Combinatorial Optimization and Computations in the Ring of Polynomials

ALEXANDER I. BARVINOK *

Version of March 12, 1993

Abstract. We reduce a problem of Combinatorial Otimization to a computational problem in the ring of polynomials. As an application, we find new polynomially solvable classes of the Traveling Salesman Problem and the Weighted 3-Packing Problem. In particular, we show that for any fixed norm in a Euclidean space and for any fixed relative error one can compute the length of a longest Hamiltonian contour joining given points in polynomial time. We also design faster PSPACE algorithms for testing Hamiltonian graphs, for the 3-Packing Problem, and the Vertex Disjoint Paths Problem. We prove that the permanent of a matrix can be computed in polynomial time if we fix the rank of the matrix.

1. Introduction

In this paper we consider further applications of exponential sums (see, for example, [2]) to the problems of Combinatorial Optimization. We design polynomial time algorithms for some particular cases of the Traveling Salesman Problem and of the Weighted 3-Packing Problem and faster PSPACE algorithms for unweighted versions of these problems as well as for the Vertex Disjoint Paths Problem. We briefly describe our approach below.

Let $\mathbb{R}^n$ be the Euclidean space equipped with the standard inner product $\langle \cdot, \cdot \rangle$. Assume that we have a (generally speaking, very large) set S of integral vectors in $\mathbb{N}^n \subset \mathbb{R}^n$. Usually, these vectors represent characteristic functions of certain combinatorially interesting subsets of the set $\{1, \ldots, n\}$. Then a general problem of Combinatorial Optimization can be posed as follows:

(1.1) Combinatorial Optimization Problem. Given vector $l = (l_1, \ldots, l_n) \in \mathbb{R}^n$, find

$$w(l) = \min\{\langle l, m \rangle : \quad m \in S\}.$$

We are interested in the computational complexity of this problem. Formally replacing in (1.1) "min" by "$\sum$" and "$+$" by "$\cdot$" we come to the following problem:

* Supported by KTH, Stockholm and DIMACS.

(1.2) Computational Problem of Algebra. Compute the value of the polynomial

$$P(x) = \sum_{m=(m_1,\ldots,m_n)\in S} x_1^{m_1}\cdots x_n^{m_n}$$

in a given point $x = L \in \mathbb{R}^n$.

The reduction of (1.1) to (1.2) is based on the following observation:

(1.3) Reduction. For $t > 0$ let us define the vector $L(t) \in \mathbb{R}^n$ by the formula $L_i(t) = \exp\{-t \cdot l_i\}$ for $i = 1, \ldots, n$. Then

$$P\big(L(t)\big) \geq \exp\{-t \cdot w(l)\} \geq \frac{1}{|S|} \cdot P\big(L(t)\big),$$

where $|S|$ denotes the cardinality of the set S. Thus the value

$$w_t = -t^{-1} \cdot \log P\big(L(t)\big)$$

for $t \geq \epsilon^{-1} \cdot \log |S|$ approximates the optimal value $w(l)$ in (1.1) with error not more than $\epsilon > 0$.

It turns out that for many important and computationally hard problems Problem 1.2 is much easier than Problem 1.1, since the polynomial P satisfies certain algebraic identities which allow to compute it. Moreover, we can find an algebraic condition for L which ensures polynomial computability of $P(L)$. Then we retranslate this algebraic condition formally replacing "$\cdot$" by "$+$" and "$\sum$" by "min" into a combinatorial condition for l which ensures polynomial solvability of Probem 1.1.

In Section 2 we introduce our main technical tool for solving Problem 1.2, namely, an invariant inner product in the ring of polynomials. As an example, we prove that the permanent of a matrix of a fixed rank is polynomially computable.

In Section 3 we introduce "Hamiltonian permanent" of a square matrix whose distinction from the usual permanent is that we take the sum over all Hamiltonian cycles. We prove that the Hamiltonian permanent of a matrix can be computed in polynomial time if we fix the rank of the matrix. We also design a faster PSPACE algorithm for finding a Hamiltonian contour in a directed graph.

In Section 4 we deal with the Traveling Salesman Problem. We define "combinatorial rank" of a matrix, formally replacing "$\sum$" by "min" and "$\cdot$" by "$+$" in the usual definition of the rank. We prove that if the combinatorial rank of the matrix of weights is fixed, then one can compute the smallest weight of a Hamiltonian contour in pseudopolynomial time.

In Section 5 we consider a natural example of a matrix with small combinatorial rank arising as the matrix of pairwise distances between points in the Euclidean space. We prove that for any fixed norm in the Euclidean space $\mathbb{R}^d$ and for any fixed number $\delta > 0$ one can compute in polynomial time the length of a longest Hamiltonian contour joining given points in $\mathbb{R}^d$ with relative error less than δ.

In Sections 6-7 we further illustrate our technique applying it to the Weighted 3-Packing Problem and to the Vertex Disjoint Paths Problem. We obtain results which are similar to the results of Sections 3-4.

In Section 8 we discuss some open problems and, in particular, a possibility of a direct combinatorial algorithm in Problem 1.1 without translation into Problem 1.2 and then retranslation back.

We use two standard computational models (see [1,3]). In the first ("arithmetic") model we perform arithmetic operations (addition, subtraction, multiplication, division, and comparison) with real numbers. The time complexity of an algorithm is the number of arithmetic operations. In the second ("bit") model we perform bitwise operations with rational numbers and the time complexity is the number of these operations. The space complexity is defined in the standard way. It is always clear from the context which computational model is used. In Sections 2-3 the main accent is on the arithmetic model whereas in Sections 4-5 we mostly use bitwise computations.

2. Inner Product in the Ring of Polynomials

Let $\mathcal{P}_n = \mathbb{R}[x_1, \ldots, x_n]$ be the ring of polynomials in variables $x_1, \ldots, x_n$ over the reals. We consider the following inner product $\langle \cdot, \cdot \rangle$ defined on the real linear space $\mathcal{P}_n$:

$$\langle P, Q \rangle = P\left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right) Q(x_1, \ldots, x_n)\Big|_{x_1 = \ldots = x_n = 0} \tag{2.1}$$

for $P, Q \in \mathcal{P}_n$.

In other words,

$$\langle P, Q \rangle = \sum_{\alpha} \alpha_1! \cdots \alpha_n! \cdot c_\alpha \cdot b_\alpha, \tag{2.2}$$

if

$$P = \sum_{\alpha} c_\alpha \cdot x^\alpha, \quad \text{and} \quad Q = \sum_{\alpha} b_\alpha \cdot x^\alpha,$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $x^\alpha = x^{\alpha_1} \cdots x^{\alpha_n}$. It is easy to see now that $\langle \cdot, \cdot \rangle$ is indeed an inner product. We give below the third equivalent definition.

