Journal of Manufacturing Systems 31 (2012) 195–203

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Technical paper

A three-fold approach for job shop problems: A divide-and-integrate strategy with immune algorithm

Beizhi Li, Shanshan Wu*, Jianguo Yang, Yaqin Zhou, Min Du

Department of Mechanical Engineering, Donghua University, Shanghai, China

A R T I C L E I N F O

Article history:
Received 22 March 2010
Received in revised form 2 March 2011
Accepted 24 May 2011
Available online 16 July 2011

Keywords:
Shop scheduling
Priority rules
Immune algorithm
Dynamic scheduling

A B S T R A C T

This paper presents a novel divide-and-integrate strategy based approach for solving large scale job-shop scheduling problems. The proposed approach works in three phases. First, in contrast to traditional job-shop scheduling approaches where optimization algorithms are used directly regardless of problem size, priority rules are deployed to decrease problem scale. These priority rules are developed with slack due dates and mean processing time of jobs. Thereafter, immune algorithm is applied to solve each small individual scheduling module. In last phase, integration scheme is employed to amalgamate the small modules to get gross schedule with minimum makespan. This integration is carried out in dynamic fashion by continuously checking the preceding module's machine ideal time and feasible slots (satisfying all the constraint). In this way, the proposed approach will increase the machine utilization and decrease the makespan of gross schedule. Efficacy of the proposed approach has been tested with extremely hard standard test instances of job-shop scheduling problems. Implementation results clearly show effectiveness of the proposed approach.

1. Introduction

Scheduling refers to assigning activities (tasks or jobs) to resources (machines, labors) over time such that certain objective(s) is optimized, subject to both temporal and physical constraints [1]. These objectives can be minimization of makespan, minimization of maximum lateness and minimization of the number of tardy jobs etc. In production and operations management, scheduling can be of many types such as resource constrained project scheduling, job-shop scheduling, and flow-shop scheduling. This paper deals with the large scale job-shop scheduling problem (JSSP). In general, job shop scheduling refers to allocation of jobs to appropriate machines satisfying the precedence and resource constraints. In its general form, JSSP is NP-complete, meaning that there is probably no efficient procedure for exactly finding shortest schedules for arbitrary instances of the problem. Job shop scheduling plays a key role in attaining low lead time, flexibility and agility. In this way it plays vital role in the production management and thus has long been targeted by practitioners and researchers.

Since some efficient methods for $n \times 2$ and $2 \times m$ job shop scheduling problems were proposed by Jackson [2] and Akers [3], a great many methods have been developed in an attempt to solve this classical problem of diverse sizes due to the wide spread of automated manufacturing systems. These methods span from enumerative algorithms aiming at finding exact solutions to approximation algorithms which can be used to solve larger problems in speed. The approximation algorithms can be divided into two categories: opportunistic problem solver and local search heuristic [4]. Priority rules are the probably earliest and the most frequently applied heuristics for solving job shop scheduling problems in practice because of their ease of implementation and their low computational complexity. Green and Appel [5] presented empirical analysis of job shop selection rule, and Panwalkar and Iskander [6] conducted review of dispatching rules. Vepsalainen and Morton [7] developed due-dates and delay-penalty based efficient dispatching rules for the weighted tardiness problem. Holthaus and Rajendran [8] proposed dispatching rules with respect to the objectives of minimizing mean flow time, and maximum flow time. Kim and Kim [9] developed a scheduling mechanism in which the job dispatching rules vary for a flexible manufacturing system. These reviews suggest that job scheduling problems may take many forms and solution methodologies can be evaluated on various performance measures. However, with mere dispatching rules, solution quality degrades as problem size further increases. Therefore, JSSP is considered computationally complex problem and within its class it is one of the hardest problems to solve optimally [10–12]. Thus, the research focus of approximation algorithms is shifted to local randomized search inspired by natural phenomena or artificial intelligence [13–17]. In which a prior knowledge is required in order to escape a local optimum. In a search procedure, it is important to design and implement a good heuristic to control the search process or obtain an initial

* Corresponding author.
E-mail address: ss.wu.021@gmail.com (S. Wu).

0278-6125/$ – see front matter © 2011 The Society of Manufacturing Engineers. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.jmsy.2011.05.005

schedule by implementing a good heuristic or having one technique embedded within a meta-strategy. Zhou et al. [18] proposed a hybrid heuristic GA where scheduling rules are integrated into the process of genetic algorithm. Park et al. [19] proposed a hybrid solution approach for population initialization in job shop scheduling. Pezzella et al. [20] developed an algorithm to integrate different strategies for generating, selecting and reproducing individuals. Zhang et al. [21] proposed and applied a new enhanced neighborhood structure to solve the job shop scheduling problem. Bozejko and Makuchowski [14] described a hybrid tabu search algorithm dedicated to a job shop problem with a no-wait constraint and makespan criterion. Nevertheless, job shop scheduling problems still remain open and attractive to many researchers as the size of increases. Some efforts have been made for solving it. Singer [22] developed a rolling horizon heuristic for large job problems. The methods divide a given instance into a number of sub-problems, each having to correspond to a time window of the overall schedule. Hodgson et al. [23] showed a simulation-based procedure to satisfy date in large job shops. Nevertheless, it has been noticed that due to the combinatorial nature of job shop scheduling problem, with the increase in number of jobs and resources, search space increases exponentially and thus generation of consistently good schedule is very difficult [10], even random search methods with heuristics have extreme difficulty in effectively searching the solution space for instances of dimensionality greater than 15 [24].

