Theoretical Computer Science 410 (2009) 4480–4488

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Maximum weight bipartite matching in matrix multiplication time☆

Piotr Sankowski

Institute of Informatics, Warsaw University, Banacha 2, 02-097, Warsaw, Poland

ARTICLE INFO

Keywords:
Matchings in graphs
Matrix multiplication
Weighted perfect matchings
Shortest paths

ABSTRACT

In this paper we consider the problem of finding maximum weight matchings in bipartite graphs with nonnegative integer weights. The presented algorithm for this problem works in $\tilde{O}(Wn^{\omega})$[1] time, where ω is the matrix multiplication exponent, and W is the highest edge weight in the graph. As a consequence of this result we obtain $\tilde{O}(Wn^{\omega})$ time algorithms for computing: minimum weight bipartite vertex cover, single source shortest paths and minimum weight vertex disjoint s-t paths. All of the presented algorithms are randomized and with small probability can return suboptimal solutions.

1. Introduction

The weighted matching problem is one of the fundamental problems in combinatorial optimization. The first algorithm for this problem in the bipartite case was proposed in the fifties of the last century by Kuhn [14]. His result has been improved several times since then; the known results are summarized in the Table 1. The bold font indicates an asymptotically best bound in the tables. In particular the algorithm presented here is faster than the algorithm of Gabow and Tarjan [8] and the algorithm of Edmonds and Karp [5] in the case of dense graphs with small integer weights. Contrary to the other algorithms in the table, our algorithm is randomized and may return suboptimal solutions with small probability.

Note, that in this summary there are no algorithms that use matrix multiplication. However, in the papers studying the parallel complexity of the problem [13,19], such algorithms are implicitly constructed. These results lead to $O(Wn^{\omega+2})$ sequential time algorithms. In this paper we improve the complexity by a factor of n^2. The improvement in the exponent by 1 is achieved with the use of the very recent results of Storjohann [25], who showed faster algorithms for computing polynomial matrix determinants. His results are summarized in Section 1.1. Further improvement is achieved by a novel reduction technique, that allows us to reduce the weighted version of the problem to an unweighted one. The four steps of the reduction are schematically presented in Fig. 1. As a step of the reduction we also compute the bipartite weighted cover of the graph. The unweighted problem is then solved with use of the $O(n^{\omega})$ time algorithms developed two years ago by Mucha and Sankowski [18]. Storjohann's result can also be used to compute the maximum weight of a perfect matching in general graphs. However, the problem of finding such a matching remains unsolved.

The weighted matching problem is not only interesting by itself, but also it can be used to solve many other problems in combinatorial optimization. In particular, the presented algorithm for finding maximum weight perfect matchings can be used to find minimum weight perfect matching, as well as, maximum and minimum weight matchings. Moreover, the minimum weight perfect matching algorithm can be used for computing the minimum weight of k vertex disjoint s-t paths, whereas the minimum weight vertex cover can be used to solve the single source shortest paths (SSSP) problem with negative edge weights. The complexity of the algorithms for computing the minimum weight of k vertex disjoint s-t paths, follow exactly the results in Table 1. The author is not aware of any special algorithms for this problem. The complexity results for the SSSP problem with negative edge weights are summarized in Table 2.

☆ Research supported by KBN grant 1P03A01830.

E-mail address: sank@mimuw.edu.pl.

[1] $\tilde{O}$ denotes the so-called "soft O" notation, i.e. $f(n) = \tilde{O}(g(n))$ iff $f(n) = O(g(n)\log^k n)$ for some constant k.

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.07.028

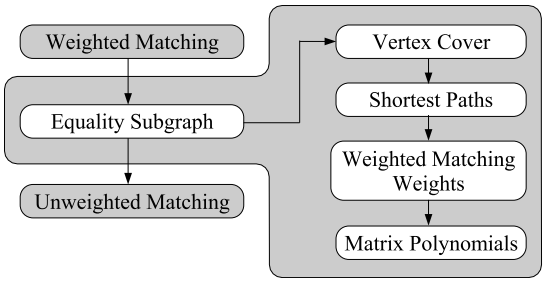


Fig. 1. The scheme of the reduction from weighted to unweighted matchings.

Table 1
The complexity results for the bipartite weighted matching.

Complexity	Author
$O(n^4)$	Khun (1955) [14] and Munkers (1957) [20]
$O(n^2m)$	Iri (1960) [10]
$O(n^3)$	Dinic and Kronrod (1969) [4]
$\boldsymbol{O(nm + n^2 \log n)}$	Edmonds and Karp (1970) [5]
$O(n^{\frac{3}{4}} m \log W)$	Gabow (1983) [7]
$\boldsymbol{O(\sqrt{n}m \log(nW))}$	Gabow and Tarjan (1989) [8]
$O(\sqrt{n}mW)$	Kao, Lam, Sung and Ting (1999) [12]
$\boldsymbol{\tilde{O}(n^\omega W)}$	This paper (randomized solution)

Table 2
The complexity results for the SSSP problem with negative weights. The bold font indicates an asymptotically best bound in the table.

