, deterministic version of the problem in which raw materials for all jobs are assumed to be ready for processing at the beginning of the schedule, and the processing times are deterministic.

The JSSP arises in low-volume production systems in which products are made to order. In these production systems, jobs usually differ considerably in their processing sequences and times. Solving the JSSP is particularly important for the efficient utilization of production resources and for satisfying due dates. However,

[*] Tel.: +20 16 2689 333; fax: +20 23 5693 025.
E-mail address: tabdelmaguid@alumni.usc.edu

0307-904X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2008.02.004

the JSSP is fairly complex. Even for the small case of three jobs and three machines, the JSSP, with the objective of minimizing the makespan, is known to be NP-hard (Sotskov and Shakhlevich [1]). Earlier complexity results for the JSSP are provided by Garey et al. [2]. This high level of complexity demands continuous efforts for developing efficient solution approaches.

1.1. Problem definition and notations

In the JSSP, each job consists of an ordered list of operations that represents its processing route. We denote $I = \{1, 2, \ldots, v\}$ as the set of all operations' indexes. The operations' indexes are assigned such that for job $k \in J$, the subset of consecutive indexes, $I_k = \{\alpha_k, \alpha_k + 1, \alpha_k + 2, \ldots, \omega_k\} \subseteq I$, includes the indexes of operations belonging to that job; where in the set I_k, the operation with the lower index is to be processed first. For operation i, the time needed to finish its processing is p_i, the job to which it belongs is denoted $jb(i)$, and its work center or machine is denoted $wc(i)$. The task of the scheduling process is to determine the start time s_i for every operation $i \in I$. In addition to the *technological or precedence constraints* which define the mandatory processing sequence of operations belonging to the same job, the following set of constraints must be satisfied by a solution to be feasible. This set of constraints is in a disjunctive (either-or) form, and it represents the condition that operations on the same machine must be processed in different time intervals

$$s_i \geqslant s_j + p_j \text{ or } s_j \geqslant s_i + p_i \quad \forall i, j \in I, \text{ where } wc(i) = wc(j) \text{ and } jb(i) \neq jb(j). \tag{1}$$

1.2. The disjunctive graph model

The disjunctive graph model (Roy and Sussman [3]) has been used as a standard network representation for the job shop scheduling problem. Based on the same concepts found in the structure of the activity-on-arc project networks, nodes in the disjunctive graph model are used to represent the event of starting the processing of an operation. Here, we use operation indexes as labels for nodes. An arc connecting two consecutive nodes, i and $i + 1$, where operations i and $i + 1 \in I_k$ for some job k, represents the activity of processing operation i and has a length of p_i. Using the language of project scheduling, the operations belonging to the same job are represented as a series of consecutive activities. In addition, two dummy source (0) and terminal ($v + 1$) nodes are defined to respectively represent the events of starting and ending the schedule. To complete the project network, dummy arcs, $(0, \alpha_k)$ and $(\omega_k, v + 1)$ for all $k \in J$ with zero lengths are added. The project network $N = (V; Z)$ for the set of nodes $V = \{i: i = 0, \ldots, v + 1\}$ and the set of directed arcs $Z = \cup_{k \in J} Z_k$, where $Z_k = \{(i, i + 1): i, i + 1 \in I_k\} \cup \{(0, \alpha_k), (\omega_k, v + 1)\}$, is sufficient to represent the technological constraints; however, it is not a complete representation for the JSSP as constraints (1) are not taken into consideration.

To represent constraints (1), disjunctive pairs of arcs are defined over the nodes of all the operations that share the same machine and belong to different jobs. The *disjunctive pair* of arcs between nodes i and j on a graph, written as $\langle i,j \rangle$, is formed by the two arcs (i,j) and (j,i) such that any path in the graph is allowed to include at most one of them. A *selection* on the disjunctive pair $\langle i,j \rangle$, denoted $Sel(\langle i,j \rangle)$, is either arc (i, j) with length p_i or arc (j,i) with length p_j. The selection (i,j) is called the *complement* of the selection (j,i) and visa versa. A predetermined selection $Sel(\langle i,j \rangle) = (i,j)$ is said to be *complemented* when it is changed to (j,i). A *complete selection* in a disjunctive graph is obtained when selections for all disjunctive pairs of arcs are decided. The disjunctive graph is defined as $G = (V; Z, W)$ for the set of disjunctive pairs $W = \cup_{m \in M} W_m$, where $W_m = \{\langle i,j \rangle: i,j \in Q_m \text{ and } jb(i) \neq jb(j)\}$ for the set of nodes $Q_m = \{i{:}i \in I \text{ and } wc(i) = m\}$.

In the disjunctive graph model, each disjunctive arc is associated with a binary decision variable whose value determines its selection. Solution algorithms that are built upon that model operate in the domains of these binary decision variables to determine a complete selection on the disjunctive graph. A complete selection should not result in a cyclic graph so as to be able to interpret it into a unique feasible schedule.

One of the early exact solution algorithms that are based on the disjunctive graph model is the implicit enumeration algorithm of Balas [4]. However, the majority of the disjunctive graph-based solution algorithms are branch-and-bound. Examples in the literature include: Charlton and Death [5], Carlier and Pinson [6], Applegate and Cook [7] and Brucker et al. [8]. Jain and Meeran [9] provide a recent review. It is known that the most effective branch-and-bound methods for the JSSP are based on the disjunctive graph model (Brucker