ELSEVIER

Computational Geometry 9 (1998) 247–256

Computational Geometry
Theory and Applications

On determining the congruence of point sets in d dimensions ☆

Tatsuya Akutsu [1]

Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108, Japan

Communicated by E. Welzl; submitted 14 November 1995; accepted 28 January 1997

Abstract

This paper considers the following problem: given two point sets A and B ($|A| = |B| = n$) in d dimensional Euclidean space, determine whether or not A is congruent to B. This paper presents an $O(n^{(d-1)/2} \log n)$ time randomized algorithm. The birthday paradox, which is well-known in combinatorics, is used effectively in this algorithm. Although this algorithm is Monte-Carlo type (i.e., it may give a wrong result), this improves a previous $O(n^{d-2} \log n)$ time deterministic algorithm considerably. This paper also shows that if d is not bounded, the problem is at least as hard as the graph isomorphism problem in the sense of the polynomiality. Several related results are described too. © 1998 Elsevier Science B.V.

Keywords: Pattern matching; Congruence; Birthday paradox; Randomized algorithm

1. Introduction

Geometric pattern matching problems have been studied extensively in computational geometry [4,5,13]. Most of such studies have been done for approximate matchings in two or three dimensions. Few studies for exact matchings in general dimensions have been done. This paper studies a basic problem of exact matching: the problem of deciding the congruence of two point sets in general dimensions.

Several studies have been done for exact matchings. $O(n \log n)$ time algorithms for determining the congruence of various objects in two dimensions were developed by Atallah [6], Highnam [14] and Manacher [15]. Sugihara developed an $O(n \log n)$ time algorithm for determining the congruence of two polyhedra in three dimensions [18]. Atkinson developed an $O(n \log n)$ time algorithm for determining the congruence of two point sets in three dimensions [7]. Alt et al. developed an $O(n^{d-2} \log n)$ time algorithm for determining the congruence of two point sets in d dimensions [5]. Rezende and Lee studied the following exact matching problem: given point sets P and S, determine whether or not P matches any subset of S by translation, rotation, reflection and global scaling [17].

☆ A preliminary version of this paper has appeared in [2].

[1] E-mail: takutsu@ims.u-tokyo.ac.jp.

PII S0925-7721(97)00010-2

In this paper, we present an $O(n^{(d-1)/2}\log n)$ time Monte-Carlo type randomized algorithm for deciding whether or not two point sets A and B ($|A| = |B| = n$) are congruent in d dimensional Euclidean space ($d > 3$). Although our algorithm is a randomized one, this improves the previous result [5] considerably. Moreover, we show that if d is not bounded, the congruence problem is at least as hard as the graph isomorphism problem in the sense of the polynomiality. Several related results are described too.

Recently, Matoušek suggested that an $O(n^{d/2+O(1)})$ time deterministic algorithm and an $O(n^{d/4+O(1)})$ time Monte-Carlo type randomized algorithm might be obtained for the congruence problem [16]. We briefly describe his idea here for the readers' sake. In this paper, the d dimensional congruence problem is reduced to the $d-1$ dimensional congruence problems by choosing $O(\sqrt{n})$ points randomly. In his method, the d dimensional congruence problem is reduced to the $d-2$ dimensional congruence problems by using the smallest distance point pairs. Since the number of smallest distance pairs is $O(n)$ and they can be computed in $O(n\log n)$ time, an $O(n^{d/2+O(1)})$ time deterministic algorithm is obtained. Combining with the birthday paradox idea used in this paper (i.e., choosing $O(\sqrt{n})$ smallest distance pairs randomly), an $O(n^{d/4+O(1)})$ time Monte-Carlo type randomized algorithm is obtained.

2. Preliminaries

Let E^d denote the d dimensional Euclidean space. For two points $\boldsymbol{p}, \boldsymbol{q} \in E^d$, $\overline{\boldsymbol{pq}}$ denotes a line segment between $\boldsymbol{p}$ and $\boldsymbol{q}$, and $\|\overline{\boldsymbol{pq}}\|$ denotes the length of it. For a point set $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, the *centroid* of P is the point given by $\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i$, and $\dim(P)$ denotes the dimensions of the affine hull of P.