Slightly abusing notation, let us denote by $\langle \cdot, \cdot \rangle$ the standard inner product on $\mathbb{R}^n$ (no confusion should arise here since it is always clear from the context which linear space is considered). So $\langle x, y \rangle = x_1 \cdot y_1 + \ldots + x_n \cdot y_n$ for $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$. Then

$$Q(x) = Q\left(\frac{\partial}{\partial y_1}, \ldots, \frac{\partial}{\partial y_n}\right) \exp\{\langle x, y \rangle\}\Big|_{y=0}.$$

Therefore by (2.1) we get

$$\langle P, Q \rangle = P\left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right) Q\left(\frac{\partial}{\partial y_1}, \ldots, \frac{\partial}{\partial y_n}\right) \exp\{\langle x, y \rangle\}\Big|_{x=y=0}. \tag{2.3}$$

Let $G \in \text{End } \mathbb{R}^n$ be a linear operator on $\mathbb{R}^n$. By G^* we denote the operator which is conjugate to G. Let us define an action of End $\mathbb{R}^n$ on $\mathcal{P}_n$ by the formula

$$G(P)(x) = P(G^*(x))$$

for all $x \in \mathbb{R}^n$, $P \in \mathcal{P}_n$, and $G \in \text{End } \mathbb{R}^n$. We note that $G(P+Q) = G(P) + G(Q)$ and $G(P \cdot Q) = G(P) \cdot G(Q)$ for all $P, Q \in \mathcal{P}_n$ and $G \in \text{End } \mathbb{R}^n$. The following result is well-known, but for the sake of completeness we give its proof here.

(2.4) Theorem. *We have*

$$\langle G(P), Q \rangle = \langle P, G^*(Q) \rangle$$

for all $P, Q \in \mathcal{P}_n$ and all $G \in \mathbb{R}^n$.

Proof. Since $\langle G(x), y \rangle = \langle x, G^*(y) \rangle$ for all $x, y \in \mathbb{R}^n$ and $G \in \text{End } \mathbb{R}^n$, and the operators $P\left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)$ and $Q\left(\frac{\partial}{\partial y_1}, \ldots, \frac{\partial}{\partial y_n}\right)$ comute by (2.3) the proof follows. □

In this paper we show that certain computational problems in Combinatorial Optimization can be reduced to the computation of $\langle P, Q \rangle$ for some polynomials $P, Q \in \mathcal{P}_n$. We will exploit Theorem 2.4 to design efficient algorithms for these problems. A simple example is given below.

Permanent of a matrix of a small rank. Let $A = (a_{ij}) : i, j = 1, \ldots, n$ be a real $n \times n$ square matrix. The expression

$$\text{per } A = \sum_{g \in S_n} \prod_{i=1}^{n} a_{i,g(i)}$$

is called *permanent* of A (see [5]). Here S_n denotes the symmetric group, i.e. the group of all permutations of the set $\{1, \ldots, n\}$.

Let us define polynomials $P, Q \in \mathcal{P}_n$ by the formulas

$$P = \prod_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot x_j, \quad Q = x_1 \cdots x_n.$$

Then by (2.2)

$$\text{per } A = \langle P, Q \rangle. \tag{2.5}$$

It is known that the problem of computation of the permanent of a given matrix is NP-hard. We shall show, however, that if we fix the rank of the matrix then its permanent can be computed in strongly polynomial time.

(2.6) Theorem. *Let us fix $r \in \mathbb{N}$. Then for any given $n \in \mathbb{N}$ and any given $n \times n$ matrix A such that rank $A \leq r$ one can compute the permanent of A using a number of arithmetic operations which is bounded by a polynomial in n (the degree of this polynomial is linear in r). If A is a rational matrix, then the size of all numbers involved in the algorithm is bounded by a polynomial in the input size.*

Proof. We use (2.5). Since rank $A \leq r$ there exist linear forms $l_i \in \mathcal{P}_n, i = 1, \ldots, n$ in r variables $x_1, \ldots, x_r$ and an operator $G \in \mathrm{End}\ \mathbb{R}^n$ such that $G(l_i) = \sum_{j=1}^{n} a_{ij} x_j$ for all $i = 1, \ldots, n$. Let us put $L = \prod_{i=1}^{n} l_i$, $L \in \mathcal{P}_n$. Note that $G(L) = P$ and that L is a polynomial in r variables $x_1, \ldots, x_r$ of degree n. To compute the forms l_i and G one needs to perform $O(n^3)$ arithmetic operations (as to solve the problem of linear algebra). So by (2.5) and (2.4) we have

$$\mathrm{per}\ A = \langle G(L), Q \rangle = \langle L, G^*(Q) \rangle.$$

Let us denote $R = G^*(Q)$. Then R is the product of linear forms $b_i = G^*(x_i) \in \mathcal{P}_n$. Let us denote by $\hat{b}_i, i = 1, \ldots, n$ the linear form obtained from b_i by the substitution $x_{r+1} = \ldots = x_n = 0$ and put $\hat{R} = \prod_{i=1}^{n} \hat{b}_i$. Then we have

$$\langle L, G^*(Q) \rangle = \langle L, R \rangle = \langle L, \hat{R} \rangle.$$

Since L and $\hat{R}$ are homogeneous polynomials in r variables $x_1, \ldots, x_r$ of degree n by expanding them into the sum of $\binom{n+r-1}{r-1}$ monomials using (2.2) we compute the desired value of per A. □

3. Hamiltonian Permanent

In this section we consider an analogue of permanent which we call "Hamiltonian permanent". Let us denote by H_n the set of all Hamiltonian cycles (i.e., permutations which consist of the single cycle) in the symmetric group S_n. For an $n \times n$ matrix A we define its Hamiltonian permanent by the formula

$$\mathrm{hamp}\ A = \sum_{g \in H_n} \prod_{i=1}^{n} a_{i,g(i)}. \tag{3.1}$$

It is easy to see that to compute the Hamiltonian permanent is NP-hard problem. However, we show that one can compute the Hamiltonian permanent of a matrix in strongly polynomial time if the rank of the matrix is fixed.

For $x = (x_1, \ldots, x_n)$ let us define a diagonal $n \times n$ matrix $D(x) = \mathrm{diag}\ \{x_1, \ldots, x_n\}$. Let us put

$$P(x_1, \ldots, x_n) = \mathrm{Tr}\ \big(D(x) \cdot A\big)^n, \tag{3.2}$$

where Tr denotes the usual trace of an $n \times n$ square matrix. So P is a homogeneous polynomial of degree n in n variables.

(3.3) Lemma. *Let us denote $Q = x_1 \cdots x_n \in \mathcal{P}_n$. Then for a square $n \times n$ matrix A we have*

$$\text{hamp } A = \frac{1}{n}\langle P, Q\rangle,$$

where $P \in \mathcal{P}_n$ is defined by (3.2).

Proof. Let us denote by $T_i(x)$ the i-th diagonal element of the matrix $T(x) = \big(D(x)\cdot A\big)^n$. Then

$$T_i(x) = \sum_I \prod_{j=1}^{n} x_{i_j} \cdot a_{i_j, i_{j+1}}, \tag{3.3.1}$$

where the sum is taken over all sequences $I = (1 \le i_1, \ldots, i_n \le n)$, where $i_1 = i$ and we put $i_{n+1} = i$. Let us denote by $t_{i,I} \in \mathcal{P}_n$ the I-th summand of (3.3.1). It is clear now that

$$\langle t_{i,I} \ , \ Q\rangle = \begin{cases} \displaystyle\prod_{j=1}^{n} a_{i_j, i_{j+1}}, & \text{if all numbers } i_1, \ldots, i_n \text{ are pairwise distinct;} \\ 0, & \text{otherwise.} \end{cases}$$

Thus all sequences I with a non-zero product $\langle t_{i,I} \ , \ Q\rangle$ are in bijective correspondence with the Hamiltonian cycles $g \in S_n$ such that $g(i_j) = i_{j+1}$ for $1 \le j \le n$ (we put $i_1 = i_{n+1} = i$). □

