Computers & Operations Research 45 (2014) 7–11

Contents lists available at ScienceDirect

Computers & Operations Research

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

A hybrid heuristic approach for single machine scheduling with release times

Federico Della Croce [a,b,*], Fabio Salassa [a], Vincent T'kindt [c]

[a] D.A.I., Politecnico di Torino, Italy
[b] CNR, IEIIT, Torino, Italy
[c] Université Francois-Rabelais, Tours, France

ARTICLE INFO

Available online 1 December 2013

Keywords:
Integer programming
Matheuristics
Positional completion times

ABSTRACT

In this work we consider the well-known one-machine total completion time sequencing problem subject to release times. We present a very large scale neighborhood search heuristic based on mathematical programming. This heuristic makes use of the positional completion time formulation of the problem in which valid inequalities are added. The proposed procedure compares favorably with the state of the art heuristics.

1. Introduction

We consider the one-machine total completion time sequencing problem subject to release times. The problem can be stated as follows. A set N of n jobs must be processed on a single machine which is always available. Each job j has a processing time p_j and a release time r_j. The machine can process one job at a time and the processing of each job j cannot start before r_j. Preemption is not allowed. The objective is to minimize the sum of completion times $\sum C_j$ where C_j denotes the completion time of job j.

The problem is known to be $\mathcal{NP}$−hard in the strong sense [13] and is denoted as $1|r_j|\sum C_j$ in the three-field classification. The $1|r_j|\sum C_j$ problem has been extensively studied in the literature. If identical release times are considered, the related $1||\sum C_j$ problem is solvable in $O(n \log n)$ time by means of the so-called SPT (Shortest Processing Time) priority rule [16]. On the other hand, if preemption is allowed, the related $1|r_j, pmtn|\sum C_j$ problem is solvable in $O(n \log n)$ time by means of the so-called SRPT (Shortest Remaining Processing Time) priority rule [1], where at any time the job with shortest remaining processing time among the available jobs is scheduled. Clearly, the optimal solution value of the $1|r_j, pmtn|\sum C_j$ problem provided by the SRPT rule constitutes a lower bound for the corresponding $1|r_j|\sum C_j$ problem. An improvement on this bound exploiting the structural properties of the preemptive solution has been proposed in [6]. Such improvement is shown to run in $O(n^2)$ time. Among the exact methods, we cite the efficient exact procedures proposed in [2,14,15]. As far as heuristic algorithms are concerned, we cite the Recovering Beam Search (RBS) approach of [5], the tabu search approach presented in [9] and the recent hybrid approach of [12]. In [12] is experimentally reported that [5] outperforms [9] in terms of the average deviation to the optimal solution and that [12] outperforms both [5,9] at the cost of an important increase of the solution time. Purpose of this work is to handle this problem by means of a matheuristic procedure.

Matheuristics are methods that recently attracted the attention of the community of researchers, suddenly giving rise to an impressive amount of work in a few years. Matheuristics lie on the general idea of exploiting the strength of both metaheuristic algorithms and exact methods as well, leading to a "hybrid" approach (see [11]), but because of their novelty there is no unique classification nor a consolidated working framework in the field; hence, it is hard to state a pure and sharp definition of these methods.

A distinguishing feature is often the exploitation of nontrivial mathematical programming tools as part of the solution process. For example, in [8] a sophisticated Mixed-Integer Linear Programming (MILP) solver is used for analyzing very large neighborhoods in the solution space.

A crucial issue also underlined in [11] is that the structure of these methods is not a priori defined and in fact a solution approach can be built in many different ways. A simple approach used in the present work consists of a two-stage procedure: a first heuristic procedure is applied to the problem for generating a starting solution and then a post processing "refinement" procedure is applied exploiting, for example, some peculiar properties of the mathematical formulation of the problem under analysis. Here, we couple a heuristic algorithm like Recovering Beam Search (RBS) [3,5] with a neighborhood search based on a MILP

* Corresponding author.
E-mail addresses: federico.dellacroce@polito.it (F. Della Croce), fabio.salassa@polito.it (F. Salassa), tkindt@univ-tours.fr (V. T'kindt).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.11.016

formulation solved by means of a commercial tool. This approach has already been successfully applied to the $F2||\sum C_j$ problem [4]. The two-stage approach is appealing because of its simplicity and for the possibility of concentrating more on modeling the neighborhood instead of building up the search procedure. Exploiting this idea we reached very good results, improving the solution quality over the state of the art heuristics.

The remainder is organized as follows. In Section 2 a MILP model based on positional completion times is recalled and strengthened by means of valid inequalities. The proposed math-heuristic procedure is described in Section 3. Extensive computational testing is discussed in Section 4. Final remarks are given in Section 5.

2. Problem formulation and valid inequalities

Various classical ILP formulations for the $1|r_j|\sum C_j$ problem have been proposed in the literature. However, in terms of efficiency, the positional variables ILP formulation (see [10]) turns out to be the best one with respect to other classical models based on disjunctive variables and constraints. This formulation is defined as follows.

