Expert Systems with Applications 37 (2010) 282–287

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Multi-objective scheduling of dynamic job shop using variable neighborhood search

M.A. Adibi [a], M. Zandieh [b,*], M. Amiri [c]

[a] Faculty of Industrial and Mechanical Engineering, Qazvin Azad University, Qazvin, Iran
[b] Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C., Tehran, Iran
[c] Department of Industrial Management, Management and Accounting Faculty, Allameh Tabatabaei University, Tehran, Iran

ARTICLE INFO

Keywords:
Dynamic job shop
Multi-objective scheduling
Variable neighborhood search
Artificial neural networks

ABSTRACT

Dynamic job shop scheduling that considers random job arrivals and machine breakdowns is studied in this paper. Considering an event driven policy rescheduling, is triggered in response to dynamic events by variable neighborhood search (VNS). A trained artificial neural network (ANN) updates parameters of VNS at any rescheduling point. Also, a multi-objective performance measure is applied as objective function that consists of makespan and tardiness. The proposed method is compared with some common dispatching rules that have widely used in the literature for dynamic job shop scheduling problem. Results illustrate the high effectiveness and efficiency of the proposed method in a variety of shop floor conditions.

1. Introduction

The job shop scheduling (JSS) problem has attracted many optimization methods because it still exists in most of manufacturing systems in various forms. JSS problem is well-known to be NP-hard and various methods like mathematical techniques, dispatching rules, artificial intelligence, artificial neural networks, neighborhood searches, fuzzy logic, and etc. are introduced to obtain an optimum (or a near to optimum) solution. But these methods are usually designed to address static JSS problem and real time events such as random job arrivals and machine breakdowns are ignored. Tacking into account these events, JSS problem shifts to a new kind of problem that is well-known as dynamic job shop scheduling (DJSS) problem. In DJSS problem, one or more conditions of the problem like number of jobs or number of operable machines are changed by any new event. Therefore the solution before the event is not good or even feasible longer. So in addition to scheduling problem, it is needed to deal with dynamic events in DJSS problems.

In DJSS problem, due to changes in problem condition during planning horizon, using a scheduling method with parameters set at their optimum value as used in most researches (Chryssolouris & Subramanian, 2001; Dominic, Kaliyamoorthy, & Saravana Kumar, 2004; Qi, Burns, & Harrison, 2000; Zhou, Nee, & Lee, 2008) can reduce performance of the selected method. Preventing such this problem, Aydin and Oztemel (2000) used reinforcement learning agents to select proper rule for scheduling according to the shop floor conditions in real time. Shugang, Zhiming, and Xiaohong (2005) have used a heuristic obtained by training a neural network offline with the genetic algorithm. Sha and Liu (2005a, 2005b) presented a model that incorporates a data mining tool for mining the knowledge of job scheduling about due date assignment in a dynamic job shop environment to adjust an appropriate parameter according to the condition of the shop floor at the instant of job arrival. Zandieh and Adibi (2009) used artificial neural network (ANN) to estimate proper parameters of their scheduling method at any rescheduling point for minimizing mean flow time. ANN models have been studied since early 1980s with an objective of achieving human like performance in many fields of science and are intended for modeling the organizational principles of nervous system (Bose & Liang, 1996). In ANNs, a network of processing elements is designed and mathematics carry out information processing for problems whose solutions require knowledge that is difficult to describe. ANNs can be used to predict appropriate parameters of scheduling method at rescheduling point using a pattern extracted from learning sample. Compared with the traditional pattern recognition, ANN can provide an exact description for multidimensional and non-linear problems (Yating, Bin, Lei, & Wenbin, 2008).

In this study, to address multi-objective DJSS problem, variable neighborhood search (VNS) (Mladenovic & Hansen, 1997) is selected as a scheduling method at any rescheduling point because it brings together a lot of desirable properties for a metaheuristic such as simplicity, efficiency, effectiveness, generality, and etc. (Zandieh & Adibi, 2009; Hansen, Mladenovic, & Moreno Perez, 2007). VNS is a new neighborhood search metaheuristic that has widely used to combinatorial optimization problems in recent years. In this paper, to enhance the efficiency and effectiveness of

* Corresponding author.
E-mail address: m_zandieh@sbu.ac.ir (M. Zandieh).

0957-4174/$ - see front matter © 2009 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2009.05.001

VNS, its parameters are updated at any rescheduling point by ANN (Zandieh & Adibi, 2009). Multi-objective performance measure that contains both makespan and total tardiness is also applied in the scheduling process.

The rest of the paper is organized as follows: in Section 2, the multi-objective job shop scheduling problem is defined in details. Variable neighborhood search algorithm is argued in Section 3. Artificial neural network is presented in Section 4. The dynamic job shop scheduling problem is defined in Section 5. Simulation study is presented in Section 6 and conclusion is located in Section 7.

2. The multi-objective job shop scheduling problem

The ordinary job shop scheduling model considers n jobs to be process on m machines ($n \times m$ operations) while minimizing some function of completion time of jobs subject to following technological constraints and assumptions;

- Each machine can perform only one operation at a time on any job.
- An operation of a job can be performed by only one machine at a time.
- Once an operation has begun on a machine, it must not be interrupted.
- An operation of a job cannot be performed until its preceding operations are completed.
- There are no alternate routings, i.e. an operation of a job can be performed by only one type of machine.
- Operation processing time and number of operable machines are known in advance.

