SUMMARY  In this paper, we study a problem of inferring blood relationships which satisfy a given matrix of genetic distances between all pairs of \( n \) nodes. Blood relationships are represented by our proposed graph class, which is called a pedigree graph. A pedigree graph is a directed acyclic graph in which the maximum indegree is at most two. We show that the number of pedigree graphs which satisfy the condition of given genetic distances may be exponential, but they can be represented by one directed acyclic graph with \( n \) nodes. Moreover, an \( O(n^2) \) time algorithm which solves the problem is also given. Although phylogenetic trees and phylogenetic networks are similar data structures to pedigree graphs, it seems that inferring methods for phylogenetic trees and networks cannot be applied to infer pedigree graphs since nodes of phylogenetic trees and networks represent species whereas nodes of pedigree graphs represent individuals. We also show an \( O(n^2) \) time algorithm which detects a contradiction between a given pedigree graph and distance matrix of genetic distances.

key words: algorithm, directed acyclic graph, distance matrix, pedigree, genetic distance

1. Introduction

A blood relationship is one of the most important factor in bioinformatics since it directly affects the inheritance of genes. Many studies and projects depend on blood relationships \([20],[23]\). For example, there are projects to develop drugs by determining genes which cause diseases from pedigrees and medical records \([22]\). Although there are some papers about the consistency checking of a given pedigree and some genetic information \([1]\), it seems that their main purpose is not to detect errors of pedigrees but those of other genetic information. Furthermore, there are almost no theoretical studies for the cases where pedigrees cannot be obtained. In such cases, methods of inferring or validating pedigrees are considered to be helpful.

Then, in this paper, we discuss problems to infer directed acyclic graphs (DAGs) whose indegrees are at most two (We call them pedigree graphs). When genetic distances between any two individuals are given, we discuss problems (i) to construct a DAG which represents all pedigrees which satisfy given genetic distances. (ii) to validate whether or not there is a contradiction between given distances and a given pedigree graph. Genetic distances can be calculated from the number of differences of SNPs (Single Nucleotide Polymorphisms). If any distances between two adjacent nodes are given, the solution can be constructed very easily. Then, from a theoretical interest, we assume that a genetic distance is obtained as the shortest path of blood relationships.

As related works, phylogenetic trees and phylogenetic networks should be mentioned. A phylogenetic tree is a tree showing the evolutionary interrelationships among various species or other entities that are believed to have a common ancestor \([24]\). Various methods to infer phylogenetic trees from genetic information have been researched \([2],[4],[5],[7],[10],[12]\). Since a phylogenetic tree does not include even undirected cycles, it cannot represent recombinations, that is, there are no nodes whose indegrees are more than one. Then, data structures which are slightly different from phylogenetic trees have been also researched recently \([21]\). One of such data structures is a phylogenetic network. The data structure of a phylogenetic network is very similar to that of a pedigree graph. A phylogenetic network is a directed graph in which degrees of nodes are at most two and there is only one node whose indegree is zero \([14]\). There are also many papers for problems to infer or compare phylogenetic networks: Inferring phylogenetic networks when 0-1 sequences are given for every species \([13],[14],[26]\), constructing phylogenetic networks from phylogenetic tree-like structures \([16]–[18]\), finding similar sub-graphs from phylogenetic networks \([6],[19]\), and so on. In these problems, they have a firm precondition that there is only one node whose indegree is zero in a phylogenetic network. However, in the problems to infer pedigree graphs, if an individual has no information about its ancestors, the indegree of the corresponding node should be zero. Then, in pedigree graphs, the number of nodes whose indegrees are zero may be more than one.

Distance Realization Problem is a problem to infer graphs or networks from given distance matrices \([15]\): problems to minimize the total length of edges \([15]\), problems to minimize the number of adding nodes \([3]\), problems in which solutions must be trees \([11]\), and so on. Most of Distance Realization Problems seem to allow adding new nodes. If adding new nodes is not allowed, one of the solutions is a complete graph which can be trivially made from the given distance matrix. However, if a triangle inequality is not satisfied, there are no solutions. If the length of the edge \( v_iv_j \) is equal to the sum of the lengths of the edges \( v_iv_k \) and \( v_kv_j \) in a solution, the graph obtained by removing the
edge $v_iv_j$ is also a solution. Our problem is regarded as a kind of Distance Realization Problems. Adding new nodes are not allowed in our problems. Solutions are overlapped representations of DAGs in which indegrees are at most two. The number of nodes whose indegrees are zero (or one) is not limited.

