Computers & Operations Research 36 (2009) 2682 – 2689

Contents lists available at ScienceDirect

Computers & Operations Research

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

Heuristics for a two-stage assembly flowshop with bicriteria of maximum lateness and makespan

Fawaz S. Al-Anzi[a], Ali Allahverdi[b,*]

[a]Department of Computer Engineering, Kuwait University, P.O. Box 5969, Safat, Kuwait
[b]Department of Industrial and Management Systems Engineering, Kuwait University, P.O. Box 5969, Safat, Kuwait

ARTICLE INFO

Available online 9 December 2008

Keywords:
Assembly flowshop
Bicriteria
Makespan
Maximum lateness
Heuristic

ABSTRACT

We consider a two-stage assembly flowshop scheduling problem with the objective of minimizing a weighted sum of makespan and maximum lateness. The problem is known to be NP-hard, and therefore, we propose heuristics to solve the problem. The proposed heuristics are Tabu search (Tabu), particle swarm optimization (PSO), and self-adaptive differential evolution (SDE). An extensive computational experiment is conducted to compare performances of the proposed heuristics. The computational experiment reveals that both PSO and SDE are much superior to Tabu. Moreover, it is statistically shown that PSO performs better than SDE. The computation times of both PSO and SDE are close to each other and they are less than 40 and 45 s, respectively, for the largest size problem considered.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

The two-stage assembly flowshop scheduling problem has received attention recently due to its applicability in real life problems. For example, Potts et al. [1] described an application in personal computer manufacturing where central processing units, hard disks, monitors, keyboards, and etc. are manufactured at the first stage, and all the required components are assembled to customer specification at a packaging station (the second stage). Lee et al. [2] described another application in a fire engine assembly plant. The body and chassis of fire engines are produced in parallel in two different departments. When the body and chassis are completed and the engine has been delivered (purchased from outside), they are fed to an assembly line where the fire engine is assembled. Yet another application is in the area of queries scheduling on distributed database systems, Allahverdi and Al-Anzi [3]. In short, many real life problems can be modeled as a two-stage assembly flowshop scheduling problem.

A two-stage assembly flowshop scheduling problem is defined as following. There are m machines at the first stage while there is only one machine at the second stage. There are n jobs available for scheduling such that each job has m+1 operations. The first m operations of a job are performed at the first stage in parallel by m machines and the final operation is conducted at the second stage. Each of the m operations of a job at the first stage is performed by a different machine and the last operation on the machine at the second stage may start only after all m operations at the first stage are completed. Each machine can process only one job at a time. It should be noted that when there is only one machine at the first stage (i.e., $m = 1$), then the problem reduces to the two-machine flowshop scheduling problem.

Different performance measures are considered in the scheduling research. These performance measures may be classified as completion time related or due date related. Makespan (C_{max}), a completion time related performance measure, is one of the most widely used performance measures. Minimizing makespan is important in situations where a simultaneously received batch of jobs is required to be completed as soon as possible. For example, a multi-item order submitted by a single customer needs to be delivered as soon as possible. The makespan criterion also increases the utilization of resources. On the other hand, minimizing maximum lateness (L_{max}) is a widely used due date related measure. This objective is particularly important in situations where there is a penalty to complete a job beyond its due date and the penalty increases with the gap between the two.

The two-stage assembly flowshop problem to minimize C_{max} is addressed by several papers including Lee et al. [2], Potts et al. [1], Hariri and Potts [4], Sun et al. [5], and Allahverdi and Al-Anzi [3]. The problem is also addressed with respect to L_{max} criterion, Allahverdi and Al-Anzi [6] and Al-Anzi and Allahverdi [7].

The research mentioned so far addressed only a single criterion of either C_{max} or L_{max} while the majority of real life problems requires the decision maker to consider both C_{max} and L_{max} before

* Corresponding author. Tel.: +965 2498 7874; fax: +965 2481 6137.
E-mail addresses: alanzif@eng.kuniv.edu.kw (F.S. Al-Anzi), allahverdi@kuniv.edu.kw (A. Allahverdi).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.11.018

arriving at a decision. The two-stage assembly scheduling problem with both C_{max} and L_{max} has not been addressed and is the topic of the current paper. The only related research that we are aware of is the work of Allahverdi and Al-Anzi [8] who addressed the problem with the objective of minimizing a weighted sum of C_{max} and mean completion time.

In this paper, we address the two-stage assembly flowshop scheduling problem with the objective of minimizing a function which is a weighted sum of C_{max} and L_{max}. The problem is described in the next section, and three heuristics are proposed in Section 3. The heuristics are evaluated through randomly generated data and the results are analyzed in Section 4. The summary of the work and possible future research directions are presented in Section 5.

2. Problem description

There is a set of n jobs simultaneously available for processing. It is assumed that preemption is not allowed, i.e., any started operation has to be completed without interruptions. There are m+1 operations related to each job where the first m operations have to be performed on stage one (each of the m operations by one of the m machines at stage one) while the last operation is performed at stage two on the assembly machine. Let

$t_{i,j}$	operation time of job i on machine j (at stage one), $i=1,\ldots,n, j=1,\ldots,m$,
p_i	operation time of job i on assembly machine (at stage two),
C_i	completion time of job i,
d_i	due date of job i,
L_i	lateness of job i

Note that job k is complete once all $t_{k,j}$ $(j=1,\ldots,m)$ on the first stage and p_k on the second stage are completed. It should be also noted that the p_k on the second stage may start only after all $t_{k,j}$ $(j=1,\ldots,m)$ on the fist stage have been completed.

