Applied Soft Computing 13 (2013) 1391–1399

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop

T.C. Wong*, S.C. Ngan[1]

Department of Systems Engineering and Engineering Management, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong, China

ARTICLE INFO

Article history:
Received 30 December 2011
Received in revised form 30 March 2012
Accepted 3 April 2012
Available online 19 April 2012

Keywords:
Assembly job shop
Lot streaming
Genetic algorithm, Particle swarm optimization
Part sharing ratio
System congestion index
Makespan

ABSTRACT

Very often, studies of job shop scheduling problem (JSSP) ignore assembly relationship and lot splitting. If an assembly stage is appended to JSSP for the final product, the problem then becomes assembly job shop scheduling problem (AJSSP). To allow lot splitting, lot streaming (LS) technique is examined in which jobs may be split into a number of smaller sub-jobs for parallel processing on different stages such that the system performance may be improved. In this study, the system objective is defined as the makespan minimization. In order to investigate the impact of LS on the system objective under different real-life operating conditions, part sharing ratio (PSR) and system congestion index (SCI) are considered. PSR is used to differentiate product-specific components from general-purpose, common components, and SCI for creating different starting conditions of the shop floor. Both PSR and CSI are useful as part sharing (also known as component commonality) is a common practice for manufacturing with assembly operations and system loading is a significant factor in influencing the shop floor performance. Since the complexity of AJSSP is NP-hard, a hybrid genetic algorithm (HGA) and a hybrid particle swarm optimization (HPSO) are proposed and developed to solve AJSSP in consideration of LS technique. Computational results show that for all test problems under various system conditions, HGA can significantly outperform HPSO. Also, equal-sized lot splitting is found to be the most beneficial LS strategy especially for medium-to-large problem size.

1. Introduction

Given a number of machines and jobs, each operation of a job must be processed on one of the machines in a predetermined and fixed sequence in a traditional job shop environment. A job is deemed as "completed" once all of its operations are successfully processed by the corresponding machines. Usually, there is no mutual relationship between jobs in the same shop floor. Also, a job which usually contains a batch of identical items cannot be split into smaller pieces. Therefore, a batch must be wholly moved among machines even some items of the batch have been already processed. In reality, these two restrictions are sometimes invalid. The first restriction assumes that each job is independent and the second assumes that each job cannot be split. To relax the first restriction, an assembly stage is attached to the job shop such that there is assembly relationship between jobs after completion at JSSP stage. Hence, the bill-of-material (BOM) of all products must be constructed to define the root components (jobs) of a product. With the BOM, the assembly relationship between different jobs can be created. If there is no part sharing, only jobs from the same BOM can be assembled. In contrast, part sharing allows job assembly from distinct BOMs. To allow job splitting, lot streaming (LS) technique is considered to remove the second restriction. With LS, it is a must to make three LS decisions for each job: (1) whether the job will be split (binary: yes or no), (2) the sub-job number, and (3) the size of each sub-job. In order to simulate a realistic shop floor, a 4-level part sharing ratio (PSR) and a 4-level system congestion index (SCI) are applied. PSR is applied to define the degree of component commonality among different products, i.e. the higher the PSR, the higher the probability that a component is shared by more than one product. SCI is examined to simulate different initial operating condition of a system, i.e. the higher the SCI, the higher the system congestion will be. By considering different levels of PSR and CSI, the impact of LS on makespan minimization can be better examined. As the problem complexity is NP-hard, a hybrid genetic algorithm (HGA) and a hybrid particle swarm optimization (HPSO) are proposed and developed to minimize the makespan of JSSP with assembly stage (AJSSP). Computational experiments are performed to examine the goodness of HGA and HPSO with respect to the objective function, and to obtain the best LS strategy under various system conditions.

* Corresponding author. Tel.: +852 3442 6155, fax: +852 3442 0172.
E-mail addresses: andywtc@graduate.hku.hk (T.C. Wong), scngan@cityu.edu.hk (S.C. Ngan).
[1] Tel.: +852 3442 8400; fax: +852 3442 0172.

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.04.007

This paper is organized as follows. The literature review is given in Section 2. In Section 3, the problem background and formulations are presented. The two hybrid algorithms are depicted in Section 4. In Section 5, computational results are reported and discussed. Conclusions are made together with future research direction in Section 6.

2. Literature review

In AJSSP, a job usually contains a batch of identical items or components. Thus, it is sometimes called a lot. In the presence of an assembly stage, all completed jobs from the BOM of the same product need to be assembled to form the final products that will be eventually delivered to customers. Under normal circumstances, the product assembly can start immediately after all jobs of the same BOM are completed at the JSSP stage. To solve AJSSP, decision makers not only need to determine the processing sequences on machines, but also the splitting conditions (i.e. three previously mentioned LS decisions) of all jobs. In general, the complexity of AJSSP must be at least the same as that of JSSP, i.e. NP-hard. Assembly can be seen as the process to construct a final product from its root components. One of the basic rules is to ensure that only root components from the same BOM can be assembled if there is no part sharing. The sophistication of a product mainly depends on its component number and the assembly levels.