The organization of this paper is as follows. In Sect. 2, we define basic terms used in this paper. In the first half of Sect. 3, we formulate the problem of inferring blood relationships which satisfy a given matrix of genetic distances as Pedigree Graph Inferring Problem and then present an $O(n^3)$ time algorithm ConstructCPG. In the second half of Sect. 3, we show an $O(n^2)$ time algorithm PedigreeValidate for Pedigree Graph Validating Problem, which is a problem to determine whether or not a given pedigree graph satisfies the given distance matrix. Finally, we conclude the paper.

2. Pedigree Graphs

Pedigrees are often represented as shown in Fig. 1 (a). In this paper, they are represented by DAGs as shown in Fig. 1 (b). A pedigree graph is a DAG in which the maximum indegree is two. The number of nodes whose indegrees are zero or one is not limited.

**distance**: The length of an edge of a pedigree graph represents the genetic distance between two nodes which are incident to the edge. For example, in Fig. 2 (a), the length of the edge between $v_1$ and $v_4$ is two ($v_1$ is denoted by 1 and $v_4$ is denoted by 4 for simplicity). Then, $v_1$ is one of the parents of $v_4$ and their genetic distance is 2. Since we assume that a genetic distance is the shortest path of blood relationships, we define distances of pedigree graphs by using direct distances and indirect distances as follows: The direct distance between $v_i$ and $v_j$ is the length of the shortest directed path between $v_i$ and $v_j$ and denoted by $dd(v_i, v_j)$. The indirect distance between $v_i$ and $v_j$ is the minimum sum of two directed paths which are initialized by a common ancestor of $v_i$ and $v_j$ and denoted by $id(v_i, v_j)$. For example, in Fig. 2 (a), since the length of the shortest path between $v_3$ and $v_4$ is 5, $dd(v_3, v_4) = 5$ holds. On the other hand, $v_3$ and $v_4$ have a common ancestor $v_1$. Since $dd(v_1, v_3) = 1$ and $dd(v_1, v_4) = 2$ hold, $id(v_3, v_4) = 1 + 2 = 3$. The distance between $v_i$ and $v_j$, denoted by $d(v_i, v_j)$, is defined as the smaller value of $dd(v_i, v_j)$ and $id(v_i, v_j)$. Thus, in Fig. 2 (a), since $id(v_3, v_4) < dd(v_3, v_4)$ holds, $d(v_3, v_4) = id(v_3, v_4)$ is 3. When neither direct distances nor indirect distances exist, we say that the distance is $\infty$. In Fig. 2 (b), although there is an undirected path between $v_3$ and $v_5$, $d(v_3, v_5) = \infty$ since they are not in blood relationships (i.e., they have no common ancestor). Similarly, $d(v_2, v_3)$ is $\infty$ although $d(v_3, v_4) = id(v_3, v_4)$ is 2 holds.

**distance matrix**: From the definition, the distances between two nodes are symmetric. The matrix in which the element $(i, j)$ corresponds the distance between $v_i v_j$ ($i < j$) is called a distance matrix. Distance matrices are denoted as shown in Figs. 2 (c) and (d) for simplicity. The distance matrices shown in Figs. 2 (c) and (d) are corresponding to pedigree graphs shown in Figs. 2 (a) and (b) respectively.

**birth order**: When $(i, j)$ ($i < j$) of the given distance matrix is $j-i$, the pedigree graph shown in Fig. 3 (a) satisfies the condition. However, pedigree graphs shown in Figs. 3 (b) and (c) also satisfy the given distance matrix. In Fig. 3 (a), $v_1$ is the oldest individual and $v_5$ is the youngest individual. On the other hand, in Fig. 3 (c), $v_5$ is the oldest individual and $v_1$ is the youngest individual. Thus, there is a case that even the oldest individual and the youngest individual cannot be distinguished. Then, we assume that a birth order is given in our problems. When a birth order is given, if there is an edge between $v_i$ and $v_j$ ($i < j$), $v_i$ must be the parent of $v_j$. In the case of Fig. 3, if a birth order is given, Fig. 3 (a) is the only one solution. We use a birth order only for determining edge directions, that is, which is the parent and which is the child. Since a rigorous birth date or birth time is not needed, there are a lot of methods of obtaining a birth order.