Complexity	Author
$O(n^4)$	Shimbel (1955) [24]
$O(n^2mW)$	Ford (1956) [11]
$\boldsymbol{O(nm)}$	Bellman (1958) [1], Moore (1959) [17]
$O(n^{\frac{3}{4}} m \log W)$	Gabow (1983) [7]
$O(\sqrt{n}m \log(nW))$	Gabow and Tarjan (1989) [8]
$\boldsymbol{O(\sqrt{n}m \log(W))}$	Goldberg (1993) [9]
$\boldsymbol{\tilde{O}(n^\omega W)}$	Yuster and Zwick (2005) [26], Sankowski (2005) [22], this paper (randomized solutions)

The rest of the paper is organized as follows. In the remainder of this introductory section, we summarize the results in linear algebra algorithms, we show the randomization technique used later in this paper and we recall the result from [18]. In Section 2 we present our reduction technique in the case of bipartite graphs. This section is divided into four parts, each containing one reduction step. The first step of the reduction, i.e., the construction of the equality subgraph from the minimum vertex cover, is based on Egerváry's theorem. The reduction from minimum vertex cover to shortest paths was given by Iri [10]. The third step is the reduction from shortest paths to matchings' weights in appropriately defined graphs. The matchings' weights can be later computed with use of the matrix polynomial algorithms. In Section 3 we review the applications of the algorithm to: other kinds of the weighted matching problems, the SSSP problem and minimum weight vertex disjoint s-t path problem. Finally, Section 4.1 concludes the paper.

1.1. Linear algebra algorithms

We denote by ω the exponent of a square matrix multiplication algorithm. The best bound on $\omega \leq 2.376$ is due to Coppersmith and Winograd [2]. The interaction between matrix multiplication and other linear algebra problems is well understood. The best known algorithms for many problems in linear algebra work in matrix multiplication time, i.e., the determinant of an $n \times n$ matrix A, or the solution to the linear system of equations, can be computed in $O(n^\omega)$ arithmetic operations. Very recently Storjohann [25] has shown that for polynomial matrices these problems can be solved with the same exponent.

Theorem 1 (*Storjohann '03*). *Let $A \in K[x]^{n\times n}$ be a polynomial matrix of degree d and $b \in K[x]^{n\times 1}$ be a polynomial vector of the same degree, then*

- *determinant* $\det(A)$,
- *rational system solution* $A^{-1}b$,

can be computed in $\tilde{O}(n^\omega d)$ operations in field K, with high probability.

For our purposes it would be convenient if one could compute the inverse matrix A^{-1} in the above time bounds. This is impossible because the rational functions in the inverse can have degrees as high as nd, so just the output size is $\Omega(n^3 d)$, and no algorithm can work faster. Thus the approach similar to the one in papers [21,18], where the inverse matrix is used to find edges belonging to a perfect matching, will not lead to $\tilde{O}(Wn^\omega)$ time algorithm.

1.2. Zippel–Schwartz lemma

In this paper we will reduce a weighted perfect matching problem to testing if some set of polynomials is non-zero. In the simpler case of perfect matchings, in order to test if a graph has a perfect matching we need to check whether an appropriately defined adjacency matrix is non-singular [16]. We can verify that the matrix is non-singular by computing its determinant, which is a polynomial of the entries of the matrix. However, this determinant may have exponentially many terms, so it cannot be computed symbolically in polynomial time. The following lemma due to Zippel [27] and Schwartz [23] can be used to overcome this obstacle.

Lemma 2. *If $p(x_1, \ldots, x_m)$ is a non-zero polynomial of degree d with coefficients in a field and S is a subset of the field, then the probability that p evaluates to 0 on a random element $(s_1, s_2, \ldots, s_m) \in S^m$ is at most $d/|S|$. We call such event a false zero.*

Corollary 3. *Let p be a prime number of length $(1+c)\log n$. If a non-zero polynomial of degree n over Z_p is evaluated on random values from Z_p, then the probability of false zero is at most $\frac{1}{n^c}$, for any $c > 0$.*

In the standard RAM model we assume the word size to be $O(\log n)$. Thus the finite field arithmetic modulo p, except division, can be implemented in constant time. The divisions can be realized in $O(\log n)$ time. Nevertheless in our algorithms divisions are not time dominating operations. These assumptions are used to establish the time bounds for our algorithms.

1.3. Unweighted matchings

As stated in the introduction this paper shows the reduction from the weighted matching problem to the unweighted one. Next the unweighted perfect matching problem is solved in $O(n^\omega)$ randomized time as stated in the following theorem [18].

