Operations Research Letters 37 (2009) 365–367

Contents lists available at ScienceDirect

Operations Research Letters

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

The counting complexity of a simple scheduling problem

Gerardo Berbeglia *

Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, H3T 2A7, Canada

ARTICLE INFO

Article history:
Received 27 March 2009
Accepted 16 May 2009
Available online 28 May 2009

Keywords:
Scheduling
Counting
#P-complete
Computational complexity
Polynomial interpolation

ABSTRACT

Let T be a set of tasks. Each task has a non-negative processing time and a deadline. The problem of determining whether or not there is a schedule of the tasks in T such that a single machine can finish processing each of them before its deadline is polynomially solvable. We prove that counting the number of schedules satisfying this condition is #P-complete.

1. Introduction

We consider the *single machine scheduling problem with processing times and deadlines* which can be described as follows. Let $T = \{1, \ldots, n\} = [n]$ be a set of tasks. With each task are associated two non-negative integers, a processing time p_i and a deadline d_i. Let S_n be the symmetric group on $[n]$. A *schedule* of the tasks consists of a permutation $\sigma \in S_n$. A schedule is said to be *feasible* if $\sum_{i=1}^{j} p_{\sigma(i)} \leq d_{\sigma(j)}$ for all $1 \leq j \leq n$. In other words, σ is feasible if a single machine can process the tasks following the order of σ and is able to finish each of them before its deadline. Determining whether or not there exists a feasible schedule can be done by ordering tasks according to their respective deadline in increasing order. If such a schedule, generally called "earliest deadline first", is not feasible then no feasible schedule exists. Several extensions to this problem are also polynomial time solvable. Two examples are the problem of finding a schedule that minimizes the number of tardy tasks [1] and the problem of determining whether or not there exists a feasible schedule which respects a given set of precedence constraints between the tasks [2]. In this article we prove that the problem of counting the number of feasible schedules for this problem belongs to the class #P-complete, a class of hard counting problems introduced by Valiant [3]. This result shows that even for a very simple scheduling problem, heuristic or approximate algorithms for counting the number of solutions may be the only useful ones. The complexity of counting solutions in scheduling was also studied by [4] but in more complex multi-criteria problems.

In Section 2 we define the counting version of the problem studied, as well as other relevant definitions. The hardness result is presented in Section 3. An open question is given in Section 4.

2. Definitions

Let Σ be an alphabet, i.e., a finite set. A counting problem $f : \Sigma^* \to \mathbb{N}$ belongs to the class #P if there exists a non-deterministic polynomial-time Turing machine T such that for each $x \in \Sigma^*$, $f(x)$ is the number of accepting computations of T. A counting problem f is said to be *polynomially Turing reducible* to another counting problem g written $f \leq_T^t g$ if there exists an deterministic polynomial-time algorithm for f, given an oracle for g. We say that a counting problem f is *hard* for the class #P if $a \leq_P^t f$ for every counting problem $a \in$ #P. The class #P-complete consists of all the counting problems f that are hard for the class #P and that belong to #P.

We define formally the counting version of the *single machine scheduling problem with processing times and deadlines* and a counting version of the *Subset Sum Problem* which is #P-complete [5].

Name: *Single Machine Schedule Counting Problem* (1-SCP)
Input: $P = (p_1, \ldots, p_n) \in \mathbb{Z}^n_{\geq 0}$ and $D = (d_1, \ldots, d_n) \in \mathbb{Z}^n_{\geq 0}$.
Output: The number of permutations $\sigma \in S_n$ such that $\sum_{i=1}^{j} p_{\sigma(i)} \leq d_{\sigma(j)}$ for all $j \in [n]$.

Name: *#Subset Sum Problem* (#SSP)
Input: $\langle S, m\rangle$ with $S = (a_1, \ldots, a_n) \in \mathbb{N}^n$, $n \in \mathbb{N}$, and $m \in \mathbb{N}$.
Output: The number of subsets $T \subseteq [n]$ such that $\sum_{j \in T} a_j = m$.