(3.4) Theorem. *Let us fix $r \in \mathbb{N}$. Then for any given $n \in \mathbb{N}$ and any given $n \times n$ matrix A such that rank $A \le r$ one can compute the Hamiltonian permanent of A using a number of arithmetic operations which is bounded by a polynomial in n (the degree of this polynomial is linear in r^2). If A is a rational matrix then the size of all numbers involved in the algorithm is bounded by a polynomial in the input size.*

Proof. We use Lemma 3.3. Since rank $A \le r$ then there exists an $n \times n$ non-degenerate matrix U such that the last $n - r$ columns of the matrix $A \cdot U$ consist of the zero elements. To compute such a matrix U one needs to perform $O(n^3)$ arithmetic operations. We have

$$\text{Tr } \big(D(x) \cdot A\big)^n = \text{Tr } \big(U^{-1} \cdot D(x) \cdot A \cdot U\big)^n.$$

Let us denote $S(x) = U^{-1} \cdot D(x) \cdot A \cdot U$. Then $S(x)$ is an $n \times n$ square matrix whose elements are linear forms in $x_1, \ldots, x_n$ and only the first r columns are non-zero. To simplify notation we denote this matrix by S. Therefore Tr $S^n =$ Tr C^n, where C is the left upper corner $r \times r$ submatrix of S.

Let us put $m = r^2$ and consider an $r \times r$ square matrix L whose elements are $x_1, \ldots, x_m$ (in an arbitrary order). Then where exists $G \in \text{End } \mathbb{R}^n$ such that $G(L_{ij}) = C_{ij}$. In this case we write $G(L) = C$. To construct such G one needs to perform $O(n^3)$ arithmetic operations. Thus we have

$$\text{hamp } A = \frac{1}{n}\langle Q, \text{Tr } C^n\rangle = \frac{1}{n}\langle Q, \text{Tr } \big(G(L)\big)^n\rangle = \frac{1}{n}\langle G^*(Q), \text{Tr } L^n\rangle.$$

The last equality holds because the trace of a power of a matrix is a polynomial in the elements of the matrix. Let us denote $R = G^*(Q) \in \mathcal{P}_n$. Then R is the product of linear forms $b_i = G^*(x_i) \in \mathcal{P}_n, i = 1, \ldots, n$. Let us denote by $\hat{b}_i$ the form obtained from b_i by the substitution $x_{m+1} = \ldots = x_n = 0$ and put $\hat{R} = \prod_{i=1}^{n} \hat{b}_i$. Then we have

$$\langle G^*(Q), \mathrm{Tr}\ L^n \rangle = \langle \hat{R}, \mathrm{Tr}\ L^n \rangle.$$

Since $\hat{R}$ and $\mathrm{Tr}\ L^n$ are homogeneous polynomials in $m = r^2$ variables $x_1, \ldots, x_m$ of degree n, by expanding them into the sum of $\binom{n+m-1}{m-1}$ monomials and using (2.2) we compute the desired value hamp A. □

The following lemma gives us a useful identity for the inner product of polynomials of a particular type.

(3.5) Lemma. *Let $P \in \mathcal{P}_n$ be a homogeneous polynomial of degree n and $Q = x_1 \cdots x_n$. For a (possibly empty) subset $\omega \subset \{1, \ldots, n\}$ let us denote by $x_\omega \in \mathbb{R}^n$ the vector whose i-th coordinate is equal to 1 if $i \in \omega$ and 0 otherwise. Then*

$$\langle Q, P \rangle = (-1)^n \cdot \sum_{\omega} (-1)^{|\omega|} P(x_\omega),$$

where the sum is taken over all subsets of $\{1, \ldots, n\}$ and $|\omega|$ denotes the cardinality of ω.

Proof. This identity holds since for $P = Q$ the sum is obviously equal to 1 and if P is a monomial whose support does not contain certain $i \in \{1, \ldots, n\}$, then the summands corresponding to $\omega - \{i\}$ and $\omega \cup \{i\}$ annihilate each other. □

The following result gives a resonably good algorithm for the computation of hamp A for a generic A.

(3.6) Theorem. *For any given $n \in \mathbb{N}$ and any given $n \times n$ matrix A one can compute the Hamiltonian permanent of A using $O(n^4 \cdot 2^n)$ arithmetic operations and $O(n^2)$ space. If A is a rational matrix, then the size of all numbers involved in the algorithm is bounded by a polynomial in the input size.*

Proof. We use (3.2), (3.3), and (3.5). Note, that $P(x) = \mathrm{Tr}\left(D(x) \cdot A\right)^n$ is a homogeneous polynomial of degree n and that we can compute in the straightforward way the value of $P(x)$ for an arbitrary $x \in \mathbb{R}^n$ using $O(n^4)$ arithmetic operations. Thus we consecuitively sum up all 2^n summands $(-1)^{|\omega|} P(x_\omega)$ by Lemma 3.5. □

Now we are going to apply our results to the Traveling Salesman Problem. First, we consider the problem of deciding if there exists a Hamiltonian contour in a given directed graph with n vertices. An exhaustive search would give us time complexity $O(n!)$, however it can be accomplished in a polynomially bounded space. The method of dynamic programming allows one to test a graph with the time complexity $O(n^2 \cdot 2^n)$, however it uses

$O(2^n)$ memory (see, for example, [4]) (since the basic idea is a reduction to subgraphs). The author can not provide any reference to an algorithm which would have a singly exponential in n time complexity and would use a polynomially bounded space. An immediate corollary of Theorem 3.6 is that such an algorithm does exist.

Without loss of generality we assume that given graph does not contain loops or multiple directed edges. Therefore, we assume that a graph Γ with n vertices $\{1,\dots,n\}$ is given by its adjacency matrix A:

$$A_{ij} = \begin{cases} 1, & \text{if } ij \text{ is a directed edge in } \Gamma, \\ 0, & \text{otherwise.} \end{cases}$$

In this case we sometimes write $\Gamma(A)$ instead of Γ.

(3.7) Theorem. *There exists an algorithm which for any given $n \in \mathbb{N}$ and any given directed graph Γ with n vertices decides whether Γ contains a Hamiltonian contour with $O(n^4 \cdot 2^n)$ time complexity and $O(n^2)$ space complexity.*

Proof. If A is adjacency matrix of a graph Γ then hamp A is equal to the number of Hamiltonian contours in Γ. To compute hamp A we use Theorem 3.6. □

Using the "divide and conquer" method one can construct a Hamiltonian contour (if one exists) with $O(n^6 \cdot 2^n)$ time complexity and $O(n^2)$ space complexity. We also note that Theorem 3.4 allows us to determine in polynomial time the parity of the number of Hamiltonian contours in a graph $\Gamma(A)$ if the rank of the matrix A over the field $\mathbb{Z}_2$ is fixed.

4. The Traveling Salesman Problem

Let us consider the following optimization problem:

(4.1) Traveling Salesman Problem.

For given real square $n \times n$ matrix $A = (a_{ij})$ find

$$w(A) = \min\Big\{ \sum_{i=1}^{n} a_{i,g(i)} : g \in H_n \Big\}.$$

We recall that H_n is the set of all Hamiltonian cycles in the symmetric group S_n.