**redundant edges**: The pedigree graph shown in Fig. 4 (a) satisfies the distance matrix shown in Fig. 4 (c). However, the pedigree graph as shown in Fig. 4 (b) which can be made by adding an edge between $v_1$ and $v_3$ also satisfies the distance matrix. Moreover, if the length of the
edge between \( v_1 \) and \( v_3 \) is 3 or more than 3, the pedigree graph satisfies the distance matrix. In such a case, it is impossible to determine the length of the edge between \( v_1 \) and \( v_3 \). When pedigree graphs \( G_1 = (V, E) \) and \( G_2 = (V, E - e_1) \) \((e_1 \in E)\) are given, if the corresponding matrices of \( G_1 \) and \( G_2 \) are same, \( e_1 \) is called a redundant edge and not included by solutions. Note that the edge \( v_3v_4 \) of Fig. 2 (a) is not a redundant edge. If the edge is eliminated, \( d(v_2, v_4) \) changes from 6 to \( \infty \).

### 3. Pedigree Graph Inferring Problem

First, we discuss the following trivial problem:

- **Input**: A distance matrix and a birth order.
- **Output**: A set of pedigree graphs, in which lengths of edges are one, which satisfy the distance matrix and the birth order.

Since lengths of edges are one, two nodes are adjacent if and only if the corresponding element of the distance matrix is one. Moreover, since the birth order is given, directions of edges are determined uniquely. Therefore, the solution is obtained in \( O(n^2) \) time. The number of solutions is only one. Thus the case in which lengths of edges are one can be solved easily.

Next, we discuss the case in which the lengths of edges are positive real values. Since, in such a case, the number of pedigree graphs which satisfies the condition of a given matrix may be exponential, we define the output of the problem as an overlapped representation of all possible pedigree graphs as explained later. Let a feasible pedigree graph (FPG) be a pedigree graph which satisfies both of following conditions:

- does not contradict the given distance matrix and birth order.
- does not have redundant edges.

Let \( e_{i_1}, e_{i_2}, \ldots, e_{i_k} \) be edges terminated by \( v_i \) in a DAG where \( \text{length}(e_{i_1}) \leq \text{length}(e_{i_2}) \leq \ldots \leq \text{length}(e_{i_k}) \) holds. Let a compiled pedigree graph (CPG) for a given distance matrix and birth order be a DAG which satisfies the following conditions:

- Every FPG is a subgraph of the CPG.
- For each \( i (1 \leq i \leq n) \)
  - If \( k_i = 0 \) (there are no edges terminated by \( v_i \)) in a CPG, every FPG does not have in-edges terminated by \( v_i \).
  - If \( k_i = 1 \) in a CPG, \( e_{i_1} \) is the only one in-edge terminated by \( v_i \) in every FPG.
  - If \( k_i \geq 2 \) in a CPG, every FPG must include the edge \( e_{i_1} \) and one of \( e_{i_2}, \ldots, e_{i_k} \) as the other in-edge of \( v_i \).

### Pedigree Graph Inferring Problem

- **Input**: A distance matrix and a birth order (equivalent to \( G_{in} \) as explained later).
- **Output**: A compiled pedigree graph (CPG).

Interestingly, a CPG exists if the number of FPGs is greater than or equal to 1 for a distance matrix and birth order. Note that the number of FPGs which correspond to a CPG may be exponential of \( n \). The existence of a CPG in such a case is ensured by the following lemmas. If a CPG is obtained, it is straightforward to construct a FPG from a CPG.

A directed graph in which there is just one edge between any two nodes is called a tournament [25]. When a distance matrix and a birth order are given, a tournament, in which weights are corresponding to the distance matrix and directions of edges are corresponding to the birth order can be uniquely obtained, and which is called a basic tournament and denoted by \( G_{in} \). Note that \( G_{in} \) is equivalent to the input of the Pedigree Graph Inferring Problem. Let \( G_{out} \) be a set which consists of all FPGs.

For example, the distance matrix shown in Fig. 5 (a) with a birth order is equivalent to the basic tournament shown in Fig. 5 (b). Since both pedigree graphs shown in Figs. 5 (c) and (d) are FPGs of \( G_{in} \), they are elements of \( G_{out} \). Note that the edge \( v_3v_4 \) in Fig. 5 (c) is not a redundant edge. If there is not an edge of \( v_3v_4 \) in Fig. 5 (c), \( d(v_2, v_4) \) becomes \( \infty \).