Let $\sigma \in S_n$ and $\hat{\sigma} \in S_{n+k}$ $(k > 0)$ be two permutations. The permutation $\hat{\sigma}$ *k-extends* the permutation σ if for all $i, j \in [n]$, $\hat{\sigma}^{-1}(i) \leq \hat{\sigma}^{-1}(j)$ whenever $\sigma^{-1}(i) \leq \sigma^{-1}(j)$. In this case, $\hat{\sigma}$ is also called a *k-extension* of σ. The following fact was proved in [5].

* Tel.: +1 514 969 9823.
E-mail address: gerardo.berbeglia@hec.ca.

0167-6377/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2009.05.004

Fact 1. The number of $\hat{\sigma} \in S_{n+k}$ that k-extends a permutation $\sigma \in S_n$ is $\binom{n+k}{k} k!$.

3. Hardness of the 1-SCP

To show that the 1-SCP is hard, we will show that with the use of an oracle that solves the 1-SCP, it is possible to solve the counting version of the *Subset Sum Problem* (#SSP) in polynomial time using the technique of polynomial interpolation [6]. The reduction resembles the reduction used in [5] to prove that counting feasible solutions of the *Traveling Salesman Problem with Pickups and Deliveries*, called #TSPPD, is #P-complete. However, both reductions have specific intricacies, and we believe that it is not possible to construct a direct and simple reduction of the #TSPPD to the 1-SCP.

Given an instance $I = \langle P = (p_1, \ldots, p_n), D = (d_1, \ldots, d_n)\rangle$ of the 1-SCP and a number $k \in \mathbb{N}$, define a new 1-SCP instance I_k, called the *k-modified problem*, as follows. First, multiply each of the elements in the sequences P and D by $n+2$. Second, consider $\omega = k + (n+2)\sum_{i=1}^{n} p_i$ and add k new tasks by adding k 1's at the end of the sequence P and k ω's at the end of the sequence D. Thus, the modified instance I_k is $\langle P' = ((n+2)p_1, \ldots, (n+2)p_n, p_{n+1} \ldots p_{n+k}), D' = ((n+2)d_1, \ldots, (n+2)d_n, d_{n+1} \ldots d_{n+k})\rangle$ where $p_{n+r} = 1$ and $d_{n+r} = k + (n+2)\sum_{i=1}^{n} p_i$ for $r \in [k]$. See Fig. 1 for an example of a 1-SCP instance and one of its modified instances.

Consider an instance I of the 1-SCP, the modified instance I_k with $k < n+2$ and let $\sigma \in S_n$. Let $h = \max\{\ell \in [n] : \sum_{j=1}^{\ell} p_{\sigma(j)} = d_{\sigma(j)}\} \cup \{0\}$. The following lemma allow us to determine how many feasible permutations $\hat{\sigma} \in S_{n+k}$ for the instance I_k are k-extensions of the given $\sigma \in S_n$.

Lemma 2. *The number of feasible permutations $\hat{\sigma} \in S_{n+k}$ for I_k which are k-extensions of $\sigma \in S_n$ are 0 if σ is not feasible for I and $\binom{n+k-h}{k} k!$ otherwise.*

Before proving the lemma, we consider the example shown in Fig. 1. In the instance I, tasks 1 and 2 must be processed first and second respectively in order to respect their deadlines. Tasks 3 and 4, can be processed in any order, so there are two feasible solutions. In the modified instance I_3, we can see that tasks 1 and 2 must still be processed first and second respectively, the rest of the other 5 tasks can be processed in any order. Thus, there are a total of $5! = 120$ feasible solutions, 60 of them are extensions of one feasible schedule of I and the other 60 are extensions of the other.

We now proceed with the proof of Lemma 2.

Proof of Lemma 2. First observe that the new k tasks represented by the new k elements added in the sequences P' and D' will never be processed late since their deadlines are sufficiently large. The problem is therefore to determine when the previous tasks might be late under the new extension. It is clear that if σ is not a feasible permutation of I, no feasible extensions can be achieved in I_k. If in the original schedule σ of I no task has reached its deadline, i.e., $\sum_{j=1}^{\ell} p_{\sigma(j)} < d_{\sigma(\ell)}$ for all $\ell \in [n]$, and thus $h = 0$, then the new k elements of P' can be positioned anywhere. This is because even if all the new tasks are placed at the beginning, then

$$\begin{aligned} k + \sum_{j=1}^{\ell}(n+2)p_{\sigma(j)} &= k + (n+2)\sum_{j=1}^{\ell} p_{\sigma(j)} \\ &\le k + (n+2)(d_{\sigma(\ell)} - 1) \\ &= k + (n+2)d_{\sigma(\ell)} - (n+2) \\ &\le (n+2)d_{\sigma(\ell)}. \end{aligned}$$