Note that the expression (3.1) for the Hamiltonian permanent of A appears if we formally replace "min" by "$\sum$" and "$\sum$" by "$\prod$" in (4.1). We are going to reduce the problem of computation of $w(A)$ to the computation of the Hamiltonian permanent of a matrix.

By $|A|$ we denote the value $\max\{|a_{ij}| : 1 \le i,j \le n\}$.

(4.2) Theorem. *Let us fix a polynomial $p(n) : n \in \mathbb{N}$. There exist a polynomial $q(n)$ and an algorithm which for any given $n \in \mathbb{N}$ and any given $n \times n$ integral matrix A such that $|A| \leq p(n)$ computes the minimal weight $w(A)$ of a Hamiltonian cycle in Problem 4.1 with $O\big(q(n) \cdot 2^n\big)$ time complexity and $O\big(q(n)\big)$ space complexity.*

Proof. We use (1.3). For $t > 0$ let us define an $n \times n$ square matrix $\mathcal{A}(t)$ by the formula:

$$\mathcal{A}_{ij}(t) = \exp\{-t \cdot a_{ij}\}.$$

We use the exponential function and logarithm with the base 2, so $\exp\{x\} = 2^x$. We have $w(A) \leq n \cdot |A|$ and for all $t > 0$

$$\text{hamp } \mathcal{A}(t) \geq \exp\{-t \cdot w(A)\} \geq \frac{1}{(n-1)!} \cdot \text{hamp } \mathcal{A}(t).$$

These inequalities hold since the Hamiltonian permanent of $\mathcal{A}(t)$ is equal to the sum of the expressions $\exp\left\{-t \cdot \sum_{i=1}^{n} a_{i,g(i)}\right\}$ taken over all $(n-1)!$ Hamiltonian cycles g in the symmetric group S_n.

Let us choose the smallest integral $t \geq 3 \cdot n \cdot \log n$, put $w_t = -t^{-1} \cdot \log \text{hamp } \mathcal{A}(t)$ and then round w_t to the nearest integer w. This value w is the desired value $w(A)$ from (4.1). To compute the Hamiltonian permanent of matrix $\mathcal{A}(t)$ we use Theorem 3.6. Note that we can compute the entries of $\mathcal{A}(t)$ in time which is polynomial in n. □

Remark. An exhaustive search in the Traveling Salesman Problem would give us time complexity $O(n!)$ whereas it can be accomplished using a polynomially bounded space. The method of dynamic programming would give us $O(n^2 \cdot 2^n)$ time complexity (essentially the number of arithmetic operations) but it would require an exponentially large space (see [4]). It seems to be an interesting problem to design an algorithm in the Traveling Salesman Problem having a time complexity which is singly exponential in n and a space complexity bounded by a polynomial in n (note, that the complexity in Theorem 4.3 depends also on the size of the weights). Theorem 4.3 gives us an algorithm for the computation of a minimal weight of a Hamiltonian cycle but using the general method "divide and conquer" one can construct an optimal Hamiltonian cycle.

We are going to consider particular polynomially solvable cases of the Traveling Salesman Problem. We noted that the expression for $w(A)$ in 4.1 transforms into Hamiltonian permanent if we replace "min" by "$\sum$" and "$\sum$" by "$\prod$". Theorem 3.4 asserts that the Hamiltonian permanent can be computed in polynomial time for a matrix of the fixed rank. We try to find an appropriate version of the "fixed rank" for the Traveling Salesman Problem. Formaly replacing "$\cdot$" by "$+$" and "$\sum$" by "min" in the usual formula for the rank we come to the following definition.

(4.3) Definition. Let $A = (a_{ij}) : i,j = 1,\dots,n$ be an $n \times n$ real matrix. We say that A has *combinatorial rank* r if $r \in \mathbb{N}$ is the smallest natural number such that there exist r real vectors $h_s, s = 1,\dots,r$ and r real vectors $v_s, s = 1,\dots,r$ such that

$$a_{ij} = \min\{h_s(i) + v_s(j), s = 1,\dots,r\} \tag{4.3.1}$$

for all $1 \le i,j \le n$.

Here $h_s(i)$ and $v_s(j)$ denote the i-th and j-th components of the vectors h_s and v_s correspondingly. In this case we write

$$\mathrm{comr}\ A = r.$$

Like the usual rank, the combinatorial rank of an $n \times n$ square matrix does not exceed n. However, the author does not know any polynomial algorithm for the computation of the combinatorial rank. At the same time, the author believes, that for any fixed r there exists a polynomial time algorithm which checks whether $\mathrm{comr}\ A = r$.

We are going to design a (pseudopolynomial) algorithm for finding the minimal weight $w(A)$ in Problem 4.1 if A is a matrix of a fixed combinatorial rank.

For a real vector $x = (x_1,\dots,x_n) \in \mathbb{R}^n$ by $|x|$ we denote $\max\{|x_i|, i = 1,\dots,n\}$. For a matrix A given in the form (4.3.1) let us denote $L(A) = \max\{|h_s|, |v_s| : s = 1,\dots,r\}$.

(4.4) Theorem. *Let us fix a number $r \in \mathbb{N}$. Then there exists an algorithm, which for any given $n \in \mathbb{N}$ and any $n \times n$ matrix A, such that the combinatorial rank of A is equal to r and A is given in the form (4.3.1) where h_s and v_s are integral vectors, computes the smallest weight $w(A)$ of a Hamiltoinan cycle in Problem 4.1 in time which is bounded by a polynomial in n and $L(A)$.*

Proof. Again, we use a modification of (1.3). For $t > 0$ let us define an $n \times n$ square matrix $\mathcal{A}(t)$ by the formula

$$\mathcal{A}_{ij}(t) = \sum_{s=1}^{r} \exp\{-t \cdot \big(h_s(i) + v_s(j)\big)\}.$$

Then for all t $\mathrm{rank}\ \mathcal{A}(t) \le r$ and

$$\mathrm{hamp}\ \mathcal{A}(t) \ge \exp\{-t \cdot w(A)\} \ge \frac{1}{r^n \cdot (n-1)!} \cdot \mathrm{hamp}\ \mathcal{A}(t).$$

Again, we use the exponential function and logarithm with the base 2. These inequalities hold since the Hamiltonian permanent of $\mathcal{A}(t)$ is equal to the sum of the expressions $\exp\left\{-t \cdot \sum_{i=1}^{n} h_{s_i}(i) + v_{s_i}(g(i))\right\}$ taken over all $(n-1)!$ Hamiltonian cycles g from the symmetric group S_n and r^n possible choices of $s_i \in \{1,\dots,r\}$.

Let us choose the smallest integer $t \ge 3 \cdot n \cdot (\log n + \log r)$, put $w_t = -t^{-1} \cdot \log \mathrm{hamp}\ \mathcal{A}(t)$ and then round w_t to the nearest integer w. This value $w = w(A)$ is the desired smallest weight of a Hamiltonian cycle in Problem 4.1. To compute the Hamiltonian permanent of $\mathcal{A}(t)$ we use Theorem 3.4. Note, that since $t \le 4 \cdot n^2$ we can compute the entries of $\mathcal{A}(t)$ in time which is polynomial in n and $L(A)$. □

Remark. The algorithm of Theorem 4.4 is psuedopolynomial whereas it would be interesting to find a (strongly) polynomial algorithm for finding the smallest weight of a Hamiltonian contour in a weighted graph if the matrix of weights has a fixed combinatorial rank. Theorem 4.4 gives us only the minimal weight of a Hamiltonian cycle but provides no way to find such a cycle. A straightforward application of the "divide and conquer" approach fails since this leads to the increasing of the combinatorial rank of the matrix of weights.

