RAIRO-Oper. Res. 50 (2016) 681–687
DOI: 10.1051/ro/2016021

RAIRO Operations Research
www.rairo-ro.org

IMPROVING THE SOLUTION COMPLEXITY OF THE SCHEDULING PROBLEM WITH DEADLINES: A GENERAL TECHNIQUE [*]

Amir Elalouf[1] and Eugene Levner[2]

Abstract. The aim of this paper is to develop improved polynomial-time approximation algorithms belonging to the family of the fully polynomial time approximation schemes (FPTAS) for a group of scheduling problems. In particular, the new technique provides a positive answer to a question posed more than three decades ago by Gens and Levner [G.V. Gens and E.V. Levner, *Discrete Appl. Math.* **3** (1981) 313–318]: "Can an epsilon-approximation algorithm be found for the minimization version of the job-sequencing-with-deadlines problem running with the same complexity as the algorithms for the maximization form of the problem?"

Mathematics Subject Classification. 41A29, 41A10, 65D15, 65Y10, 68Q25.

Received June 2, 2015. Accepted February 23, 2016.

1. Introduction

The scheduling problem with deadlines, introduced more than 30 years ago by Lawler and Moore [10], is formulated as follows. Let there be n independent jobs $J_1, J_2, \dots, J_n$ to be processed on one machine. Associated with each job, J_i, $i = 1, \dots, n$, are its processing time, p_i, and deadline, d_i, $i = 1, \dots, n$. If the processing of job J_i, is not completed by its deadline, d_i, then a penalty w_i is to be paid, $i = 1, \dots, n$. The problem is to schedule the jobs in such an order as to minimize the total penalty, that is, to find a permutation $\pi = (\pi(l), \pi(2), \dots, \pi(n))$ of $\{1, 2, \dots, n\}$ that minimizes

$$W(\pi) = \sum_{i=1,\dots,n} w_{\pi(i)} x_{\pi(i)}$$

where $x_{\pi(i)} = [if p_{\pi(1)} + p_{\pi(2)} + \cdots + p_{\pi(i)} > d_{\pi(i)}$ then 1 else 0].

Lawler and Moore [10] proposed a pseudo-polynomial exact method for the solution of this problem. However, the problem, denoted MIN-JSDP, is known to be NP-hard [4,9]. Sahni [12,13] successfully solved the maximization version of this problem, using a fully polynomial time approximation scheme (FPTAS) running in $O(n^2/\epsilon)$ time, where ϵ is the required solution accuracy. However, Sahni's algorithm cannot be directly generalized to solve MIN-JSDP. This is because the algorithm is based on the use of a tight bound b such that $b \leq W^* \leq cb$, with W^* being the minimum value of the penalty and c a small constant; in the case of MIN-JSDP, it is nontrivial to obtain such a bound b.

Keywords. Job-sequencing-with-deadlines scheduling problem, approximation algorithm, FPTAS.

[*] *This paper is extended and improved version to the paper submitted to the 2nd International Conference on Control, Decision and Information Technologies (CoDIT'14), Metz, France, November 3-5, 2014.*

[1] Bar Ilan University, Ramat Gan, Israel. amir.elalouf@biu.ac.il

[2] Ashkelon Academic College, Ashkelon, Israel.

Gens and Levner [5] developed an FPTAS algorithm for the MIN-JSDP with complexity $O(n^3/\epsilon)$. The algorithm is based on the ϵ-grouping technique, which does not require knowledge of a tight bound. A similar ϵ-grouping technique was developed by Hassin [7], who designed an FPTAS for the restricted shortest path problem. Gens and Levner [6] improved their FPTAS for the MIN-JSDP, obtaining a worst-case complexity of $O(n^2/\epsilon + n^2 \log n)$; this complexity was still greater than that of Sahni's [12, 13] algorithm for the maximization version of the problem. This outcome led Gens and Levner [6] to ask whether it is possible to develop an ϵ-approximation algorithm for MIN-JSDP whose complexity is the same as that of the algorithm for the maximization form of the problem.