Theorem 4 (*Mucha and Sankowski '04*). *A perfect matching in a graph can be computed in $O(n^\omega)$ time, with high probability.*

For simplicity in the remainder of this paper we assume that in the considered graphs there is always a perfect matching. When necessary this assumption can be checked with use of this theorem.

2. Weighted matchings in bipartite graphs

A weighted bipartite $2n$-vertex graph G is a tuple $G = (U, V, E, w)$. The vertex sets are given by $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. $E \subseteq U \times V$ denotes the edge set, and the function $w : E \to \mathbb{Z}_+$ ascribes weights to the edges. We denote by W the maximum weight in w.

In the *maximum weight bipartite perfect matching problem* we seek a perfect matching M in a weighted bipartite graph G to maximize the total weight $w(M) = \sum_{e \in M} w(e)$.

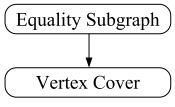
A *weighted cover* is a choice of labels $y(v_1), \ldots, y(v_n), y(u_1), \ldots, y(u_n)$ such that $y(v) + y(u) \geq w(vu)$ for all $vu \in E$. The *minimum weight cover problem* is that of finding a cover of minimum weight. The following theorem states that the vertex cover problem and the maximum weight matching problem are dual.

Theorem 5 (*Egerváry '31*). *Let $G = (U, V, E, w)$ be a weighted bipartite graph. The maximum weight of a perfect matching of G is equal to weight of the minimum weight cover of G.*

In order to find weighted perfect matchings we explore this duality. In the next two subsections we will show how to solve one of these problems when we already have the solution to the other.

2.1. From equality subgraph to vertex cover

The following lemma is one of the ingredients of the Hungarian method. The *equality subgraph* G_p for a weighted graph G and its vertex cover p is defined as $G_p = (U, V, E')$, where $E' = \{uv : uv \in E \text{ and } p(u) + p(v) = w(uv)\}$. The following lemma is a direct consequence of Egerváry's Theorem.



Lemma 6. *Consider a weighted bipartite graph $G = (U, V, E, w)$ and a minimum vertex cover p of G. The matching M is a perfect matching in G_p iff it is a maximum weight perfect matching in G.*

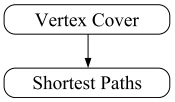
Thus if we show how to find a minimum weight bipartite vertex cover p in $\tilde{O}(Wn^\omega)$ time, then we will be able to find the equality subgraph G_p. Next we find the maximum weight bipartite matching in G by finding a perfect matching in G_p with use of Theorem 4. A similar observation to Lemma 6, but phrased with use of allowed edges, was formulated in [6].

2.2. From vertex cover to shortest paths

The following technique was given by Iri [10] and shows how to find a vertex cover using one shortest path computation. At first sight it might seems useless, because we need to know a maximum weight perfect matching first. However, in the next section we show that this knowledge is not needed and the distances given in the lemma can be computed even then. Let M be a maximum weight perfect matching in a weighted bipartite graph $G = (U, V, E, w)$. Construct a directed graph $D = (U \cup V \cup \{s\}, A, w_d)$, and

- for all edges $uv \in E$, with $u \in U$ and $v \in V$, add a directed edge (u, v) to A, with weight $w_d((u, v)) := -w(uv)$,
- for all edges $uv \in M$, with $u \in U$ and $v \in V$, add a directed edge (v, u) to A, with weight $w_d((v, u)) := w(uv)$,
- add zero weight edges (s, v) for each $v \in V$.

Let $\text{dist}_G(u, v)$ be the distance from vertex u to vertex v in the graph G.



Lemma 7 (*Iri '60*). *Set* $y(u) := \text{dist}_D(s, u)$ for $u \in U$ *and* $y(v) := -\text{dist}_D(s, v)$ for $v \in V$, *then* y *is a minimum weight vertex cover in* G.

Proof. First note that there are no negative-weight cycles in D, because otherwise we could increase the weight of M by swapping edges along such cycle. Note also that M is a perfect matching, so all vertices in U are reachable from s. Thus all distances $\text{dist}_D(s, u)$ are correctly defined.

