Computers & Industrial Engineering 57 (2009) 983–995

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A resource-constrained assembly job shop scheduling problem with Lot Streaming technique

T.C. Wong [a,*], Felix T.S. Chan [c], L.Y. Chan [b]

[a] Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong
[b] Department of Industrial and Manufacturing Systems Engineering, University of Hong Kong, Pokfulam Road, Hong Kong
[c] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

ARTICLE INFO

Article history:
Received 26 May 2008
Received in revised form 15 January 2009
Accepted 4 April 2009
Available online 10 April 2009

Keywords:
Assembly job shop
Genetic algorithm
Lot Streaming
Particle swarm optimization
Resource constraint

ABSTRACT

To ensure effective shop floor production, it is vital to consider the capital investment. Among most of the operational costs, resource must be one of the critical cost components. Since each operation consumes resources, the determination of resource level is surely a strategic decision. For the first time, the application of Lot Streaming (LS) technique is extended to a Resource-Constrained Assembly Job Shop Scheduling Problem (RC_AJSSP). In general, AJSSP first starts with Job Shop Scheduling Problem (JSSP) and then appends an assembly stage for final product assembly. The primary objective of the model is the minimization of total lateness cost of all final products. To enhance the model usefulness, two more experimental factors are introduced as common part ratio and workload index. Hence, an innovative approach with Genetic Algorithm (GA) is proposed. To examine its goodness, Particle Swarm Optimization (PSO) is the benchmarked method. Computational results suggest that GA can outperform PSO in terms of optimization power and computational effort for all test problems.

1. Introduction

In classical Job Shop Scheduling Problem (JSSP), there are $(1, \ldots, n)$ jobs must be processed on $(1, \ldots, m)$ machines in predefined operation sequences. Decision makers then need to determine the processing sequence of jobs on all machines such that each job can visit all machine once with all system constraints satisfied. A job is regarded as "completed" if all of its operations are finished and it implies that each job is independent. Usually, a job is a batch of identical items in which the whole batch cannot be split. It means that the batch must be wholly transferred from machines to machines even some items have been already processed. The first restriction assumes that each job is independent and the second assumes that each job cannot be split. In reality, these two restrictions are not always valid. To relax the first restriction, an assembly stage is attached such that JSSP becomes Assembly Job Shop Scheduling Problem (AJSSP). In this connection, each job is an entity of the Bill-Of-Material (BOM) of all products. The assembly relationship between different jobs is defined by the BOMs. If there is no common part, only jobs from the same BOM can be assembled. In contrast, only common parts from distinct BOMs may be assembled. Lot Streaming (LS) technique depicts the process of splitting jobs into smaller sub-jobs so that successive operations of the same job can be overlapped on different stages. In consideration of resource constraints, AJSSP then becomes a Resource-Constrained AJSSP (RC_AJSSP). For the first time, the application of LS is extended to RC_AJSSP. Applying LS, three decisions must be made for each job: (1) Whether the job is split, (2) The sub-job number and (3) The size of each sub-job. To model a realistic shop floor, 2-level resource constraint, 4-level common part ratio and 4-level workload index are investigated. In this study, a Genetic Algorithm (GA) based approach is proposed. To examine the goodness of our proposed approach, another evolutionary algorithm is considered as the benchmarked method. Recently, some papers have studied the comparison between GA and Particle Swarm Optimization (PSO) on scheduling-related problems (e.g. Gaafar, Masoud, & Nassef, 2008; Guo, Li, Mileham, & Owen, 2009; Sha & Hsu, 2008). However, the results do not indicate an obvious distinction between GA and PSO. According to our knowledge, PSO has not yet been fully applied to scheduling problems especially with assembly. In this connection, this study attempts to fill the gap and computational experiments are performed to explicitly compare GA and PSO in terms of optimization power and computational effort.

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 proposed and benchmarked approaches 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.

* Corresponding author.
E-mail addresses: andywtc@graduate.hku.hk (T.C. Wong), mffchan@inet.polyu.edu.hk (F.T.S. Chan), plychan@hku.hk (L.Y. Chan).

0360-8352/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2009.04.002

2. Literature review

As an extension of JSSP, AJSSP first starts with JSSP and then appends an assembly stage after job completion. A job or lot is a batch of identical items or components. Since there is an assembly stage, all completed jobs from the BOM of the same product should be assembled. 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 job processing sequences on machines, but also the sequences of assembly operations among jobs. In general, the complexity of AJSSP must be at least the same as that of JSSP. Assembly is usually defined as the process to construct a final product from its root components. One basic rule is that components can be assembled only if they belong to the same BOM. The complexity of a product mainly depends on the number of its root components and the assembly levels.

