Computers & Operations Research 67 (2016) 207–215

Contents lists available at ScienceDirect

Computers & Operations Research

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

Flexible open shop scheduling problem to minimize makespan

Danyu Bai [a,b,*], Zhi-Hai Zhang [b], Qiang Zhang [c]

[a] School of Economics & Management, Shenyang University of Chemical Technology, Shenyang 110142, PR China
[b] Department of Industrial Engineering, Tsinghua University, Beijing 100084, PR China
[c] Software College, Northeastern University, Shenyang 110819, PR China

ARTICLE INFO

Available online 11 November 2015

Keywords:
Scheduling
Flexible open shop
Makespan
Asymptotic analysis
Differential evolution algorithm

ABSTRACT

This study investigates the static and dynamic versions of the flexible open shop scheduling problem with the goal of minimizing makespan. The asymptotic optimality of the general dense scheduling (GDS) algorithm is proven by the boundedness hypothesis. For large-scale problems, the GDS-based heuristic algorithms are presented to accelerate convergence. For moderate-scale problems, the differential evolution algorithm is employed to obtain high-quality solutions. A series of random experiments are conducted to demonstrate the effectiveness of the proposed algorithms.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

The open shop scheduling model originated in large automotive garages for vehicle repairs [11]. This model is also applied in inspection industry and health-care system. For example, medical diagnosis of coronary heart disease (CHD) generally requires three examinations, namely, blood test, ultrasonic cardiogram and coronary computed tomography (CT) scan. A CHD patient may undergo these procedures in any order. However, any two procedures cannot be performed simultaneously because blood drawing, echocardiography and CT rooms belong to different departments in the outpatient service of a hospital. Several nurses or facilities are assigned to each of these rooms in a parallel fashion to accelerate the diagnostic testing. Such diagnostic model is called a generalized open shop or a flexible open shop (FOS).

Technically, FOS is an extension of the classical open shop (COS) and parallel-machine models, in which n jobs must be executed once at each of the $c \geq 2$ stages (or machine centers) without interruption (the notation mentioned in [21] that describes the flexible shop scheduling is followed). A stage consists of a number of parallel machines, and at least one of these stages includes more than one machine. A job has to be processed at each stage by using only one of the machines. The sequence of each stage that processes jobs and the route of each job passing through the stages can be chosen arbitrarily. The objective is to find a schedule that simultaneously determines machine processing orders and job visiting routes to optimize some criteria, such as makespan or total completion time.

For the COS makespan problem, Bai and Tang [3] proved the asymptotic optimality of the dense scheduling (DS) algorithm in the sense of probability limit. This study generalizes the DS algorithm, *i.e.*, the general dense scheduling (GDS) algorithm, and the result of the asymptotic optimality for the FOS makespan problem. In contrast to the classical method for performance evaluation (*i.e.*, referring to worst-case analysis that estimates the maximum relative error between the solution generated by an approximation algorithm and the optimal solution for any possible problem instance, even those that are unlikely to appear in the real world), the asymptotic analysis characterizes the convergence effect of the algorithm when the problem size is sufficiently large [26]. Especially, an asymptotically optimal algorithm can be executed as an optimal schedule as the number of jobs approaches to infinity, which more conforms to an industrial production environment where thousands of jobs are usually processed on one or more machines. With the guarantee of asymptotic optimality, for large-scale problems, the GDS-based heuristics are designed to quickly solve the static and dynamic versions.

Given that intelligent optimizer is more efficient to obtain high-quality solution within a specified time, for moderate-scale problems, a differential evolution (DE) algorithm, therefore, is introduced to solve the problem. The DE algorithm is a population-based global evolutionary method that uses simple mutation and crossover operators to generate new candidate solutions in which a one-to-one competition scheme is applied to greedily select new candidates [27]. The advantages of this algorithm include simple structure, easy implementation, quick convergence, and robustness, all of which make it be one of the most powerful optimizers. As each chromosome in the traditional version is built for a continuous search space, a job-permutation-based representation is provided such that the algorithm can be executed in a discrete

[*] Corresponding author. Tel.: +86 10 62772847; fax: +86 10 62794399.
E-mail address: mikebdy@163.com (D. Bai).

http://dx.doi.org/10.1016/j.cor.2015.10.012
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

domain. To the best of the authors' knowledge, no study has yet utilized the DE algorithm to solve the FOS scheduling problem. At the end of the article, numerical simulations are conducted to demonstrate the convergence of the GDS-based heuristics and the performance of the DE algorithm.

The rest of the article is organized as follows. Section 2 presents a brief survey on the FOS problem and the DE algorithm. Section 3 provides the formulation of the problem. Sections 4 and 5 analyze the asymptotic optimality of the GDS algorithm and describe the GDS-based heuristics, respectively. Section 6 describes the concrete procedures of the DE algorithm. Sections 7 and 8 present the computational results and the conclusions, respectively. Several results of the parallel-machine scheduling problem with release dates are given in the Appendix.

2. Literature survey

The standard three-field representation [10] is employed in the following discussion to formally describe the scheduling problems.

2.1. Flexible open shop scheduling

Unlike the fruitful results for flexible flow shop scheduling problems [23], few studies have been reported on the FOS scheduling problems. These works mainly discussed polynomial solvable cases, worst-case analysis on approximation algorithms, and application of intelligent optimization (except the DE algorithm), which are summarized below.

