Computers & Operations Research 39 (2012) 2152–2160

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion

Guanlong Deng, Xingsheng Gu *

Key Laboratory of Advanced Control and Optimization for Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China

A R T I C L E I N F O

Available online 7 November 2011

Keywords:
No-idle flow shop
Scheduling
Differential evolution
Speed-up
Insert neighborhood
Local search

A B S T R A C T

This paper presents a hybrid discrete differential evolution (HDDE) algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion, which is not so well studied. The no-idle condition requires that each machine must process jobs without any interruption from the start of processing the first job to the completion of processing the last job. A novel speed-up method based on network representation is proposed to evaluate the whole insert neighborhood of a job permutation and employed in HDDE, and moreover, an insert neighborhood local search is modified effectively in HDDE to balance global exploration and local exploitation. Experimental results and a thorough statistical analysis show that HDDE is superior to the existing state-of-the-art algorithms by a significant margin.

1. Introduction

The flow shop scheduling problem (FSP) involves n jobs processed on m machines. Each job has to be sequentially processed on the first machine, the second machine and so on until the last machine. The flow shop problem is then described as finding processing sequences of the n jobs on the m machines so that a given performance criterion is optimized. If the sequences of processing jobs are the same for all machines, then the problem becomes permutation flow shop scheduling problem (PFSP), which is usually considered in the flow shop research field. No-idle constraint requires there is no idle time between the processing of any operations on each machine. If the no-idle constraint is added, the no-idle PFSP is defined. The no-idle PFSP with makespan criterion, denoted by $Fm/no\text{-}idle, prmu/C_{max}$, is to determine a job sequence that minimizes the makespan, namely, the completion time of the last job on the last machine.

It should be noted that the no-idle constraint makes the problem significantly different, which is validated in an example by Ruiz et al. [1], though it is known that the $F2/no\text{-}idle, prmu/C_{max}$ is equivalent to the two-machine Johnson problem $F2/prmu/C_{max}$ [2], which can be solved in $O(n \log n)$ time by Johnson's algorithm [3]. Besides, the $Fm/no\text{-}idle, prmu/C_{max}$ should be distinguished from the $Fm/no\text{-}idle/C_{max}$, though the $F3/no\text{-}idle, prmu/C_{max}$ is equivalent to the $F3/no\text{-}idle/C_{max}$ [4] and some researchers reduced FSP to PFSP without stressing the permutation constraint.

The no-idle PFSP is a complex combinatorial optimization problem with strong industrial background. The no-idle constraint is imposed when cost of an idle machine is high or idle time is not allowed once a machine is started; examples are in the fire glass processing [5] and the foundry production [6]. Therefore, research on the no-idle PFSP is of great significance in both theory and application.

Adiri and Pohoryles [2] were the first to address the no-idle PFSP to minimize the sum of job completion time, i.e., $Fm/no\text{-}idle, prmu/\sum C_j$. Vachajitpan [7] was the first to consider the makespan objective in his paper, where the mixed integer programming (MIP) and corresponding branch and bound (B&B) method solved some problems with small sizes. Woollam [8] firstly considered several heuristics, including the NEH heuristic [9], to solve no-idle PFSP. Concerning computational complexity, the $F3/no\text{-}idle, prmu/C_{max}$ has been proved to be NP-hard by Baptiste and Lee [10]. Before that, Garey et al. [11] proved the NP-hardness of the $F3//C_{max}$. In Baptiste and Lee's paper, they also proposed a B&B method for the $Fm/no\text{-}idle, prmu/C_{max}$. Saadani et al. [6] solved the $F3/no\text{-}idle, prmu/C_{max}$ with a heuristic. Saadani et al. [12] also proposed a traveling salesman problem (TSP)-based approach (referred to as SGM) to solve the $Fm/no\text{-}idle, prmu/C_{max}$. Kamburowski [13] proposed a network representation of the makespan that provided a better insight into the $F3/no\text{-}idle, prmu/C_{max}$, and he pointed out an anomaly that prolonging some processing times might result in the possible reduction in the makespan. Kalczynski and Kamburowski [5] proposed a heuristic named KK for the $Fm/no\text{-}idle, prmu/C_{max}$ with alleged computational complexity of $O(mn^2)$. The heuristic was shown to outperform the heuristic SGM as well as NEH heuristic adapted for no-idle problem.