The distances $\text{dist}_D(s, u)$ define a Belman–Ford like potential function such that $w_d((u, v)) \geq \text{dist}_D(s, v) - \text{dist}_D(s, u)$. For each edge $uv \in E$ we have a directed edge (u, v) in D and

$$w_d((u, v)) \geq \text{dist}_D(s, v) - \text{dist}_D(s, u),$$
$$-w(uv) \geq -y(v) - y(u).$$
$$w(uv) \leq y(v) + y(u).$$

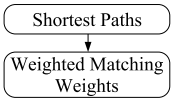
Thus y is a vertex cover. Now consider edges $uv \in M$. We have a corresponding directed edge (v, u) in D and

$$w_d((v, u)) \geq \text{dist}_D(s, u) - \text{dist}_D(s, v),$$
$$w(uv) \geq y(u) + y(v).$$

Summing up the above inequality for all edges in M we obtain that the weight of y is not greater than $w(M)$. Hence from Theorem 5 we obtain that y is a minimum. □

2.3. From shortest paths to matching weights

In this section we show how to compute the distances in D by computing the maximum weight of a perfect matching in several graphs that are variants of G. Consider a weighted bipartite graph $G = (U, V, E, w)$. Add a new vertex $s = u_{n+1}$ to U and a new vertex $t = v_{n+1}$ to V. Connect s with all vertices from V with zero weight edges, and connect the vertex t with a vertex u in U with zero weight edge as well. Let us denote by $G(u)$ the resulting graph and by $M(u)$ the maximum weight perfect matching in this graph. Note that $G(u)$ has a perfect matching because G has a perfect matching. By $G(*)$ we denote the graph where the vertex t is connected with all vertices in U.



Lemma 8. *Let* $G = (U, V, E, w)$ *be a weighted bipartite graph and let* M *be the maximum weight perfect matching* M *in* G, *then* $\text{dist}_D(s, u) = w(M) - w(M(u))$, *for all* $u \in U$.

Proof. Consider the matchings $M(u)$ and M. Direct all edges in M from V to U and all edges in $M(u)$ from U to V. Moreover, direct edges adjacent to s to point from s and edges adjacent to t to point to t. In this way we obtain a directed path p from s to t and a set $\mathcal{C}$ of even length alternating cycles (see Fig. 2). The path p corresponds to the path in graph D from s to u. And the cycles in $\mathcal{C}$ correspond to cycles in D. All these cycles have a zero weight because M and $M(u)$ are maximum. Thus the weight of p is exactly $w(M) - w(M(u))$ and $\text{dist}(r, u) \leq y_u$.

Now note that from each path p from s to u in D we can construct a perfect matching $M'(u)$ in $G(u)$. We simply take $M'(u) = p \oplus M$. Thus $w(M) - w(M'(u)) = w(p)$, and so $\text{dist}(r, u) \geq y_u$. □

Note that if the minimum vertex cover is computed correctly for U then for V it can be determined by the following equation,

$$y(v) := \max_{i \in U} \{w(vi) - y(i)\} \text{ for } v \in V. \tag{1}$$

Thus we can restrict ourselves to computing the vertex cover only for vertices in U, as in Lemma 8. The crucial observation in the above lemma is that it does not require the matchings M or $M(u)$, but it requires only their weights.

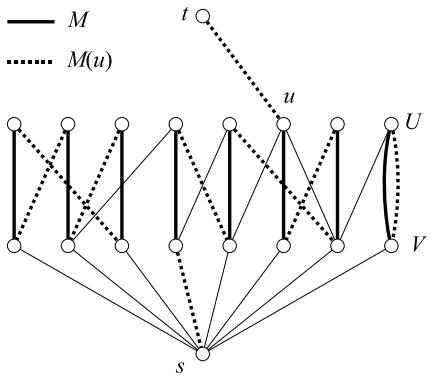


Fig. 2. Graph $G(u)$ with the matching M and $M(u)$. The bold edges have positive weights whereas the dashed edges have negative weights. The sum $M \cup M(u)$ gives a path from s to t.

2.4. From matching weights to matrix polynomials

In this subsection we show how to compute the weights of the matchings $M(u)$ with use of the matrix polynomials. We start by showing a way of computing the weight of a maximum weight perfect matching.

For a weighted bipartite graph $G = (U, V, E, w)$, define a $n \times n$ matrix $\tilde{B}(G, x)$ by

$$\tilde{B}(G,x)_{i,j} = \begin{cases} z_{i,j}x^{w(v_iv_j)} & \text{if } v_iv_j \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where $z_{i,j}$ are distinct variables corresponding to edges in G. The following is based on the observation given by Karp, Upfal and Wigderson [13], but needs to be adopted to work over closed fields.

Lemma 9. *The degree of x in $\det(\tilde{B}(G, x))$ is the weight of the maximum weight perfect matching in G. Moreover, all constant coefficients in $\det(\tilde{B}(G, x))$ are non-zero over finite field Z_p for $p \geq 2$.*

Proof. The determinant is given as the sum over all possible permutations of the products along each permutation

$$\det(\tilde{B}(G,x)) = \sum_{\pi \in \Pi_n} \operatorname{sgn}(\pi) \prod_{i=1} n\tilde{B}(G,x)_{i,\pi_i}.$$

It is not hard to see that each non-zero element of the sum corresponds to a perfect matching in G consisting of edges $i\pi_i$ for $1 \leq i \leq n$. Note, that the entries of $\tilde{B}(G, x)$ corresponding to the edges of the matching are multiplied all together, and so by the definition of $\tilde{B}(G, x)$ the degree of x in the result equals to the weight of the matching in G. Hence, the maximum degree of x equals to the maximum weight of the perfect matching. Moreover, each element of the sum is different because the permutations are different and so are the variables $z_{i,j}$ in each product. So, each constant coefficient in $\det(\tilde{B}(G, x))$ is 1 and is non-zero over Z_p for $p \geq 2$. □

The following is a direct consequence of Theorem 1, Corollary 3 and the above lemma.