Let us denote the length of a directed edge \((v_i, v_j)\) by \( \text{length}(v_i, v_j) \).

**LEMMA 1**: Let \((v_j, v_i) \) \((j < i)\) be the shortest edge whose terminal node is \( v_i \) in \( G_{in} \). Then, \((v_j, v_i) \) also exists in every FPG.
LEMMA 3: The indegree of graphs shown in (c) and (d) are FPGs of $G_{in}$ in the Pedigree Graph Inferring Problem. Furthermore, both of pedigree graphs shown in (c) and (d) are FPGs of $G_{in}$. In other words, they are elements of $G_{out}$.

Proof: There is no edge which is shorter than $(v_j, v_i)$ and is terminated by $v_i$ in $G_{in}$. Assume that there is not $(v_j, v_i)$ in a FPG. Since the birth order is given, there is not $(v_j, v_i)$. Furthermore, since lengths of edges are positive, $d(v_j, v_i)$ is longer than $\text{length}(v_j, v_i)$. It contradicts the assumption. Since the direction of the edge $v_jv_i$ is determined uniquely, there is a directed edge $v_jv_i$ in every FPG. □

LEMMA 2: Let $d$ be the length of the shortest edge terminated by $v_i$ in $G_{in}$. Then, the number of edges which are terminated by $v_i$ and whose length is $d$ is at most two in $G_{in}$.

Proof: Assume that there are three such edges. By Lemma 1, such edges also exist in every FPG. Since indegrees of pedigree graphs are at most two, it is a contradiction. □

LEMMA 3: The indegree of $v_i$ of a FPG is zero if and only if all $d(v_i, v_j) (i < j)$ are $\infty$ in $G_{in}$.

Proof: Assume that $d(v_i, v_j)$ is finite for some $i$ and $j (i < j)$. $d(v_i, v_j)$ is equal to either $dd(v_i, v_j)$ or $id(v_i, v_j)$. However, by the birth order, the path between $v_i$ and $v_j$ must include the edge terminated by $v_j$. Then, the indegree of $v_j$ is not zero. Moreover, if $d(v_i, v_j) = \infty$ for any $i$ and $j (i < j)$, it is clear that the indegree of $v_j$ is zero. □

The outline of the $O(n^3)$ time algorithm for Pedigree Graph Inferring Problem is as follows: First, a basic tournament is constructed from the given distance matrix and the birth order. Next, a CPG is obtained by removing unnecessary edges. We select appropriate in-edges for each node of $G_{in}$ from $O(n)$ edges and use them as edges of the CPG. However, it is necessary to prove that such selecting operations can be done separately for each node of $G_{in}$.

For example, in Fig. 6, the basic tournament shown in (b) is constructed from the distance matrix of (a) and the birth order. For the basic tournament of Fig. 6(b), there are two types of FPGs. One includes an edge pair $(v_1, v_4)$, $(v_2, v_4)$ and the other includes $(v_2, v_4), (v_3, v_4)$. A naive algorithm separating these two cases before the selecting operations at other nodes may increase the computation time exponentially. However, the following lemma shows that the selecting operations can be done separately. Let $V_{small} = \{v_1, v_2, \ldots, v_1\}$ and $V_{large} = \{v_1, v_2, \ldots, v_{n+k}\}$. Moreover, let $G_1 = (V_{small}, E_1), G_2 = (V_{small}, E_2)$ be pedigree graphs. Let $\text{length}(P)$ be the length of a undirected path $P$. Let $v_j, v_j \in V_{small}$ and $v_j, v_j \in V_{large}$.

LEMMA 4: When distances between any $v_j, v_j$ of $G_1$ and $v_j, v_j$ of $G_2$ are same, for any $E_3$ which yields pedigree graphs $G_3 = (V_{large}, E_3)$ and $G_4 = (V_{large}, E_3)$, distances between any $v_j, v_j$ of $G_3$ and $v_j, v_j$ of $G_4$ are same.

Proof: It suffices to prove the lemma for the case where $v_j, v_j \in V_{large} - V_{small}$ since it can be applied to the other cases by putting $\phi$ to $P_a, P_c, P_d$ and $P_j$.