The aim of this paper is to provide a positive answer to that question. In the next section we describe the problem. In Section 4 we present the improved FPTAS algorithm and estimate its complexity. Section 5 concludes the paper.

2. Problem formulation

It is well-known (see [10, 15] that, when seeking to identify a schedule that solves the MIN-JSDP as defined above, it is possible to limit the search to schedules such that: (i) jobs that are to be on time are processed in increasing order of their deadlines, with the late jobs following them in arbitrary order; and (ii) there is no idle time between jobs. The optimal schedule is included among the schedules that meet these two criteria. In what follows, we sort all the jobs in increasing order of their deadlines (earliest deadline first) and let

$$x_i = [\text{if the job } J_i \text{ is to be late then 1 else 0}].$$

Then the MIN-JSDP can be written as the following integer programming problem:

$$\text{minimize } W(\boldsymbol{x}) = \textstyle\sum_{i=1,\ldots,n} w_i x_i$$

$$\text{subject to} \quad \sum_{i=1,\ldots,j} t_i(1 - x_i) \le D_j, \quad j = 1, \ldots, n,$$

$$x_i = 0 \text{ or } 1, \quad i = 1, \ldots, n,$$

$$\text{where } \boldsymbol{x} \text{ stands for } (x_1, \ldots, x_n).$$

The problem can be rewritten as follows:
MIN-JSDP:

$$\text{minimize} \quad W(\boldsymbol{x}) = \sum_{i=1,\ldots,n} w_i x_i$$

$$\text{subject to} \quad T(\boldsymbol{x}) = \sum_{i=1,\ldots,j} p_i x_i \le B_j, \quad j = 1, \ldots, n,$$

$$x_i = 0 \text{ or } 1, \quad i = 1, \ldots, n,$$

$$\text{where } B_j = \textstyle\sum_{i=1,\ldots,j} p_i - d_j, \quad j = 1, \ldots, n.$$

3. A new FPTAS algorithm

Our approach to constructing a new FPTAS for the MIN-JSDP follows a computational scheme that we recently developed for solving the restricted shortest path problem (see [1, 11]). The algorithm consists of three main stages:

Stage A. Find a preliminary lower bound LB and an upper bound UB on the optimal solution such that $UB/LB \le n$.

Stage B. Find improved lower and upper bounds UB and LB on the optimal solution such that $UB/LB \le 2$. This step is key to reducing the complexity of the algorithm in comparison with that of [6]. Specifically, Gens and Levner [6] identified these bounds in $O(n^2 \log n)$ time; we will present a technique that enables us to calculate these bounds with lower complexity. We will also show that this technique can be used to improve the solution complexity of other types of scheduling problems.

Stage C. Partition the interval $[LB,\, UB]$ into $\lceil n/\epsilon \rceil$ equal sub-intervals, delete sufficiently close solutions in the sub-intervals, and then find an ϵ-approximation solution using full enumeration of the "representatives" taking only one "representative" from each sub-interval.

Let us consider these stages in detail.

A. Finding the preliminary bounds

This step is presented in Gens and Levner [6] and is executed as follows: Initialize every x_i to 0. Order (temporarily) all the jobs according to non-decreasing penalties: $w_{i(1)} \le w_{i(2)} \le \cdots \le w_{i(n)}$. Then set $x_{i(1)} = 1$, $x_{i(2)} = 1$, $x_{i(3)} = 1, \ldots$ in that order until all the n constraints of the MIN-JSDP are satisfied.

Let k^* be the smallest number of such x_i's, and let $I = i(k^*)$. As shown in Gens and Levner [6], $w_I \le W^* \le k^* w_I \le n w_I$, and therefore we can take $LB = w_I$, and $UB = k^* w_I$. If we cannot find such an I index, the problem has no feasible solution; then we terminate the algorithm and return "no solution". The running time needed to find the value of w_I is clearly $O(n^2)$.