In this research, a multi-objective performance measure is applied as the objective function to construct schedules. The objective function consists of makespan and tardiness. Makespan is defined as the total time that is required to process a group of jobs. Tardiness is defined as the difference between the completion time and due date for each job in which the completion time occurs after the due date. It is observed that the variance of the makespan is much smaller than that of tardiness and noted that this can have a detrimental effect on the quality of solutions (Rangsaritratsamee, Ferrell, & Kurz, 2004). So Eq. (1) is used to be minimized in this research.

$$f = 5 \times Makespan + 2 \times Tardiness = 5[\underset{alli}{Max}(d_i) - \underset{alli}{Min}(s_i)] + 2\sum_{alli} \Psi_i(d_i - DD_i), \quad (1)$$

where d_i, departure time of job i; DD_i, due date time of job i; s_i, starting time of job i; $\Psi_i = 1$ if $(d_i - DD_i) > 0; = 0$ otherwise.

Operation-based representation and deduction of schedule (Amirthagadeswaran & Arunachalam, 2006) is used in this article. This representation method encodes a schedule as a sequence of operations. Each number stands for one operation. The specific operation represented by a number is interpreted according to the order of the numbers in the string. Each of the $n$jobs will appear $m$times spread over the entire string. For instance, we are given a state of [2 1 3 2 3 3 1 1 2], where $\{1,2,3\}$ represents $\{job_1, job_2, job_3\}$, respectively. Obviously, there are totally nine operations, but three different integers, each is repeated three times. The first integer, 2, represents the first operation of the second job, O_{21}, to be processed first on the corresponding machine. Likewise, the second integer, 1, represents the first operation of the first job, O_{11}. Thus, the set of [2 1 3 2 3 3 1 1 2] is understood as $[O_{21}, O_{11}, O_{31}, O_{22}, O_{32}, O_{33}, O_{12}, O_{13}, O_{23}]$, where O_{ij} stands for the jth operation of ith job.

3. Variable neighborhood search

Variable neighborhood search is a new local search technique that tries to escape from local optimum by changing the neighborhood structure (NS) that is the manner in which the neighborhood is defined. In its basic form, VNS explores a set of neighborhoods of the current solution, makes a local search from a neighbor solution to a local optimum, and moves to it only if there has been an improvement (Perez, Vega, & Martin, 2003).

The VNS algorithm that is used in this article begins with an initial solution, $x \in S$, where S is the set of search space and uses a two-nested loop. The inner one alters and explores using two main functions namely 'shake' and 'local search', respectively. The outer loop works as a refresher reiterating the inner loop. Local search explores for an improved solution within the local neighborhood, while shake diversifies the solution by switching to another local neighborhood. The inner loop iterates as long as it keeps improving the solutions, where an integer, k, controls the length of the loop. Once an inner loop is completed, the outer loop reiterates until the termination condition is met. The steps of VNS are illustrated in Fig. 1.

In Fig. 1, $N_k(k = 1, 2 \ldots, k_{max})$ denote neighborhood structures for both shake and local search functions. To avoid costing too much computational time, the best number of neighborhood structure is often two (Liao & Cheng, 2007) which is followed by our algorithm for each function (shake and local search). The two neighborhoods employed in our algorithm are *Insertion* and *Swap*. *Insertion* randomly identifies two particular operations and places one operation in the position that directly precedes the other operation. *Swap* randomly identifies two particular operations and places each operation in the position previously occupied by the other operation.

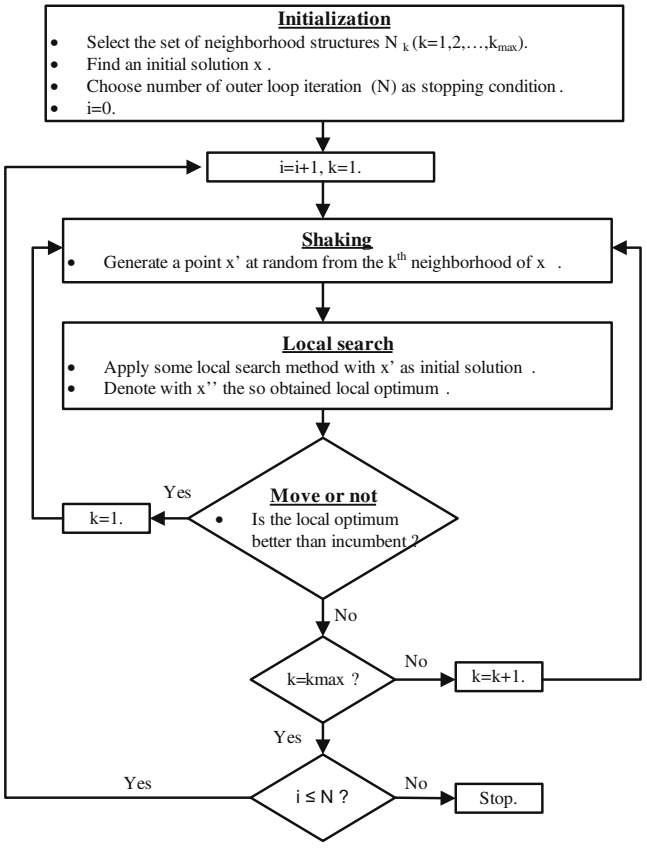

Fig. 1. VNS algorithm.