Operations Research Perspectives 2 (2015) 13–23

Contents lists available at ScienceDirect

Operations Research Perspectives

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

Trajectory Scheduling Methods for minimizing total tardiness in a flowshop

Xiaoping Li [a,b,*], Long Chen [a,b], Haiyan Xu [a,b], Jatinder N.D. Gupta [c]

[a] School of Computer Science and Engineering, Southeast University, Nanjing 211189, China

[b] Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, 211189, China

[c] College of Business Administration, University of Alabama in Huntsville, Huntsville, Al, USA

ARTICLE INFO

Article history:
Received 13 May 2014
Received in revised form
29 December 2014
Accepted 29 December 2014
Available online 13 January 2015

Keywords:
Scheduling
Heuristic
Permutation flow shop
Total tardiness

ABSTRACT

In this paper, Trajectory Scheduling Methods (TSMs) are proposed for the permutation flowshop scheduling problem with total tardiness minimization criterion. TSMs belong to an iterative local search framework, in which local search is performed on an initial solution, a perturbation operator is deployed to improve diversification, and a restart point mechanism is used to select the new start point of another cycle. In terms of the insertion and swap neighborhood structures, six composite heuristics are introduced, which exploit the search space with a strong intensification effect. Based on purely insertion-based or swap-based perturbation structures, three compound perturbation structures are developed that construct a candidate restart point set rather than just a single restart point. The distance between the current best solution and each start point of the set is defined, according to which the diversification effect of TSMs can be boosted by choosing the most appropriate restart point for the next iteration. A total of 18 trajectory scheduling methods are constructed by different combinations of composite heuristics. Both the best and worst combinations are compared with three best existing sequential meta-heuristics for the considered problem on 540 benchmark instances. Experimental results show that the proposed heuristics significantly outperform the three best existing algorithms within the same computation time.

1. Introduction

The permutation flow shop scheduling problem (PFSP) is important and prevalent in modern manufacturing systems (for example, the flexible manufacturing environment) and in traditional industry settings (such as chemical, food, and metal processing). Not completing a job by its due date would lead to: (1) incurring tardiness costs which depend on the penalty clauses in the contract if there are any; (2) loss of goodwill which results in an increased probability of losing the customer for some or all future jobs; and (3) a damaged reputation which would turn other customers away [1]. Therefore, minimizing total tardiness which is closely related to the due dates agreed by all partners is of great importance in manufacturing systems. In this paper, the PFSP to minimize total tardiness considered, which is known to be NP-hard in the strong sense [2] and can be denoted as $F|prmu|\sum T_j$ [3].

For decades, many exact methods, heuristics, and meta-heuristics have been proposed for the considered problem [4]. Exact methods are effective only for small size problems. Branch & bound procedures [5–9] are exact methods for few jobs (usually less than 20 jobs) being scheduled on two machines. However, exact methods are seldom efficient in practical environments because usually there are more than 20 jobs to be scheduled on more than three machines. Therefore, heuristics and meta-heuristics have been investigated. Generally, there are two types of heuristics: constructive heuristics and composite ones. The constructive heuristic NEH_{EDD} [10] has been the most widely used, which was adapted from NEH [11] using the EDD (Earliest Due Date) rule to produce the seed. NEH_{EDD} is always adopted by composite heuristics or meta-heuristics to generate initial solutions. For example, NEH_{EDD} is utilized to generate initial solutions of the typical composite heuristics [12]. Meta-heuristics are always adopted for combinatorial optimization problems, which provide high level strategies for exploring search spaces using different methods [13]. They generally obtain better solutions than simple constructive heuristics but require significantly more computation time. Meta-heuristics can be classified into population-based and trajectory (or single point) methods [13].