Let $P_3 = P_a + P_b + P_c$ be the shortest path between $v_j, v_j$ in $G_3$. Let $P_4 = P_a + P_b + P_j$ be the shortest path between $v_j, v_j$ in $G_4$ (See Fig. 7). Let $P_b \subseteq E_1, P_c \subseteq E_2$, and $P_a, P_c, P_d, P_j \subseteq E_3$. (There is a possibility that $P_b$ and $P_c$ are $\phi$). Assume that $\text{length}(P_3) \neq \text{length}(P_4)$. (The generality is not lost by $\text{length}(P_1) > \text{length}(P_3)$). By the assumptions of $G_1$ and $G_2$, there exists $P_e$, whose initial node and terminal node are same as $P_e$, which satisfies $\text{length}(P_e) = \text{length}(P_3)$ and $P_e \subseteq E_1$. Then, $P_5 = P_d + P_e + P_j$ satisfies $P_5 \subseteq E_1 + E_3$ and $\text{length}(P_5) < \text{length}(P_3)$. If $P_5$ corresponds either a direct or indirect distance of $v_j, v_j$, it contradicts that $\text{length}(P_3)$ is the shortest path of $v_j, v_j$, and $\text{length}(P_5) = \text{length}(P_4)$ can be proved.

The following proof shows that $P_5$ corresponds either a direct or indirect distance of $v_j, v_j$.

- When $P_3$ is composed of a directed path:
  The generality is not lost by assuming that each of $P_b, P_c$ is composed of a directed path and $P_a = \phi$. Moreover, $P_e$ is composed of at most two directed paths. Then, at the connection between $P_a$ and $P_j$ in $P_5 = P_a + P_j$, the directions of two edges are same. By the birth order, once a directed path goes out from $V_{small}$, it never comes back inside $V_{small}$. Thus $P_5$ satisfies the condition.

- When $P_3$ is composed of two directed paths:
  Each of $P_a, P_c, P_d$ and $P_j$ is composed of a directed path. $P_b$ is composed of two directed paths. Therefore,
$P_g$ is composed of at most two directed paths.

1. When $P_g$ is composed of a directed path:
The directions of edges are different either at the connection of $P_d$ and $P_g$, or at the connection of $P_g$ and $P_f$. Thus, $P_5$ corresponds an indirect distance of $v_i v_j$.

2. When $P_g$ is composed of two directed path:
Since the directions of edges are same at the connection of $P_d$ and $P_g$, and $P_g$ and $P_f$, $P_5$ corresponds an indirect distance of $v_i v_j$.

Thus, $\text{length}(P_3) = \text{length}(P_4)$. 

In the following part of this paper, we show ConstructCPG which can output the solution of Pedigree Graph Inferring Problem in $O(n^3)$ time. Before explaining the algorithm formally, we use examples in order to explain it intuitively.

Suppose that a distance matrix shown in Fig. 6 (a) and a birth order are given as the input of Pedigree Graph Inferring Problem. The corresponding basic tournament can be constructed as shown in Fig. 6 (b). The algorithm selects appropriate in-edges of each node of the basic tournament in order to construct a CPG. From Lemma 4, the selections can be done independently. For the simplicity of the explanation, the algorithm treats nodes in the ascending order, that is, $v_1, v_2, \ldots, v_n$. From Lemma 3, the indegrees of $v_1$ and $v_2$ of the CPG which satisfies the distance matrix of Fig. 6 (a) are zero as shown in Fig. 8 (a) and the parents of $v_3$ are $v_1$ and $v_2$. Next, the algorithm selects appropriate edges which are terminated by $v_4$. In the basic tournament shown in Fig. 6 (b), the lengths of edges which are terminated by $v_4$ are 8, 3, and 5. From Lemma 1, if there exists a solution, $(v_2, v_4)$, whose length is 3, must be included in every FPG. Actually, as shown in Fig. 8 (b), the pedigree graph which includes $(v_2, v_4)$ does not contradict the distance matrix.