Let $C_{[j]}$ be the completion time of the jth job processed and let x_{ij} be a 0/1 variable, where $i,j \in \{1,\ldots,n\}$. A variable x_{ij} is equal to 1 if job i is in position j, and 0 otherwise.

$$\min \sum_{j=1}^{n} C_{[j]} \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,\ldots,n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,\ldots,n \tag{3}$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \tag{4}$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \quad \forall j = 2,\ldots,n \tag{5}$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \quad \forall j = 2,\ldots,n \tag{6}$$

$$x_{ij} \in \{0,1\}, \quad C_{[j]} \geq 0 \tag{7}$$

where constraints (2) and (3) state that a job is chosen for each position in the sequence and each job is processed exactly once. Constraint (4) sets the completion time of the first job. Constraints (5) and (6) forbid each job to start before the job in the previous position completes and before its release date.

The above formulation can be strengthened by adding valid inequalities derived from properties proposed in [2,6]. We make use of the following notation. Let the set N of jobs be reindexed in such a way that $(1,2,\ldots,n)$ is a Shortest Processing Time (SPT) sequence, hence $i<j$ whenever $p_i < p_j$ or $p_i = p_j$ and $r_i \leq r_j$ (with ties broken arbitrarily). Let S denote the SRPT schedule and let OPT denote the optimal sequence.

In [2], it was shown that, for all positions j, the completion time of the jth job in OPT is greater than or equal to the completion time of the jth job in S, that is $C_{[j]}(OPT) \geq C_{[j]}(S)$.

On the other hand, in [6], an improvement over the SRPT lower bound has been proposed by application of a logical reasoning on positions in an optimal schedule. We recall here how this improvement is computed. We know that a job k completed in position $[j]$ in S will occupy some position in the optimal sequence OPT and this issue can be formalized according to the following three exhaustive cases: job k occupies position $l \geq [j+1]$, job k occupies position $h \leq [j-1]$, or job k occupies position $[j]$. In [6] it is shown (and we refer to that paper for details) that, as one of the above cases must hold, it is possible to compute in $O(n^2)$ time a coefficient θ_j (for all positions j) indicating a lower bound on the increase in OPT of the completion times of the jobs placed in positions $[j-1]$, $[j]$ and $[j+1]$ with respect to the jobs occupying the same positions in the $SRPT$ sequence S, that is $C_{[j-1]}(OPT) + C_{[j]}(OPT) + C_{[j+1]}(OPT) \geq C_{[j-1]}(S) + C_{[j]}(S) + C_{[j+1]}(S) + \theta_j$. Notice that position $j-1$ makes no sense for $j=1$ and position $j+1$ makes no sense for $j=n$.

Combining the above results, the following Property holds.

Property 1. *The following are valid inequalities that can be added to the problem model.*

$$C_{[j]} \geq C_{[j]}(S) \quad \forall j \in 1,\ldots n \tag{8}$$

$$C_{[1]} + C_{[2]} \geq C_{[1]}(S) + C_{[2]}(S) + \theta_1 \tag{9}$$

$$C_{[j-1]} + C_{[j]} + C_{[j+1]} \geq C_{[j-1]}(S) + C_{[j]}(S) + C_{[j+1]}(S) + \theta_j \quad \forall j = 2\ldots n-1 \tag{10}$$

$$C_{[n-1]} + C_{[n]} \geq C_{[n-1]}(S) + C_{[n]}(S) + \theta_n \tag{11}$$

We conducted preliminary experiments on a QuadCore 2.67 GHz PC with 3 GB RAM to evaluate the impact of the above valid inequalities when solving the ILP formulation by CPLEX 12.3 solver. Table 1 reports the benchmark on solution quality and time of the pure model and the enhanced one over the whole set of 200 testbed instances with 50 jobs taken from [9] and kindly provided by the authors. These instances are generated using the scheme provided in [7] that we recall below. The processing times are drawn at random from the discrete uniform distribution on [1, 100]. The release dates are uniformly distributed between [0, $50.5nR$] where ten R values (0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0, 3.0) are considered. Globally 10 problem classes (in the following denoted as $Class_1$–$Class_{10}$) are generated. For each class, 20 instances are generated. A time limit of 120 s is imposed. In the table, the entry *Solved instances* indicates the total number of each instances solved to optimality within the time limit.

As we can see, the pure model is able to solve optimally just 11 instances over 200. Viceversa, the enhanced model is able to solve optimally all instances requiring approximately 2 s on the average and 26 s in the worst-case. Indeed, adding the valid inequalities strongly helps in solving more efficiently the problem to optimality. We also performed some experiments to evaluate the efficiency of CPLEX 12.3 to solve the enhanced ILP formulation compared to the dedicated best available exact approaches. We compare this approach (indicated as *ILP*) with the algorithm of [14] (indicated as *TDCE*) and with the algorithm of [15] (indicated as *ST* and using the standard parameters indicated in the reference). Notice that the algorithm of [14] generalizes the one of [2] and is strictly superior to it in terms of performances. We consider here the sets of 50-job and 100-job instances taken from [9] and

Table 1
Impact of the valid inequalities on 50 jobs instances.

Instance size	Model (1)–(7)	Model (1)–(11)		
	Solved instances	Solved instances	Average CPU time (s)	Maximum CPU time (s)
50 jobs	11/200	200/200	2.09	26.32