Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops

Korhan Karabulut

Software Engineering Department, Yasar University, Izmir, Turkey

ARTICLE INFO

Article history:
Received 5 March 2015
Received in revised form 4 May 2016
Accepted 9 June 2016
Available online 10 June 2016

Keywords:
Flowshop problem
Scheduling
Tardiness
Iterated greedy algorithm
Random search

ABSTRACT

The permutation flowshop scheduling problem is an NP-hard problem that has practical applications in production facilities and in other areas. An iterated greedy algorithm for solving the permutation flowshop scheduling problem with the objective of minimizing total tardiness is presented in this paper. The proposed iterated greedy algorithm uses a new formula for temperature calculation for acceptance criterion and the algorithm is hybridized with a random search algorithm to further enhance the solution quality. The performance of the proposed method is tested on a set of benchmark problems from the literature and is compared to three versions of the traditional iterated greedy algorithm using the same problem instances. Experimental results show that, the proposed algorithm is superior in performance to the other three iterated greedy algorithm variants. Ultimately, new best known solutions are obtained for 343 out of 540 problem instances.

1. Introduction

Efficient scheduling is very important in production, manufacturing, information processing, and transportation systems (Pinedo, 2012). The permutation flowshop problem (PFSP) is a combinatorial optimization problem in which there is a set $N = \{1, 2, \ldots, n\}$ of n jobs to be processed on a set $M = \{1, 2, \ldots, m\}$ of m machines. Jobs are executed in the same order on all machines starting from machine 1 to machine m. The objective is to find a permutation of jobs that optimizes a given criterion. The most widely used optimization objective is the minimization of the maximum completion time, also known as, the makespan or C_{max}. In real life applications, scheduling problems may have due dates which make the problem more difficult. Maximum lateness, number of tardy jobs, total tardiness and total weighted tardiness are among several objectives for scheduling problems that involve due dates (Pinedo, 2012). Tardiness of a job is defined as $T_j = \max\{C_{mj} - d_j, 0\}$ where C_{mj} is the completion time of job j on the last machine and d_j is its due date. There is a positive tardiness if the job j finishes after its due date; otherwise tardiness for the job will be 0. The completion time of job j on machine i is calculated as $C_{ij} = \max\{C_{i-1,j}, C_{i,j-1}\} + p_{ij}$, $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$ where p_{ij} is the processing time of job j on machine i, $C_{0,j} = 0$ for $j = 1, 2, \ldots, n$ and $C_{i,0} = 0$ for $i = 1, 2, \ldots, m$. The objective used in this paper is the minimization of total tardiness of all jobs which is calculated as $\sum_{j=1}^{n} T_j$. This problem can be denoted as $F/prmu/\sum T_j$ (Pinedo, 2012).

PFSP with total tardiness objective is shown to be *NP*-hard even for one machine (Du & Leung, 1990). There exist heuristic methods as well as exact methods in the literature. Due to the complexity of the problem, exact methods are employed only for small number of jobs and machines. Branch-and-bound algorithms are proposed for the single machine case (Della Croce, Tadei, Baracco, & Grosso, 1998) as well as the two machine case (Kim, 1993; Pan, Chen, & Chao, 2002; Pan & Fan, 1997; Schaller, 2005; Sen, Dileepan, & Gupta, 1989). Chung, Flynn, and Kirca (2006) was able to solve problems up to 15 jobs and 2 machines, and 20 jobs and 8 machines. Most recently, Baker (2013) reports the results of a comparison with Chung et al.'s branch-and-bound algorithm and a commercial Excel add-in (RSP), where RSP was able to solve all the generated problems having 16 jobs with 4 and 8 machines, all of the 8-machine problems with 18 jobs except for one, and all of the 4-machine problems with 22 jobs except for two; within an hour of CPU time on a computer that has an Intel Core2 2.7 GHz processor with 8 GB of RAM.

Heuristic algorithms construct solutions from scratch, based on priority rules such as earliest due date (EDD), minimum slack (SLACK), shortest processing times (SPT) and modified due date (MDD) (Vallada, Ruiz, & Minella, 2008). Such heuristics can be found in Gelders and Sambandam (1978), Ow (1985) and Raman (1995). NEH method (Nawaz, Enscore, & Ham, 1983) is the best performing heuristic for the makespan criterion. Kim (1993)

E-mail address: korhan.karabulut@yasar.edu.tr

http://dx.doi.org/10.1016/j.cie.2016.06.012
0360-8352/© 2016 Elsevier Ltd. All rights reserved.

provides an adapted version of the NEH heuristic for the total tardiness problem called NEH_{edd} where jobs are initially sorted by using earliest due date (EDD) rule, i.e., by their due dates, in an increasing order. NEH_{edd} is also used as an initial solution for other heuristic and metaheuristic algorithms. For example, NEH_{edd} is used in Kim, Lim, and Park (1996) to generate the starting solution for two improvement heuristics named ENS1 and ENS2 that use insert and swap neighborhoods respectively. Most recently, Viagas and Framinan (2015) introduced new heuristics based on NEH_{edd} that can find 25% better results than the original NEH_{edd}.

Several metaheuristic methods have been used for total tardiness minimization. Onwubolu and Mutingi (1999) proposed a genetic algorithm (GA), Armentano and Ronconi (1999) used tabu search, Hasija and Rajendran (2004) proposed a simulated annealing algorithm with local search and Onwubolu and Davendra (2006) used a differential evolution algorithm.

