Discrete Applied Mathematics 157 (2009) 2541–2547

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Counting feasible solutions of the traveling salesman problem with pickups and deliveries is #*P*-complete

Gerardo Berbeglia [a,*], Geňa Hahn [b]

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

[b] *Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montreal, H3C 3J7, Canada*

ARTICLE INFO

Article history:
Received 6 June 2007
Received in revised form 13 November 2008
Accepted 3 March 2009
Available online 5 April 2009

Keywords:
Traveling salesman problem with pickups and deliveries
Counting solutions
#*P*-complete
Complexity

ABSTRACT

Deciding whether or not a feasible solution to the *Traveling Salesman Problem with Pickups and Deliveries* (TSPPD) exists is polynomially solvable. We prove that counting the number of feasible solutions of the TSPPD is hard by showing that the problem is #*P*-complete.

1. Introduction

The problem of computing the number of solutions of certain problems has been studied in various areas such as combinatorics, artificial intelligence, statistical physics, operations research and constraint programming. In a constraint programming context, the number of solutions of a specific subproblem, or a relaxation of the problem, can be used for developing branching strategies [11]. Several authors counted feasible solutions in vehicle routing problems, for example [3,12,5]. Healy and Moll [5] presented an application of counting feasible solutions for solving the *Dial-a-Ride Problem*. Their technique consists of a local search method with a secondary metric. The additional metric is the number of feasible solutions neighbour to the current solution.

The performance of exact and heuristic algorithms for vehicle routing problem sometimes depends on the number of feasible solutions of the specific instance to solve. For illustration, consider the following three examples. Bourgeois et al. [2] developed three heuristics for solving the *Black and White Traveling Salesman Problem* and observed that one of the heuristics was able to find a feasible solution more frequently than the other two on tightly constrained instances. The other two heuristics, however, tended to produce better solutions than the first one when a feasible solution could be found. Aminu and Eglese [1] developed an exact constraint programming algorithm for the *Chinese Postman Problem with Time Windows*. They concluded that the constraint programming approach works much better on instances with relatively small number of feasible solutions. Mak and Boland [8] solved the *Asymmetric Traveling Salesman Problem with Replenishment Arcs* with an integer linear programming software using a polynomial size formulation and they compared the results with those obtained by column generation. They observed that their method is very efficient for problems with a large number of feasible solutions. These are hard for column generation because they contain too many columns. These three examples

* Corresponding author. Tel.: +1 514 343 6111; fax: +1 514 343 7121.
E-mail addresses: gerardo.berbeglia@hec.ca, gerardo@crt.umontreal.ca (G. Berbeglia), hahn@iro.umontreal.ca (G. Hahn).

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.03.003

suggest that advance knowledge, or at least an estimate, of the number of feasible solutions of a given instance of a vehicle routing problem can be used to determine which method is likely to perform best for solving the problem. In this article we prove that for the *Traveling Salesman Problem with Pickups and Deliveries* (TSPPD) a method for counting efficiently the exact number of solutions is unlikely to exist. More precisely, we prove that this problem is complete for #*P*, a class of counting problems introduced by Valiant [14] in the late 1970's. Consequently, heuristic methods or approximation algorithms for counting the number of solutions of the TSPPD can be useful.

Along the way to a proof of #*P*-completeness of the #TSPPD, we define a simple permutation problem (#PP) and prove that it is #*P*-complete. Both these results should be useful for proving that other counting problems are hard, specially those related to scheduling. An important ingredient in our proof that #PP is #*P*-hard is an interpolation which resembles Valiant's technique [15]. The combinatorial matrix used in our polynomial interpolation, however, is not the Vandermonde matrix used by Valiant.

The remainder of the article is organized as follows. In Section 2 we define the #TSPPD and the counting complexity classes #*P* and #*P*-complete. In Section 3 we prove some basic properties that will be used for proving our main results, including the #*P*-completeness of the counting version of the Subset Sum Problem. The main results are proven in Section 4. We close with open questions in Section 5.

