A lower bound for approximating the geometric minimum weight matching

Gautam Das* Michiel Smid†

January 6, 1999

Abstract

Given a set $S$ of $2n$ points in $\mathbb{R}^d$, a perfect matching of $S$ is a set of $n$ edges such that each point of $S$ is a vertex of exactly one edge. The weight of a perfect matching is the sum of the Euclidean lengths of all edges. Rao and Smith have recently shown that there is a constant $r > 1$, that only depends on the dimension $d$, such that a perfect matching whose weight is less than or equal to $r$ times the weight of a minimum weight perfect matching can be computed in $O(n \log n)$ time. We show that this algorithm is optimal in the algebraic computation tree model.

**Keywords**: Minimum weight matching, lower bounds, computational geometry.

1 Introduction

Let $S$ be a set of $2n$ points in $\mathbb{R}^d$, where $d \geq 1$ is a (small) constant. We consider sets of edges having the points of $S$ as vertices. Such a set $M$ is called a perfect matching of $S$, if each point of $S$ is a vertex of exactly one edge in $M$. In other words, a perfect matching is a partition of $S$ into $n$ subsets of size two. The weight $\text{wt}(M)$ of a perfect matching $M$ is defined as the sum of the Euclidean lengths of all edges in $M$. The minimum weight matching $\text{MWM}(S)$ of $S$ is the perfect matching of $S$ that has minimum weight.

*Department of Mathematical Sciences, The University of Memphis, Memphis, TN 38152, USA. E-mail: dasg@msci.memphis.edu.

†Department of Computer Science, University of Magdeburg, D-39106 Magdeburg, Germany. E-mail: michiel@ig.cs.uni-magdeburg.de.
The best known algorithm that computes a minimum weight matching is due to Vaidya [6]; its running time is bounded by $O(n^{5/2}(\log n)^4)$ if $d = 2$, and $O(n^{3-1/c^d})$ if $d > 2$, for some constant $c > 1$.

Rao and Smith [5] considered the easier problem of approximating the minimum weight matching. Let $r > 1$ be a real number. A perfect matching $M$ of $S$ is called an $r$-approximate $MWM$, if $wt(M) \leq r \cdot wt(MWM(S))$. Rao and Smith have shown that an $r$-approximate $MWM$, for

$$r = c \cdot \exp(8 \cdot 2^{1-1/(d-1)} \sqrt{d})$$

where $c$ is a constant, can be computed in $O(n \log n)$ time.

In this paper, we will show that Rao and Smith’s algorithm is optimal in the algebraic computation tree model. That is, we will prove the following theorem.

**Theorem 1** Let $d \geq 1$ be an integer. Every algebraic computation tree algorithm that, when given a set of $2n$ points in $\mathbb{R}^d$ and a real number $r > 1$, computes an $r$-approximate $MWM$, has worst-case running time $\Omega(n \log n)$.

Note that this lower bound even holds for dimension $d = 1$. Moreover, it holds for any approximation factor $r$, even one that depends on $n$. For example, computing a $2^{2n}$-approximate $MWM$ has worst-case running time $\Omega(n \log n)$.

Our proof of Theorem 1 uses Ben-Or’s theorem [1]. The proof technique that we use is related to those used in Chen, Das and Smid [2], and Das, Kapoor and Smid [3].

## 2 The proof of Theorem 1

In this section, we prove Theorem 1 for the case when $d = 1$. Clearly, this implies an $\Omega(n \log n)$ lower bound for any dimension $d \geq 1$.

We assume that the reader is familiar with the algebraic computation tree model. (See Ben-Or [1], and Preparata and Shamos [4].) Our lower bound will use the following well known result.

**Theorem 2 (Ben-Or [1])** Let $V$ be any set in $\mathbb{R}^n$ and let $B$ be any algorithm that belongs to the algebraic computation tree model and that accepts $V$. Let $\#V$ denote the number of connected components of $V$. Then the worst-case running time of $B$ is $\Omega(\log \# V - n)$. 

2
Let $\mathcal{A}$ be an arbitrary algebraic computation tree algorithm that, when given as input a sequence of $2n$ real numbers $x_1, x_2, \ldots, x_{2n}$ and a real number $r > 1$, computes an $r$-approximate $MWM$ for the $x_i$’s. We will use Theorem 2 to prove that $\mathcal{A}$ has worst-case running time $\Omega(n \log n)$.

Note that algorithm $\mathcal{A}$ solves a computation problem. In order to apply Theorem 2, we need a decision problem, i.e., a problem having values YES and NO. Below, we will define such a decision problem; in fact, we will define the corresponding subset $V \subseteq \mathbb{R}^{2n}$ of YES-inputs.

Fix the integer $n$ and the real number $r > 1$. We define an algorithm $\mathcal{B}$ that takes as input any sequence of $2n$ real numbers. On input sequence $x_1, x_2, \ldots, x_{2n}$, algorithm $\mathcal{B}$ does the following.

**Step 1.** Check if $x_i = i$, for all $i$, $1 \leq i \leq n$. If not, output NO, and terminate. Otherwise, go to Step 2.

**Step 2.** Let $\epsilon := 1/(2rn)$. Run algorithm $\mathcal{A}$ on the input $x_1, x_2, \ldots, x_{2n}, r$.

Let $M$ be the $r$-approximate $MWM$ that is computed by $\mathcal{A}$. Check if all edges of $M$ have length $\epsilon$. If so, output YES. Otherwise, output NO.

