http://dx.doi.org/10.21622/ace.2022.02.1.043

Algorithms for Comparing Large Pedigree Graphs

Nahla A. Belal[1], Lamiaa A. Amar[2] and Hany H. Sherief[2]

[1]College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime Transport (AASTMT), Smart Village, B 2401, Giza, Egypt

[2]Alexandria University, Faculty of Science, Al Azaritah WA Ash Shatebi, Qism Bab Sharqi, Alexandria, Egypt

Email: { nahlabelal@aast.edu, lamy2003@hotmail.com, hanysherief@gmail.com }

Received on, 08 May 2022 - Accepted on, 13 June 2022 - Published on, 29 June 2022

ABSTRACT

The importance of pedigrees is translated by geneticists as a tool for diagnosing genetic diseases. Errors resulting during collection of data and missing information of individuals are considered obstacles in deducing pedigrees, especially larger ones. Therefore, the reconstructed pedigree graph evaluation needs to be undertaken for relevant diagnosis. This requires a comparison between the derived and the original data. The present study discusses the isomorphism of huge pedigrees with labeled and unlabeled leaves, where a pedigree has hundreds of families, which are monogamous and generational. The algorithms presented in this paper are based on a set of bipartite graphs covering the pedigree and the problem addressed is parameter tractable. The Bipartite graphs Covering the Pedigree (BCP) problem is said to possess a time complexity of $f(k).mod(X)^{O(1)}$ where f is the computing function that grows exponentially. The study presents an algorithm for the BCP problem that can be categorized as a polynomial-time-tractable evaluation of the reconstructed pedigree. The paper considers pedigree graphs that consist of both labeled and unlabeled leaves that make use of parameterized and kernelization algorithms to solve the problem. The kernelization algorithm executes in $O(k^3)$ for the BCP graphs.

Keywords: Pedigree Graphs, Isomorphism, Parameterized Algorithm, Kernelization Algorithms, Bipartite graphs Covering Peigrees (BCP) Problem

I. INTRODUCTION

Recently, an enormous amount of populace genotype data produced from a Single-Nucleotide Polymorphism (SNP) in genomes were collected. The amount of such data are expected to rise rapidly in the near future. The populace genotype data relationships are normally represented using a family tree or pedigree. A pedigree depicts the relationship amongst the family members and portrays a specific trait, abnormality, or a disease commonly shared by them through inheritance [1]. A pedigree is always of great interest and importance to scientists. For example, pedigree analysis can be be used for initiating a country-wise exchange of germplasm [2]. Further, it can also serve to devise strategies accordingly. Approaches to reconstruct pedigree are multi-fold. One such way is to construct using genetic data of survivors and backtracking to identify the ancestral origins [3, 4, 5, 6]. Reconstruction methods incorporate comparing the derived pedigree against the original, as both possess the same set of individuals and genetic data but are mutually exclusive in their inferred ancestors. The existing methods used to compare pedigrees are very low on accuracy [7, 8, 9].

http://dx.doi.org/10.21622/ace.2022.02.1.043

A team of researchers have developed a method for comparing the topology of pedigrees with labeled individuals [10], where they studied the following two main pedigree graph comparison aspects:

- Evaluation of the reconstructed pedigree in comparison with the original, and
- Examination of the inferred and true ancestors for isomorphism or exhibition of dissimilitude

The authors employed the edit distance algorithm. Both problems are said to be computationally complex and NP-hard. The edit-distance problem is APX-hard, meaning that there is a Polynomial Time Approximation Scheme (PTAS) reduction from every problem in APX to the former and is highly specific in the case of graph matching.

Chen, in his research presented models for a pedigree with labeled leaves [11]. Labeled leaves represent individuals that have available DNA on which genotyping or sequencing can be performed without relying on genealogical structure. He also compared two pedigrees with labeled leaves only. Avoiding pedigrees with leaves missing the data of the biological father, that are unlabeled leaves, in turn, has considerably lowered the accuracy in the results [12]. Jiang, on the other hand, presented his research work to evaluate pedigrees with unlabeled leaves and proved it to be GI-hard and Fixed-Parameter Tractable (FPT) [13]. The large running time of this method is the major drawback hampering its comparison to arbitrary 2-generation pedigrees. Similarly, Amar et al. studied the isomorphism for large unlabeled sub-pedigree, but only up to 7-generation, by using two traits and proved that the fixed parameters are tractable [14]. Recently, [15] developed the PedigreeNet viewer which was able to generate a pedigree network of 4706 maize lines and 5487 relationships.