Then, as shown in Fig. 8 (c), the algorithm examines whether or not the pedigree graph which includes $(v_3, v_4)$ contradicts the distance matrix. Although $d(v_1, v_4) = 6$ in the pedigree graph shown in Fig. 8 (c), the corresponding value in the distance matrix is 8. Thus, a contradiction is actually detected. When an edge is added in the process of constructing a CPG, there is a case that the distance of two nodes in the corresponding pedigree graph is shorter than the corresponding value of the distance matrix. In such a case, the algorithm may avoid the contradiction by increasing the length of the edge using the method shown in Lemma 5. In this example, the contradiction is avoided by changing the length of $(v_3, v_4)$ to 7 as shown in Fig. 8 (d). Note that $(v_3, v_4)$ of Fig. 8 (d) is not a redundant edge. If $(v_3, v_4)$ is removed, $d(v_1, v_4)$ becomes $\infty$ although it is 8 in the distance matrix. It should also be noted that a contradiction cannot be avoided in the case where the distance of two nodes in a pedigree graph is longer than the corresponding value of the distance matrix. In such a case, it is determined that the edge should be included in neither FPGs nor a CPG. Thus, when
both \((v_2, v_4)\) and \((v_3, v_4)\) are included by the corresponding pedigree graph, the conditions of the distance matrix and the birth order are satisfied.

From Lemma 1, \((v_2, v_4)\) should be included by every FPG. Moreover, there is a possibility that the condition of the distance matrix is satisfied when \((v_1, v_4)\) is selected as the other edge terminated by \(v_4\) in a pedigree graph. In fact, there is no contradiction when \((v_1, v_4)\) and \((v_2, v_4)\) are selected as the two in-edges terminated by \(v_4\) in a FPG. In other words, at the point of time shown in Fig. 8 (e), ConstructCPG should keep three in-edges terminated by \(v_4\) in order to construct a CPG. This means that the shortest edge \((v_2, v_4)\) must be included by every FPG and that both \((v_1, v_4)\) and \((v_3, v_4)\) can be the candidate for the other in-edge terminated by \(v_4\) in FPGs.

Next, the algorithm selects edges terminated by \(v_5\) of \(G_{in}\). The lengths of edges terminated by \(v_5\) are \(2, 5, 3, 1\) and 2. From Lemma 1, when there exists a solution, the shortest edges \((v_1, v_5)\) and \((v_4, v_5)\) should be included by every FPG. Since pedigree graphs constructed from the CPG shown in Fig. 8 (f) which includes \((v_1, v_5)\) and \((v_4, v_5)\) satisfy the condition of the distance matrix and the birth order, both \((v_1, v_5)\) and \((v_4, v_5)\) should be included in the CPG.

Thus, ConstructCPG outputs the CPG of Fig. 8 (f) as the solution. As explained above, an indegree of a CPG may be more than two. In such a case, FPGs can be constructed by selecting the shortest edge and selecting the other edge arbitrary at each node. Note that if the number of the shortest edges is two, both of them should be included in the CPG. If the number of the shortest edges is more than two, there is not a solution from Lemma 2. Details of ConstructCPG are as follows:

**algorithm ConstructCPG**

begin

1. Construct the basic tournament (= \(G_{in}\)).
2. Remove edges whose lengths are \(\infty\) in \(G_{in}\).
3. Sort in-edges by lengths in the ascending order at each node \(v_i\) of \(G_{in}\) so that \(length(e_{i_1}) \leq length(e_{i_2}) \leq \ldots \leq length(e_{i_n})\) holds.
4. By the method of Lemma 3, find nodes whose indegrees are zero in \(G_{in}\). Apply following procedures to nodes, denoted by \(v_i (i = 1, \ldots, n)\), whose indegrees are not zero in \(G_{in}\).
   a. If the indegree of \(v_i\) is more than two and \(length(e_{i_1}) \neq length(e_{i_2})\) holds, stop after reporting “There is no solution”. When \(length(e_{i_1}) = length(e_{i_2})\), eliminate all \(e_{i_j}\) \((j \geq 3)\) and finish the process for \(v_i\). Since there is a possibility that no pedigree graphs satisfy the distance matrix, examine whether or not there is a contradiction when edges terminated by \(v_i\) are \(e_1\) and \(e_2\). When \(length(e_{i_1}) \neq length(e_{i_2})\), apply following procedures. Calculate the shortest distances which use \(e_{i_j}\) between \(v_i\) and every \(v_k\) \((1 \leq k \leq i-1)\). Apply following procedures to \(e_{i_j}\) \((j \geq 2)\).
   i. Calculate the shortest distances which use \(e_{i_j}\) between \(v_i\) and every \(v_k\) \((1 \leq k \leq i-1)\).
   ii. Calculate every \(d(v_j, v_i) (1 \leq k \leq i-1)\) of the corresponding pedigree graph by comparing the case where \(e_{i_j}\) is used and that where \(e_{i_j}\) is used. If \(d(v_k, v_i)\) is longer than the corresponding value of the distance matrix, eliminate \(e_{i_j}\) from \(G_{in}\) since the contradiction cannot be avoided. On the other hand, if \(d(v_k, v_i)\) is shorter than the corresponding value of the distance matrix, increase the length of the edge by the method of Lemma 5. If either the contradiction still exists or \(e_{i_j}\) is redundant, eliminate \(e_{i_j}\).
5. Output \(G_{in}\) as the CPG.