By intensive scanning of literature, we found that researchers have paid little attention to realistic circumstances while solving the JSSP. A few of those circumstances are as follows:

1. The JSSP considered before mainly aims at scheduling one module at a time, while in real floor shop, there are many occasions when two or more modules are to be considered simultaneously. In this case, it is required to set a priority to these modules.
2. In the past, large-scale problem instances have not been considered sufficiently by researchers. They considered not more than few dozens of jobs in a module. While, in some industries scale of job shop scheduling problems could be hundreds or even more. For instance, in textile sectors, a factory in a developing country usually has thousands of machines and receives thousands of orders on a monthly basis. This requires huge computational time when finding good solutions.
3. Moreover, in the past approaches, machine's capacity is not utilized effectively and most of the time they remain idle. This is significant waste of resources in the terms of machine utilization.

If in a large-scale scheduling module all jobs are considered simultaneously then it would result in huge increase of computational burden, cognitive load and poor solution quality. In most cases, each machine left with a certain periods of idle time after solution for one scheduling module is obtained. Thus, in successive scheduling modules, these feasible idle times can be utilized to minimize the gross makespan of large scheduling module. Therefore, we attempt to take advantage of features of both dispatching rules and meta-heuristic algorithms for solving large scale job shop problem. Moreover, little work has been done on solving with meta-heuristic methods incorporating dispatching rules for large job shop scheduling.

In order to deal with the above mentioned shortcomings, this paper proposes a novel divide-and-integrate strategy based meta-heuristic method for job shop scheduling problem. Among aforementioned local search methods, one favorable attempt to solve the JSSP is immune algorithm. Hart et al. [25] depicted an artificial immune system approach to producing robust schedules for a dynamic job shop scheduling problem. Ong et al. [26] applied the clonal selection principle of the human immune system to solve the flexible job-shop problem with recirculation. Naderi et al. [27] presented a novel meta-heuristic method based on the artificial immune algorithm incorporating new features for an extended problem of job shop scheduling. Despite huge capability of exploration and exploitation of artificial immune system, it is not used much to resolve JSSP yet. Therefore, this paper presents an innovative way of using remarkable exploration and exploitation capabilities of AIS to resolve job shop scheduling problem.

The proposed divide-and-integrate strategy based immune algorithm (DISIA) is three-folded, and the major building blocks of this approach are:

1. Division of the original problem in small modules by deploying heuristic;
2. Use of intelligent meta-heuristic algorithm to get the optimal/near-optimal solution for each module;
3. Integration the solutions into one gross solution with the aid of integration scheme.

Heuristic rules are applied to divide the large problem instances into various small modules. This is done due to the notion that small instances can be solved more effectively and efficiently as compared to larger one. After dividing the large problem instance into relatively small modules, next is to optimize them with the aid of an effective meta-heuristic. In order to accomplish this task in real time we utilized the exploration and exploitation capabilities of immune algorithm. In the third and last stage, solutions of small individual models are integrated by deploying integration schema.

The remainder of this paper is divided into five sections. In Section 2, problem environment and its mathematical aspects are presented. In Section 3, the proposed algorithm DCSIA, including integration schema is described in detail. Numerical computations along with the results and discussions are presented in Section 4. Finally, Section 5 concludes the paper.

2. Problem description

The dynamic job shop scheduling problem considers K scheduling modules of jobs ($J_{11}, J_{12}, \ldots, J_{1N1}; J_{21}, J_{22}, \ldots, J_{2N2}; \ldots; J_{k1}, J_{k2}, \ldots, J_{kNk}$) to be processed on machines $M_1, M_2, \ldots, M_M$. Each scheduling module has different number of jobs. The kth scheduling module consists of jobs $J_{k1}, J_{k2}, \ldots, J_{kNk}$ (N_k detonates the total number of jobs in the kth scheduling module). Each job j has to be processed by a predefined sequence of operations ($j = 1, 2, \ldots, h_{ki}$), O_{kij} denotes the jth operation for job i in the kth scheduling module, h_{ki} denotes the number of operations of job i, p_{kij} denotes the processing time of operation O_{kij} on $M(O_{kij})$. The arrival time of materials is denoted by t_{Aki}.

For the sake of clarity of the modeling, assumptions made in this research are the following:

(1) Arrival times and release times of each job in every scheduling module are known.
(2) Setup times and machines change times between operations are negligible.
(3) There are no precedence constraints among the operations of different jobs.
(4) Machines are independent from each other.
(5) Jobs are independent from each other.
(6) At a given time, an operation can only be processed on one machine. Job preemptions are not allowed.
(7) At a given time, a machine can only execute one operation. It becomes available to other operations once processing of current operation is over.
(8) The successive job module does not start processing until the ongoing job module is completed.