A lot of practical work recently carried out show the massive importance of pedigrees. Among the most recent work done is Xie et al. [16], where the studied a chinese pedigree for retinal vasculopathy and how cerebral leukoencephalopathy can be deduced. Also, in [17] developed an R package that uses pedigree and genomic data to analyze genetic connectedness. Pedigree analysis also extends to plants, in [18], the authors studied 7 generation pedigrees on peaches.

This research focuses on the algorithm for large pedigree isomorphism problem that requires the connected components in a given Family F to constitute a graph. Vertices of the graph can be separated into two sets that are dissimilar and independent, with edges connecting these two disjoint sets of vertices. The disjoint sets of vertices and edges interconnecting them contain labeled and unlabeled leaves, and are represented as an isomorphism problem, between the reconstructed and the original pedigree. In this paper, an algorithm is proposed, for the bipartite graphs constituting the pedigree for which leaves are labeled and unlabeled. The algorithm is executed in polynomial time, given that the bipartite graphs in the family are bound by a constant. Along with the above, a kernelization problem whose time complexity increases linearly is presented that returns an instance of size $O(k^3)$.

http://dx.doi.org/10.21622/ace.2022.02.1.043

The following are the definitions based on which this research has been conducted:

Definition 1 A pedigree $G = (P, X, g, l)$ consists of a directed and acyclic graph where P denotes set of vertices and edges represented as: $P = (I(P);E(P))$;
the gender function denoted as g and represented as: $g : I(P) \rightarrow \{male, female\}$
X the set of labelled vertices and that which is denoted as $X \subseteq I(P)$ and
l denoted for injective labelling as $l : X \rightarrow N$
where N is the graph size and the set of vertices in P have an in-degree of either 0 or 2 and if $(a, v1)$ and $(b, v1) \in$ E
then g(a) $\neq$ g(b).

The set of vertices and directed edges represent the parent-child relationship in this graph and the graph is acyclic in nature. A monogamous vertex "v" in graph P and pedigree G is defined to be that vertex which mates with the opposite gender vertex exactly once and is formally defined as a $v \neq v^{'}$ where (v, x) and $(v^{'}, x) \in$ to $E(P)$ for some x in the set of vertices $I(P)$ is exactly one. If all the vertices in that graph mate with exactly one other vertex of opposite gender then the pedigree is said to be monogamous.

A pedigree $G = (P, X, g, l)$ is generational if there exists $Q : I(P) \rightarrow N$ and $Q(v) = 1$, where v has an in-degree of 0, and there exists an edge (u, v) in $E(P)$, such that the in-degree of v is more than 0 and $Q(v) = Q(u) + 1$. Then Q indicates the generation for v which is nothing but a numeral.

Definition 2 Two pedigree graphs $G = (P, X, g, l)$, G0 $= (P^{'}, X^{'}, g^{'}, l^{'})$, with labelled and unlabelled leaves are said to be isomorphic when the following condition holds true. There exists a gender function $g : I(P) \rightarrow (male, female)$, for the labelled and unlabelled leaves, the gender is unknown and there exists a bijection, i.e. one to one and onto mapping between the two sets. It is represented as : $\psi : V \rightarrow V^{'}$, if for every $v \in V$ the gender of v i.e. $g(v)$ matches the gender of v in the other set, i.e $g'(v)$ and $(x, v) \in E$ if, ψ(x) ψ(v) $\in E^{'}$.

Definition 3 Maximum : 3-Dimensional Matching : Given a labelled pedigree graph P constituting of sets of disjoint paths of size n, termed A, B and C, and a set of triples $Y \subseteq A \times B \times C$. M is defined as the subset of Y such that every 2 triples in M are disjoint and cardinality of $mod[M]$ is maximized. Given an instance of two pedigree P and $P^{'}$. An arbitrary fixed order of elements is taken into consideration i.e. $Y = A \cup B \cup C$ and use the assigned order. For each triplet $t = (xi, yj, zk)$ a sub pedigree containing (x_i, y_j, z_k) of the pedigree P as follows:

