Computers & Operations Research 40 (2013) 521–525

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

Some heuristics for no-wait flowshops with total tardiness criterion

Gengcheng Liu, Shiji Song*, Cheng Wu

Department of Automation, Tsinghua University, Beijing, 100084, PR China

ARTICLE INFO

Available online 7 August 2012

Keywords:
No-wait flowshop
Heuristic
Total tardiness

ABSTRACT

No-wait flowshop scheduling problem is widely investigated because of its practical application and specific properties. However, the total tardiness criterion has not been much considered. In this paper, we propose six heuristic approaches for no-wait flowshops with total tardiness criterion, among which the modified NEH algorithm (MNEH) is verified to be the best. Also, a speed-up technique is introduced to MNEH to reduce the computational time in certain cases. By numeral experiments and analysis, we evaluate the performances of various heuristics. Finally we find out that MNEH is a satisfactory algorithm dealing with this problem.

1. Introduction

The no-wait flowshop scheduling problem with total tardiness criterion is considered in this paper. In no-wait flowshops, each job has to be processed from the first machine to the last without any interruption, and the job sequence is unique on all machines. No-wait flowshop model adapts to a wide range of industrial applications, such as steel production, chemical industry, food processing, etc. A comprehensive survey on the research and application of no-wait flowshop scheduling problem can be found in Hall and Sriskandarayah's review paper [1].

Literature proved that no-wait flowshops with more than two machines are NP-hard [1]. For no-wait flowshops, both heuristic and meta-heuristic approaches have been investigated to find high quality solutions. Dispatching rules such as SPT and EDD, which are the simplest heuristics, can be applied to no-wait flowshop scheduling problems [2]. Laha and Chakraborty proposed a constructive heuristic for minimizing makespan in no-wait flowshop scheduling [3]. Framinan et al. designed an efficient heuristic for total flowtime minimization in no-wait flowshops [4]. The NEH heuristic algorithm was first proposed by Nawaz, Enscore and Ham in solving the permutation flowshop scheduling problems for minimizing makespan [5]. Literatures have conveyed that NEH algorithm performs best for permutation flowshops among the heuristic approaches [6]. The main ideology of NEH can also be applied in no-wait flowshop problems. Various meta-heuristic algorithms dealing with no-wait flowshops, e.g. tabu search, simulated annealing and differential evolution, are discussed in [7–10]. In addition, there exists a property in no-wait flowshops [11]: the total processing time of any two adjacent jobs is a fixed value no matter what the whole schedule is, thus it can be calculated beforehand. This technique is applied to the algorithms above to reduce the computational time.

There have been a lot of research works on no-wait flowshop scheduling problems with various criterions, e.g. makespan [3], total flow time [4], and maximum lateness [7]. Moreover, combinations of the criterions have been considered. Allahverdi and Aldowaisan focused on no-wait flowshops with bi-criteria of makespan and total flow time [8]. Allahverdi and Aldowaisan and Pan et al. considered the bi-criteria of makespan and maximum lateness [9,10]. However, until now, there are few works considering the criterion of total tardiness for no-wait flowshops, while actually this criterion is quite meaningful in just-in-time production research. Ref. [12] is the only paper we could found involving total tardiness in a bi-criteria problem, and the authors proposed a scatter search approach to solve this problem.

As for other types of scheduling problems, studies with total tardiness criterion are not rare, such as single-machine problems, flowshops and flowshops with blocking. Congram et al. proposed an iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem [13]. Bülbül et al. developed heuristics for the problem of scheduling customer orders in a flowshop with the objective of minimizing the sum of tardiness, earliness and intermediate inventory holding costs [14]. Framinan and Leisten proposed a simple approach based on a variable greedy algorithm to minimize total tardiness in permutation flowshops [15]. Ronconi and Henriques investigated some heuristic algorithms for total tardiness minimization in a flowshop with blocking [16]. Dhingra and Chandna considered a bi-criteria (total weighted squared tardiness and makespan) m-machine SDST flowshop scheduling using modified heuristic genetic algorithm [17]. From these literatures, we can see that heuristics are the main-stream approaches for solving scheduling problems with total tardiness criterion.

* Corresponding author.
E-mail address: shijis@mail.tsinghua.edu.cn (S. Song).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.07.019

In this paper, we will investigate the performances of several simple heuristics in no-wait flowshops with total tardiness criterion. And based on the classic NEH algorithm, we will propose a novel heuristic, "modified NEH algorithm" (MNEH for short), to solve the problem. The remaining of the paper is organized as follows: Section 2 gives the problem description, Section 3 lists several heuristic algorithms and ends up with a modified NEH algorithm (MNEH), numeral experiments and the result analysis are in Section 4, and Section 5 gives the final conclusions.

2. Problem description

In an n*m no-wait flowshop, we have a set of jobs $J=\{J_1,J_2,\ldots,J_n\}$ and a set of machines $M=\{M_1,M_2\ldots,M_m\}$. Each job has a due date and different processing times on different machines. The authors of [11] found that a no-wait flowshop is equivalent to a Traveling Salesman Problem, thus we can define a so-called "distance" between each two jobs, and this distance will not vary as the schedule of jobs changes.