2. Definitions

Let $G = (V, A)$ be a complete (that is, $(u, v) \in A$ and $(v, u) \in A$ for all $u \neq v \in V$) directed graph with vertex set $V = \{0, \ldots, n\} = \{0\} \cup [n]$ (with $[n] = \{1, 2, \ldots, n\}$ to simplify notation later). Vertex 0 represents the depot and the remaining vertices represent customers. Two non-negative integers, d_i and p_i, are associated with each customer i. The quantity d_i represents the number of units that need to be delivered from the depot to customer i, and the quantity p_i represents the number of units that must be brought from customer i back to the depot. Each arc (i, j) has a non-negative length, or cost, c_{ij}. A vehicle of capacity q is located at the depot and must be used to supply and collect the required number of units from customers. A *feasible solution* of the *Traveling Salesman Problem with Pickups and Deliveries* (TSPPD) is a simple directed cycle $P = (0, a_1, \ldots, a_n)$ in G such that $(a_1, \ldots, a_n)$ is a permutation of $[n]$ (the set of customers) and the vehicle load never exceeds q along the cycle, i.e. $\sum_{k=1}^{n} d_k + \sum_{j=1}^{i}(p_{a_j} - d_{a_j}) \leq q$ for all $i \in [n]$. The *cost* of a feasible solution $P = (0, a_1, \ldots, a_n)$ is equal to the sum of the costs of the edges of the cycle, i.e., $c_{0,a_1} + \sum_{i=1}^{n-1} c_{a_i,a_{i+1}} + c_{a_n,0}$. The TSPPD consists of determining a feasible solution in G with minimum cost. We denote by #TSPPD the problem of counting all feasible solutions of a given TSPPD instance.

Let $Q_D = \sum_{i=1}^{n} d_i$ and $Q_P = \sum_{i=1}^{n} p_i$ denote the total number of delivery and pickup units respectively. Let $\delta_i = p_i - d_i$ be the difference between the vehicle load after and before visiting customer i. Determining whether a TSPPD instance has a feasible solution is polynomial time solvable. One just has to check whether $q \geq \max\{Q_D, Q_P\}$. If yes, there is at least one feasible cycle consisting of visiting the customers in increasing order of δ_i. If not, no solution exists since the vehicle leaves the depot with load Q_D and returns to it with load Q_P. This is a corollary of a stronger result proved by Mosheiov [9].

We need to introduce some terminology and notation. Let S_n be the symmetric group on $[n]$. We say that a permutation $\hat{\sigma} \in S_{n+k}$, $(k > 0)$ *k-extends* a permutation $\sigma \in S_n$ if for all $i, j \in [n]$, $\hat{\sigma}(i) \leq \hat{\sigma}(j)$ whenever $\sigma(i) \leq \sigma(j)$. We call $\hat{\sigma}$ a *k-extension* of σ. The number of $\hat{\sigma} \in S_{n+k}$ that k-extends a permutation $\sigma \in S_n$ can be calculated as follows. Consider n objects ordered along a line. The number of ways of inserting additional k objects in the line, but without changing the relative positions of the original n objects, is $\binom{n+k}{k} k!$ and this is exactly the number of k-extensions.

The complexity classes #*P* and *NP* [10] can be defined by the introduction of an *NP*-relation on the class of decidable problems, perhaps best thought of as languages over the alphabet {0, 1} so that we have a precise notion of the length (size) of the input string (problem instance) x. We give here a definition adapted from Trevisan [13]. A relation R is an *NP*-relation, if there is a polynomial time decision algorithm A such that $(x, y) \in R \Leftrightarrow A(x, y) = 1$ and there is a polynomial p such that $(x, y) \in R \Rightarrow |y| \leq p(|x|)$ (think of a problem instance x and its polynomially verifiable solution y; see below). Given a relation R, we denote by #R the problem of counting for how many y, for a given x, is $(x, y) \in R$. The class #*P* is the class of problems of the form #R where R is a *NP*-relation. Any *NP*-relation R defines a language $L_R = \{x : \exists y, (x, y) \in R\}$. A language L belongs to the class *NP* if there is an *NP*-relation R such that $L = L_R$. Thus, for any *NP* language L_R, if $x \in L_R$, any y such that $(x, y) \in R$ can be seen as a "witness" of the membership of x in L. Given an *NP*-relation R, the problem #R can be seen as the problem of counting the number of witnesses for a given input of an *NP* problem. Given two relations A and B, the problem #A is said to be *polynomially Turing reducible* to the problem #B written $\#A \leq_T^p \#B$ if there is a polynomial time algorithm for #A, given an oracle for #B. A problem #C is *hard* for the class #*P* if $\#A \leq_T^p \#C$ for every problem $\#A \in \#P$. A problem is #*P-complete* if it is hard for the class #*P* and it belongs to #*P*. See [7] for more.

3. Tools

In this section we define the matrix $A(n)$ and we prove that it is invertible. This property is needed later in our polynomial interpolation. At the end of this section, we prove that the counting version of the well known *Subset Sum Problem* is #*P*-complete. This fact, which we will use for proving the hardness of the #TSPPD, is well known; curiously, no proof appears to be available in the literature which is the reason for providing it here (Theorem 4).