We also give a standard reformulation of 4.4. We consider the case of a rational matrix A given by its representation (4.3.1) where $h_s, v_s : s = 1, \dots, r$ are rational vectors.

(4.5) Corollary. *Let us fix a number $r \in \mathbb{N}$. There exists an algorithm which for any given $n \in \mathbb{N}$, any rational matrix A, such that combinatorial rank of A is equal to r and A is given by (4.3.1), and any rational $\epsilon > 0$ computes the minimal weight $w(A)$ of a Hamiltonian cycle in Problem 4.1 with error less than ϵ in time which is bounded by a polynomial in n, ϵ^{-1}, $L(A)$, and the input size.*

Proof. Let N be a smallest integer which is larger than $4 \cdot \epsilon^{-1} \cdot n$. For the representation (4.3.1) of the matrix A let us construct vectors $\bar{h}_s, \bar{v}_s : s = 1, \dots, r$ consisting of rational numbers with the denominator N and such that $|h_s - \bar{h}_s|, |v_s - \bar{v}_s| < (2 \cdot n)^{-1} \cdot \epsilon$ for all $s = 1, \dots, r$. Let us define the matrix $\bar{A} = (\bar{a}_{ij})$ by the formulas:

$$\bar{a}_{ij} = \min\{\bar{h}_s(i) + \bar{v}_s(j) : s = 1, \dots, r\}.$$

Then the matrix $B = N \cdot \bar{A}$ satisfies the conditions of Theorem 4.4 and $L(B) \leq (4 \cdot \epsilon^{-1} \cdot n + 1) \cdot (L(A) + 1)$. So we can apply the algorithm of Theorem 4.4 to the matrix B. The value $w = N^{-1} \cdot w(B)$ is the desired approximation of $w(A)$. □

5. Longest Hamiltonian Contour in Euclidean Space

In this section we consider certain natural examples of matrices having small combinatorial rank. These matrices arise as the matrices of pairwise distances between given points in various metrics in the Euclidean space. The first such example (L^∞ norm in $\mathbb{R}^d$) was suggested by S. Onn. Inspired by this example, the author considered L^1 norm in the Euclidean space and then A. Gerards suggested a far reaching generalization.

Let $\|\cdot\|$ be a certain norm in the Euclidean space $\mathbb{R}^d$. Assume that we have a set $P_1, \dots, P_n$ of points in $\mathbb{R}^d$ and let us consider the $n \times n$ matrix $A = a_{ij}$ of pairwise distances between these points, so

$$a_{ij} = \|P_i - P_j\|. \tag{5.1}$$

Let us consider the unit ball

$$\mathbf{B} = \{x \in \mathbb{R}^d : \|x\| \leq 1\}.$$

If $\mathbf{B}$ is a polytope then the norm $\|\cdot\|$ is called *polyhedral* norm. The following result was suggested by A. Gerards.

(5.2) Theorem. *Let $\|\cdot\|$ be a polyhedral norm in the Euclidean space $\mathbb{R}^d$. Assume that $A = (a_{ij})$ is the matrix of pairwise distances between given points $P_1, \ldots, P_n \in \mathbb{R}^d$ defined by (5.1). Then the combinatorial rank of the matrix $-A$ does not exceed the number of facets of the polytope* $\mathbf{B}$.

Proof. Let us denote by $\langle \cdot, \cdot \rangle$ the standard inner product in $\mathbb{R}^d$ and assume that the ball $\mathbf{B}$ is given by the system of linear inequlities:

$$\mathbf{B} = \{x \in \mathbb{R}^d : |\langle u_s, x \rangle| \leq 1, s = 1, \ldots, f\},$$

for some vectors $u_1, \ldots, u_f$ in $\mathbb{R}^d$. In fact, the vectors u_s and $-u_s$ ranges over all normal vectors to the facets of $\mathbf{B}$. We introduce the vectors $v_s \in \mathbb{R}^n, s = 1, \ldots, f$ putting $v_s(i) = \langle P_i, u_s \rangle$.

Then we have

$$\|P_i - P_j\| = \min\{\lambda \geq 0 : P_i - P_j \in \lambda \cdot \mathbf{B}\} =$$

$$= \min\{\lambda \geq 0 : |v_s(i) - v_s(j)| \leq \lambda \text{ for all } s = 1, \ldots, f\} = \max\{|v_s(i) - v_s(j)| : s = 1, \ldots, f\}.$$

Therefore

$$-a_{ij} = -\|P_i - P_j\| = \min\{v_s(i) - v_s(j), v_s(j) - v_s(i) : s = 1, \ldots, f\}.$$

Thus comr $-A \leq 2 \cdot f$, and, moreover, the matrix $-A$ can be efficiently written in the form (4.3.1). □

Thus, if $\|\cdot\|$ is the L^∞ norm then for any n points $P_1, \ldots, P_n$ in $\mathbb{R}^d$ the combinatorial rank of the matrix $-A : a_{ij} = \|P_i - P_j\|$ does not exceed $2 \cdot d$ and for the L^1 norm the combinatorial rank of this matrix does not exceed 2^d.

Since we are dealing with computational complexity, we will consider polyhedral norms such that the unit ball $\mathbf{B}$ is a *rational* polytope given by a system of linear inequalities

$$\mathbf{B} = \{x \in \mathbb{R}^d : |\langle u_s, x \rangle| \leq 1, s = 1, \ldots, f\}$$

for some rational vectors $u_1, \ldots, u_f$ and we assume that the points $P_1, \ldots, P_n$ are given by their rational coordinates. The distance between two points P_i and P_j

$$\|P_i - P_j\| = \max\{|\langle P_i - P_j, u_s \rangle|, s = 1, \ldots, f\}$$

can be computed using $O(n)$ arithmetic operations.

(5.3) Corollary. *Let us fix a polyhedral norm $\|\cdot\|$ in the Euclidean space $\mathbb{R}^d$. Then there exists an algorithm which for any given $n \in \mathbb{N}$, any n points $P_1, \ldots, P_n$ in $\mathbb{Q}^d$ given by their coordinates, and any given rational $\epsilon > 0$ compute the length of the longest Hamiltonian contour joining these points with error less than ϵ in time bounded by a polynomial in n, $\max\{\|P_i\|, i = 1, \ldots, n\}$, ϵ^{-1}, and the input size.*

Proof. Follows by Theorem 5.2 and Corollary 4.5. □

Let $\|\cdot\|$ be an arbitrary norm in $\mathbb{R}^d$. Then we can approximate the unit ball $\mathbf{B}$ by a centrally symmetric rational convex polytope and thus approximate the norm $\|\cdot\|$ by a certain polyhedral norm. This imply the following result.