end

The idea of the method of Lemma 5 is as follows: Suppose that edges terminated by \(v_i\) in a candidate of a FPG are \(e_{i_1}\) and \(e_{i_2}\). Let \((d_{given_1}, d_{given_2}, \ldots, d_{given_k})\) be the distances which \(v_i\) should satisfy for \(v_1, v_2, \ldots, v_i-1\). Let \((d_{j_1}, d_{j_2}, \ldots, d_{j_k})\) be the distances which use \(e_{i_1}\) between \(v_i\) and \(v_1, v_2, \ldots, v_{i-1}\). Moreover, let \((d_{j_1}, d_{j_2}, \ldots, d_{j_k})\) be the distances which use \(e_{i_2}\) between \(v_i\) and \(v_1, v_2, \ldots, v_{i-1}\).

The algorithm compares \(d_{j_1}, d_{j_2}, \ldots, d_{j_k}\) and \(d_{given_1}, d_{given_2}, \ldots, d_{given_k}\). The algorithm terminates with “\(+\)” if \(d_{j_k}\) is as long as \(d_{given_k}\), the algorithm terminates with “\(-\)” if \(d_{j_k}\) is shorter than \(d_{given_k}\).

From Lemma 1, there are no “\(-\)” in \((d_{j_1}, d_{j_2}, \ldots, d_{j_k})\). If \((d_{j_1}, d_{j_2}, \ldots, d_{j_k})\) are all “\(-\)”, since edges except for \(e_{i_1}\) are redundant, the algorithm should eliminate all edges terminated by \(v_i\) in \(G_{in}\) except for \(e_{i_1}\). Then, we discuss only
the case in which \((d_{11}, d_{12}, \ldots, d_{1n})\) are composed of “=”, “+” and “-”. \((d_{21}, d_{22}, \ldots, d_{2n})\) are composed of “=”, “+” and “-”.

**LEMMA 5:** When \(d_{ik}(1 \leq k \leq i-1)\) includes “-”, \(e_{ij}\) can be included by a FPG without contradictions only if the length of \(e_{ij}\) is modified to \(\text{length}(e_{ij})+\max(d_{\text{given}}-d_{ik}|1 \leq k \leq i-1)\).

**Proof:** If \(\text{length}(e_{ij})\) is shorter than the value shown above, \(d_{ik}(1 \leq k \leq i-1)\) still includes “-”. Then, it contradicts the distance matrix when \(e_{ij}\) is included by a FPG. On the other hand, if \(\text{length}(e_{ij})\) is longer than the value shown above, \(d_{ik}(1 \leq k \leq i-1)\) are all “+”. Furthermore, \((d_{11}, d_{12}, \ldots, d_{1n})\) includes at least one “+”. Thus, if \(e_{ij}\) is included by a FPG, the distance becomes too long and it is a contradiction. □

ConstructCPG outputs a CPG, which is an overlapped representation of all FPGs of \(G_{in}\). Note that \(G_{in}\) is equivalent to the input of Pedigree Graph Inferring Problem.

**THEOREM 1:** Pedigree Graph Inferring Problem can be solved in \(O(n^3)\) time by ConstructCPG.

**Proof:** Let \(v_i\) and \(v_j\) be nodes which initialize \(e_{ii}\) and \(e_{ij}\) respectively. The distances which use \(e_{ii}\) (or \(e_{ij}\)) between \(v_i\) and \(v_j\) \((1 \leq k \leq i-1)\) can be calculated by adding \(\text{length}(e_{ii})\) (or \(\text{length}(e_{ij})\)) to the distances from \(v_i\) (or \(v_j\)). Thus, the distance calculation can be done in \(O(n^2)\) time. It takes \(O(n^2)\) time to construct a basic tournament from a distance matrix. The other operations can also be done in \(O(n^3)\) time. □

The inputs of Pedigree Graph Inferring Problem are a distance matrix and a birth order and the output of the problem is a CPG. On the other hand, Pedigree Graph Validating Problem is as follows: The inputs of the problem are a distance matrix, a birth order and a pedigree graph. The output is whether or not there is a contradiction.