A mapping T of E^d onto itself is said to be *isometric* if $\|\overline{\boldsymbol{pq}}\| = \|\overline{T(\boldsymbol{p})T(\boldsymbol{q})}\|$ for all points $\boldsymbol{p}$ and $\boldsymbol{q}$. Let $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ and $B = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ denote point sets in E^d respectively. For A, we define $T(A) = \{T(\boldsymbol{a}_i) \mid \boldsymbol{a}_i \in A\}$. If there exists an isometric mapping which satisfies $B = T(A)$, A and B are said to be *congruent*. If A and B are congruent, we write $A \cong B$. $\boldsymbol{a}_i \in A$ and $\boldsymbol{b}_j \in B$ are called *equivalent* if $\boldsymbol{b}_j = T(\boldsymbol{a}_i)$ holds for some isometric mapping T such that $B = T(A)$. An isometric mapping T can be written in the form $T : \boldsymbol{p} \mapsto M\boldsymbol{p} + \boldsymbol{c}$ where M is a $d \times d$ orthonormal real matrix, i.e., $M^{\mathrm{T}} = M^{-1}$, and $\boldsymbol{c}$ is a d-vector. T is called a mapping of the *first* (respectively *second*) *kind* if $\det(M) = +1$ (respectively -1). Since any T of the second kind can be written as $\boldsymbol{p} \mapsto MJ\boldsymbol{p} + \boldsymbol{c}$ where $J(x_1, x_2, \ldots, x_d) = (-x_1, x_2, \ldots, x_d)$ and $\det(M) = +1$ [5], we only consider isometric mappings of the first kind.

In this paper, we also consider the congruence of labeled point sets. In such a case, A and B are congruent ($A \cong B$) if there exists an isometric mapping T such that $B = T(A)$ holds and $\mathrm{label}(T(\boldsymbol{a}_i)) = \mathrm{label}(\boldsymbol{a}_i)$ holds for all $\boldsymbol{a}_i \in A$. $\boldsymbol{a}_i \in A$ and $\boldsymbol{b}_j \in B$ are called equivalent if $\boldsymbol{b}_j = T(\boldsymbol{a}_i)$ holds for such an isometric mapping.

For each (labeled) point set $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, a point sequence $C(P) = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n)$ is called a *canonical form* if it satisfies the following conditions: $C(P) \cong P$ when $C(P)$ is treated as a point set; $C(A) = C(B)$ if and only if $A \cong B$. Note that, once canonical forms are computed, whether $A \cong B$ or not can be determined in $O(n)$ time by simply comparing $C(A)$ and $C(B)$ (i.e., by testing whether or not two sequences of real numbers are identical).

3. A randomized algorithm for congruence

In this section, we present a randomized algorithm for deciding the congruence of two point sets in d dimensions, where we assume that d ($d > 3$) is a fixed constant. Note that in this section and in the next section, we adopt a random access machine (RAM) as a model of computation. Furthermore, we assume that the machine can represent arbitrary real numbers and can exactly perform all the geometric computations involved (e.g., determining angles, distances, etc.) without round-off-errors.

3.1. Birthday paradox

The birthday paradox is well-known in combinatorics [10]. It states that on the average, 24 persons are needed for at least two of them to have the same birthday, assuming all birth dates to be equally distributed over a year. If there were n days in a year, $\Theta(\sqrt{n})$ persons would be needed. The birthday paradox has been applied to several algorithms [10,12]. For applying the birthday paradox to the congruence problem, the following observation is useful: if $A \cong B$, and a set of $O(\sqrt{n})$ points A' (respectively B') is chosen randomly from A (respectively B), there exists at least one pair of equivalent points $(\boldsymbol{a}_i, \boldsymbol{b}_j) \in A' \times B'$ with high probability. Once an equivalent point pair is given, the congruence problem in d dimensions can be reduced to the congruence problem in $(d-1)$ dimensions using a similar reduction as in [5]. Thus reducing the problem recursively, we can solve the congruence problem.

3.2. Algorithm

The following procedure `CheckCongruence`$(\{A_1, \ldots, A_h\}, \{B_1, \ldots, B_k\}, d)$ determines whether or not there exists at least one pair (A_i, B_j) such that $A_i \cong B_j$, where A_i's and B_j's are labeled point sets in d dimensions. Note that we can assume without loss of generality that $\dim(A_i) = d$ (respectively $\dim(B_i) = d$) for all A_i (respectively B_i) because $\dim(A)$ can be determined in linear time for fixed d using the well known orthonormalization method of Schmidt, and $\dim(A) = \dim(B)$ if $A \cong B$.