(5.4) Theorem. *Let us fix a norm in $\mathbb{R}^d$ and a number $\delta > 0$. Then for any given $n \in \mathbb{N}$ and any n points in $\mathbb{Q}^d$, given by their coordinates, one can compute the length of the longest Hamiltonian contour joining these points with relative error less than δ in time bounded by a polynomial in the input size.*

Proof. Without loss of generality we assume that δ is sufficiently small, say $\delta < 1/3$. Let $\mathbf{B} = \{x \in \mathbb{R}^d : \|x\| \leq 1\}$ be the unit ball in the norm $\|\cdot\|$. We construct a rational centrally symmetric polytope $\mathbf{B}_1$ such that

$$(1 - \delta/3) \cdot \mathbf{B} \subset \mathbf{B}_1 \subset (1 + \delta/3) \cdot \mathbf{B}.$$

Then the Minkowski's functional

$$\|x\|_1 = \min\{\lambda \geq 0 : x \in \lambda \cdot \mathbf{B}_1\}$$

is a polyhedral norm in $\mathbb{R}^d$ such that

$$(1 - \delta/3) \cdot \|x\|_1 \leq \|x\| \leq (1 + \delta/3) \cdot \|x\|_1$$

for all $x \in \mathbb{R}^d$.

Note that this construction does not depend on the particular choice of the points $P_1, \ldots, P_n$ and since the norm $\|\cdot\|$ and δ are fixed it can be done in advance, before starting the algorithm.

Without loss of generality we may assume that

$$\|P_1 - P_2\|_1 = \max\{\|P_i - P_j\|_1 : 1 \leq i, j \leq n\}.$$

Applying an appropriate translation, and then a dilatation we can assume that $P_1 = 0$ and $\|P_2 - P_1\|_1 = 2$, so $\|P_1 - P_i\|_1 = \|P_i\|_1 \leq 2$ for all i and the length of a longest Hamiltonian contour is not smaller than 2. Now we use Corollary 5.4 for the norm $\|\cdot\|_1$ with $\epsilon = \delta/3$. The computed number approximates the length of a longest Hamiltonian contour with relative error less than δ. □

In the next two sections we give another two examples illustrating our technique. We consider the Weighted 3-Packing Probem and the Vertex Disjoint Paths Problem. Both of them are NP-hard. Our consideration here are almost parallel to those of Sections 3-4. Therefore sometimes we only sketch a proof.

6. Weighted 3-Packing Problem

Let us choose $n \in \mathbb{N}, n = 3 \cdot m$ for $m \in \mathbb{N}$. Let us choose a cubic tensor $C \in (\mathbb{R}^n)^{\otimes 3}, C = \{c(i,j,k) : 1 \leq i, j, k \leq n\}$. We assume that the tensor c is *symmetric* i.e., the value $c(i,j,k)$ is independent of the order of i, j, k. In other words, we assume that for each

(unordered) 3-subset $I = \{i, j, k\}$ of the set $\{1, \ldots, n\}$ we assign a real number $c(I)$ which we call the weight of the subset. In principle, we allow subsets with multiple elements but actually they will not affect our consideration. To each partition $\{I_1, \ldots, I_m\}$ of the set $\{1, \ldots, n\}$ onto unordered 3-sets such that each element of the set $\{1, \ldots, n\}$ is covered exactly once we associate the weight of this partition, i.e., $\sum_{i=1}^{m} c(I_j)$. Then the Weighted 3-Packing Problem can be formulated as follows:

(6.1) Weighted 3-Packing Problem. For given symmetric tensor $C \in (\mathbb{R}^n)^{\otimes 3}$ find

$$w(C) = \min\Big\{\sum_{j=1}^{m} c(I_j) : J = \{I_1, \ldots, I_m\}\Big\},$$

where $J = \{I_1, \ldots, I_m\}$ ranges over all unordered partitions of the set $\{1, \ldots, n\}$ onto disjoint unordered 3-sets such that each element of the set $\{1, \ldots, n\}$ is covered exactly once.

It is known (see [3]) that the (Weighted) 3-Packing Problem is NP-hard. Following the same ideas as before let us substitute formally "min" by "$\sum$" and "$\sum$" by "$\prod$". We come to the following homogeneous polynomial in variables $c(I)$ of degree m.

(6.2) Hyperhafnian. For a symmetric tensor $C \in (\mathbb{R}^n)^{\otimes 3}$ put

$$\mathrm{HAF}\ C = \sum_{J=\{I_1,\ldots,I_m\}} \prod_{j=1}^{m} c(I_j),$$

there the sum is taken over all unordered partitions J of the set $\{1, \ldots, n\}$ onto disjoint 3-sets such that each element $1, \ldots, n$ is covered exactly once. We call this polynomial *hyperhafnian* of the tensor C since its analogue for the 2-matchings sometimes is called hafnian (see [5]).

As in the case of Hamiltonian permanent (see Section 3) it turns out that the hyperhafnian can be computed in strongly polynomial time if we fix the rank of the tensor.

(6.3) Definition. Let $C \in (\mathbb{R}^n)^3$ be a cubic tensor. We say that the tensor C has *rank* r if $r \in \mathbb{N}$ is the smallest natural number such that there exist $3 \cdot r$ vectors $h_s, v_s, u_s \in \mathbb{R}^n$ such that

$$c(i, j, k) = \sum_{s=1}^{r} h_s(i) \cdot v_s(j) \cdot u_s(k) \tag{6.3.1}$$

for all $1 \le i, j, k \le n$.

Unlike the rank of a matrix, the rank of a cubic tensor is hardly computable. However, the author believes that for any fixed r the condition $\mathrm{rank}\ C = r$ can be checked in polynomial time.

(6.4) Theorem.

1. There exists an algorithm, which for any given $n \in \mathbb{N}$ and any given symmetric tensor $C \in (\mathbb{R}^n)^{\otimes 3}$ computes the hyperhafnian of C using $O(n^4 \cdot 2^n)$ arithmetic operations and $O(n^3)$ space. If the tensor C is rational, then the size of all numbers involved in the algorithm is bounded by a polynomial in the input size.

2. Let us fix $r \in \mathbb{N}$. There exists an algorithm, which for any given $n \in \mathbb{N}$ and any given symmetric tensor $C \in (\mathbb{R}^n)^{\otimes 3}$ such that rank $C = r$ *and the tensor C is given by its representation (6.3.1) computes the hyperhafnian of C using a number of arithmetic operations which is bounded by a polynomial in n (the degree of this polynomial is linear in r). If the vectors in the representation (6.3.1) are rational, then the size of all numbers involved in algorithm is bounded by a polynomial in the input size.*

Proof. 1) With the tensor $C = \{c(i,j,k) : 1 \le i,j,k \le n\}$ we associate a cubic polynomial

$$p_C(x) = \sum_I c(I) \cdot x^I. \tag{6.4.1}$$

Here $I = \{i,j,k\}$ ranges over all 3-subsets of the set $\{1,\ldots,n\}$ and $x^I = x^i \cdot x^j \cdot x^k$. Putting $Q = x_1 \cdots x_n \in \mathcal{P}_n$, $P = p_C^m \in \mathcal{P}_n$ we get

$$\text{HAF } C = \frac{1}{m!} \cdot \langle P, Q \rangle.$$

To conclude the proof of 1) we use this representation of HAF C and Lemma 3.5. Note that to compute the value of $P(x)$ in a given point $x \in \mathbb{R}^n$ one needs to perform $O(n^4)$ arithmetic operations.