Readers are referred to Vallada et al. (2008) for a detailed review and comparative evaluation of numerous exact, heuristic and metaheuristic methods for the PFSP with total tardiness objective. Vallada et al. (2008) also proposed a benchmark suite that is used in this work, to be able to compare the algorithms on a common test set.

In most recent works, Vallada and Ruiz (2009) report the results for parallel and serial execution of several cooperative metaheuristics for both total tardiness and makespan criterion. Framinan and Leisten (2008) use a variable iterated greedy algorithm where the number of jobs that are removed from the current permutation is varied from 1 to $n-1$. Vallada and Ruiz (2010) present a GA with path relinking, and Kellegöz, Toklu, and Wilson (2010) use an elite guided steady-state GA.

Chen and Li (2013) use an integrated iterated local search (IILS) that is based on ILS. M'Hallah (2014) presents an iterated local search algorithm hybridized with a variable neighborhood descent algorithm. Cura (2015) proposes a new evolutionary approach with a new mating scheme designed for the problem that can achieve better results as the size of the problem increases.

The IG algorithm presented by Ruiz and Stützle (2007) for the PFSP is a simple and easy to implement, yet powerful and effective metaheuristic. The algorithm has two phases: destruction and construction. In the destruction phase, a number of randomly selected jobs are removed from the current solution. In the construction phase, each removed job is reinserted to all available positions by using the NEH heuristic. A local search algorithm may also be applied after these two phases. For diversification, Metropolis criterion, which is also the selection mechanism in simulated annealing, is used as the acceptance rule.

Variable Neighborhood Search (VNS) (Mladenovic & Hansen, 1997) is another metaheuristic where a systematic change of different neighborhoods are applied to a solution at hand. Usually, local search phase of VNS is costly for large search spaces. Reduced VNS (RVNS) (Hansen & Mladenovic, 2001) is a simplified VNS in which there is no local search step. In RVNS, random neighborhoods of the new solution candidate are investigated for given number of times and the incumbent solution replaces the current best only if it is better. Variable Neighborhood Descent (VND) (Hansen & Mladenovic, 2001) is another extension of VNS where there is no shaking step and neighborhood change is made in a deterministic way.

Random search is a simple metaheuristic and is easy to implement and can be adapted to different problems. Random search starts from an initial solution and is based on adding a random vector to the current solution, replacing the current solution if the new solution is better and repeating this procedure until the stopping criterion is met. Although this scheme is simple, it is shown to guarantee to find the global optimum by Baba (1981) and Solis Francisco, Wets, and Roger (1981).

In the following sections, an IG algorithm that uses RLS, which is a combination of random search and reduced VNS in the local search phase for solving the PFSP with total tardiness minimization criterion is presented. The proposed algorithm, called IG_RLS, uses efficient implementation of speedups and provides the state-of-art results by finding the new best-known solutions for 343 out of 540 instances for the problem instances proposed by Vallada et al. (2008).

The rest of the paper is organized as follows: Section 2 gives the details of the proposed algorithm. The computational results are presented in Section 3. Conclusions are provided in Section 4.

2. The iterated greedy algorithm

The IG_RLS algorithm uses permutation representation for candidate solutions. The pseudocode of the algorithm is given in Fig. 1.

The initial solution is constructed using the NEH_{edd} heuristic. NEH is considered as the best heuristic for makespan minimization. NEH starts with sorting jobs by their total processing times in decreasing order, taking the first two jobs and obtaining a partial permutation that minimizes the partial makespan. Then, the heuristic proceeds with inserting the remaining jobs into every possible slot in the partial solution, considering jobs one by one in the obtained order. The slot that minimizes the makespan is selected for insertion of the current job. In NEH_{edd} heuristic, the jobs are sorted by their due dates in an increasing order. Again, as in the original NEH, after selecting the ordering of the first two jobs, each possible slot is considered for all the remaining jobs and the slot that minimizes the total tardiness is selected.

RLS local search is applied to the initial solution. Then, in a loop, the current job sequence is destructed and constructed. The destruction size (d) is set to 4 as found experimentally in Ruiz and Stützle (2007). In the destruction phase, d randomly selected jobs are removed from the current permutation. In the construction phase, each removed job, in the order in which it was removed, is inserted into all possible positions in the partial permutation. The resulting partial permutation that minimizes total tardiness is selected. The reinsertion procedure is computationally expensive, especially for large instances, since all of the positions are considered. Li, Wang, and Wu (2009) describe a modified NEH procedure for total flowtime minimization that could save up to 50% of the time required for calculating the total flowtime. The main idea behind the speedup mechanism is reusing the flowtime calculations for the unchanged part of the permutation after an insert or swap operation and calculating the flowtime only for the changed part of the permutation. A matrix of flowtimes for each job on each machine has to be calculated for the initial permutation in order to

```
procedure IG_RLS
    $\pi_0 \leftarrow NEH_{edd}$
    $\pi_{best} \leftarrow \pi \leftarrow LocalSearch(\pi_0)$
    $d \leftarrow 4$
    while(Not Termination)
        $\pi' \leftarrow DestructionConstruction(\pi, d)$
        $\pi'' \leftarrow LocalSearch(\pi')$
        if $f(\pi'') < f(\pi)$ then
            $\pi \leftarrow \pi''$
            if $f(\pi'') < f(\pi_{best})$ then
                $\pi_{best} \leftarrow \pi''$
            endif
        else if $random() < exp\{-((f(\pi'') - f(\pi))/Temperature)\}$ then
            $\pi \leftarrow \pi''$
        endif
    end while
    return $\pi_{best}$
end procedure
```

Fig. 1. The IG_RLS algorithm.