Genetic algorithms (GAs) are the most common population-based methods for the $F|prmu|\sum T_j$ problem. The GA developed in [14] generates initial individuals randomly and outperforms

* Corresponding author at: School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. Tel.: +86 25 52090916; fax: +86 25 52090916.
E-mail address: xpli@seu.edu.cn (X. Li).

http://dx.doi.org/10.1016/j.orp.2014.12.001

the DE (Differential Evolutionary) algorithm [15] proposed later. GAPR, GAPR2 and GADV [16] are three GA methods presented recently, which seem to be the best existing sequential algorithms for the considered problem. These three GAs use the EDD rule or both the EDD dispatching rule and the NEH_{EDD} heuristic to generate one or two initial individual(s) while the other initial individuals are generated randomly. Furthermore, based on these three GAs, three cooperative genetic algorithms (CGAPR, CGAPR2, and CGADV) were investigated in [17]. These investigations were performed on 4, 8, and 12 parallel computers and found to be relatively more effective than the GAPR, GAPR2, and GADV, which used only one computer.

The popular Trajectory Scheduling Methods (TSMs) start with an initial solution and improve it by a suitable strategy. Tabu Search (TS) and Simulated Annealing (SA) are commonly adopted complex strategies in TSMs. The two TS algorithms proposed in [18] and [10] utilize the heuristic developed in [19] and the EDD rule, respectively, to generate initial solutions. The four TS and four SA algorithms presented in [12] produce initial solutions using NEH_{EDD} and apply several local search methods for further improvement. The TS constructed in [20] adopts the Modified Due Dates rule to generate the initial solution. The two SAs introduced in [21] use the Earliest Apportioned Due Date rule as the initial heuristic. The SA algorithm developed in [22] generates the seed by a constructive heuristic and improves the current solution by several local search methods. A complex strategy was introduced in [23], which uses Ow's algorithm [19] to generate the initial solution and integrates SA with TS to obtain a high quality solution. Besides SA and TS, Iterated Local Search (ILS) is an effective trajectory meta-heuristic for combinatorial optimization. Though ILS has been applied to job shop scheduling problems [24], the PFSP with makespan minimization [25], and the PFSP with total flow time minimization [26], it has not yet been used to solve the $F|prmu|\sum T_j$ problem according to the extensive and comprehensive review on heuristics and metaheuristics for the m-machine flowshop problem with total tardiness minimization by E. Vallada, R. Ruiz and G. Minella [4].

In this paper, Trajectory Scheduling Methods (TSMs) are proposed for PFSP with total tardiness minimization. There are three components in each of the TSMs: Composite Heuristic, Adaptive Perturbation, and Restart Point Selection. For the considered problem, six composite heuristics and three compound perturbation methods are developed and compared. NEH_{EDD}, the most widely used rule for initial solutions, is adopted to generate the start point. An adaptive perturbation operator is presented to produce a set of candidate restart points. By defining the distance between a pair of solutions, a restart point selection criterion is introduced to select the most promising restart point of the next iteration from the candidate set.

The rest of the paper is organized as follows. Section 2 gives the description of the $F|prmu|\sum T_j$ problem. Section 3 discusses the proposed Trajectory Scheduling Methods. Empirical evaluation and comparison results of the proposed heuristics with existing algorithms are shown in Section 4. Finally, Section 5 concludes the paper with a summary of our findings and some fruitful directions for future research.

2. Problem description

To define the $F|prmu|\sum T_j$ problem, consider the following scenario: a set of n jobs are processed on m machines where each job requires m operations processed on m machines $M_1, \ldots, M_m$ sequentially with the same order. Each operation has a predetermined processing time and each machine can process one operation exclusively at a time. Preemption of jobs is not allowed.