Corollary 10. *The weight of the maximum weight bipartite perfect matching can be computed in $\tilde{O}(Wn^{\omega})$ time, with high probability.*

Proof. The determinant of $\tilde{B}(G, x)$ can be written as:

$$\det(\tilde{B}(G,x)) = \sum_{i=0}^{n} p_i(\{z_{i,j}\})x^i,$$

where $p_i(\{z_{i,j}\})$ are the polynomials in variables $z_{i,j}$. Note, that in order to compute the degree of x in $\det(\tilde{B}(G, x))$ we need to find the highest i such that $p_i(\{z_{i,j}\})$ is non-zero.

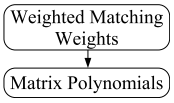
In order to compute the matching weight

- we set p to be a prime number of length $3 \log n$.
- we substitute random values from Z_p, for all variables $z_{i,j}$ in $\tilde{B}(G, x)$ in order to obtain matrix $B(G, x)$,
- we compute its determinant with use of Theorem 1.

Note that the operations in Z_p for p of logarithmic length can be carried over in constant time in RAM model. Hence, the time needed to compute the determinant $\det(B(G, x))$ is bounded by $\tilde{O}(Wn^{\omega})$. Now, with use of Corollary 3 the probability of obtaining a false zero for any of the p_i's is less then $\frac{1}{n^2} \times n = \frac{1}{n}$ because the degree of p_i in the variables $z_{i,j}$ is not higher than the degree of the determinant which in turn is not higher than n. □

The above corollary gives the way of computing the weight of the maximum weight perfect matching $w(M)$, but we also need to compute the weights $w(M(u))$, for all $u \in U$.

For a given matrix A, we denote by $A^{i,j}$ the matrix A with elements in i-th row and j-th column set to zero except that $A_{i,j} = 1$. From the definition of the adjoint we have that $\mathrm{adj}(A)_{i,j} = \det(A^{j,i})$ and $A^{-1}\det(A) = \mathrm{adj}(A)$. Moreover, let us denote by e_i the i'th basis vector in cartesian coordinates.



Lemma 11. *Given a weighted bipartite graph $G = (U, V, E, w)$, compute $z = \det(\tilde{B}(G(*), x))\tilde{B}(G(*), x)^{-1}e_{n+1}$, then $\deg_x(z_i) = w(M(u_i))$.*

Proof. We have

$$z_i = \det(\tilde{B}(G(*), x))\left(\tilde{B}(G(*), x)^{-1}e_{n+1}\right)_i = \left(\mathrm{adj}(\tilde{B}(G(*), x))e_{n+1}\right)_i$$
$$= \mathrm{adj}(\tilde{B}(G(*), x))_{n+1,i} = \det(\tilde{B}(G(*), x)^{i,n+1}) = \det(\tilde{B}(G(u_i), x)),$$

where we have noticed that the matrices $\tilde{B}(G(*), x)^{i,n+1}$ and $\tilde{B}(G(u_i), x)$ are equal. Now from Lemma 9 we obtain that

$$\deg_x(z_i) = \deg_x\left(\det(\tilde{B}(G(u_i), x))\right) = w(M(u_i)). \quad \square$$

As a consequence of Theorem 1, Corollary 3 and the above corollary, we obtain that the values $w(M(u))$ can be computed in $\tilde{O}(Wn^{\omega})$ time. We only need to take a prime number p of length $4\log n$ in order to guarantee that a false zero appears in z_i with probability less than $\frac{1}{n}$. Joining this observation with Lemma 8, Lemma 7 and Lemma 6 we obtain

Theorem 12. *Given a weighted bipartite graph $G = (U, V, E, w)$ with edge weights in the range $\{0, 1, \ldots, W\}$, a minimum weight vertex cover of G and a maximum weight perfect matching in G, can be computed in $\tilde{O}(Wn^{\omega})$ time, with high probability.*

3. Applications of the matching algorithm

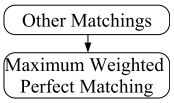
3.1. Maximum and minimum matchings

In the *minimum weight bipartite perfect matching problem* we seek a perfect matching M in a weighted bipartite graph G to minimize the total weight $w(M)$. The instance of the minimization problem can be turned into the maximization problem by defining a new weight function $w' : E \to \mathcal{Z}_+$ as

$$w'(e) = -w(e) + W.$$

By taking negative weights we turn the minimization problem into a maximization problem and by adding W we guarantee that edge weights are positive. This does not change the solution because the weight of all perfect matchings is increased by nW.