Procedure `CheckCongruence`$(\{A_1, \ldots, A_h\}, \{B_1, \ldots, B_k\}, d)$
begin
  if $d = 3$ **then**
    begin
      if there is a pair (A_i, B_j) such that $A_i \cong B_j$ (#1)
      then output 'YES' **else** output 'NO';
      halt
    end;
  for all A_i **do**
    Choose a point set $A_i' \subset A_i$ randomly such that A_i' does not contain the centroid of A_i and $|A_i'| = \min(|A_i| - 1, \lceil K\sqrt{n} \rceil)$, where K is a constant to be determined later;
  for all B_i **do** Choose $B_i' \subset B_i$ randomly in the same way;
  for all A_i and $\boldsymbol{a}_j \in A_i'$ **do** $A_{ij} \leftarrow \mathrm{proj}(A_i, \boldsymbol{a}_j)$; (#2)

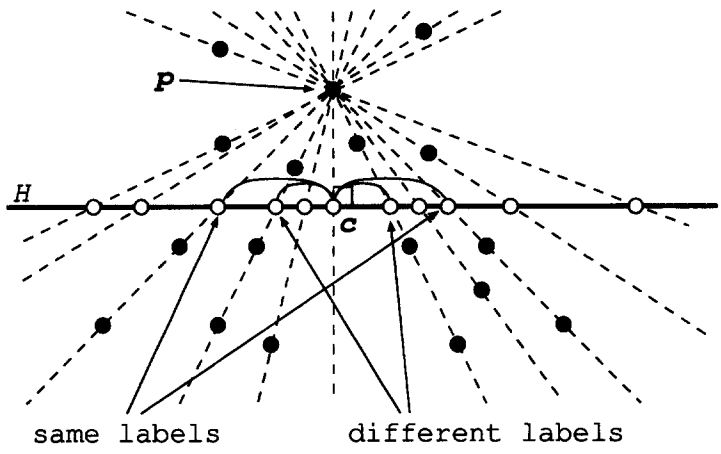

Fig. 1. Projection from d dimensions to $d-1$ dimensions.

for all B_i and $\boldsymbol{b}_j \in B'_i$ **do** $B_{ij} \leftarrow \mathrm{proj}(B_i, \boldsymbol{b}_j)$; (#3)
CheckCongruence$(\{A_{11}, A_{12}, \ldots, A_{21}, \ldots\}, \{B_{11}, B_{12}, \ldots, B_{21}, \ldots\}, d-1)$
end

In the above procedure, a d dimensional point set A_i (respectively B_i) is reduced to the $(d-1)$ dimensional point set $\mathrm{proj}(A_i, \boldsymbol{p})$ (respectively $\mathrm{proj}(B_i, \boldsymbol{p})$) where $\boldsymbol{p}$ is not the centroid of A_i (respectively B_i). This projection must satisfy the property that $\mathrm{proj}(A_i, \boldsymbol{p})$ and $\mathrm{proj}(B_j, \boldsymbol{q})$ are congruent if and only if A_i and B_j are congruent and $\boldsymbol{p}$ and $\boldsymbol{q}$ are equivalent. Such a point set $\mathrm{proj}(A_i, \boldsymbol{p})$ can be computed in the following way (see also Fig. 1). A similar procedure is used in [5].

Let $\boldsymbol{c}$ be the centroid of A_i. Let H be the hyperplane such that $\boldsymbol{c} \in H$ and $\overline{\boldsymbol{pc}}$ is perpendicular to H, and let H' be the hyperplane such that $\boldsymbol{p} \in H'$ and H' is parallel to H. If $\boldsymbol{a} \in A_i$ lies on H' (degenerated case), we replace $\boldsymbol{a}$ with $\boldsymbol{a} + \delta\overrightarrow{\boldsymbol{pc}}$ where δ is a sufficiently small constant. Then, $\mathrm{proj}(A_i, \boldsymbol{p})$ is a set of points $\boldsymbol{q} \in H$ such that $\overline{\boldsymbol{pq}}$ is parallel to $\overline{\boldsymbol{pa}}$ for some $\boldsymbol{a} \in A_i$. Next, each point in $\mathrm{proj}(A_i, \boldsymbol{p})$ is labeled so that $\boldsymbol{q}$ and $\boldsymbol{r}$ are labeled with the same integer number if and only if $Q \cup \{\boldsymbol{p}, \boldsymbol{c}\} \cong R \cup \{\boldsymbol{p}, \boldsymbol{c}\}$ holds by an isometric mapping which does not move $\boldsymbol{p}$ or $\boldsymbol{c}$, where $Q = \{\boldsymbol{a} \in A_i \mid \boldsymbol{a}$ is projected to $\boldsymbol{q}\}$ and $R = \{\boldsymbol{a} \in A_i \mid \boldsymbol{a}$ is projected to $\boldsymbol{r}\}$. This labeling procedure is done simultaneously for all projected points in the same depth of the recursion so that equivalent points can have the same label. Note that this labeling procedure can be done in $\mathrm{O}(Ln\log(Ln))$ time using an optimal sorting algorithm (e.g., merge sort), where L is the total number of sets (A_{ij}'s and B_{ij}'s) in the same depth of the recursion.

