European Journal of Operational Research 222 (2012) 31–43

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Local search methods for the flowshop scheduling problem with flowtime minimization

Quan-Ke Pan [a,b], Rubén Ruiz [c,*]

[a] State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, PR China
[b] College of Computer Science, Liaocheng University, Liaocheng 252059, PR China
[c] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universitat Politècnica de València, Ciudad Politécnica de la Innovación, Edificio 8G, Acc. B. Camino de Vera S/N, 46021 Valencia, Spain

ARTICLE INFO

Article history:
Received 23 October 2011
Accepted 25 April 2012
Available online 3 May 2012

Keywords:
Scheduling
Flowshop
Flowtime
Local search
Metaheuristics

ABSTRACT

Flowshop scheduling is a very active research area. This problem still attracts a considerable amount of interest despite the sheer amount of available results. Total flowtime minimization of a flowshop has been actively studied and many effective algorithms have been proposed in the last few years. New best solutions have been found for common benchmarks at a rapid pace. However, these improvements many times come at the cost of sophisticated algorithms. Complex methods hinder potential applications and are difficult to extend to small problem variations. Replicability of results is also a challenge. In this paper, we examine simple and easy to implement methods that at the same time result in state-of-the-art performance. The first two proposed methods are based on the well known Iterated Local Search (ILS) and Iterated Greedy (IG) frameworks, which have been applied with great success to other flowshop problems. Additionally, we present extensions of these methods that work over populations, something that we refer to as population-based ILS (pILS) and population-based IG (pIGA), respectively. We calibrate the presented algorithms by means of the Design of Experiments (DOE) approach. Extensive comparative evaluations are carried out against the most recent techniques for the considered problem in the literature. The results of a comprehensive computational and statistical analysis show that the presented algorithms are very effective. Furthermore, we show that, despite their simplicity, the presented methods are able to improve 12 out of 120 best known solutions of Taillard's flowshop benchmark with total flowtime criterion.

1. Introduction

Finite capacity scheduling entails the determination of the processing order of a series of jobs that have to be processed on the available machines in a production shop. A first classification of scheduling problems can be derived according to the way machines are distributed in the factory. When several machines are arranged in series and jobs must visit all these machines in the same order we have what is called a flowshop. These problems have been subjected to detailed studies since the pioneering work of Johnson (1954). More specifically, a flowshop problem comprises a set N of n jobs that must be processed on a set M of m machines. These m machines are arranged in series and each job $j \in N$ is broken down into m tasks, one per machine. A job models a given production lot of a product or client order that must be manufactured. All jobs visit machines in the same order and p_{ij} denotes the known, non-negative and deterministic amount of time that job j needs at machine i. At any given time, a job is either waiting for processing or being processed by exactly one machine. Similarly, machines are either idle or occupied by a job. Baker (1974, Chapter 6, pp. 136–137) further details all restrictions that apply: All jobs are independent and available for processing at time 0. Machines never break down and are always ready. Once started at a machine, jobs are processed until completion with no preemption allowed, etc. A schedule is obtained after devising a permutation of the jobs for every machine, resulting in $(n!)^m$ possible solutions. The setting is usually simplified and only permutation schedules are examined, resulting in the permutation flowshop scheduling problem (PFSP) where job passing is not allowed, i.e., all jobs visit the machines in the same order. This reduces the number of solutions to $n!$ The objective in the PFSP is to find a permutation such that a given criterion is optimized. Most studied criteria are based on the completion times of the jobs at machines. More specifically, let $\pi = \{\pi(1), \pi(2), \ldots, \pi(n)\}$ be a possible permutation or solution to the problem. The completion time of job j at position $\pi(j)$ at machine i is denoted by $C_{i,\pi(j)}$ and it is computed as follows:

$$C_{i,\pi(j)} = \max\{C_{i-1,\pi(j)}, C_{i,\pi(j-1)}\} + p_{i,\pi(j)} \quad (1)$$

where $j = 1, \ldots, n$, $i = 1, \ldots, m$, $C_{i,\pi(0)} = 0$, and $C_{0,\pi(j)} = 0$.

* Corresponding author.
E-mail address: rruiz@eio.upv.es (R. Ruiz).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.04.034

The completion time of a job j in the shop is then $C_{m,j}$ or C_j for short. With completion times, many different objectives are defined. The most studied criterion is the minimization of the makespan or $C_{\max}$, where $C_{\max} = \max_{j=1,\ldots,n} C_j$. This paper studies the total flowtime minimization, which has also been studied intensively. Total flowtime is defined as:

$$TFT = \sum_{j=1}^{n} C_j \tag{2}$$

When there are no release dates, total flowtime and total completion time are equivalent objectives. Total flowtime minimization reduces the work in progress or WIP and results in a stable utilization of resources. Jobs "stay" in the shop a reduced amount of time (Framinan et al., 2005). This is of particular importance to industries where reducing inventory or holding costs is of paramount importance.