* Corresponding author. Tel.: +862164253463; fax: +862164253121.
E-mail addresses: dglag@163.com (G. Deng), xsgu@ecust.edu.cn (X. Gu).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.10.024

The authors also showed that NEH performed better than SGM and indicated that B&B method of Baptiste and Lee [10] is only applicable for very small instances. In addition, Kalczynski and Kamburowski [14] developed the network representation of makespan in the *Fm*/*no-idle*, *prmu*/C_{max}, and gave some efficiently solvable special cases and some lower bounds on makespan. As of late, Baraz and Mosheiov [15] presented an $O(mn^3)$ greedy algorithm (referred to as GH_BM in this paper as in [1]) for the *Fm*/*no-idle*, *prmu*/C_{max}. GH_BM was shown to perform better than SGM heuristic, but the authors bypassed the KK heuristic. In two similar papers, Pan and Wang [16,17] proposed a discrete particle swarm algorithm (HDPSO) and a discrete differential evolution (DDE$_{LS}$) for the same problem, respectively. In these two papers, a speed-up method was proposed to evaluate the whole insert neighborhood of a job permutation with $(n-1)^2$ neighbors. This reduced the computational complexity of evaluating the whole insert neighborhood from $O(mn^3)$ to $O(mn^2)$. Both algorithms were shown to be better than GH_BM, KK, and NEH, the result also indicated that KK was better than NEH and that NEH was better than GH_BM. A comprehensive survey and research were done by Ruiz et al. [1]. Their results indicated that the iterated greedy method (IG$_{LS}$) by Ruiz and Stutzle [18] outperformed existing methods (including HDPSO and DDE$_{LS}$). Goncharov and Sevastyanov [4] proposed several polynomial time heuristics based on a geometrical approach for the *Fm*/*no-idle*/C_{max}; however, neither computational experiments nor comparisons were provided.

The differential evolution (DE) algorithm, proposed by Storn and Price [19], is a simple yet powerful population-based stochastic search technique for real parameter optimization. It has successfully been applied to diverse domain of science and engineering. In DE, new candidates are generated by mutation and crossover operators and a one-to-one competition scheme greedily deciding whether the new candidates will survive in the next generation. As for scheduling problems, DE has been applied in permutation flow shop scheduling [20], no-wait permutation flow shop scheduling [21], etc. However, there has to be conversion from real domain to discrete domain because of the discrete characteristics of scheduling. In their two papers, Tasgetiren et al. [22] proposed a discrete differential evolution (DDE) algorithm for scheduling problems, and Pan and Wang [17] presented a DDE with local search (aforementioned DDE$_{LS}$) in no-idle permutation flow shop scheduling.

This paper focuses on presenting an effective hybrid discrete differential evolution (HDDE) algorithm for the *Fm*/*no-idle*, *prmu*/C_{max}. In the proposed HDDE, individuals are to be represented as job permutations, and new candidates are to be generated in a simple but effective way. Besides, a novel speed-up method distinct from accelerations in [16,17] will be proposed to evaluate the whole insert neighborhood of a job permutation with $(n-1)^2$ neighbors, which is inspired by the network representation of makespan [14]. Based on the speed-up method, a modified effective insert neighborhood local search will be imbedded in HDDE to balance global exploration and local exploitation. To validate the effectiveness and efficiency of the proposed algorithm, an extensive computational campaign will be provided.

The remainder of the paper is organized as follows: Section 2 provides the formulation of the *Fm*/*no-idle, prmu*/C_{max}. In Section 3, a novel speed-up method for the insert neighborhood is proposed. Section 4 elaborates the proposed HDDE. Section 5 gives a comprehensive computational results and comparisons. Finally, we summarize the contribution and draw some conclusions of this paper in Section 6.