Potts et al. [1] and Allahverdi and Al-Anzi [6] showed that permutation schedules are dominant with respect to C_{max} and L_m, respectively. Therefore, permutation schedules are also dominant for the problem addressed in this paper. Thus, we restrict our search for the optimal solution to permutation schedules. In other words, the sequence of jobs on all of the machines, including the assembly machine, is the same.

Let the bracket denote the job in a given position. For example, $p_{[j]}$ denotes the processing time of the job in position j on the second (assembly) machine. It can be shown that the completion time of the job in position j is as follows:

$$C_{[j]} = \max\left\{\max_{k=1,\ldots,m}\left\{\sum_{i=1}^{j} t_{[i,k]}\right\}, C_{[j-1]}\right\} + p_{[j]} \quad \text{where } C_{[0]}=0$$

Hence, the makespan (C_{max}) and maximum lateness (L_{max}) are defined as

$$C_{max} = C_{[n]}$$

$$L_i = C_i - d_i$$

and hence,

$$L_{max} = \max(L_1, L_2, \ldots, L_n)$$

If the weight given to C_{max} is denoted by α (and that of L_{max} by $1-\alpha$), then the value of the objective function (VOF) can be computed as

$$VOF = \alpha C_{max} + (1-\alpha)L_{max}$$

where $0<\alpha<1$. Notice that when $\alpha>0.5$, more weight is given to the criterion of C_{max} while more weight is given to the criterion of L_{max} when $\alpha<0.5$. The objective is to find a schedule which minimizes the value of VOF.

The described problem has no polynomial solution since it is known that the problem for $\alpha=1$ (i.e., when the objective is to minimize C_{max}) is NP-hard in the strong sense even for $m=2$ (see Lee et al. [2]). Therefore, there are two approaches that can be used to solve the problem. One approach is to develop an implicit enumeration technique such as branch-and-bound algorithm to solve for relatively small problems while the other approach is to develop heuristics. In this paper, we opted to develop the latter approach. The next section describes the developed heuristics.

3. Heuristics

In this section, we propose three heuristics for the problem. The first proposed heuristic is a Tabu search (Tabu) heuristic, the second is a particle swarm optimization (PSO) heuristic, and the third is an self-adaptive differential evolution (SDE) heuristic. All the three heuristics start with some given initial sequences and iteratively improve until a stopping criterion is met.

One of the initial sequences used in all the three heuristics is the earliest due date (EDD) sequence since it is known that the EDD sequence performs well for L_{max} criterion. Another initial sequence that performs well for the other performance measures, i.e., C_{max}, is desirable. It is known that if there is only one machine at the first stage (i.e., $m=1$), then the two-stage assembly flowshop scheduling problem is reduced to the regular two-machine flowshop scheduling problem for which the well known Johnson algorithm [9] is optimal with respect C_{max} performance measure. For the problem that we address $m>1$, we consider m artificial problems where each problem consists of a single machine on the first stage and the assembly machine at the second stage. Let Johnson-i denote the sequence obtained by applying the well known Johnson algorithm to the artificial problem consisting of the ith $(i=1,\ldots,m)$ machine on the first stage and the assembly machine on the second stage. Therefore, we will have m sequences, i.e., Johnson-1, Johnson-2, …, Johnson-m. Let JNS denote the sequence from among Johnson-i which has the minimum VOF. Another words, VOF(JNS) = min(VOF(Johnson-1), VOF(Johnson-2),…,VOF(Johnson-m)) where VOF(Johnson-i) denotes the value of the objective function of the sequence Johnson-i.

3.1. Tabu

The Tabu heuristic randomly chooses the best sequence from the neighborhood of a current sequence by making changes to the current sequence. A new change is allowed to the current sequence if this change is not in the previous h changes, i.e., Tabu list. A list of length h of position pairs, i.e., a pair of the form (i, j) where the jobs in positions i and j were exchanged, are kept in a list for checking. The list h is called the *Tabu list*. The changes are repeated iteratively until a stopping criterion is met.

For example, consider scheduling of five jobs and assume that at some point of time we have three sequences of $\boldsymbol{S}_1=[3, 5, 2, 4, 1]$, $\boldsymbol{S}_2=[5, 4, 1, 2, 3]$, and $\boldsymbol{S}_3=[5, 2, 1, 4, 3]$. In this example, it is easy to see that sequences $\boldsymbol{S}_2$ and $\boldsymbol{S}_3$ are closer to each other than sequences $\boldsymbol{S}_1$ and $\boldsymbol{S}_3$ because $\boldsymbol{S}_2$ can be obtained from $\boldsymbol{S}_3$ by exchanging jobs 2 and 4 in the sequences while to obtain $\boldsymbol{S}_3$ from $\boldsymbol{S}_1$ one needs to reorder 4 jobs. In this context, we define the distance between two sequences as the number of mismatches between the two sequences. In the above example, the distance between $\boldsymbol{S}_2$ and $\boldsymbol{S}_3$ is 2 while that of $\boldsymbol{S}_1$ and $\boldsymbol{S}_3$ is 4. According to this definition, the minimum distance than can be achieved is 2 for

Download English Version:

https://daneshyari.com/en/article/475334

Download Persian Version:

https://daneshyari.com/article/475334

Daneshyari.com