The PFSP with total flowtime criterion is denoted as $n/m/P/\sum C_j$ or as $F/prmu/\sum C_j$ according to the well known existing scheduling notations (Pinedo, 2009, and many others). $F/prmu/\sum C_j$ has been proved to be NP-hard in the strong case for $m \geqslant 2$ after the results of Gonzalez and Sahni (1978). Although some exact methods have been reported in the literature (Ignall and Schrage, 1965; Bansal, 1977; Stafford, 1988 and others), they are limited to small problem instances as solving times quickly become impractical for realistically-sized cases. As a result, research has focused on the development of heuristics that produce reasonable solutions with low time and memory requirements. Some heuristics have been presented by Rajendran (1993), Rajendran and Ziegler (1997) and Li and Wu (2005), to name just a few. With the advent of powerful desktop computers, and now for more than two decades, special emphasis has been given to the study of metaheuristics, capable of producing near optimal solutions, albeit normally at the cost of longer calculations. Some examples are the genetic algorithm of Tang and Liu (2002), ant colony optimization (ACO) of Rajendran and Ziegler (2004) and the differential evolution of Pan et al. (2008), among many others.

Metaheuristics provide excellent results and constitute the state-of-the-art methods available for the PFSP with total flowtime criterion. However, many metaheuristics are fairly sophisticated and depend on several parameters and settings that might be problem and even instance dependent. Most of the time, the presented methods are so specifically tailored for the problem at hand that slight variations of the scheduling setting require extensive changes in the algorithms or even render them inapplicable. In some cases, published algorithms are so intricate that an independent coding is unlikely to obtain the same reported effectiveness or efficiency without contacting the authors to obtain detailed information and/or source codes. All this severely hinders potential practical applications. Therefore, simple, general and easily adaptable algorithms are highly desirable. However, such simplistic methods might produce lower quality solutions and a difficult compromise arises between simplicity and performance.

The Iterated Local Search (ILS) and Iterated Greedy (IG) frameworks, described by Lourenço et al. (2010) and Ruiz and Stützle (2007), respectively, constitute two simple templates for combinatorial optimization. They have resulted in state-of-the-art results for several problems, including the permutation flowshop. Following the successful application of the above two local search based frameworks, this paper presents four algorithms: an IGA, an ILS, and two population-based extensions, dubbed as population-based IGA (pIGA), and population-based ILS (pILS), respectively. The main focus is on simplicity, extensibility and ease of coding and replication of results. The presented methods employ some powerful, yet simple operators in order to improve performance. The results of the presented algorithms are compared to those of recently published metaheuristics. The computational results and statistical analyses show, as we will detail, that the presented algorithms are new state-of-the-art methods for the problem under consideration.

The rest of the paper is organized as follows. Section 2 reviews the literature of the PFSP with total flowtime minimization criterion. Section 3 presents the four local search based algorithms in detail. The proposed algorithms are calibrated in section 4. A comprehensive comparison of the presented algorithms is shown, along with statistical analyses, in Section 5. Finally, we conclude the paper in Section 6.

2. Literature review

The PFSP with total flowtime criterion was first studied by Ignall and Schrage (1965) and by Gupta (1972). This is more than a decade later than the pioneering work of Johnson (1954) for makespan minimization in the PFSP. Due to the difficulty faced by exact methods to solve medium size or large instances, efforts have been mainly dedicated to finding high quality solutions in a reasonable computational time by using heuristic or metaheuristic optimization techniques. Framinan et al. (2005) provide a comprehensive review and evaluation of heuristics for the PFSP with total flowtime criterion. Here we mention just the most cited heuristics. Rajendran (1993), Rajendran and Ziegler (1997), Liu and Reeves (2001), Li and Wu (2005) and, more recently, Laha and Sarin (2009) present high performing simple heuristics. Other more elaborated methods are those of Allahverdi and Aldowaisan (2002), Framinan et al. (2005), and Li et al. (2009). In any case, in order to attain a better solution quality for the problem under consideration, modern metaheuristics have been increasingly applied in recent years. One of the earliest known applications of genetic algorithms (GA) is due to Vempati et al. (1993). In this case, a simple GA was presented but only applied to small instances of size 25×10 (25 jobs and 10 machines) maximum. Later, Yamada and Reeves (1998) presented a genetic local search algorithm (GA_{LS}) providing good quality solutions for five sets of Taillard (1993) instances (20×5, 20×10, 20×20, 50×5 and 50×10) but needing large computational times. Gupta et al. (2000) designed a tabu search (TS) based approach that was compared against the heuristics of Rajendran (1993) obtaining better results for the tested instances. Rajendran and Ziegler (2004) proposed two ant colony optimization (ACO) algorithms, called M-MMAS and PACO, respectively, for makespan and total flowtime minimization. PACO showed better performance than M-MMAS and the best heuristic proposed by Liu and Reeves (2001). Later, Rajendran and Ziegler (2005) have introduced a new ACO algorithm based on similar concepts to those of M-MMAS and PACO with slightly better performance in some scenarios. Tasgetiren et al. (2007) extended a continuous particle swarm optimization (PSO) method to the PFSP with both makespan and total flowtime criteria. With this method, 57 out of 90 best known solutions reported by Liu and Reeves (2001) and Rajendran and Ziegler (2004) for Taillard (1993) benchmarks were improved. However, the PSO was soon surpassed by the combinatorial PSO (CPSO) of Jarboui et al. (2008) and also by the discrete differential evolution (DDE_{RLS}) and iterated greedy algorithms (IG_{RLS}) of Pan et al. (2008).

Quite recently, it seems that there has been an intensified interest in this problem as quite a number of new metaheuristics have been published. Tseng and Lin (2009) proposed a hybrid genetic local search algorithm (denoted as HGA_{T1}) by employing GA to do the global search and two methods, Insertion Search and Insertion Search with Cut-and-Repair, to do the local search. The authors demonstrated improved performance of their proposed HGA_{T1} over the PSO of Tasgetiren et al. (2007), GA_{LS} of Yamada and Reeves