Lemma 1. *Let $n \geq 2$, $n \in \mathbb{N}$ and $j \in \mathbb{N}$, $1 \leq j \leq n$. Then,*

$$\sum_{i=0}^{n-1}(-1)^{n+i-1}\binom{n-1}{i-1}\binom{n+i-j}{i} = \begin{cases}\binom{2n-j}{n} & \text{if } j > 1\\ \binom{2n-1}{n} - 1 & \text{if } j = 1.\end{cases}$$

Proof. Assume n is even. Then

$$\sum_{i=1}^{n-1}(-1)^{n+i-1}\binom{n-1}{i-1}\binom{n+i-j}{i} = \sum_{i=1}^{n}(-1)^{i-1}\binom{n-1}{i-1}\binom{n+i-j}{n-j} - (-1)\binom{n-1}{n-1}\binom{2n-j}{n} \tag{1}$$

$$= -\binom{n-j+1}{1-j} + \binom{2n-j}{n} \tag{2}$$

$$= \begin{cases}\binom{2n-j}{n} & \text{if } j > 1\\ \binom{2n-1}{n} - 1 & \text{if } j = 1.\end{cases}$$

The equivalence $\sum_{i=1}^{n}(-1)^{i-1}\binom{n-1}{i-1}\binom{n+i-j}{i} = -\binom{n-j+1}{1-j}$ used between Eqs. (1) and (2) is a special case of a binomial identity proved in Graham et al. [4] (identity 5.24, page 169). The case where n is odd can be proved analogously, mutatis mutandis. □

Definition 2. Assume $n \geq 2$. Let $A(n)$ be the matrix defined as

$$A(n)_{ij} = \binom{n+i-j}{i} \quad i, j \in [n]. \tag{3}$$

Lemma 3. *The matrix $A(n)$ has full rank.*

Proof. We proceed by induction in n.

If $n = 2$, the matrix

$$A(2) = \begin{pmatrix}\binom{2}{1} & \binom{1}{1}\\ \binom{3}{2} & \binom{2}{2}\end{pmatrix} = \begin{pmatrix}2 & 1\\ 3 & 1\end{pmatrix}$$

has full rank.

Assume that $A(n-1)$ has full rank for $n > 2$. We first show that the $(n-1) \times (n-1)$ upper right submatrix of $A(n)$ is equal to $A(n-1)$. This is proved by following the definition of $A(n)$: $A(n)_{i(j+1)} = \binom{n+i-j-1}{i} = \binom{n-1+i-j}{i} = A(n-1)_{ij}$ for $i, j \in [n-1]$. Therefore, by the induction hypothesis, we can assume that the first $n-1$ rows of $A(n)$ are linearly independent. Let us denote the i-th row vector of $A(n)$ by $w^i = (w^i_1, \ldots, w^i_n)$, $i = 1, \ldots, n$. We prove that w^n cannot be written as a linear combination of $w^1, \ldots, w^{n-1}$. We proceed by contradiction. Assume there are real numbers $\alpha_1, \ldots, \alpha_{n-1}$ such that

$$w^n_j = \sum_{i=1}^{n-1}\alpha_i w^i_j \quad (j = 1, \ldots, n). \tag{4}$$

Let φ^i be the i-th row vector of $A(n-1)$. Using the fact that the $(n-1) \times (n-1)$ upper right submatrix of $A(n)$ is equal to $A(n-1)$, we have that for all $i \in [n-1]$ and $j \in \{2, \ldots, n\}$, $w^i_j = \varphi^i_{j-1}$. Thus, (4) is equivalent to (5) and (6):

$$w^n_1 = \sum_{i=1}^{n-1}\alpha_i w^i_1 \tag{5}$$

$$w^n_j = \sum_{i=1}^{n-1}\alpha_i \varphi^i_{j-1} \quad (j = 2, \ldots, n). \tag{6}$$

By the induction hypothesis, the set $\{\varphi_1, \ldots, \varphi_{n-1}\}$ is a basis of the vector space $\mathbb{R}^{n-1}$ and so the real numbers $\alpha_1, \ldots, \alpha_{n-1}$ satisfying (6) are unique (see, e.g., Hoffman and Kunze [6]). Applying the definition of the vectors w and φ and the definition of the matrix $A(n)$, Eqs. (5) and (6) can be written as follows:

$$\binom{2n-1}{n} = \sum_{i=1}^{n-1} \alpha_i \binom{n+i-1}{i} \tag{7}$$

$$\binom{2n-j}{n} = \sum_{i=1}^{n-1} \alpha_i \binom{n+i-j}{i} \quad (j = 2, \ldots, n). \tag{8}$$

If we let $\alpha_i = (-1)^{n+i-1}\binom{n-1}{i-1}$, Eq. (8) is satisfied, by Lemma 1. However, Eq. (7) is not since

$$\sum_{i=1}^{n-1}(-1)^{n+i-1}\binom{n-1}{i-1}\binom{n+i-1}{i} = \binom{2n-1}{n} - 1 \neq \binom{2n-1}{n}.$$

This proves that the unique numbers $\alpha_1, \ldots, \alpha_{n-1}$ satisfying (6) do not satisfy (7) and thus the assumption that the last row vector of $A(n)$ i.e., w_n, is a linear combination of the first $n-1$ row vectors of $A(n)$ leads to a contradiction. Therefore, $A(n)$ has full rank. □

We now define the counting version of the *Subset Sum Problem* and we prove that it is #*P*-complete by providing a polynomial reduction from #3-*SAT* which is parsimonious, i.e., it preserves the number of solutions. Our proof is based on known techniques for reducing the 3-*SAT* to the *Subset Sum Problem*.