In the *maximum weight bipartite matching problem* we want to construct a matching M (not necessary perfect) in a weighted bipartite graph G, that maximizes the total weight $w(M)$. In order to obtain the instance of the perfect matching problem we simply have to add zero weight edges between all not connected vertices in G. The maximum weight perfect matching in the resulting graph corresponds to the maximum weight matching in G.



Theorem 13. *Given a weighted bipartite graph $G = (U, V, E, w)$ with edge weights in the range $\{0, 1, \ldots, W\}$ a minimum weight perfect matching, as well as, maximum weight matching in G, can be computed in $\tilde{O}(Wn^{\omega})$ time, with high probability.*

3.2. Single source shortest paths

A *weighted directed graph* G is a tuple $G = (V, E, w)$, where the vertex set is given by $V = \{v_1, \ldots, v_n\}$, $E \subseteq V \times V$ denotes the edge set, and the function $w : E \to \mathcal{Z}$ ascribes the lengths to the edges. We denote by W the maximum absolute value of the edge length. In the *single source shortest paths problem* want to compute the distances from a given vertex v to all other vertices.

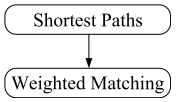
The following reduction is well known [15]. Define a bipartite graph $G' = (U', V', E', w')$ in the following way:

$$U' = \{u'_1, \ldots, u'_n\},$$
$$V' = \{v'_1, \ldots, v'_n\},$$
$$E' = \{u'_i v'_j : (v_i, v_j) \in E\} \cup \{u'_i v'_i : 1 \le i \le n\},$$
$$w'(u'_i v'_j) = \begin{cases} -w((v_i, v_j)) & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

A perfect matching in G' corresponds to a set of cycles in the graph G. If a maximum weight of a perfect matching in G' is greater than zero, there is a negative weight cycle in G. Therefore, we can detect if G has a negative length cycle in $\tilde{O}(Wn^{\omega})$ time. If this is not the case, we compute a minimum vertex cover y' of G'. There is always a zero weight matching in G', so the weight of y' is zero. Note also that $y'(u'_i) = -y'(v'_i)$, because the edges $u'_i v'_i$ are tight. Thus we have for an edge (v_i, v_j)

$$y'(u'_i) + y'(v'_j) \geq w'(u'_i v'_j)$$
$$y'(u'_i) - y'(u'_j) \geq -w((v_i, v_j))$$
$$w((v_i, v_j)) \geq y'(u'_j) - y'(u'_i).$$

We see that $p : V \to \mathbb{Z}$, defined as $p(v_i) := y'(u'_i)$, is a good potential function in G. Hence we can use it to map the edge weights to positive values conserving the shortest paths (for details see [3]). With the use of the nonnegative weights we can compute the shortest paths with Dijkstra's algorithm.



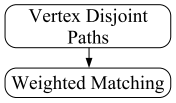
Theorem 14. *Given a weighted directed graph $G = (V, E, w)$ with edge weights in the range $\{0, 1, \ldots, W\}$ and a vertex $v \in V$, a negative cycle in G can be detected, or the distances from v can be computed, in $\tilde{O}(Wn^{\omega})$ time, with high probability.*

3.3. Minimum weight disjoint paths

Let $G = (V, E, w)$ be a weighted directed graph with two distinguished vertices $s, t \in V$. In the *minimum weight vertex disjoint s-t paths* problem we want to find the minimum weight of k vertex disjoint paths in G from s to t. The vertices s and t are common for the paths. For given G and k, let us define a bipartite graph $G^k = (U^k, V^k, E^k, w^k)$, in the following way

$$U^k = \{u_1^k, \ldots, u_n^k\} \cup \{s_1^k, \ldots, s_k^k\},$$
$$V^k = \{v_1^k, \ldots, v_n^k\} \cup \{t_1^k, \ldots, t_k^k\},$$
$$\begin{aligned} E^k := &\{u_i^k v_j^k : (v_i, v_j) \in E,\ v_i \neq s,\ v_j \neq t\} \\ &\cup \{u_i^k v_i^k : 1 \leq i \leq n\} \\ &\cup \{s_i^k v_j^k : (s, v_j) \in E,\ 1 \leq i \leq k\} \\ &\cup \{u_j^k t_i^k : (v_j, t) \in E,\ 1 \leq i \leq k\}. \end{aligned}$$
$$w^k(u_i^k v_j^k) = \begin{cases} w((v_i, v_j)) & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Consider a graph G' obtained from G by adding an edge (t, s). Similarly as above, a perfect matching in G^k corresponds to a set C of cycles in G'. These cycles are, in general, vertex disjoint, but the vertices s and t may be common. The vertices s_i^k and t_i^k cannot be matched in G^k with themselves. Hence C must contain the edge (t, s) exactly k times, because there are k copies of s and t. By removing the edge (t, s) from C we obtain k vertex disjoint paths from s to t. The minimum weight of the s-t paths corresponds to the minimum weight of the perfect matching in G^k. Moreover G^k contains a perfect matching if and only if G contains at least k vertex disjoint s-t paths.