- A female individual is created with (x_i, y_j, z_k) for a leaf of P.
- Create parents w_h^t, m^t for (x, y, z) and h is the generation of the pedigree for $2 \leq$ mod $(A \cup B \cup C)$
- For each m_1^t, m_2^t and w_2^t are the parents who are created upto generation h and for each w_1^t create parents u_1^t and v_1^t upto generation h. P now contains Y sub pedigrees and each has a unique leaf.
- Next $P^{'}$ is initialized as a copy of P and the stated condition is $1 \leq h \leq$ mod $(A \cup B \cup C)$.
- t_1, t_2, t_3 sharing h items in mod $(A \cup B \cup C)$ and u_1^t.....u_n^t are merged and v_1^t.....v_n^t are merged to form sub-pedigree with max isomorphism.

http://dx.doi.org/10.21622/ace.2022.02.1.043

Problem Definition Suppose there exists a derived pedigree graph. Hence, a framework to compare large real and induced pedigrees will be discussed. A sample pedigree is shown in Figure 1. To compare the two pedigrees, the sub pedigrees of this graph have to be taken into consideration. The sub pedigrees are represented in Figure 2. Where, Figure 2(a) and 2(c) constitute labelled leaves, whereas 2(b), 2(d) and 2(e) show unlabelled leaves. These sub pedigrees exude generational and monogamous trait. Considering the pedigree graph shown in Figure 1, it will be subsequently proved in 3 that that there exists isomorphism upto the 3rd generation. If there exists an isomorphism between the deduced and the original pedigree with both labeled and unlabeled leaves, it will be validated. This can help geneticist to use it as an essential tool to diagnose the genetic disease. Thus, the problem statement is as follows: The reconstructed or obtained graph of a large pedigree can be validated by evaluating the existence of isomorphic traits between its sub-pedigrees that are bound by the linearly increasing time complexity of $O(k^3)$.

The problem described can be formally defined as follows.

INSTANCE

A family $F_0 = (V_{F0}, E_{F0})$ consisting of k_0 disjoint sub-families, and a real pedigree graph $P = (V_p, E_p)$, with the same number of individuals $jV_{F0}j = jV_Pj$, and a large sub-family $K_{p,d}$.

SOLUTION

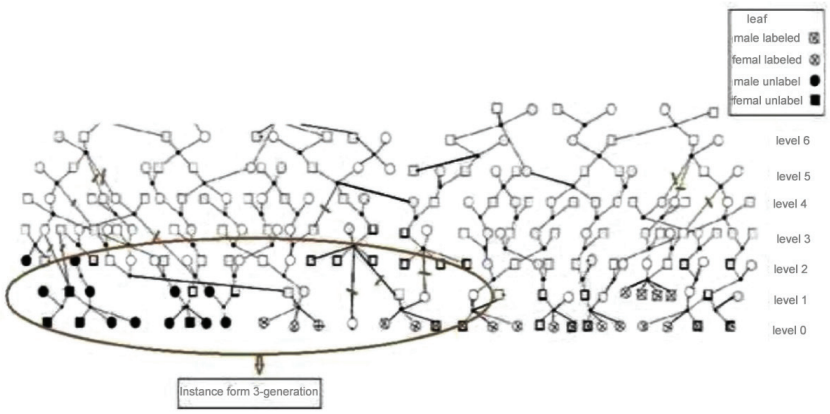

Figure 1: A large pedigree containing labeled and unlabled leaves.

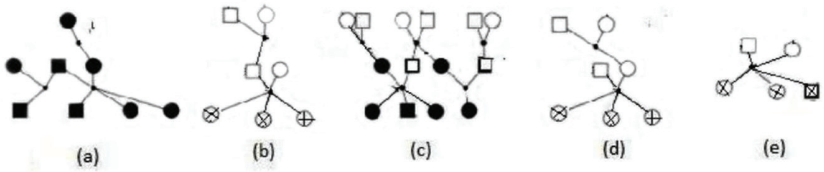

Figure 2: (a) and (c) constitute of labelled leaves whereas (b), (d) and (e) have unlabelled leaves.

F_0 is a sub-pedigree of G up to three-generations. This study discusses the isomorphism of huge pedigrees with labelled and unlabeled leaves, where the pedigree has hundreds of families, that are monogamous and generational.

http://dx.doi.org/10.21622/ace.2022.02.1.043

The main contributions of the paper are as summarized in the points below:

- The problem of comparing large pedigree graph is defined.
- Algorithms for solving the problem are designed.
- Faster kernelization algorithms for solving the problem are designed.
- Theorems and lemmas for proving and analyzing the proposed algorithms are given.
- The importance of pedigrees to geneticists is highlighted.