2) Let us construct $3 \cdot r$ linear forms

$$H_s(x) = \sum_{i=1}^n h_s(i) \cdot x_i, V_s(x) = \sum_{i=1}^n v_s(i) \cdot x_i, U_s(x) = \sum_{i=1}^n u_s(i) \cdot x_i, s = 1,\ldots,r,$$

where h_s, v_s and u_s are vectors from the representation (6.3.1). Let us put

$$\bar{p}_C(x) = \frac{1}{6} \cdot \sum_{s=1}^r H_s(x) \cdot V_s(x) \cdot U_s(x).$$

Since the tensor C is symmetric, the polynomial $\bar{p}_C$ differs from the polynomial p_C in (6.4.1) only in terms involving x_i^2 or x_i^3. Therefore,

$$\text{HAF } C = \frac{1}{m!} \cdot \langle \bar{P}, Q \rangle,$$

where $\bar{P} = \bar{p}_C^m$. Now we can construct a linear operator $G \in \text{End } \mathbb{R}^n$ such that $G(x_{3s-2}) = H_s(x)$, $G(x_{3s-1}) = V_s(x)$, and $G(x_{3s}) = U_s(x)$ for $s = 1,\ldots,r$. Put

$$L = \Big(\sum_{s=1}^r x_{3s-2} \cdot x_{3s-1} \cdot x_{3s}\Big)^m.$$

Then we have

$$\langle Q, \bar{P} \rangle = \langle Q, G(L) \rangle = \langle G^*(Q), L \rangle.$$

Let us denote $R = G^*(Q)$. Then R is the product of linear forms $b_i = G^*(x_i), i = 1, \ldots, n$. Let us denote by $\hat{b}_i, i = 1, \ldots, n$ the linear form obtained from b_i by the substitution $x_{r+1} = \ldots = x_n = 0$ and put $\hat{R} = \prod_{i=1}^{n} \hat{b}_i$. Then

$$\langle L, G^*(Q) \rangle = \langle L, R \rangle = \langle L, \hat{R} \rangle.$$

Since L and $\hat{R}$ are homogeneous polynomials in $3 \cdot r$ variables of degree n by expanding them into the sum of $\binom{n + 3 \cdot r - 1}{r - 1}$ monomials using (2.2) we compute the desired value. □

Now we turn to the optimization problem 6.1. As in Section 4, we introduce a combinatorial analogue of the rank of a tensor.

(6.5) Definition. We say that 3-valent tensor $C \in (\mathbb{R}^n)^{\otimes 3}$ has *combinatorial rank* r if r is the smallest integer such that there exist $3 \cdot r$ vectors h_s, v_s and $u_s, s = 1, \ldots, r$ in $\mathbb{R}^n$ such that

$$c(i,j,k) = \min\Big\{ h_s(i) + v_s(j) + u_s(k), s = 1, \ldots, r \Big\} \tag{6.5.1}$$

for all $i, j, k \in \{1, \ldots, n\}$.

Let us denote $|C| = \max\{|c(i,j,k)| : 1 \le i, j, k \le n\}$. If C is given by (6.5.1) we denote $L(C) = \max\{|h_s|, |v_s|, |u_s|, s = 1, \ldots, r\}$.

(6.6) Theorem.

1) Let us fix a polynomial $p(n) : n \in \mathbb{N}$. There exist a polynomial $q(n)$ and an algorithm which for any given $n \in \mathbb{N}$ and any given integral tensor $C \in (\mathbb{Z}^n)^{\otimes 3}$ such that $|C| \le p(n)$ computes the minimal weight $w(C)$ of a partition in Problem 6.1 with the time complexity $O\big(q(n) \cdot 2^n\big)$ and space complexity $O\big(q(n)\big)$.

2) Let us fix a number $r \in \mathbb{N}$. Then there exists an algorithm, which for any given $n \in \mathbb{N}$ and any tensor $C \in (\mathbb{R}^n)^{\otimes 3}$ such that the combinatorial rank of C is equal to r and C is given by (6.5.1), where h_s, v_s, and u_s are integral vectors, computes the minimal weight $w(C)$ of a partition in Problem 6.1 in time bounded by a polynomial in n and $L(C)$.

Proof. To prove 1), let us define for $t > 0$ a tensor $\mathcal{C}_t \in (\mathbb{R}^n)^{\otimes 3}$ putting $\mathcal{C}_t(i,j,k) = \exp\{-t \cdot c(i,j,k)\}$ for $1 \le i, j, k \le n$. Then by (1.3)

$$\mathrm{HAF}\ \mathcal{C}_t \ge \exp\{-t \cdot w(C)\} \ge \frac{6^m \cdot m!}{n!} \cdot \mathrm{HAF}\ \mathcal{C}_t.$$

Now we proceed as in Theorem 4.2 using Theorem 6.4, 1) for the computation of the hyperhafnian.

2) For $t > 0$ let us put

$$\mathcal{C}_t(i,j,k) = \sum_{s=1}^{r} \sum_{\sigma} \exp\{-t \cdot (h_s(\sigma(i)) + v_s(\sigma(j)) + u_s(\sigma(k)))\},$$

where σ ranges over all 6 permutations of the set $\{i,j,k\}$. Then $\mathcal{C}_t(i,j,k)$ is a symmetric tensor and rank $\mathcal{C}_t \leq 6 \cdot r$ for all $t > 0$. By (1.3) we have

$$\text{HAF } \mathcal{C}_t \geq \exp\{-t \cdot w(C)\} \geq \frac{6^m \cdot m!}{(6 \cdot r)^m \cdot n!} \cdot \text{HAF } \mathcal{C}_t.$$

Now we proceed as in Theorem 4.4 using Theorem 6.4, 2) for the computation of the hyperhafnian. □

Remark. The problem of finding a perfect 3-matching in a uniform 3-hypergraph Γ with n vertices can be easily reduced to Problem 6.1 where tensor C consists of $0-1$ elements. Namely, we put $c(i,j,k) = 0$ if $\{i,j,k\}$ is an edge of $\Gamma = \Gamma(C)$ and $c(i,j,k) = 1$ otherwise. Then Theorem 6.7 1) provides an algorithm for testing whether there exists a perfect matching in Γ. The time complexity of this algorithm is singly exponential in n whereas the space complexity is polynomial in n. The author can not provide any reference to any other algorithm in this problem with similar bounds of complexity.

7. Vertex Disjoint Paths Problem

Let Γ be a directed graph with n vertices $\{1,\ldots,n\}$ and A be its adjacency matrix (see also Section 3). In this case we write sometimes $\Gamma(A)$ instead of Γ. We consider the following problem.

(7.1) Vertex Disjoint Paths Problem. For given $n \times n$ square $0-1$ matrix A and for given pairs of vertices $(b_1,e_1),\ldots,(b_k,e_k)$ of graph $\Gamma(A)$ decide whether there exist k vertex disjoint directed paths $\pi_1,\ldots,\pi_k$ in $\Gamma(A)$ joining b_s and e_s for $s = 1,\ldots,k$.

We present here an algorithm for the Vertex Disjoint Paths Problem whose time complexity is singly exponential in n and space complexity is polynomial in n.