Generally speaking, two types of AJSSP can be addressed. The first type only allows machining to the root components but not the assemblies (e.g. Chan, Wong, & Chan, 2008a; Guide, Srivastava, & Kraus, 2000; Kim & Kim, 1996). The second type allows machining both to the root components and the assemblies (e.g. Gravel, Price, & Gagne, 2000; Guo, Wong, Leung, Fan, & Chan, 2006; Mckoy & Egbelu, 1998; Mckoy & Egbelu, 1999; Mohanasundaram, Natarajan, Viswanathkumar, Radhakrishnan, & Rajendran, 2002; Pathumnakul & Egbelu, 2006; Pongcharoen, Hicks, & Braiden, 2004; Pongcharoen, Hicks, Braiden, & Stewardson, 2002; Thiagarajan & Rajendran, 2003). In other words, assemblies are also required to be processed on machines before final product assembly. It is noted that the current research problem is dedicated to the first type only. Among the first type, Chan et al. (2008a) proposed a hybrid GA with simple dispatching rules using LS technique. In their study, GA was employed to determine LS conditions while dispatching rules were used to solve AJSSP after LS conditions had been determined. The computational results suggested that minimum slack time heuristic with equal size LS strategy was the best recommended approach for solving small-medium sized AJSSP. Although LS was clearly studied, the application of simple dispatching rules to shop floor scheduling might provide sub-optimum solutions. Kim and Kim (1996) compared two evolutionary algorithms for minimizing the earliness penalty of sub-jobs and the tardiness penalty of the final products. For simplicity, LS was neglected. Guide et al. (2000) 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 reported that simple dispatching rules might outperform complex optimization methods under medium-high uncertainty levels. However, LS was also ignored. Since there is insufficient study dedicated to the first type, this research intends to fill the gap. If necessary, the current problem can be easily modified to fit the scope of the second type. Nevertheless, differences between the two types are not very significant.

Firstly introduced by Reiter (1966), LS technique is a methodology of splitting a job into a number of smaller sub-jobs such that its successive operations can be overlapped on different stages. Prior to the application of LS, it is a must to define the nature of job size and the type of sub-job. In general, the job size can be discrete or continuous. Discrete job size means a job contains an integer number of identical items. Continuous job size can be any real number. The sub-job type can be consistent and variable. Consistent sub-job type restricts that the sub-job size is fixed between successive stages. Variable sub-job type means that the sub-job size may vary. According to Trietsch and Baker (1993), there are 4 types of LS models. In this paper, only the second LS model is adopted, i.e. consistent sub-jobs with intermittent idling such that idle time is allowed between sub-jobs on stages. Moreover, the total number of sub-jobs and the sub-job size are fixed once LS is applied. For a more comprehensive review on LS, please refer to Chang and Chiu (2005). 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. Several related studies about LS to PMSP can be referred to Kim, Shim, Kim, Choi, and Yoon (2004), Süer, Pico, and Santiago (1997), Tahar, Yalaoui, Chu, and Amodeo (2006) and Yalaoui and Chu (2006). However, there is no LS model developed for scheduling problems with assembly stage. Hence, the primary research goal of this paper is to fill the gap. By doing so, the impact of LS on AJSSP can be investigated such that the results may provide useful insights 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
$DM_h(t)$	demand of product h at time t. Value = max. when t is 0
D_h	due date of product h
CN_h	total number of components of product h
At_h	assembly time of product h
$Ct_h(t)$	completion (delivery) time of product h at different time t
CF_i	current fixture type loaded on machine i
Q_j	batch size of lot j
Q_{js}	batch size of sth sub-lot of lot j
F_j	fixture type assigned to lot j
UF_j	total unit of fixture type of lot j
T_{jk}	tool type assigned to kth operation of lot j
UT_{jk}	total unit of tool type of kth operation of lot j
MS_{jk}	machine for kth operation of lot j
Pt_{jk}	processing time of kth operation of lot j
α_{jsk}	value = 1 if setup is required for kth operation of sth sub-lot of lot j. Otherwise, value = 0.
SU_{jk}	setup time of kth operation of lot j
SU_i	total setup time on machine i due to fixture changeover
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
Ct_{jsk}	completion time of kth operation of sth sub-lot of lot j
SN_j	sub-lot number of lot j
mc_i	machining cost per hour of machine i
lc_h	late cost per unit per hour of product h
ic_j	inventory cost per unit per hour of lot j
NF	number of operation delays due to fixture shortage
NT	number of operation delays due to tool shortage
$WIP_i(t)$	work-in-process inventory of machine i at time t
TPT_i	total processing time required on machine i
BOM_{hj}	value = 1 if lot j belongs to the bill-of-material of product h. otherwise, value = 0.
RL	resource level, high or low
GEN	maximum number of generation
PS	population size
CR	crossover rate
MR	mutation rate
Present	the current position of a particle
PBest	the best position reached so far by a particle
GBest	the best position reached so far by all particles