Let $T_\mathcal{A}(n)$ and $T_\mathcal{B}(n)$ denote the worst-case running times of algorithms $\mathcal{A}$ and $\mathcal{B}$, respectively. Then, it is clear that

$$T_\mathcal{B}(n) \leq T_\mathcal{A}(n) + cn,$$

for some constant $c$. Therefore, if we can show that $T_\mathcal{B}(n) = \Omega(n \log n)$, then it follows immediately $T_\mathcal{A}(n) = \Omega(n \log n)$.

Let $V$ be the set of all points $(x_1, x_2, \ldots, x_{2n})$ in $\mathbb{R}^{2n}$ that are accepted by algorithm $\mathcal{B}$. We will show that $V$ has at least $n!$ connected components. As a result, Theorem 2 implies the $\Omega(n \log n)$ lower bound on the running time of $\mathcal{B}$.

**Lemma 1** Let $\pi$ be any permutation of $1, 2, \ldots, n$, and let $\epsilon = 1/(2rn)$. Then the point

$$P := (1, 2, \ldots, n, \pi(1) + \epsilon, \pi(2) + \epsilon, \ldots, \pi(n) + \epsilon)$$

is contained in the set $V$.

**Proof.** Let $M^*$ be the $MWM$ of the elements $1, 2, \ldots, n, \pi(1) + \epsilon, \pi(2) + \epsilon, \ldots, \pi(n) + \epsilon$. Since $0 < \epsilon < 1/2$, it is easy to see that $M^*$ consists of the edges $(i, i + \epsilon), 1 \leq i \leq n$. 

3
Consider what happens when algorithm \( B \) is run on input \( P \). Clearly, this input “survives” Step 1. Let \( M \) be the \( r \)-approximate MWM that is 
computed in Step 2. We will show below that \( M = M^* \). Having proved this, 
it follows that algorithm \( B \) accepts the input \( P \), i.e., \( P \in V \).

Suppose that \( M \neq M^* \). Then \( M \) contains an edge of the form \((i, j)\), 
\((i, j + \epsilon)\), or \((i + \epsilon, j + \epsilon)\), for some integers \( i \) and \( j \), \( i \neq j \). (We consider 
edges to be undirected.) Since \( 0 < \epsilon < 1/2 \), it follows that this edge and, 
also the matching \( M \), has weight more than 1/2. Clearly, the optimal 
matching \( M^* \) has weight \( nc = 1/(2r) \). Therefore, \( wt(M) > 1/2 = r \cdot wt(M^*) \). 
This is a contradiction, because \( M \) is an \( r \)-approximate MWM.

**Lemma 2** The set \( V \) has at least \( n! \) connected components.

**Proof.** Let \( \pi \) and \( \rho \) be two different permutations of \( 1, 2, \ldots, n \). Consider 
the points 

\[
P := (1, 2, \ldots, n, \pi(1) + \epsilon, \pi(2) + \epsilon, \ldots, \pi(n) + \epsilon)
\]

and 

\[
R := (1, 2, \ldots, n, \rho(1) + \epsilon, \rho(2) + \epsilon, \ldots, \rho(n) + \epsilon).
\]

in \( \mathbb{R}^{2n} \). By Lemma 1, both these points are contained in the set \( V \). We will 
show that they are in different connected components of \( V \).

Let \( C \) be an arbitrary curve in \( \mathbb{R}^{2n} \) that connects \( P \) and \( R \). Since \( \pi \) and 
\( \rho \) are distinct permutations, there are indices \( i \) and \( j \) such that \( \pi(i) < \pi(j) \) 
and \( \rho(i) > \rho(j) \). Hence, the curve \( C \) contains a point \( Q \), 

\[
Q = (p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_n),
\]

such that \( q_i = q_j \). We claim that \( Q \) is not contained in \( V \). This will prove 
that \( P \) and \( R \) are in different connected components of \( V \).

To prove the claim, first assume that there is an index \( k \), \( 1 \leq k \leq n \), such 
that \( p_k \neq k \). Then point \( Q \) is rejected by algorithm \( B \) and, therefore, \( Q \not\in V \). 
Hence, we may assume that 

\[
Q = (1, 2, \ldots, n, q_1, q_2, \ldots, q_n).
\]

Let us see what happens if we run algorithm \( B \) on input \( Q \). This input 
“survives” Step 1. Let \( M \) be the \( r \)-approximate MWM that is constructed 
in Step 2.

If \( M \) contains an edge of the form \((p_k, p_\ell) = (k, \ell)\), then algorithm \( B \) 
rejects point \( Q \), because such an edge has length more than \( \epsilon \). Hence, we 
may assume that each edge of \( M \) has the form \((p_k, q_\ell) = (k, q_\ell) \). Let \( a \) and 
\( b \) be the integers, \( 1 \leq a, b \leq n \), such that \((a, q_i) \) and \((b, q_j) \) are edges of \( M \). 
Since (i) \( a \) and \( b \) are distinct integers, (ii) \( q_i = q_j \), and (iii) \( 0 < \epsilon < 1/2 \), one 
of these two edges must have length more than \( \epsilon \). Hence, algorithm \( B \) rejects 
point \( Q \). This completes the proof. ■
References