Thus, the deadline for each task $\ell \in [n]$ in the instance I_k is respected. Using Fact 1, the total number of extensions is $\binom{n+k}{k} k!$. Finally, if, for some $\ell \in [n]$, $\sum_{j=1}^{\ell} p_{\sigma(j)} = d_{\sigma(j)}$, then $h > 0$ and in this case no new task can be positioned before the hth position since the task originally in the position h will be late, but they can be positioned at any place after this. This gives a total of $\binom{n+k-h}{k} k!$ extensions. □

The relation between I and I_k, shown by Lemma 2, gives rise to a series of linear equations shown in (1). Variables y_k represent the number of feasible permutations (schedules) $\hat{\sigma} \in S_{n+k}$ for the instance I_k with $1 \le k \le n+1$, x_j $(j > 0)$ is the number of feasible permutations $\sigma \in S_n$ of the instance I such that $j = \max\{\ell \in [n] : \sum_{i=1}^{\ell} p_{\sigma(i)} = d_{\sigma(i)}\}$, and x_0 is the number feasible permutations $\sigma \in S_n$ of the instance I such that $\sum_{i=1}^{h} p_{\sigma(i)} < d_{\sigma(h)}$ for all $h \in [n]$. We call $x = (x_0, \ldots, x_n)$ the *characteristic vector* of the instance I of the 1-SCP.

$$y_k = \sum_{j=0}^{n} \binom{n+k-j}{k} k! x_j \quad k = 1, \ldots, n+1. \tag{1}$$

Lemma 3. *The system of linear equation (1) is linearly independent.*

This lemma is proved in [5] (Lemma 5).

We now propose a transformation from an instance of the #SSP to an instance of the 1-SCP. Let $A = \langle S, m\rangle$, with $S = (a_1, \ldots, a_n)$, be an instance of the #SSP. Consider the instance of the 1-SCP such that $P = (p_1, \ldots, p_{n+1})$ and $D = (d_1, \ldots, d_{n+1})$ are sequences of length $n+1$ whose elements are defined as follows. For each i with $1 \le i \le n$, set $p_i = a_i$ and $d_i = 1 + \sum_{j=1}^{n} a_i$ and let $p_{n+1} = 0$ and $d_{n+1} = m$.

The following lemma, shows how to compute the number of solutions of an instance of the *Subset Sum Problem* given the characteristic vector of the instance of the 1-SCP obtained through the described transformation.

Lemma 4. *Let A be an instance of the #SSP and let I be the 1-SCP instance obtained from A as shown above. Let x be the characteristic vector of I. Then,*

$$\sum_{j=2}^{n+1} \frac{x_j}{(j-1)!(n+1-j)!}$$

is the number of solutions of A.

Proof. Consider first the 1-SCP instance I obtained from the transformation above. Except for the task represented by the numbers p_{n+1} and d_{n+1}, which from now on we call the task $n+1$, all the other tasks are finished strictly before their deadline for any given schedule $\sigma \in S_{n+1}$. This is because the total processing time of the $n+1$ tasks is $\sum_{j=1}^{n} a_i$, which is smaller than their deadline.

We now construct a function $\psi(\sigma)$ which takes any feasible permutation of the 1-SCP instance I and returns either the empty set or a subset $T \subseteq [n]$ such that $\sum_{a_i \in T} a_i = m$. Let $\sigma \in S_{n+1}$. We set $\psi(\sigma) = \emptyset$ whenever task $n+1$ is finished in the schedule σ strictly before its deadline, i.e., $\sum_{i=1}^{j} p_{\sigma(i)} < d_{\sigma(j)}$ for $1 \le j \le n+1$. If task $n+1$ is finished exactly at its deadline i.e., $\sum_{i=1}^{\ell} p_{\sigma(i)} = d_{\sigma(\ell)}$ for some $1 \le \ell \le n+1$, we set $\psi(\sigma) = T$ where $T = \{\sigma(i) | 1 \le i < \ell\}$. Note that ℓ is necessarily the position of the task $n+1$ in the schedule σ and that T is such that $\sum_{i \in T} a_i = m$. Denote by SS_t the number of subsets T of $[n]$ with cardinality t such that $\sum_{j \in T} a_j = m$. Clearly, #SSP(A)$= \sum_{t=1}^{n} SS_t$. Consider any $T \subseteq [n]$ with cardinality t such that $\sum_{j \in T} a_j = m$. We will now calculate $|\psi^{-1}(T)|$, that is, the number of feasible permutations ρ of I such that $\psi(\rho) = T$. Observe that a permutation ρ is such that $\psi(\rho) = T$ if and only if (i) $T = \{\rho(i) | 1 \le i \le t\}$ and (ii) $\sigma(t+1) = n+1$. Therefore, $|\rho^{-1}(T)|$,