Name: #*Subset Sum Problem* (#SSP).

Input: $\langle B, s, \pi\rangle$, where B is a finite set, $\pi : B \to \mathbb{N}$ is a function and $s \in \mathbb{N}$.

Output: The number of subsets $T \subseteq B$ such that $\sum_{i\in T} \pi(i) = s$.

Theorem 4. *The #Subset Sum Problem is #P-complete.*

Proof. It is well known that #3-*SAT* is #*P*-complete, see, for example, [14]. Since there exists a polynomial-time non-deterministic Turing machine that first chooses a subset $T \subseteq B$ and then accepts it only when $\sum_{i\in T} \pi(i) = s$, #SSP is in #*P*. We now describe a parsimonious polynomial reduction from #3-*SAT* to #SSP.

Let ϕ be an instance of #3-*SAT*, that is, a sentence in conjunctive normal form in which no clause has more than three terms. Let $X = \{x_i, : i \in [n]\}$ be the set of variables that occur in ϕ and let $C = \{c_j : j \in [m]\}$ be the set of clauses. For each variable x_i let $\hat{x}_{i0}, \hat{x}_{i1}$ be the corresponding literals, that is, $\neg x_i$ and x_i, respectively. Recall that the Kroenecker delta is (for our purposes) a function $\delta : \mathbb{N} \times \mathbb{N} \longrightarrow \{0, 1\}$ defined by $\delta(i, j) = 1$ if and only if $i = j$. Define $\rho : [n] \times [m] \times \{0, 1\} \longrightarrow \{0, 1\}$ by $\rho(i, j, \ell) = 1$ if and only if the literal $\hat{x}_{i\ell}$ occurs in the clause c_j.

For each literal $\hat{x}_{i\ell}$ define an integer $v_{i\ell}$ of $n + m$ decimal digits by setting the kth digit $v_{i\ell}^k = \delta(i, k)$ for $k \in [n]$ and $v_{i\ell}^k = \rho(i, k, \ell)$ for $k \in \{n + j : j \in [m]\}$. For each clause c_j and $j \in [m]$ define two new literals $\hat{c}_{j1}$ and $\hat{c}_{j2}$; and two integers $c_{j\ell}$ of $n + m$ decimal digits with $\ell = 1, 2$ by $c_{j\ell}^{n+j} = \ell$ and $c_{j\ell}^k = 0$ otherwise. Let $B = \{\hat{x}_{i\ell} : i \in [n],\ \ell \in \{1, 0\}\} \cup \{\hat{c}_{i\ell} : i \in [m],\ \ell \in \{1, 2\}\}$. For $y \in B$ let $\pi(y)$ (also written π_y) be the corresponding integer. That is, if $y = \hat{x}_{i\ell}$, $\pi(y) = v_{i\ell}$ and if $y = \hat{c}_{i\ell}$ then $\pi(y) = c_{i\ell}$. Let $s = s_1 \ldots s_n s_{n+1} \ldots s_{n+m}$ be the $n + m$-digit integer with $s_i = 1$ for $i \in [n]$ and $s_{n+j} = 4$ for $j \in [m]$.

Observe first that for any $S \subseteq B$, we have $\sum_{y\in S} \pi_y^k \leq 6$ for $k \in \{n + j : j \in [m]\}$ while $\sum_{y\in S} \pi_y^k \leq 2$ for $k \in [n]$. Thus $\sum_{y\in S} \pi_y$ can be obtained digit by digit. Let $\tau : X \longrightarrow \{0, 1\}$ be a truth assignment with 0 indicating that the variable is false, 1 that it is true. This partitions X into $X_0 \cup X_1$ with $X_\ell = \{x \in X : \tau(x) = \ell\}$, $\ell = 0, 1$. Let $|c_j|_\tau$ be the number of literals in c_j set to 1 by τ; this is a number in $\{0, 1, 2, 3\}$ in general and in $\{1, 2, 3\}$ if c_j is satisfied. Let $S_\tau = \{\hat{x}_{i\ell} \in B : \ell \in \{0, 1\}, x_i \in X_\ell\} \cup \{\hat{c}_{j1} : j \in [m], |c_j|_\tau = 3\} \cup \{\hat{c}_{j2} : j \in [m], |c_j|_\tau = 2\} \cup \{\hat{c}_{ji} : j \in [m], i \in \{0, 1\}, |c_j|_\tau = 1\}$.