Theorem 15. *Given a weighted directed graph $G = (V, E, w)$ with edge weights in the range $\{0, 1, \ldots, W\}$, a minimum weight of k vertex disjoint s-t paths can be computed in $\tilde{O}(Wn^{\omega})$ time, with high probability.*

4. Conclusions and open problems

4.1. Weighted matchings in general graphs

Storjohann's theorem can be applied in the general case as well. Let us define for a weighted graph $G = (V, E, w)$ a matrix

$$\tilde{A}(G, x)_{i,j} = \begin{cases} z_{i,j} x^{w(ij)} & \text{if } (i, j) \in E \text{ and } i < j, \\ -z_{j,i} x^{w(ij)} & \text{if } (i, j) \in E \text{ and } i > j, \\ 0 & \text{otherwise.} \end{cases}$$

The following is the generalization of Lemma 9 to general graphs. The following lemma is again an adoption of results from [13]. However, as it has not been formulated over Z_p previously we give here the whole proof.

Lemma 16. *The degree of x in $\det(\tilde{A}(G,x))$ is twice the weight of the maximum weight perfect matching in G. Moreover, all constant coefficients in $\det(\tilde{B}(G,x))$ are non-zero over finite field Z_p for $p \geq 3$.*

Proof. Let us write the determinant as the following sum

$$\det(\tilde{A}(G,x)) = \sum_{\pi \in \Pi_n} \operatorname{sgn}(p) \prod_{i=1}^{n} \tilde{A}(G,x)_{i,\pi_i}. \tag{2}$$

Each permutation π corresponds to a directed cycle cover $\mathcal{C}$ of G given by edges (i, π_i). If $\mathcal{C}$ contains an odd length cycle we can reverse the direction of this cycle, and change the sign of the contribution given to (2)—from antisymmetry of $\tilde{A}(G,x)$. Thus a non-zero contribution is given only by permutations corresponding to even length cycle covers.

A maximum weight perfect matching M in G can be transformed into a cycle cover $\mathcal{C}$ of G composed of cycles of length 2. Each edge of M appears in $\mathcal{C}$ twice and so

$$\deg_x(\det(\tilde{A}(G,x))) \geq 2w(M).$$

Let us now take the highest degree term in (2) corresponding to an even length cycle cover $\mathcal{C}$. From this cycle cover we can obtain two perfect matchings M_1 and M_2 by alternately taking edges from each cycle. The degree of this term is equal to $w(M_1) + w(M_2)$. Thus

$$\deg_x(\det(\tilde{A}(G,w,x))) \leq w(M_1) + w(M_2) \leq 2w(M),$$

by the maximality of M.

Now note that each term with fixed set of variables $z_{i,j}$ in determinant $\det(\tilde{A}(G,x))$ is a sum of several terms resulting from all possible orientations of the cycles in the given undirected cycle cover of the graph. For a cycle cover with k cycles there are 2^k possible orientations. Hence each constant coefficient is a power of 2 and so is non-zero in Z_p for $p \geq 3$. □

Similarly as in the bipartite case we obtain the following corollary.

Corollary 17. *The weight of the maximum weight perfect matching can be computed in $\tilde{O}(Wn^{\omega})$ time, with high probability.*

The corollary can be shown in exactly the same way as Corollary 10.

4.2. Conclusions

We have shown that the algebraic approach to finding a perfect matching can be used in the case of weighted bipartite graphs as well. We have presented an algorithm solving this problem in $\tilde{O}(Wn^{\omega})$ time. For small edge weights this improves over previous fastest algorithms for this problem that work in $\tilde{O}(n^{2.5}\log(nW))$ time [8] and in $\tilde{O}(nm)$ time [5]. We have also shown how to compute the maximum weight of a matching in general graphs in $\tilde{O}(Wn^{\omega})$ time. Nevertheless, finding this matching remains an open problem. Similar ideas cannot be used for the general case because the approach of Lemma 6 does not work there. It is impossible to construct a subgraph of a graph that will contain all the maximum weight perfect matchings and no other perfect matchings. The following counterexample was given by Eppstein [6]: let $G = K_6$ with edge weights two on two disjoint triangles and unit edge weights on the remaining edges. For each weight one edge is in a unique maximum weight matching, so no edges can be removed from G. Thus in the general case one has to use a different approach. The algorithm presented here uses the fact that the dual problem is pretty simple and it can be solved using the adjoint matrix. In the general case we have to compute so called Edmonds sets, i.e. blossoms. The problem here is that they are not defined unambiguously, so it seems improbable that they could be obtained in a simple way from the adjoint of $\tilde{A}(G,x)$. This is a similar situation as in the case of the NC algorithms for perfect matchings in planar graphs, where the algorithm for the general case is not known. It also seems that we are facing here exactly the same problem, i.e., how to compute Edmonds sets in polylogarithmic time.