3.3. Analysis

First we analyze the time complexity and the space complexity. Note that CheckCongruence$(\{A\}, \{B\}, d)$ is invoked in order to decide the congruence of A and B in d dimensions, where $|A| = |B| = n$.

Lemma 3.1. `CheckCongruence`$(\{A\},\{B\},d)$ *works in* $O(n^{(d-1)/2}\log n)$ *time, using* $O(n^{(d-1)/2})$ *space.*

Proof. `CheckCongruence` is executed at most $d-2$ times recursively. Note that the number of point sets appearing in the arguments of the recursive execution of ith depth is

$$O\left(\left(\sqrt{n}\right)^{i-1}\right) = O\left(n^{(i-1)/2}\right).$$

Here, we analyze the time complexity for each recursive step, where m denotes the number of point sets contained in the arguments (i.e., $m = h + k$). To analyze the time complexity, we only consider the crucial parts (#1), (#2) and (#3).

In part (#1), we do not directly compare every pair (A_i, B_j). Instead, we first compute the canonical forms of A_i's and B_i's and then we sort all the canonical forms in the lexicographic order. Using the sorted list, we partition a set of A_i's and B_i's into blocks so that each block consists of point sets having identical canonical forms. Since $A_i \cong B_j$ holds if and only if both A_i and B_j belong to the same block, it is easy to test whether or not there is a pair (A_i, B_j) such that $A_i \cong B_j$. The canonical form of a (labeled) point set of size N in three dimensions can be computed in $O(N \log N)$ time [1,3]. Thus the canonical forms can be computed in $O(mn \log n)$ time. Since the size of each canonical form is $O(n)$, comparison of two canonical forms can be done in $O(n)$ time, and thus sorting can be done in $O(mn \log m)$ time. Therefore part (#1) can be executed in $O(mn \log(mn))$ time.

The time required for (#2) and (#3) is $O(mn^{3/2}\log(mn^{3/2}))$ since $O(m\sqrt{n})$ point sets are projected. Note that these parts are not executed in the last step ($(d-2)$th step) of the recursion.

Now we consider the total computation time. The total computation time for (#1) is

$$O\left((d-2)n^{(d-3)/2} n \log\left(n^{(d-3)/2} n\right)\right) = O\left(n^{(d-1)/2}\log n\right),$$

since d is assumed to be a constant. The total computation time for (#2) and (#3) is

$$O\left((d-3)n^{(d-4)/2} n^{3/2} \log\left(n^{(d-4)/2} n^{3/2}\right)\right) = O\left(n^{(d-1)/2}\log n\right).$$

Therefore the time complexity is $O(n^{(d-1)/2}\log n)$ in total.

Since $O((d-2)n^{(d-3)/2})$ sets are constructed in total, the space complexity is $O(n^{(d-1)/2})$. □

Next, we analyze the probability that the procedure succeeds. The following lemma is proved in a straight-forward way.

Lemma 3.2. *If* `CheckCongruence`$(\{A\},\{B\},d)$ *outputs 'YES', then* $A \cong B$.

The following lemma is a variant of the birthday paradox.

Lemma 3.3. *If two subsets* $S_1 \subset S$ *and* $S_2 \subset S$, *each of which consists of at least* $\sqrt{\ln(1/(1-q))\, n}$ *elements, are chosen randomly from* S ($|S| = n$), *then* $|S_1 \cap S_2| \neq \emptyset$ *holds with probability at least* q.