**Pedigree Graph Validating Problem**

- **Input:** A distance matrix, a birth order, a pedigree graph.
- **Question:** Does the pedigree graph satisfy the given distance matrix and the birth order?

This problem can be solved in \(O(n^2)\) time by applying slightly modified ConstructCPG. The part of ConstructCPG which detects a contradiction in the process of constructing a CPG can also be applied to Pedigree Graph Validating Problem. Let us call such an algorithm PedigreeValidate. Let \(e_{ij}\) (or \(e_{ii}\)) be a directed edge \((v_i, v_j)\) (or \((v_i, v_j)\)) in a given pedigree graph. Note that there is a case where \(v_j\) (or \(v_i\)) does not exist. PedigreeValidate is as follows:

**algorithm PedigreeValidate**

**begin**

For \(i = 1\) to \(n\):

- **STEP1:** Calculate distances which uses \(e_{ij}\) between \(v_i\) and \(v_j\) \((1 \leq a \leq i-1)\).
- **STEP2:** Calculate distances which uses \(e_{ii}\) between \(v_i\) and \(v_j\) \((1 \leq a \leq i-1)\).
- **STEP3:** Calculate \(d(v_i, v_j)\) \((1 \leq a \leq i-1)\) by comparing distances obtained in STEP1 and STEP2. If \(d(v_i, v_j)\) \((1 \leq a \leq i-1)\) is different from the corresponding value of the given distance matrix, return “false”.

Return “true”.

**end**

**THEOREM 2:** Pedigree Graph Validating Problem can be solved in \(O(n^2)\) time by PedigreeValidate.

**Proof:** For each \(i (1 \leq i \leq n)\), PedigreeValidate calculates the distances between \(v_i\) and \(v_j\) \((1 \leq a \leq i-1)\) for both cases where \(e_{ij}\) is used and \(e_{ii}\) is used. After obtaining \(d(v_i, v_j)\) \((1 \leq a \leq i-1)\) by comparing these two values, it is examined whether or not the distance matrix is satisfied. It takes \(O(n)\) time for each \(v_i\) \((1 \leq i \leq n)\). Thus, the algorithm can examine whether or not there is a contradiction in \(O(n^2)\) time. □

4. Concluding Remarks

In this paper, we formulated a problem to infer pedigrees from genetic distances and a birth order as Pedigree Graph Inferring Problem. Although a CPG, which is the output of Pedigree Graph Inferring Problem, is not a pedigree graph itself, it is straightforward to construct pedigree graphs from a CPG. We also formulated a problem to examine whether or not there is a contradiction among given genetic distances, a birth order and a pedigree graph as Pedigree Graph Validating Problem. We showed that Pedigree Graph Inferring Problem can be solved in \(O(n^3)\) time by ConstructCPG and Pedigree Graph Validating Problem can be solved in \(O(n^2)\) time by PedigreeValidate.

References


Takeyuki Tamura received B.E., M.E. and Ph.D. degrees in informatics from Kyoto University, Japan, in 2001, 2003, and 2006, respectively. He joined Bioinformatics Center, Institute for Chemical Research, Kyoto University as a postdoctoral fellow in April, 2006. His research interests are bioinformatics and the theory of combinatorial optimization for graphs and networks.

Hiro Ito received the B.E., M.E., and Dr. of Engineering degrees in Applied Mathematics and Physics from the Department of Engineering, Kyoto University in 1985, 1987, and 1995, respectively. From 1987 to 1996, he was a member of NTT Laboratories. From 1996 to 2001, he was in the Department of Information and Computer Sciences at Toyohashi University of Technology. Since 2001, he has been an associate professor in the Department of Communications and Computer Engineering, Graduate School of Informatics at Kyoto University. He has been engaged in research on the theory of combinatorial optimization for graphs and networks and discrete mathematics. Dr. Ito is a member of the Operation Research Society of Japan and the Information Processing Society of Japan.