(7.2) Theorem. *There exists an algorithm which for any given $n \in \mathbb{N}$, any given directed graph Γ with n vertices, any given pairs of vertices $(b_1,e_1),\ldots,(b_k,e_k)$ in Γ desides whether there exist k vertex disjoint paths $\pi_s, s = 1,\ldots,k$ joining the vertices b_s and e_s for $s = 1,\ldots,k$. The running time of this algorithm is $O\big((kn)^3 \cdot 4^n\big)$ and it uses a space bounded by a polynomial in n.*

Proof. Without loss of generality, we assume that all the vertices $b_1,\ldots,b_k,e_1,\ldots,e_k$ are pairwise distinct. Let $x_1,\ldots,x_n$ and $y_1,\ldots,y_n$ be independent variables. Let us define

diagonal matrices $D(x) = \text{diag}\,\{x_1, \ldots, x_n\}$ and $D(y) = \text{diag}\,\{y_1, \ldots, y_n\}$. Let us define the following (non-homogeneous) polynomial P of degree $2 \cdot k \cdot n$:

$$P(x,y) = \prod_{s=1}^{k} \sum_{m=1}^{n} \Big(D(x) \cdot A \cdot D(y)\Big)^{m}_{b_s, e_s}.$$

The term

$$\Big(D(x) \cdot A \cdot D(y)\Big)^{m}_{b_s, e_s}$$

denotes the (b_s, e_s)-th entry of a power of a matrix and it is equal to the sum of monomials $x_{i_1} \cdot y_{i_2} \cdot x_{i_2} \cdots y_{i_{m-1}} \cdot x_{i_{m-1}} \cdot y_{i_m}$ for all directed paths $\pi = i_1, i_2, \ldots, i_{m-1}, i_m$ in $\Gamma(A)$ where $i_1 = b_s$ and $i_m = e_s$. Therefore Problem 7.1 has a solution if and only if the polynomial $P(x,y)$ contains a non-zero monomial whose degree in each variable x_i and y_j is at most 1. Let us introduce the following differential operator

$$\mathcal{D}\Big(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\Big) = \prod_{i=1}^{n} \Big(I + \frac{\partial}{\partial x_i} + \frac{\partial}{\partial y_i} + \frac{\partial^2}{\partial x_i \partial y_i}\Big), \tag{7.2.1}$$

where I denotes the identity operator.

We conclude that Problem 7.1 has a solution, if and only if

$$\mathcal{D}\Big(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\Big) P(x_1, \ldots, x_n; y_1, \ldots y_n)\Big|_{x_1 = \ldots = x_n = y_1 \ldots = y_n = 0} \neq 0. \tag{7.2.2}$$

To compute the expression (7.2.2), we replace the differential operators $\dfrac{\partial}{\partial x_i}$, $\dfrac{\partial}{\partial y_i}$, and $\dfrac{\partial^2}{\partial x_i \partial y_i}$ for $i = 1, \ldots, n$ in (7.2.1) by the operators of finite differences

$$\frac{1}{\epsilon} \cdot (T_{x_i}(\epsilon) - I), \frac{1}{\epsilon} \cdot (T_{y_i}(\epsilon) - I), \text{ and } \frac{1}{\epsilon^2} \cdot (T_{y_i}(\epsilon) - I)(T_{x_i}(\epsilon) - I)$$

correspondingly. Here T_{x_i} is a shift: $x_i \longmapsto x_i + \epsilon$ along the variable x_i and T_{y_i} is a shift $y_i \longmapsto y_i + \epsilon$ along the variable y_i.

After that, we remove the parentheses in the right hand side of (7.2.1) and get a an operator $\mathcal{T}(\epsilon)$ which is a linear combination of 4^n shift operators of the type:

$$S(\epsilon) \cdot T(i_1 \cdot \epsilon, \ldots, i_n \cdot \epsilon; j_1 \cdot \epsilon, \ldots, j_n \cdot \epsilon),$$

where $S(\epsilon)$ is a rational function in ϵ and $i_1, \ldots, j_n \leq n$. Here $T(\delta_1, \ldots, \delta_n; \gamma_1, \ldots, \gamma_n)$ is a shift: $x_i \longmapsto x_i + \delta_i$, $y_i \longmapsto y_i + \gamma_i$ for $i = 1, \ldots, n$. Applying the operator $\mathcal{T}(\epsilon)$ to $P(x,y)\big|_{x=y=0}$ we get a rational function $Q(\epsilon)$ which is equal to a sum of 4^n expressions of the type

$$S(\epsilon) \cdot P(i_1 \cdot \epsilon, \ldots i_n \cdot \epsilon; j_1 \cdot \epsilon, \ldots, j_n \cdot \epsilon) \tag{7.2.3}$$

The function $Q(\epsilon)$ is actually a polynomial in ϵ since there exists a limit $\lim_{\epsilon \longrightarrow 0} Q(\epsilon)$ which is equal to the value of (7.2.2). Thus we have to compute the constant term of the polynomial $Q(\epsilon)$. Note that for any given x, y the value of $P(x, y)$ can be computed using $O(n^4 \cdot k)$ arithmetic operations and that deg $Q \leq 2 \cdot k \cdot n$. So we compute the values of $Q(\epsilon)$ computing 4^n summands of type (7.2.3) for $\epsilon = 1, \ldots, 2 \cdot k \cdot n + 2$ and then since Q is a polynomial of degree at most $2 \cdot k \cdot n + 1$ from the resulting system of $2 \cdot k \cdot n + 2$ linear equations we determine the constant term of Q. □

Using the general approach "divide and conquer" one can easily construct these vertex disjoint paths (if they exist). Our approach also can be adjusted to the weighted version as is done in Section 4.

8. Discussion

Our approach for solving problems of Combinatorial Optimization basically can be described as follows: using exponential sums we "lift" our problem to the problem of the computation of the value of a certain polynomial and then "project" it down. Formally, we translate a formula by replacing "min" by "$\sum$" and "+" by "$\cdot$" and then (after certain work) retranslate it back. This "loop" causes certain inconvenience, so, for example, the complexity of algorithms in Theorems 4.2, 4.4, and 6.4 is exponential in the size of numbers, and what is even more unsatisfactory, we can often compute in polynomial time only the optimal value of a functional, but not an optimal solution. Therefore, it would be very useful to explore the possibility of direct combinatorial computations in terms of "+"and "min". An obvious obstacle is that because of this translation we can use all field's operations $+, -, \cdot, :$, whereas the structure based on "min" and "+" is much poorer. For example, to compute the Hamiltonian permanent of a matrix we use Gaussian elimination, which can hardly be simulated in the semiring generated by "min" and "+". However, it is well-known that in some simplest situations there is no need in "translations". For example, the connection between the Shortest Path Problem and matrix multiplication is well studied (see, for example [1]).

Another natural problem is to improve these algorithms and to get rid of some of their most disturbing shortcomings (like mentioned above). This probably requires a certain synthesis of algebraic and combinatorial technique.

Also it would be interesting to find an efficient algorithm for the computation of combinatorial rank of a matrix or a tensor.

Acknowledgements

The results of Section 5 are obtained in collaboration with Shmuel Onn and Bert Gerards. I am grateful for their valuable suggestions.

References

1. A Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, Mass: Addison-Wesley, 1974.
2. A. Barvinok, Optimization Problems on Matroids and Exponential Sums, in: *Integer Programming and Combinatorial Optimization - Proceedings of a Conference held at Carnegie Mellon University, May 25-27, ed. E. Balas, G. Cornuéjols and R. Kannan,* Carnegie Mellon University, 1992, 316-333.
3. M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
4. E. Lawler, et al., eds. *The Traveling Salesman Problem: a guided tour of combinatorial optimization*, Chichester, New York: Wiley, 1985.
5. H. Minc, *Permanents*, Reading, Mass.: Addison-Wesley, 1978.

A.I. Barvinok
Department of Mathematics
Royal Institute of Technology
S-100 44 Stockholm
Sweden

e-mail: barvinok@math.kth.se