It is easy to see that $\sum_{y\in S_\tau} \pi_y = s$ when τ satisfies ϕ. Indeed, exactly one of v_{i0}, v_{i1} is chosen by τ whenever τ satisfies ϕ, while the choice of the $c_{i\ell}$ guarantees that $\sum_{k=n+1}^{n+m} \pi_y^k = 4$. It is equally easy to see that the mapping that sends τ to S_τ is injective and surjective for it is clear how to obtain a truth assignment from a set T summing up to s and why for different sets different truth assignments arise. Thus the transformation induces a bijection between the set of truth assignments satisfying ϕ and subsets T of B whose values sum up to s, $\sum_{i\in T} \pi(i) = s$.

This proves that there is a polynomial-time reduction from #3-*SAT* to #SSP which is parsimonious. □

In the sequel, we will assume, without loss of generality, that in the #SSP, $B = [n]$ and that $\pi(i) = a_i$ for $i \in [n]$. Therefore, the input of the #SSP can be thought to be $\langle S, m\rangle$ where S is a vector $S = (a_1, \ldots, a_n) \in \mathbb{N}^n$ and $m \in \mathbb{N}$.

4. Main results

The most important ingredient in the #*P*-completeness proof of the #TSPPD, namely the #*P*-completeness of the *Permutation Problem* defined below, is of independent interest. It requires some preparation and a few lemmas.

4.1. Hardness of the Permutation Problem

We define the *Permutation Problem* (#PP) as follows.

Name: *Permutation Problem* (#PP).

Input: $\langle S, m\rangle$ with $S = (a_1, \ldots, a_n) \in \mathbb{Z}^n$, $n \in \mathbb{N}$, and $m \in \mathbb{Z}_{\geq 0}$.

Output: The number of permutations $\sigma \in S_n$ such that $\sum_{j=1}^{i} a_{\sigma(j)} \leq m$ for all $i \in [n]$.

Given a sequence $S = (a_1, \ldots, a_n)$ of integers and an integer m, a permutation $\sigma \in S_n$ is said to be (S, m)-*valid* (or, simply, *valid*) if $\sum_{j=1}^{i} a_{\sigma(j)} \leq m$ for all $i \in [n]$. Thus, the output of the #PP when given an instance $I = \langle S, m\rangle$ is the number of (S, m)-*valid* permutations.

We will prove that the #*P*-complete problem #SSP is polynomially Turing reducible to the #PP. The idea is the following. Given an instance of the #SSP, we first construct an instance I of the #PP. Then we define a series of #PP instances associated to I, each called a *modified problem* of I. We prove that solving each of the modified instances of I allows us to obtain an integer vector containing information about I, called the *characteristic vector*, and that the number of solutions to the original instance of #SSP can be computed using this vector. We begin by defining the *k-modified problem*.

Given $k \in \mathbb{N}$ and an instance $I = \langle S, m\rangle$ of the #PP, we define an instance I_k, called the *k-modified problem*, as follows. If $S = (a_1, \ldots, a_n)$, we multiply the number m and each of the elements in the sequence S by $n + 2$ and we add k 1's to the original sequence. Thus, the modified instance I_k of #PP is $\langle S', (n + 2)m\rangle$ where $S' = ((n + 2)a_1, (n + 2)a_2, \ldots, (n + 2)a_n, b_1, \ldots, b_k)$ and $b_r = 1$ for $r \in [k]$.

Let $\sigma \in S_n$ be a permutation of S and assume $k < n + 2$. How many $(S', (n + 2)m)$-valid permutations $\hat{\sigma} \in S_{n+k}$ are k-extensions of a given $\sigma \in S_n$? If σ is not (S, m)-valid, no valid extensions can be achieved. If $\sum_{j=1}^{\ell} a_{\sigma(j)} < m$, for all $\ell \in [n]$, each $b_r = 1, r \in [k]$, can be positioned anywhere. This is because $\sum_{j=1}^{\ell}(n + 2)a_{\sigma(j)} + k \leq (n + 2)(m - 1) + k \leq (n + 2)m$. This gives a total of $\binom{n+k}{k} k!$ extensions. Finally, if, for some $\ell \in [n]$, $\sum_{j=1}^{\ell} a_{\sigma(j)} = m$, let $h = \max\{\ell \in [n] : \sum_{j=1}^{\ell} a_{\sigma(j)} = m\}$. In this case each $b_r = 1, r \in [k]$ cannot be positioned before the hth position since then the sum would exceed $(n+2)m$, but they can be positioned any place after. This gives a total of $\binom{n+k-h}{k} k!$ extensions. This relation between I and I_k gives rise to a series of linear equations shown in (9). Variables y_k represent the number of $(S', (n + 2)m)$-valid permutations $\hat{\sigma} \in S_{n+k}$, x_j $(j > 0)$ is the number of (S, m)-valid permutations $\sigma \in S_n$ such that $j = \max\{\ell \in [n] : \sum_{i=1}^{\ell} a_{\sigma(i)} = m\}$, and x_0 is the number of (S, m)-valid permutations $\sigma \in S_n$ such that $\sum_{i=1}^{h} a_{\sigma(i)} < m$ for all $h \in [n]$. The vector $x = (x_0, \ldots, x_n)$ is called the *characteristic vector* of the instance I of the permutation problem.