B. Improving the bounds

This step has two building blocks: a test procedure denoted by Test(v,ϵ) (Algorithm 1) and a narrowing procedure denoted by NARROW (Algorithm 2), which uses Test(v,ϵ) as a sub-procedure.

Test(v, ϵ) is an approximate dichotomous search for the minimum total penalty value W^*. It is implemented as a parametric dynamic programming algorithm that has the following property: for given positive parameters w and ϵ, it reports either that $W^* \le v$ or that $W^* \ge v(1-\epsilon)$. This approach was first used by Gens and Levner [6] in solving the knapsack problem and was later explored by Warburton [16], Hassin [8], Ergun *et al.* [7] and Levner *et al.* [7], among others [8], for different versions of the constrained routing problem.

Test(v,ϵ) will be repeatedly applied as a sub-procedure in the NARROW algorithm to narrow the gap between the bounds UB and LB up to $UB/LB \le 2$. This stage is very similar to the method described in Ergun *et al.* [3], with some minor adaptations for the specific characteristics of the scheduling problem. For completeness of exposition, we provide a brief description of both building blocks.

Algorithm 1. Algorithm Test(v, ϵ).

```
input : $\epsilon > 0$, $v$; $\delta = \epsilon v/n$; $(w_i,p_i,d_i)$, $i = 1,\ldots,n$, such that $d_1 \le d_2 \le \cdots \le d_n$; $B_j = \sum_{i=1,\ldots,j} t_i - d_j$,
        $j = 1,\ldots,n$.
output: Report either that $W^*$, is less than or equal to $v$, or that $W^*$ is larger than $v(l-\epsilon)$.
1  Step 1. [initialization]. $S(0) = \{(0,0)\}$; $W(0) = \emptyset$.
2  Step 2. [Generate $S(1)$ to $S(n)$].
3  for $i = l$ to $n$ do
4      $V(i) = \emptyset$
5      foreach pair $(W,\, P)$ in $S(i-1)$ do
6          if $P + t_i \ge B_i$ then
7              $V(i) \leftarrow V(i) \cup \{(W + w_i, P + p_i)\}$
8          end
9      end
10     Form $G(i-1)$, the set of pairs $(W,\, P)$ in $S(i-1)$ such that $P \ge B_i$. Form $S(i)$ by merging $G(i-1)$ and $V(i)$.
       During the merge eliminate excessive $\delta$-close and $w$-redundant pairs. If $S(i)$ is empty, go to Step 3.
11 end
12 Step 3. If any one of $S(l),\ldots,S(n)$ is empty, then report that $W^* > v(l-\epsilon)$; otherwise report that $W^* \le v$.
```

Associate each vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ with a pair (W, P), where $W = W(\boldsymbol{x})$ and $P = P(\boldsymbol{x})$ are as defined in Section 2. Denote by $S(k)$ the set of pairs (W, P), arranged in increasing order of P values, where every pair in $S(k)$ corresponds to a schedule of jobs $1, \ldots, k$.

If two pairs, (W_1, P_1) and (W_2, P_2), fulfill $P_1 \leq P_2$ and $W_1 \leq W_2$, then the pair (W_2, P_2) is considered to be dominated by (W_1, P_1) and may be discarded. When we are interested in finding solutions satisfying $W(\boldsymbol{x}) < v$ (v being a parameter), a pair (W, P) with $W > v$ is called v-*redundant* and can be discarded.

If two pairs, (W_1, P_1) and (W_2, P_2), fulfill $0 \leq W_2 - W_1 \leq \delta$, then the pairs are called δ-close, and one of the pairs can be discarded. To discard δ-close pairs from a set $S(k)$, we perform the following: (a) partition the interval $[l, w]$ into $\ulcorner n/\epsilon \urcorner$ equal subintervals of size no greater than $\delta = \epsilon v/n$; (b) if a given subinterval contains more than one pair from $S(k)$, then discard all δ-close pairs from the subinterval, with the exception of a single "representative", namely, the pair with the largest (in this subinterval) T coordinate.