Proof. Let $P(n,m)$ denote the probability that $|S_1 \cap S_2| \neq \emptyset$ holds if S_1 and S_2 such that $|S_1| = |S_2| = m$ are chosen randomly from S. Then the following inequality holds:

$$P(n,m) = 1 - \left(\frac{n-m}{n}\right)\left(\frac{n-m-1}{n-1}\right)\left(\frac{n-m-2}{n-2}\right)\cdots\left(\frac{n-2m+1}{n-m+1}\right) \geqslant 1 - \left(\frac{n-m}{n}\right)^m.$$

Thus it is sufficient that

$$1 - \left(\frac{n-m}{n}\right)^m \geqslant q$$

holds. Using the following inequalities:

$$m \ln\left(\frac{n-m}{n}\right) \leqslant \ln(1-q),$$

$$\ln\left(1-\frac{m}{n}\right) \leqslant \frac{\ln(1-q)}{m},$$

$$\frac{1}{m}\ln\left(\frac{1}{1-q}\right) \leqslant \left(\frac{m}{n}\right) + \frac{1}{2}\left(\frac{m}{n}\right)^2 + \frac{1}{3}\left(\frac{m}{n}\right)^3 + \cdots$$

$$(\text{from } \ln(1-x) = -x - x^2/2 - x^3/3 - \cdots),$$

it is sufficient that

$$\frac{m}{n} \geqslant \frac{1}{m}\ln\left(\frac{1}{1-q}\right)$$

holds. Thus the lemma holds. □

Theorem 3.4. *The congruence of two point sets in d dimensions can be tested in $O(n^{(d-1)/2}\log n)$ time using $O(n^{(d-1)/2})$ space by a Monte-Carlo type randomized algorithm, where error occurs only in the case that two point sets are congruent and the error probability can be made smaller than any fixed constant $p > 0$.*

Proof. Let

$$K = \sqrt{\ln\left(\frac{1}{1-(1/2)^{1/(d-3)}}\right)}$$

in procedure `CheckCongruence`. Then, if there is a pair (A_i, B_j) such that $A_i \cong B_j$, a pair of equivalent points $(\boldsymbol{a}, \boldsymbol{b})$ is contained in $A_i' \times B_j'$ with probability at least $(1/2)^{1/(d-3)}$ (from Lemma 3.3). Thus `CheckCongruence`$(\{A\}, \{B\}, d)$ outputs 'YES' with probability at least

$$\left(\left(\frac{1}{2}\right)^{1/(d-3)}\right)^{d-3} = \frac{1}{2} \quad \text{if } A \cong B.$$

Repeating this procedure $\lceil \log(1/p) \rceil$ times the error probability can be made smaller than p for arbitrary fixed constant $p > 0$. □

We can make a parallel version (an RNC algorithm) of `CheckCongruence`. To make a parallel version, the parts of sorting and computing canonical forms of three dimensional point sets are crucial. It is well known that sorting can be done optimally even on an EREW PRAM [9]. A canonical form of a three dimensional point set S ($|S| = n$) can be computed in $O((\log n)^3)$ time using $O(n/\log n)$ processors on a CREW PRAM [3]. Thus `CheckCongruence` can be parallelized in a nearly optimal way.

4. Application to subset matching

Recently, Rezende and Lee considered the following exact matching problem (the *subset matching* problem) [17]: given point sets $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$ and $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\}$ such that $m \leqslant n$ in E^d, determine whether or not there is a subset $S \subset Q$ such that $P \cong S$. They gave an $O(n \log n + mn^d)$ time algorithm for this problem. In this section, we focus on the one dimensional case of this problem and show that we can develop an $o(mn)$ time randomized algorithm for a special case using the random sampling technique employed in the previous section.

For P, we define

$$\mathrm{sd}(P) = \max_r \left|\left\{(\boldsymbol{p}_i, \boldsymbol{p}_j) \mid i < j \text{ and } |\overline{\boldsymbol{p}_i\boldsymbol{p}_j}| = r\right\}\right|.$$

That is, $\mathrm{sd}(P)$ denotes the maximum number of point pairs having the same distances. We consider a special case that $\mathrm{sd}(P) \leqslant c$ holds for some constant c. Then the following procedure solves the problem in $O((n^2/m + m^2)\log m)$ time with high probability. Note that it is $o(mn)$ if $n^{1/2+\varepsilon} < m < n^{1-\varepsilon}$ holds for any small constant $\varepsilon > 0$.