$$y_k = \sum_{j=0}^{n} \binom{n + k - j}{k} k! x_j \quad k = 1, \ldots, n + 1. \tag{9}$$

The above equation is trivially equivalent to

$$y_k = \sum_{j=1}^{n+1} \binom{(n + 1) + k - j}{k} k! x_{j-1} \quad k = 1, \ldots, n + 1 \tag{10}$$

which leads to the following lemma.

Lemma 5. *The system of linear equations (9) is linearly independent.*

Proof. Observe that the matrix associated with the system of linear equations (10) is equal to the product of the matrix $A(n + 1)$ (see Definition 2) by a diagonal matrix whose diagonal values are non-zero. Thus, the linear independence of the system follows from Lemma 3 which proves that the matrix $A(n + 1)$ has full rank. □

For the next result we need a simple lemma.

Lemma 6. *Let $S = (a_1, \ldots, a_n) \in \mathbb{N}^n$, $n \in \mathbb{N}$, $m \in \mathbb{N}$ and $\rho \in S_{n+1}$. Let also $a_{n+1} = -1 - \sum_{i=1}^{n} a_i$. Then there is at most one $t \in [n + 1]$ such that $\sum_{j=1}^{t} a_{\rho(j)} = m$.*

Proof. Suppose that there are $s \in [n + 1]$ and $t \in [n + 1]$ with $s < t$ such that $\sum_{j=1}^{t} a_{\rho(j)} = \sum_{j=1}^{s} a_{\rho(j)} = m$. Then $m = \sum_{j=1}^{t} a_{\rho(j)} = \sum_{j=1}^{s} a_{\rho(j)} + \sum_{j=s+1}^{t} a_{\rho(j)} = m + \sum_{j=s+1}^{t} a_{\rho(j)}$, that is, $\sum_{j=s+1}^{t} a_{\rho(j)} = 0$. But $\sum_{j=s+1}^{t} a_{\rho(j)} < \sum_{j=1}^{n+1} a_{\rho(j)} = -1$ if $a_{n+1} \in \{a_{\rho[s+1]}, \ldots, a_{\rho[t]}\}$ and $\sum_{j=s+1}^{t} a_{\rho(j)} > 0$ otherwise, since $a_i \in \mathbb{N}$ for $i \in [n]$. □

Lemma 7. *Let $I = \langle S, m\rangle$, $S = (a_1, \ldots, a_n)$, be an instance of #SSP and set $a_{n+1} = -1 - \sum_{i=1}^{n} a_i$. Consider the instance of #PP, $I' = \langle S', m\rangle$ with $S' = (a_1, \ldots, a_n, a_{n+1})$. Let x be the characteristic vector of I'. Then*

$$\sum_{t=1}^{n} \frac{x_t}{t!(n - t)!}$$

is the number of solutions of I.

Proof. We will construct a function ψ from the (S', m)-valid permutations to the subsets of $\{1, \ldots, n\}$. Let $\rho \in S_{n+1}$ be an (S', m)-valid permutation. If there is a $t \in [n+1]$ (unique by Lemma 6) such that $\sum_{j=1}^{t} a_{\rho(j)} = m$, set $\psi(\rho) = \{\rho(j) : j \in [t]\}$. Otherwise, i.e., if $\sum_{j=1}^{i} a_{\rho(j)} < m$ for all $i \in [n+1]$, we set $\psi(\rho) = \emptyset$.

To see that $\psi(\rho) \subseteq [n]$, it is sufficient to prove that $n+1 \notin \psi(\rho)$. If $n+1 \in \psi(\rho)$, there exists $i \in [t]$ such that $\rho(i) = n+1$ and thus $m = \sum_{j=1}^{t} a_{\rho(j)} = \sum_{j=1, j\neq i}^{t} a_{\rho(j)} + a_{n+1} \leq -1$(since $a_j > 0$ for $j \in [n]$) $< m$, which is absurd.