2. Formulation of the *Fm*/*no-idle, prmu*/C_{max}

The no-idle PFSP with makespan criterion can be described as follows: there are a set of n jobs $N=\{1, 2,\ldots,n\}$ and a set of m machines $M=\{M_1, M_2,\ldots,M_m\}$. Each job is available at time zero and has to be processed first on machine M_1, next on machine M_2 and so on until on machine M_m. The time required for job j ($j=1, 2,\ldots,n$) on machine M_i ($i=1, 2,\ldots,m$) is given as p_{ij}. Each machine can process at most one job at a time and each job can be processed only on one machine at a time. The processing of a given job at a machine cannot be interrupted once started. The sequence in which the jobs are to be processed is the same for each machine. To satisfy the no-idle constraint, idle time between consecutive operations on each machine is not allowed, which means each machine must process all jobs without interruption from the start of the first job to the completion of the last job. In this paper, the makespan criterion is considered, so the problem is then to find a schedule minimizing the makespan.

Let the job permutation $\pi=(\pi(1),\pi(2),\ldots,\pi(n))$ represent the schedule of jobs to be processed, and $C_{max}(\pi)$ be the makespan of π. To calculate $C_{max}(\pi)$, we adopt some findings of paper [5,14], which will be applied to inspire the speed-up method later.

Let $C_{max}(\pi;A,B)$ be the makespan of π in the permutation flow shop with two machines A and B in series and let a_j and b_j be the processing times of job j on A and B, respectively. $C_{max}(\pi;A,B)$ is the length of critical path in the network shown in Fig. 1 and can

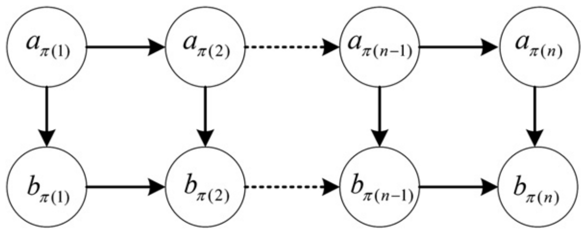

Fig. 1. Network for computing the makespan $C_{max}(\pi;A,B)$.

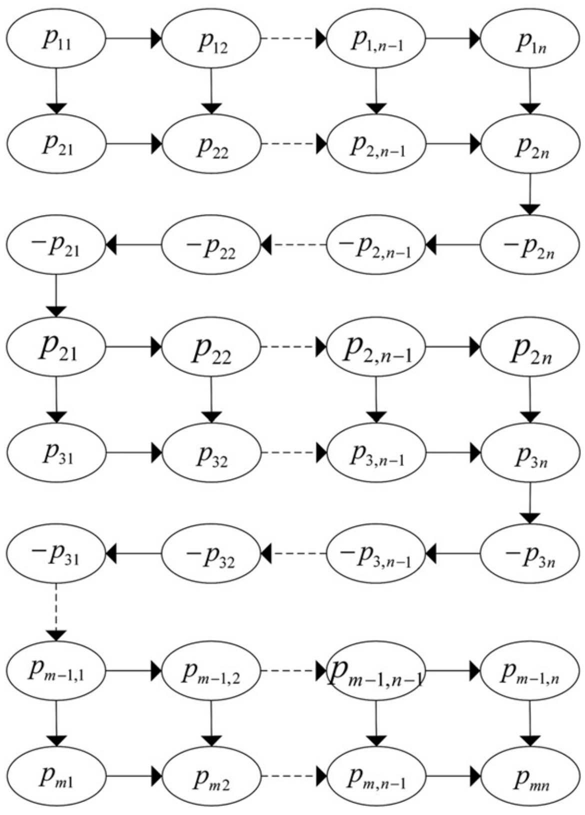

Fig. 2. Network for computing the makespan $C_{max}(\pi)$.