The most intriguing open problem is whether the dependence of the complexity on W in our algorithm can be reduced. Such reduction would imply also a faster algorithm for the SSSP problem.

References

[1] R. Bellman, On a routing problem, Quarterly of Applied Mathematics 16 (1) (1958) 87–90.
[2] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, Journal of Symbolic Computation 9 (3) (1990) 251–280.
[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge Mass, 1990.
[4] E.A. Dinic, M.A. Kronrod, An algorithm for the solution of the assignment problem, Soviet Mathematics Doklady 10 (1969) 1324–1326.
[5] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, Journal of ACM 19 (2) (1972) 248–264.
[6] David Eppstein, Representing all Minimum Spanning Trees with Applications to Counting and Generation. Technical Report ICS-TR-95-50, 1995.
[7] H.N. Gabow, Scaling algorithms for network problems, Journal of Computer and System Sciences 31 (2) (1985) 148–168.
[8] H.N. Gabow, R.E. Tarjan, Faster scaling algorithms for network problems, SIAM Journal on Computing 18 (5) (1989) 1013–1036.
[9] Andrew V. Goldberg, Scaling algorithms for the shortest paths problem, SIAM Journal on Computing 24 (3) (1995) 494–504.
[10] M. Iri, A new method for solving transportation-network problems, Journal of the Operations Research Society of Japan 3 (1960) 27–87.
[11] L.R. Ford Jr., Network Flow Theory. Paper P-923, The RAND Corperation, Santa Moncia, California, August 1956.

[12] M.-Y. Kao, T.W. Lam, W.-K. Sung, H.-F. Ting, A decomposition theorem for maximum weight bipartite matchings with applications to evolutionary trees, SIAM Journal on Computing 31 (1) (2001) 18–26.
[13] R.M. Karp, E. Upfal, A. Wigderson, Constructing a perfect matching is in random nc, Combinatorica 6 (1) (1986) 35–48.
[14] H.W Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1955) 83–97.
[15] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York, 1976.
[16] L. Lovász, On determinants, matchings and random algorithms, in: L. Budach (Ed.), Fundamentals of Computation Theory, Akademie-Verlag, 1979, pp. 565–574.
[17] E.F. Moore, The shortest path through a maze, in: Proceedings of the International Symposium on the Theory of Switching, Harvard University Press, 1959, pp. 285–292.
[18] Marcin Mucha, Piotr Sankowski, Maximum matchings via Gaussian elimination, in: Proceedings of the 45th annual IEEE Symposium on Foundations of Computer Science, 2004, pp. 248–255.
[19] K. Mulmuley, U.V. Vazirani, V.V. Vazirani, Matching is as easy as matrix inversion, Combinatorica 7 (1) (1987) 105–113.
[20] J. Munkres, Algorithms for the assignment and transportation problems, Journal of SIAM 5 (1) (1957) 32–38.
[21] M.O. Rabin, V.V. Vazirani, Maximum matchings in general graphs through randomization, Journal of Algorithms 10 (1989) 557–567.
[22] P. Sankowski, Shortest paths in matrix multiplication time, in: Proceedings of the 13th Annual European Symposium on Algorithms, in: LNCS, vol. 3669, 2005, pp. 770–778.
[23] J. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, Journal of the ACM 27 (1980) 701–717.
[24] A. Shimbel, Structure in communication nets, in: Proceedings of the Symposium on Information Networks, Polytechnic Press of the Polytechnic Institute of Brooklyn, Brooklyn, 1955, pp. 199–203.
[25] Arne Storjohann, High-order lifting and integrality certification, Journal of Symbolic Computing 36 (3–4) (2003) 613–648.
[26] R. Yuster, U. Zwick, Answering distance queries in directed graphs using fast matrix multiplication, in: Proceedings of the 46th Annual Symposium on Foundations of Computer Science, IEEE, 2005, pp. 90–100.
[27] R. Zippel, Probabilistic algorithms for sparse polynomials, in: International Symposium on Symbolic and Algebraic Computation, in: Lecture Notes in Computer Science, vol. 72, Springer-Verlag, Berlin, 1979, pp. 216–226.