Denote by SSP_t the number of subsets $T \subseteq [n]$ with cardinality t such that $\sum_{j\in T} a_j = m$. Clearly, $\#\mathrm{SSP}(I) = \sum_{t=1}^{n} \mathrm{SSP}_t$. Suppose $T \subseteq [n]$ has cardinality t and $\sum_{j\in T} a_j = m$. Any $\rho \in S_{n+1}$ such that $\psi(\rho) = T$ must satisfy the following conditions:

- $\{\rho(j) : j \in [t]\} = T$ (Note that $T \neq \emptyset$ since $m > 0$ and therefore $t > 0$);
- $\rho(t+1) = n+1$.

Indeed, if $\rho(t+1) \neq n+1$, then $\sum_{j=1}^{t+1} a_{\rho(j)} = a_{\rho(t+1)} + \sum_{j=1}^{t} a_{\rho(j)} > m$ and so ρ is not an (S', m)-valid permutation. Observe that any permutation ρ satisfying these two conditions is (S', m)-valid since $\sum_{j=1}^{h} a_{\rho(j)} \leq m$ for any $h \in [n+1]$. Thus, $|\psi^{-1}(T)|$, that is, the number of (S', m)-valid 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 $|T|!(n-|T|)! = t!(n-t)!$. Using the fact that x_t is the number of (S, m)-valid permutations $\sigma \in S_n$ such that $t = \max\{\ell \in [n] : \sum_{i=1}^{\ell} a_{\rho(i)} = m\}$, we can conclude that the number of solutions to $I = \langle S, m\rangle$ is $\sum_{t=1}^{n} \mathrm{SSP}_t = \sum_{t=1}^{n} x_t/(t!(n-t)!)$. □

Theorem 8. *#PP is #P-complete.*

Proof. The #PP clearly belongs to the class *#P*, since a polynomial-time non-deterministic Turing machine M can choose a permutation $\rho \in S_n$ and accept it only when the permutation is valid. For completeness, let $I = \langle S, m\rangle$, $S = (a_1, \ldots, a_n)$, be an instance of #SSP. Construct an instance $I' = \langle S', m\rangle$ of #PP as in the preceding lemma, that is, let $a_{n+1} = -1 - \sum_{i=1}^{n} a_i$ and let $S' = (a_1, \ldots, a_n, a_{n+1})$. Given an oracle for solving the permutation problem we can obtain the characteristic vector x for I' in polynomial time as follows. Using the oracle, we solve the k-modified problem of $I'n+1$ times, once for each k between 1 and $n+1$. By Lemma 5, we compute the characteristic vector of I' by solving the system of linear equations (9). Finally, by Lemma 7, we know that the number of solutions to $I = \langle S, m\rangle$ is $\sum_{t=1}^{n} x_t/(t!(n-t)!)$. The fact that this procedure can be performed in polynomial time proves that $\#\mathrm{SSP} \leq_T^P \#\mathrm{PP}$. □

4.2. Hardness of the #TSPPD

Using now the fact that the #PP is #*P*-complete, we prove that the #TSPPD is #*P*-complete.

The #TSPPD can be defined in a compact way as follows.

Name: #TSPPD.

Input: $\langle D, P, q\rangle$ with $q \in \mathbb{N}$, $D = (d_1, \ldots, d_n) \in \mathbb{Z}_{\geq 0}^n$, $P = (p_1, \ldots, p_n) \in \mathbb{Z}_{\geq 0}^n$ and $n \in \mathbb{N}$.

Output: the number of permutations $\sigma \in S_n$ such that $\sum_{j=1}^{n} d_j + \sum_{j=1}^{i} (p_{\sigma(j)} - d_{\sigma(j)}) \leq q$ for all $i \in [n]$.

The sequences D and P represent the deliveries and pickup demands respectively and q represents the capacity of the vehicle. The property required for the permutations is satisfied only by the set of permutations that give a feasible cycle (i.e., the capacity constraint is respected). A permutation that represents a feasible cycle is said to be *feasible*.

Theorem 9. *#TSPPD is #P-complete.*

Proof. First note that #TSPPD belongs to the class *#P*. A polynomial-time non-deterministic Turing machine M can choose a permutation $\rho \in S_n$ and accept it if such permutation produces a feasible cycle. We will now prove that $\#\mathrm{PP} \leq_T^P \#\mathrm{TSPPD}$.