Let $\mathbb{J} = \{J_1, \ldots, J_n\}$ be the job set and $\pi(n)$ be a schedule of the n jobs, i.e., a permutation of the n jobs, denoted as $(\pi_{[1]}, \ldots, \pi_{[n]})$. $\pi_{[k]} \in \mathbb{J}$ is the kth ($k = 1, \ldots, n$) job in $\pi(n)$. For convenience, a dummy job $\pi_{[0]}$ is added to the beginning of $\pi(n)$ with zero processing time and zero due date, i.e., the sequence can also be represented as $\pi(n) = (\pi_{[0]}, \pi_{[1]}, \ldots, \pi_{[n]})$. All the permutations of the n jobs are denoted as Ω, i.e., $\Omega = \{\pi(n)\}$. Let $C_{i,\pi_{[k]}}$ denote the completion time of job $\pi_{[k]}$ on machine i, and $C_{i,\pi_{[0]}} = 0$. $t_{i,j}$ represents the processing time of job j ($j = 1, 2, \ldots, n$) on machine i ($i = 1, 2, \ldots, m$). For $k = 1, \ldots, n$, $C_{i,\pi_{[k]}} = C_{1,\pi_{[k-1]}} + t_{1,\pi_{[k]}}$ when $i = 1$ and $C_{i,\pi_{[k]}} = max\{C_{i-1,\pi_{[k]}}, C_{i,\pi_{[k-1]}}\} + t_{i,\pi_{[k]}}$ when $i = 2, \ldots, m$. The tardiness of job $\pi_{[k]}$ is $T_{\pi_{[k]}} = max\{C_{m,\pi_{[k]}} - d_{\pi_{[k]}}, 0\}$, where $d_{\pi_{[k]}}$ is the due date of job $\pi_{[k]}$. The total tardiness of $\pi(n)$ can be denoted as

$$\widetilde{T}(\pi(n)) = \sum_{k=1}^{n} T_{\pi_{[k]}} = \sum_{k=1}^{n} max\{C_{m,\pi_{[k]}} - d_{\pi_{[k]}}, 0\}. \quad (1)$$

Obviously, the time complexity of calculating $\widetilde{T}(\pi(n))$ is $O(mn)$. The objective of the considered problem is to find the permutation $\pi^*(n) = \arg\min_{\pi(n)\in\Omega}\{\widetilde{T}(\pi(n))\}$ among the $n!$ solutions.

3. The proposed trajectory scheduling methods

Trajectory Scheduling Method (TSM) is composed of three components: Composite Heuristic, Adaptive Perturbation, and Restart Point Selection. A Composite Heuristic starts from an initial solution which is iteratively improved by an Iterated Improvement until it is stalled at a local optimum. The local optimum is perturbed by the Adaptive Perturbation. A new restart point is selected by the Restart Point Selection and the Iterated Improvement is performed again. The procedure is repeated until a given termination condition is satisfied.

Initially, both the current solution π^c and the current best solution π^b are generated by NEH_{EDD}. An Iterated Improvement procedure in a Composite Heuristic starts from π^c where the neighborhood is constructed by a neighborhood structure. If the best solution π^ℓ of the neighborhood is better than π^c, π^ℓ is selected as the new π^c. In every iteration, π^b is replaced with π^c if π^c is better than π^b. This neighborhood searching procedure is repeated until π^c is not better than π^b. Distinct from traditional perturbation operators each of which generates only one restart point, an Adaptive Perturbation method is developed to produce a set of candidate restart points. To select the most appropriate restart point from the candidate set, the distance from π^b to every candidate is calculated by the Restart Point Selection. If the termination condition is not satisfied, the procedure is repeated. The framework of the proposed trajectory scheduling methods is depicted as Algorithm 1.

Algorithm 1 Framework of Proposed Trajectory Scheduling Methods

1: Generate the start point π^c by NEH_{EDD}. $\pi^b \leftarrow \pi^c$.
2: **repeat**
3: Improve π^c by some *Composite Heuristic*. $\pi^b \leftarrow \pi^c$ if π^c is better than π^b.
4: Produce a set of candidate restart points by *Adaptive Perturbation* on π^c.
5: The best solution of the candidate set is selected as the new start point π^c according to the *Restart Point Selection*.
6: **until** (The termination criterion is satisfied)
7: **return** π^b.

3.1. Composite heuristics

According to the framework given in [27], a heuristic contains three phases: index development, solution construction