Procedure `SubsetMatch`(P, Q)
begin
  Choose randomly $U \subset Q$ such that $|U| = \lceil Kn/m \rceil$;
  for all $\boldsymbol{q} \in U$ **do**
    begin
      for all $1 \leqslant i \leqslant m$ **do** $A[i] \leftarrow 0$;
      for all $1 \leqslant j \leqslant n$ **do**
        for all i such that $(\exists k)(\boldsymbol{p}_k - \boldsymbol{p}_i = \boldsymbol{q}_j - \boldsymbol{q})$ **do** $A[i] \leftarrow A[i] + 1$; ($)
      if $(\exists i)(A[i] = m)$ **then begin** output 'YES'; stop **end**
    end;
  output 'NO'
end

Here we analyze this procedure. We assume that there exists at least one subset $S \subset Q$ such that $S \cong P$. Otherwise the procedure always outputs 'NO'.

Note that if $\boldsymbol{p}_i \in P$ is equivalent to $\boldsymbol{q} \in S$, $A[i]$ is incremented m times in the procedure. Therefore, the procedure outputs 'YES' if there exists $\boldsymbol{q} \in U \cap S$. We can see that this condition holds with high probability from the following lemma.

Lemma 4.1. *Let S be a subset of Q where $|S| = m$ and $|Q| = n$. If we randomly choose a subset $U \subset Q$ such that*

$$|U| = \left\lceil \frac{n}{m} \ln \frac{1}{1-q} \right\rceil, \quad (\exists \boldsymbol{q} \in U)(\boldsymbol{q} \in S)$$

holds with probability at least q.

Proof. If $|U| = t$, the probability that U contains at least one element in S is

$$1 - \left(\frac{n-m}{n}\right)\left(\frac{n-m-1}{n-1}\right)\cdots\left(\frac{n-m-t+1}{n-t+1}\right) \geqslant 1 - \left(\frac{n-m}{n}\right)^t.$$

As in Lemma 3.3,

$$1 - \left(\frac{n-m}{n}\right)^t \geqslant q \quad \text{holds for } t = \left\lceil \frac{n}{m} \ln \frac{1}{1-q} \right\rceil. \qquad \square$$

Next, we analyze the time complexity. Part (\$) is the crucial part for analyzing the time complexity. To answer the query $(\exists k)(\boldsymbol{p}_k - \boldsymbol{p}_i = \boldsymbol{q}_j - \boldsymbol{q})$ quickly, we sort all the point pairs from P according to their distances. Then, given $\boldsymbol{q}_j$ and $\boldsymbol{q}$, such $\boldsymbol{p}_k$ and $\boldsymbol{p}_i$'s can be enumerated in $O(\log m)$ time because we assume that $\mathrm{sd}(P) \leqslant c$ holds for some constant c. Since part (\$) is executed $O(n^2/m)$ times, the procedure works in $O((n^2/m)\log m)$ time. Since the point pairs can be sorted in $O(m^2 \log m)$ time, we obtain the following theorem.

Theorem 4.2. *Assume that* $\mathrm{sd}(P)$ *is bounded by a constant. Then the one dimensional subset matching problem can be solved by a Monte-Carlo type randomized algorithm in* $O((n^2/m + m^2)\log m)$ *time, where error occurs only in the case that there exists a subset of* Q *congruent to* P*, and the error probability can be made smaller than any fixed constant* $p > 0$.

5. Hardness results

Although we have presented an improved algorithm for the congruence problem, the following theorem shows that it is hard if d is not bounded (i.e., the dimension is considered part of the input). Note that in what follows, we consider not only the isometric mappings of the first kind but also those of the second kind.

Theorem 5.1. *Assume that there exists a polynomial time deterministic (respectively randomized) algorithm for the congruence problem even if the dimension is considered part of the input. Then, there exists a polynomial time deterministic (respectively randomized) algorithm for the graph isomorphism problem.*

Proof. We prove it showing a polynomial time reduction from the graph isomorphism problem to the congruence problem.

From each input graph $G(V, E)$ where $V = \{v_1, \ldots, v_n\}$, we construct the following point set in E^n:

$$\{\boldsymbol{a}_i \mid a_{ij} = \delta_{ij}\} \cup \left\{ \frac{\boldsymbol{a}_i + \boldsymbol{a}_j}{2} \;\middle|\; \{v_i, v_j\} \in E \right\} \cup \{\mathbf{o}\},$$

where $\mathbf{o}$ denotes the origin, a_{ij} denotes the jth component of $\boldsymbol{a}_i$, and $\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ otherwise.