Now we can describe the procedure Test(v, ϵ) (Algorithm 1) where $V(k)$ and $G(k)$ will denote auxiliary sets of vectors $\boldsymbol{x}$, $k = 1, \ldots, n$. The complexity of Test(v, ϵ) as well as its required space are $O(n^2/\epsilon)$. This fact and the validity of this procedure are proven in Gens and Levner [6].

We now present the narrowing procedure, NARROW (Algorithm 2), which originates from a similar procedure suggested by Ergun *et al.* [3] for solving the restricted shortest path problem. Specifically, when running Test(v, ϵ), we choose ϵ as a function of UB/LB, such that the value of ϵ changes from iteration to iteration. For the reader's convenience, to distinguish this iteratively changing value from the allowable error (denoted by ϵ) in the FPTAS, which will be presented below, we denote the former by θ. Thus, in what follows, the procedure Test(v, ϵ) will be referred to as Test(v, θ).

The idea is that when the bounds UB and LB are far from each other, we choose a large θ; when UB and LB get closer, we can choose a smaller θ. More precisely, as in Ergun *et al.* [3], in each iteration of Test(v, θ), we set $\theta \leftarrow 1 - \sqrt{LB/UB}$, whereas a new w value in each iteration depends upon θ and equals $v = \sqrt{LB \cdot UB/(1-\theta)}$. The difference between NARROW and the similar narrowing algorithm presented in Ergun *et al.* [3] lies in the content of the testing sub-procedure Test(v, θ). The complexity of NARROW (Algorithm 2) is $O(n^2)$. The proof is similar to that of Lemma 5 in Ergun *et al.* [3] and is omitted here.

Algorithm 2. Algorithm NARROW.

```
   input  : $LB$ and $UB$ such that $UB/LB \leq n$.
   output: $LB$ and $UB$ such that $UB/LB \leq 2$.
1  if $UB/LB \leq 2$ then
2      go to line 2
3  end
4  Set $\theta \leftarrow 1 - \sqrt{LB/UB}$
5  Set $v = \sqrt{LB \cdot UB/(1-\theta)}$
6  Run Test$(v, \theta)$
7  if Test$(v, \theta)$ returns that $W^* < v$ then
8      Set $UB \leftarrow v$
9      else Set $LB \leftarrow v(1-\theta)$
10 end
11 Go to line 2
12 Return the improved $LB$ and $UB$ end
```

C. Fast-approximation algorithm

We start Stage C with LB and UB values satisfying $UB/LB \leq 2$, and obtain an ϵ-approximation schedule. This Algorithm 3 is of the dynamic programming type, so it is very similar to the above algorithm Test(v, ϵ). Nevertheless, for completeness of exposition, we present its full description and estimate its computational complexity. We present the details in Algorithm 3. As in Test(v, ϵ) (Algorithm 1) above, we delete all the dominated pairs in all the sets $S(k)$. In addition, we delete ϵ-close pairs as follows:

(a) In each set $S(k)$, partition the interval $[0, UB]$ into $\lceil (UB/LB)\,(n/\epsilon)\rceil$ equal sub-intervals of size no greater than $\delta = \epsilon LB/n$;
(b) If, for a given sub-interval, there are multiple pairs from $S(k)$ for which the value of W falls into the subinterval, discard all excessive δ-close pairs, leaving only one representative pair in the subinterval, *i.e.*, the pair with the smallest P-coordinate (in this subinterval).
(c) Discard any pair $(W,\ P)$ with $W > UB$.