Lawer et al. [13] showed that the optimal schedule of problem $FO_c|prmp|C_{max}$ (*prmp* denotes the preemptive assumption) can be found in polynomial time, where the parallel machines are unrelated at each stage. de Werra et al. [31] developed strong polynomial time algorithms for the $FO_2|prmp|C_{max}$ problem, in which each job requires operating on either a single machine or simultaneously on all machines at the same stage. Matta and Elmaghraby [15] modeled a $FO_c|prop|C_{max}$ problem, where *prop* denotes the proportionate assumption, and provided polynomial time algorithms for two special cases.

Chen and Strusevich [6] pointed out the NP-hardness of problem $FO_2||C_{max}$ even for the simplest case: one stage has a single machine, whereas the other has two, which indicates that minimizing the makespan in the FOS (static and dynamic) problem is unsolvable in polynomial time. For the *c*-stage case, the authors proved that the heuristic schedule designed by combining the dense schedule and an algorithm *H* for parallel machines is at most $1+\rho_H$ times that of the optimal makespan, where ρ_H is the worst-case ratio of algorithm *H*. For the two-stage case, they established a worst-case ratio that is strictly less than 2 for the GDS- longest processing time (GDS-LPT) heuristic (*i.e.*, combining the DS algorithm and the LPT rule). Schuurmana and Woeginger [24] demonstrated that the GDS algorithm is a 2-approximation algorithm for problem $FO_c||C_{max}$, and proposed an improved approximation algorithm with a worst-case ratio that arbitrarily approaches to 3/2 for the two-stage case. Sevastianov and Woeginger [25] obtained a polynomial time approximation scheme (PTAS) for problem $FO_c||C_{max}$, where the total number of machines is bounded by a constant.

Matta [14] proposed a genetic algorithm for problem $FO_c|prop|C_{max}$, where the chromosome representation of a schedule enables any permutation of its genes to yield a feasible solution. A tabu search approach is presented for problem $FO_c|prop|C_{max}$, where the neighborhood search function is defined over a network representation of feasible solutions [1]. Naderi et al. [16] formulated the $FO_c||\Sigma C_j$ problem as an effective mixed integer linear programming model and employed memetic algorithms to solve it.

2.2. The DE algorithm in scheduling problems

Given that each chromosome is a vector of floating point numbers, which is invalid for mapping a discrete search space, the canonical DE algorithm cannot be directly applied to solve scheduling problems with an inherently discrete nature. Generally, two kinds of approaches are employed, *i.e.*, convert permutations into real numbers or represent individuals as job permutations, such that the algorithm can be executed in a discrete domain. The applications of this algorithm on scheduling problems remain considerably limited, mainly concentrated on single machine, flow shop and job shop problems, which are summarized below.

For the single machine scheduling model, Nearchou [17] adopted a V-shaped representation of the solution space within the DE algorithm to minimize the total cost of earliness and tardiness. Tasgetiren et al. [28] enhanced the performance of the DE algorithm by including a local search with an insertion neighborhood to solve the $1|s_{j,\ h}|\Sigma w_j T_j$ problem, where the setup times are sequence dependent. For the parallel-machine scheduling model, Wang et al. [30] proposed a hybrid DE algorithm by containing block mutation and crossover within the global search procedure to minimize makespan with splitting jobs.

For the flow shop scheduling model, Onwubolu and Davendra [18] presented an effective DE algorithm in which the forward and backward transformation schemes are utilized to handle discrete variables and optimize the criteria: makespan, flowtime and tardiness. Pan et al. [20] combined the DE algorithm and local search with an iterative insertion neighborhood to minimize the objectives: makespan and flowtime. Deng and Gu [7] integrated a so-called perturbed local search in the DE algorithm to promote the solution of the $F_m|no\text{-}idle|C_{max}$ problem. Pan et al. [19] embedded a multi-objective local search in the DE algorithm to stress the balance between global and local exploitations for the $F_m|no\text{-}wait|C_{max}/T_{\max}$ problem. Qian et al. [22] proposed a hybrid DE algorithm in which the largest- order- value rule is presented to convert the continuous values of individuals into job permutations and optimize the multi-objective flow shop scheduling problem with limited buffers. Wang et al. [29] developed the job-permutation-based mutation and crossover operators for the DE algorithm to solve the $F_m|block|C_{max}$ problem. Al-Anzi and Allahverdi [2] introduced a self-adaptive DE algorithm for a two-stage assembly flow shop scheduling problem to minimize maximum lateness with setup times.

For the job shop scheduling model, Zhang et al. [33] devised a special local search that generates the neighborhood by systematically perturbing the operation processing times for enhancing the convergence performance of the DE algorithm to minimize the expected total tardiness. Hu et al. [12] modified the DE algorithm to address the job shop scheduling problem with fuzzy processing time and due date. Yuan and Xu [32] embedded a local search based on the critical path in the DE algorithm to enhance the solution of the flexible job shop makespan problem.

3. Preliminaries

The FOS is a machine environment with *c* stages, where stage *k*, $k=1, 2,\ldots,c$, includes $m_k \geq 1$ identical machines in parallel. A job set $N=\{1, 2,\ldots,n\}$ is required for processing at these stages. The order in which job *j*, $j \in N$, passes through the *c* stages is arbitrary. The processing of job *j* at stage *k* is denoted as operation $O(k, j)$. Each operation $O(k, j)$ has a processing time $p(k, j)$, where all the processing times are bounded by a constant $P_{\max}$. Job *j* must be processed at each stage on any one of the machines. Let i_k denote the *i*th machine at stage *k*, $i=1, 2,\ldots,m_k$. The sum of machines in the system satisfies $\sum_{k=1}^{c} m_k = m$, where *m* is a constant, $1 < m \ll n$.