Then it is easy to see that the input graphs are isomorphic if and only if the constructed point sets are congruent. $\square$

Using a similar reduction, we can show that the subset matching problem is NP-hard.

Theorem 5.2. *The subset matching problem is NP-hard if the dimension is considered part of the input.*

Proof. In this case, we use a reduction from the subgraph isomorphism problem. Note that the subgraph isomorphism problem is known to be NP-complete [11].

From input graphs S and G, we construct point sets P_S and P_G using the same construction as in Theorem 5.1. Then P_S is congruent to a subset of P_G if and only if S is isomorphic to a subgraph of G. □

6. Conclusion

In this paper, we have presented a randomized algorithm for deciding the congruence of point sets in d dimensions, which improved the previous result considerably. However, our algorithm is not necessarily optimal as noted in Section 1. So it would be interesting to develop much more efficient algorithms.

We have also shown that the congruence problem is hard if the dimension is considered part of the input. This hardness result suggests that approximate matching problems in high dimensions are hard too since they seem to be harder than exact matching problems. However, it does not mean that we can not develop practical pattern matching algorithms in high dimensions. It seems that the congruence problem can be solved efficiently in most cases, since the d dimensional congruence problem can be reduced to the $(d-1)$ dimensional problem efficiently by computing a special point (except the centroid) invariant under isometric mappings, and such a special point seems to be computed efficiently in most cases. For the graph isomorphism problem, several algorithms which in most cases work in polynomial time have been developed [8]. Thus it would be interesting to develop pattern matching algorithms which in most cases work in polynomial time for all dimensions.

References

[1] T. Akutsu, Algorithms for determining geometrical congruity in two and three dimensions, in: Proc. 3rd Internat. Sympos. Algorithms Comput., Lecture Notes in Computer Science 650, Springer, Berlin, 1992, pp. 279–288.

[2] T. Akutsu, On determining the congruity of point sets in higher dimensions, in: Proc. 5th Internat. Sympos. Algorithms Comput., Lecture Notes in Computer Science 834, Springer, Berlin, 1994, pp. 38–46.

[3] T. Akutsu, A parallel algorithm for determining the congruence of point sets in three-dimensions, IEICE Trans. Inform. Systems E78-D (1994) 321–325.

[4] H. Alt, M. Godau, Measuring the resemblance of polygonal curves, in: Proc. 8th ACM Sympos. Comput. Geom., 1992, pp. 102–109.

[5] H. Alt, K. Melhorn, H. Wagener, E. Welzl, Congruence, similarity, and symmetries of geometric objects, Discrete Comput. Geom. 3 (1988) 237–256.

[6] M.J. Atallah, On symmetry detection, IEEE Trans. Comput. 34 (1985) 663–666.

[7] M.D. Atkinson, An optimal algorithm for geometrical congruence, J. Algorithms 8 (1987) 159–172.

[8] L. Babai, L. Kučera, Canonical labeling of graphs in linear average time, in: Proc. 20th IEEE Sympos. Found. Comput. Sci., 1979, pp. 39–46.
[9] R. Cole, Parallel merge sort, in: Proc. 27th IEEE Sympos. Found. Comput. Sci., 1986, pp. 511–516.
[10] P. Flajolet, D. Gardy, L. Thimonier, Birthday paradox, coupon collectors, cashing algorithms and self-organizing search, Discrete Appl. Math. 39 (1992) 207–229.
[11] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, New York, 1979.
[12] H. Gazit, J. Reif, A randomized parallel algorithm for planar graph isomorphism, in: Proc. ACM Sympos. Parallel Algorithms and Architectures, 1990, pp. 210–219.
[13] P.J. Heffernan, S. Schirra, Approximate decision algorithm for point sets congruence, in: Proc. 8th ACM Sympos. Comput. Geom., 1992, pp. 93–101.
[14] P.T. Highnam, Optimal algorithms for finding the symmetries of a planar point set, Inform. Process. Lett. 18 (1986) 219–222.
[15] G. Manacher, An application of pattern matching to a problem in geometrical complexity, Inform. Process. Lett. 5 (1976) 6–7.
[16] J. Matoušek, Private communication.
[17] P.J. de Rezende, D.T. Lee, Point set pattern matching in d-dimensions, Algorithmica 13 (1995) 387–404.
[18] K. Sugihara, An $n \log n$ algorithm for determining the congruity of polyhedra, J. Comput. System. Sci. 30 (1984) 36–47.