Once a schedule π is determined, we can calculate each job's completion time, and with regard to its due date, the job's tardiness will be calculated. By summing up all the tardiness values, we get the total tardiness, which is the optimization index considered in our work. Following is a list of definitions and calculating expressions used in the remaining of the paper:

p_{ij}: processing time of job j on machine i.
d_j: due date of job j.
D_{jk}: "distance" between job j and job k.

$$D_{jk}=\max_{1\le i\le m}\left(\sum_{h=1}^{i}p_{hj}-\sum_{h=1}^{i-1}p_{hk}\right) \quad (1)$$

P_j: total processing time of job j on all machines.

$$P_j=\sum_{i=1}^{m}p_{ij} \quad (2)$$

$C_{[j]}(\pi)$: completion time of the jth job in the sequence π.

$$C_{[j]}(\pi)=\sum_{l=2}^{j}D_{[l-1,l]}(\pi)+P_{[j]}(\pi) \quad (3)$$

$T_{[j]}(\pi)$: tardiness of the jth job in the sequence π.

$$T_{[j]}(\pi)=\max\{C_{[j]}(\pi)-d_{[j]}(\pi),0\} \quad (4)$$

TT: total tardiness.

$$TT=\sum_{j=1}^{n}T_{[j]}(\pi)=\sum_{j=1}^{n}\max\{C_{[j]}(\pi)-d_{[j]}(\pi),0\} \quad (5)$$

From the Gantt chart below, we can recognize those definitions quite intuitively. For a 3*3 no-wait flowshop (as Fig. 1 shows), D_{12} is the interval between the first job's starting time and the second job's starting time, the completion time of the third job in the sequence is calculated as $C_3=D_{12}+D_{23}+P_3$, and its tardiness is calculated as $T_3=\max\{C_3-d_3,0\}$.

In this paper and other previous literatures as well, the due dates are generated from a uniform distribution: $d_j\sim U[P(1-T-R/2), P(1-T+R/2)]$, where P is a lower bound of the maximum completion time and is calculated as $P=LB(C_{\max})=(\min_{1\le j\le n}\sum_{h=1}^{m-1}p_{hj})+\sum_{j=1}^{n}p_{mj}$, and T,$R\in[0,1]$ are two parameters. A lower value of T generates looser due dates, while high value generates very tight due dates. A lower value of R makes jobs' due dates closer together and high value makes due dates more scattered. Since P is a relatively loose bound, the due dates are not easy to be satisfied, which indicates that in most cases, a job's tardiness will be a positive value.

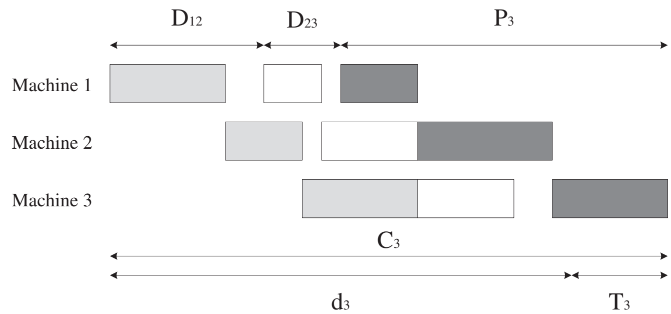

Fig. 1. Gantt chart of a 3*3 no-wait flowshop.

3. Heuristic algorithms

The main difficulty of no-wait flowshop scheduling problem with total tardiness criterion lies in the calculation of optimization index. Meta-heuristic approaches are often too time-consuming to deal with this problem, while heuristic algorithms may find satisfying solution in acceptable computation time.

3.1. Dispatching rules

Dispatching rules are the simplest heuristic approaches for scheduling problems. They sort the jobs according to a certain rule involving a quite simple index of each job, such as its processing time or due date. Below are two feasible dispatching rules for no-wait flowshops:

SPT (Smallest Processing Time): The job sequence is obtained by sorting the jobs in the increasing order of their total processing times P_j.
EDD (Earliest Due Date): The job sequence is obtained by sorting the jobs in the increasing order of their due dates d_j.

The computation complexity of SPT and EDD rules is the same as a sorting algorithm. If the quickest sorting strategy is adopted here, the time cost of dispatching rules is $O(n\log n)$.

Now we will give discussions about the two rules in some extreme cases. First, if we let $R\downarrow 0$, thus the due dates of all jobs tend to be almost the same, then the EDD rule has little difference with a random schedule. Second, if we select a large value of T ($T\approx 1$), which indicates that the due dates are too tight to be satisfied, then $TT=\sum_{j=1}^{n}\max\{C_{[j]}(\pi)-d_{[j]}(\pi),0\}\approx\sum_{j=1}^{n}C_{[j]}(\pi)-\sum_{j=1}^{n}d_{[j]}(\pi)$, and total tardiness criterion is approximately equivalent to total flow time criterion. In this case, the SPT rule is likely to have better performance, due to its optimality for $1//\sum C_j$ problems [2].

3.2. Simple constructive heuristics

Unlike dispatching rules, constructive heuristics generate the whole job sequence progressively. In simple constructive heuristics, jobs are picked out one by one and then appended to the end of the partial sequence. Three constructive heuristics are proposed as follows.