Let $\langle S, m\rangle$ be an instance of #PP with $S = (a_1, \ldots, a_n)$. Let $H = -\sum_{i=1, a_i \leq 0}^{n} a_i$. We define an instance $I = \langle P, D, q\rangle$ of the #TSPPD with $D = (d_1, \ldots, d_n)$ and $P = (p_1, \ldots, p_n)$ by setting $d_i = -\min\{a_i, 0\}$, $p_i = \max\{a_i, 0\}$ for all $i \in [n]$, and $q = H + m$. Clearly, instance I can be constructed in polynomial time. See Fig. 1 for an example. Note first that the #PP on the instance $\langle S, m\rangle$, counts the number of (S, m)-valid permutations $\sigma \in S_n$. On the other hand, the #TSPPD on the constructed instance $I = \langle P, D, q\rangle$, counts the number feasible permutations, i.e., permutations $\sigma \in S_n$ such that $\sum_{j=1}^{n} d_j + \sum_{j=1}^{i} (p_{\sigma(j)} - d_{\sigma(j)}) \leq q$ for all $i \in [n]$.

A permutation $\sigma \in S_n$ is (S, m)-valid if for each $j \in [n]$,

$$\sum_{i=1}^{j} a_{\sigma(i)} \leq m$$

$$\Leftrightarrow H + \sum_{i=1}^{j} (\max\{a_{\sigma(i)}, 0\} + \min\{a_{\sigma(i)}, 0\}) \leq H + m$$

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
2	3	-4	-2	6	1	0	-2

$m = 4$

(a) Instance I of the #PP.

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
2	3	0	0	6	1	0	0

d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
0	0	4	2	0	0	0	2

$q = 4 + 8 = 12$

(b) Instance of #TSPPD obtained from I.

Fig. 1. Example of a transformation from the #PP to the #TSPPD.

$$\Leftrightarrow H + \sum_{i=1}^{j} (p_{\sigma(i)} - d_{\sigma(i)}) \leq q$$

$$\Leftrightarrow \sum_{i=1}^{n} d_i + \sum_{i=1}^{j} (p_{\sigma(i)} - d_{\sigma(i)}) \leq q.$$

Thus, a permutation $\sigma \in S_n$ is (S, m)-valid if and only if σ is feasible for the #TSPPD instance I. This shows there is a bijection between the set of (S, m)-valid permutations and the set of feasible permutations in the #TSPPD instance I, proving the theorem. □

5. Open questions

We leave two open questions related to the problem. Given that the #TSPPD is #P-complete, we can ask ourselves whether or not there exists a *fully polynomial randomized approximation scheme* for the problem. Another question is whether or not the #TSPPD remains hard even if the input numbers (i.e., capacity of the vehicle, deliveries and pickups amounts) are written in unary notation.

Acknowledgements

We are grateful to Mark Jerrum for suggesting the modified problem as a way to prove the #P-completeness of the #TSPPD as well as other valuable comments. We also give thanks to Gilbert Laporte for reading the drafts and helping us improve the quality of this work. Thanks are also due to Leslie Valiant for quickly answering our technical query.

This work was partially supported by the Ministère de l'Éducation, du Loisir et du Sport du Québec and by a grant from the National Science and engineering research council of Canada (NSERC).

References

[1] U.F. Aminu, R.W. Eglese, A constraint programming approach to the Chinese postman problem with time windows, Computers and Operations Research 33 (2006) 3423–3431.
[2] M. Bourgeois, G. Laporte, F. Semet, Heuristics for the black and white traveling salesman problem, Computers and Operations Research 30 (2003) 75–85.
[3] J.-F. Cordeau, M. Iori, G. Laporte, J.J. Salazar González, A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading, Networks (2008) (in press).
[4] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics, Addison-Wesley, New York, 1989.
[5] P. Healy, R. Moll, A new extension of local search applied to the dial-a-ride problem, Transportation Science 14 (1980) 130–154.
[6] K. Hoffman, R. Kunze, Linear Algebra, Prentice Hall, NJ, 1971.
[7] M. Jerrum, Counting, Sampling and Integrating: Algorithms and Complexity, in: Lectures in Mathematics - ETH Zürich, Birkhäuser Verlag, Basel, 2003.
[8] V. Mak, N. Boland, Polyhedral results and exact algorithms for the asymmetric travelling salesman problem with replenishment arcs, Discrete Applied Mathematics 155 (2007) 2093–2110.
[9] G. Mosheiov, The traveling salesman problem with pick-up and delivery, European Journal of Operational Research 79 (1994) 299–310.
[10] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1995.
[11] G. Pesant, Counting solutions of CSPs: A Structural Approach, International Joint Conferences on Artificial Intelligence, IJCAI-05: 2005, pp. 260–265.
[12] K.S. Ruland, E.Y. Rodin, The pickup and delivery problem: Faces and branch-and-cut algorithm, Computers and Mathematics with Applications 33 (1997) 1–13.
[13] L. Trevisan, Computational Theory, Lecture 6, available at http://www.cs.berkeley.edu/~luca/cs278-08/lecture06.pdf, 2008.
[14] L.G. Valiant, The complexity of computing the permanent, Theoretical Computer Science 8 (1979) 189–201.
[15] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM Journal of Computing 8 (1979) 410–421.