Algorithm 3. An ϵ-approximation algorithm $AA(LB,\ UB,\ \epsilon)$.

```
    input  : $UB$, $LB$, $\epsilon$, $\delta = \epsilon LB/n$.
    output: An $\epsilon$-approximation schedule such that the corresponding expected time is at most $(1+\epsilon)W^*$
 1  Step 1. [initialization]
 2  Set $S(1) = \{(0,0)\}$, $S(k) \leftarrow \emptyset$ for $k = 2,\ldots,n$. Step 1. [Generate $S(2)$ to $S(n)$]
 3  for $k = 1$ to $n$ do
 4      $G \leftarrow \emptyset$
 5      for each pair $(W,P) \in S(k-1)$ do
 6          $G \leftarrow G \cup \{(W + w(k), P + p(k))\}$
 7      end
 8      $S(k) \leftarrow$ merge $(S(k), G)$; during the merging eliminate the dominated pairs and $\delta$-close pairs
 9  end
10  Step 3. [Determine an approximate solution]
11  find min $W$ in $S(n)$, denote it by answer
12  Return answer as the $\epsilon$-approximation penalty and use backtracking to find the schedule itself.
```

Theorem 3.1. *The complexity of the entire three-stage FPTAS is $O(n^2/\epsilon)$.*

Proof. Since the length of the subinterval is $\delta = \epsilon LB/n$, we have $O(n(\frac{UB}{LB})(1/\epsilon))$ subintervals in the interval $[0,\ UB]$, and since $UB/LB \leq 2$, there are $O(n/\epsilon)$ sub-intervals in the interval $[LB,\ UB]$. Therefore, there are $O(n/\epsilon)$ representative pairs in any set G and $S(k)$. Constructing each G in line 3 requires $O(n/\epsilon)$ elementary operations because G is constructed from a single $S(k)$. Merging the sorted sets G and $S(k)$ in line 3, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is $O(n/\epsilon)$). In Step 2, we have $O(n)$ iterations.Thus, the total complexity of Algorithm AA is $O(n^2/\epsilon)$. Since Stage C has the same complexity as Stage B, and dominates Stage A of the algorithm, the overall complexity of the approximation algorithm is $O(n^2/\epsilon)$. □

4. Other Scheduling Problems

To illustrate the general approach we present two examples: the restricted shortest path problem and the two-machine just-in-time scheduling problem.

Example 1 ([7]). A directed graph is given with n vertex and m edges. Each edge has a length and a transition time. The objective is to find, for a given T value, a shortest path such that its total transition time is less than or equal to T. The complexity of the original Hassin's algorithm is $O(nm \log\log UB/LB + nm/\epsilon)$. It is easy to see, that the suggested technique permits to reduce the FPTAS complexity to $O(mn/\epsilon)$. The interested reader can find further details in Levner *et al.* [11] and Elalouf *et al.* [1].

Example 2 ([14])**.** A set of n independent, non-preemptive jobs is available for processing at time zero. All jobs have to be processed on two machines and follow the same route through the machines. Let d_j represent the due date of job J_j; w_j the gain (income) of completing job J_j just-in-time (that is, exactly at time d_j); and p_{ij} the processing time of job J_j on machine M_i for $i = 1, 2$ and $j = 1, \ldots, n$. The objective is to find a schedule with the maximum weighted number of jobs completed exactly at time d_j. Whereas the Shabtay–Bensousan's FPTAS has complexity of $O(n^4/\epsilon + n^4 \log UB/LB)$, the technique of the present paper reduces the latter to $O(n^3/\epsilon)$ (we refer to Elalouf *et al.* [2] for further details).

Concluding this section, let us describe general conditions for the scheduling problems for which the technique described above can be used to improve the complexity of approximation procedures. Specifically, these problems are characterized by the following properties:

Let S be a general scheduling problem with n jobs for which the following conditions hold:

(i) The optimal solution (OPT) can be found, *e.g.*, by dynamic programing, in $O(n^y UB)$, where UB is the upper bound on OPT, and y is a fixed number.

(ii) An initial upper bound on the solution UB and a lower bound LB such that $UB/LB < n$ can be found in $O(n^{y+1})$ time.