$\omega = k + (n+2)\sum_{i=1}^{n} p_i = 3 + 30 = 33$

p_1	p_2	p_3	p_4
1	2	1	1

d_1	d_2	d_3	d_4
1	3	5	6

(a) Instance I of the 1-SCP. There are only 2 feasible solutions.

p_1	p_2	p_3	p_4	p_5	p_6	p_7
6	12	6	6	1	1	1

d_1	d_2	d_3	d_4	d_5	d_6	d_7
6	18	30	36	33	33	33

(b) The modified instance I_3.

Fig. 1. Example of an instance I and its modified problem I_3.

that is, the number feasible permutations ρ such that $\psi(\rho) = T$, is equal to the number of ways of permuting the elements in T, times the number of ways of permuting the elements in $[n] \setminus T$. This is equal to $|T|!(n-|T|)! = t!(n-t)!$. Now using the definition of the characteristic vector x, and observing that $x_1 = 0$ since $m > 0$, we obtain

$$\#\mathrm{SSP}(A) = \sum_{j=1}^{n} SS_j = \sum_{j=2}^{n+1} \frac{x_j}{(j-1)!(n+1-j)!}. \quad \square$$

We are now ready to prove the main result.

Theorem 5. *The #1-SCP is #P-complete.*

Proof. It is clear that the 1-SCP belongs to the class #P, since it is possible to construct a non-deterministic polynomial-time Turing machine that chooses a permutation $\sigma \in S_n$ and accepts it only when the associated schedule is feasible. We now prove that the 1-SCP is hard for the class #P by showing that giving an oracle that solves the 1-SCP, it is possible to solve the #SSP in polynomial time. Let A be an instance of the #SSP and let I be the 1-SCP instance obtained from A by the transformation already described. Using the oracle to solve the 1-SCP we can solve the k-modified problem of the instance I for each $1 \leq k \leq n+1$. By Lemma 3 we are able to compute the characteristic vector by solving the system of linear equations (1). Finally, by Lemma 4 we know that #SSP(A)$= \sum_{i=2}^{n+1} x_i/((i-1)!(n+1-i)!)$. This procedure can be performed in polynomial time and therefore the 1-SCP is hard for the class #P. $\square$

4. Open question

We leave open the question of whether the problem remains #P-complete in the case where the input numbers are written in unary notation. For this case our proof of #P-completeness is no longer valid since the #SSP can be solved in pseudo-polynomial time using dynamic programming.

Acknowledgements

The author would like to give thanks to Gilbert Laporte for reading the drafts of this article and for helping him to improve the quality of this work. This work was supported by the Canada Research Chair in Distribution Management and the Canada Research Chair in Logistics and Transportation. The support is gratefully acknowledged.

References

[1] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, Management Science 15 (1968) 102–109.
[2] E.L. Lawler, Optimal sequencing of a single machine subject to precedence constraints, Management Science 19 (1973) 544–546.
[3] L.G. Valiant, The complexity of computing the permanent, Theoretical Computer Science 8 (1979) 189–201.
[4] V. T'kindt, K. Bouibede-Hocine, C. Esswein, Counting and enumeration complexity with application to multicriteria scheduling, 4OR 3 (2005) 1–21.
[5] G. Berbeglia, G. Hahn, Counting feasible solutions of the traveling salesman problem with pickups and deliveries is #P-complete, Discrete Applied Mathematics 157 (2009) 2541–2547.
[6] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM Journal of Computing 8 (1979) 410–421.