There are mainly two types of AJSSP studies in the literature. The first type only allows machining to the components but not the assemblies (e.g. [1–3,42]). The second type allows machining to the assemblies (e.g. [4–12]). In other words, assemblies are required to be processed before assembly operation. It is noted that the research problem of our present study focuses on the first type only. Among the first type, Chan et al. [1] have proposed a hybrid GA with simple dispatching rules using LS technique. In their study, GA is employed to determine LS conditions whereas dispatching rules are used to solve AJSSP after LS conditions have been determined. The computational results suggest that minimum slack time heuristic with an equal size LS strategy is the best approach for solving small-to-medium sized AJSSP. On the other hand, although LS is thoroughly studied, the application of simple dispatching rules to shop floor scheduling could provide sub-optimum solutions. Therefore, Chan et al. [42] have proposed an improved version of this hybrid GA to solve AJSSP using GAs to determine both the LS conditions and the processing sequences in AJSSP after LS conditions have been determined. The results suggest that GA outperforms simple dispatching rules in terms of optimization power. On the other hand, Kim and Kim [2] have compared two evolutionary algorithms for minimizing the earliness penalty of sub-jobs and the tardiness penalty of the final products. For simplicity, LS is neglected. Guide et al. [3] have discussed the application of simple dispatching rules in repair shop with no spares. In their paper, the repair shop contains a disassembly area, a repair area and an assembly area. Since each repair component has its unique and fixed operation sequence, the repair shop actually can be regarded as AJSSP. They have reported that simple dispatching rules may outperform complex optimization methods under medium-to-high uncertainty levels. However, LS again is ignored. Since there is insufficient study dedicated to AJSSP of the first type, the objective of this research is to fill the gap. We note that if necessary, the current problem can be easily revised into AJSSP of the second type.

Reiter [13] introduced the LS technique for splitting a job into a number of smaller sub-jobs such that its successive operations can be overlapped on different machines or stages. As a result, the shop floor performance would be improved with smaller in-progress inventory and makespan. But the nature of job size and the type of sub-job need to be defined prior to the application of LS. In general, the job size can be discrete or continuous. The sub-job type can be consistent and variable. Consistent sub-job type restricts that the sub-job size is fixed between successive machines or stages while variable sub-job type means that the sub-job size may vary. According to Trietsch and Baker [14], there are 4 types of LS models: (I) *equal size* sub-jobs *without* intermittent idling, meaning that jobs are split into sub-jobs with even size, and are processed on the same machine continuously, (II) *equal size* sub-jobs *with* intermittent idling, allowing idle time between sub-jobs on the same machine, (III) *varied size* sub-jobs *without* intermittent idling, meaning that jobs are split into sub-jobs of uneven size, and that there is no idle time between sub-jobs on the same machine, and (IV) *varied size* sub-jobs *with* intermittent idling. In this paper, we integrate types II and IV models into a single model, such that the size of each sub-job is a decision variable and idle time is allowed between different jobs and sub-jobs on the same machine. For example, batches of solid metal plates of various sizes are being processed on a CNC drilling machine. Setup time must be allowed for fixture changeover between successive metal plates of different sizes while the machine is being kept idle. Also, setup is assumed to be non-anticipatory [15], i.e. fixture changeover can be performed only if both machine and job are available. Moreover, the total number of sub-jobs and the sub-job size are fixed throughout the entire scheduling period once LS conditions have been determined. More details about LS can be found in the study of Chang and Chiu [16]. Recently, some LS models have been extended to parallel machine scheduling problem (PMSP) in which jobs only need to be processed on one machine given that all machines are functionally identical. The readers can refer to Kim et al. [17], Süer et al. [18], Tahar et al. [19], and Yalaoui and Chu [20] for several related studies about applying LS to PMSP. However, there has been insufficient development of the LS model for scheduling problems with assembly stage. As compared to GA, particle swarm optimization (PSO) has been just applied to solve some classical production scheduling problems (e.g. [21–25]). Besides the classical problems, PSO has been also proposed to solve some hybrid problems (e.g. [26–31]). In general, the results are promising. In addition to optimizing the traditional system performance such as makespan, earliness, tardiness, flow time, etc., some LS models have been developed to examine the impact of LS on the system cost (e.g. [32,33]). However, the cost setting would be case-sensitive and cannot be generalized across different scheduling problems. Thus, this study intends to compare the capability of HGA and HPSO in minimizing the makespan under AJSSP environment with LS technique. By doing so, the results may provide useful insight on the potential application of LS to other assembly-related problems.

3. Problem background and formulations

3.1. Notations

p	Total number of products
m	Total number of machines
n	Total number of lots
n^*	Total number of sub-lots
DM_h	Demand of product h
CN_h	Total number of components of product h
At_h	Assembly time of product h
Ct_h	Completion time of product h
Q_j	Batch size of lot j
Q_{js}	Batch size of sth sub-lot of lot j
MS_{jk}	Machine for kth operation of lot j
Pt_{jk}	Processing time of kth operation of lot j
SU_{jk}	Setup time of kth operation of lot j
St_{jk}	Start time of kth operation of L_{hj}
St_{jsk}	Start time of kth operation of sth sub-lot of lot j
Ct_{jk}	Completion time of kth operation of lot j