Theorem 4.1. *When using the above technique, the complexity of the FPTAS for problem S is $O(n^{y+1}/\epsilon)$ (where ϵ is the required accuracy of the solution).*

Proof. The FPTAS described above consists of three stages, each with complexity of $O(n^{y+1}/\epsilon)$:

According to assumption (i), Stage A is done in $O(n^{y+1})$. For Stage B, using the NARROW procedure described in Section 3 requires the Test procedure to be run at most 7 times; the complexity of the Test procedure is $O(n^{y+1})$, since we can replace UB with n. For Stage C, the approximation can be done in $O(n^{y+1}/\epsilon)$ by replacing UB with n/ϵ. The complexity of Stage C dominates that of Stages A and B, and the overall complexity of the approximation algorithm is $O(n^{y+1}/\epsilon)$. $\square$

It is easy to see that the sufficient conditions of the theorem hold for the scheduling and routing problems considered in Sections 3–4 of this paper.

5. Concluding remarks

Regarding the MIN-JSDP problem, since the complexity of the new FPTAS algorithm is $O(n^2/\epsilon)$, we immediately arrive at the conclusion that the question posed by Gens and Levner [6] has a positive answer: Indeed, it is possible to obtain an ϵ-approximation solution for MIN-JSDP whose complexity is the same as that of the solution for the maximization form of the problem. It is a challenging direction for future research to find new fully polynomial approximation algorithms with low polynomial degrees for wider classes of combinatorial problems.

References

[1] A. Elalouf, E. Levner and E. Cheng, Routing and dispatching of multiple mobile agents in integrated enterprises. *Int. J. Prod. Econ.* **145** (2013) 96–106.

[2] A. Elalouf, E. Levner and H. Tang. An improved FPTAS for maximizing the weighted number of just-in-time jobs in a two-machine flow shop problem. *J. Scheduling* **16** (2013) 429–435.

[3] F. Ergun, R. Sinha and L. Zhang, An improved FPTAS for restricted shortest path. *Inf. Proc. Lett.* **83** (2002) 287–291.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co. (1979).

[5] G.V. Gens and E.V. Levner, Approximation algorithm for some scheduling problems. *Eng. Cybernet. Soviet J. Comput. Syst. Sci.* **16** (1978) 38–46.

[6] G.V. Gens and E.V. Levner, Fast approximation algorithm for job sequencing with deadlines. *Discrete Appl. Math.* **3** (1981) 313–318.

[7] R. Hassin, Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.* **17** (1992) 36–42.

[8] I. Kacem, H. Kellerer and Y. Lanuel, Approximation algorithms for maximizing the weighted number of early jobs on a single machine with non-availability intervals. *J. Combin. Optim.* (2015) **30** 403–412.

[9] R.M. Karp, Reducibility among combinatorial problems. In *Complexity of Computer Computations*, edited by R.E. Miller and L.W. Thatcher. Plenum Press (1972) 85–104.

[10] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems. *Manage. Sci.* **16** (1969) 77–84.

[11] E. Levner, A. Elalouf and T.C.E. Cheng, An improved FPTAS for mobile agent routing with time constraints. *J. Universal Comput. Sci.* **17** (2011) 1854–1862.

[12] S. Sahni, Algorithms for scheduling independent tasks. *J. Assoc. Comput. Mach.* **23** (1976) 116–127.

[13] S. Sahni, General techniques for combinatorial approximation. *Oper. Res.* **25** (1977) 920–936.

[14] D. Shabtay and Y. Bensoussan, Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems. *J. Scheduling* **15** (2012) 39–47.

[15] W.E. Smith, Various optimizers for single-stage production. *Nav. Res. Logist. Quart.* **3** (1956) 59–66.

[16] A. Warburton, Approximation of Pareto optima in multiple objective, shortest path problems. *Oper. Res.* **35** (1987) 70–79.