This paper is organized in a way that succeeding sections present the problem and the results followed by implications and limitations of the study. Section 2 summarizes the methods used in this paper. The algorithms are presented in Section 3 along with their analysis. Finally, Section 4 concludes and sets future directions.

2. METHODS

Kernelization algorithms aim at data reduction, by reducing the input to a function of the parameter. Hence, they will be used to remove the sub-families from the reconstructed pedigree till end and the similarity between the sub-pedigrees of the original and the reconstructed pedigree will be established. The algorithms execute in polynomial time across large pedigree graphs and the total number of individuals in sub-pedigrees would be bounded by $O(k^3)$.

The researchers have considered a set of BCP graphs. The parameterized algorithms are used to solve the BCP problem and the kernelization algorithm is applied repeatedly, whenever there is a sub-family that has to be removed. At the end of the process, no sub-families can be removed. At this stage the instances of each pedigree are evaluated to identify the isomorphism. This methodology can help in diagnosing the genetic diseases across the generations in large pedigrees containing both labeled and unlabeled leaves. Using bipartite graphs for which the CUPB algorithm runs in polynomial time; instead of the Fixed Parameter Tractable algorithm, significantly reduces the time required to compare pedigrees till the second generation. However, this is not possible owing to large amounts of missing datasets leading to difficulty in reconstruction. Also, due to the low-level accuracy while reconstructing the pedigree and detecting the isomorphic trait.

Section 3 presents the algorithm, a set of Bipartite graphs Covering the Pedigree (BCP), and it is shown to be fixed-parameter tractable that can be solved in $|n|0(1)$. Moreover, parameterized algorithms and kernelization algorithms for the problem defined are also presented.

3. RESULTS

This section presents the algorithms developed to solve the defined problem, along with the analysis of each algorithm.

http://dx.doi.org/10.21622/ace.2022.02.1.043

3.1. A Set of Bipartite Graphs Covering the Pedigree Graphs (BCP)

Before formulating the problem solution, it is required to introduce some definitions before using the concept. Let (F_0, P) be an illustration of a BCP problem, where the total number of individuals in the family F_0 is equal to the number of individuals in pedigree P. It is required to prove that the total number of subfamilies in F_0 forms a sub-pedigree of P. As per the research conducted in [14], F_0 becomes a sub pedigree of P if and only if removal of K - 1 edges from pedigree P transforms it into F_0. The approach taken in this paper is the creation of sub families, succeeded by pairing of sub families using PairBipartite, which ensures that if the family F^* comprises of a sub-family that is constituted of a connected component, then it is possible to identify a sub-family in F_0 and conclude its image in F^*. The proof to the above is illustrated below:

Input Two families F_0 and F^* have the same number of individuals, where F_0 is a collection of disjoint sub families.

Parameter K, where $K = K_0$ - F^* and K_0 and K^* are the number of connected components in F_0 and F^*. The number of edges in F^* that are not in F_0 would be equal to the number of edges which when removed would transform F^* to F_0 family.

Lemma 1 *Let F_i be a family with K tuple leaves $(l_1, l_2, ...l_k)$ in the same family. Let the labelled leaves be denoted from $1 \leq i \leq s$ and the unlabelled leaves be $s + 1 \leq i \leq K$. If u and v are two individuals and $u \neq v$, and u and v are connected by edges $(l_{u1}, ...l_{us})$, $(l_{vs+1}, ...l_{vk})$, and are not equal, then u and v are parents of the children $(l_{ui})_{i=1\ to\ s}$ and $(l_{vi})_{i=s+1\ to\ k}$.*

The sub family algorithm then takes into account the leaves and their corresponding information of parents and labels and separates them into sub families $k_{m,n}$.

3.1.1. Sub-Family Algorithm $(F, k_{m,n})$

Input: Instance of sub-pedigree F, where g is the gender function and $g =$ {male, female} or $g =$ null. X belongs to the set of vertices $(I(P))$, which are leaves, w_i are the labelled leaves, $1 \leq i \leq$K, and t is the set of unlabelled leaves. A set of sub-families $k_{m,n}$ is created as follows :

1. Let u be an individual in a sub-pedigree F
2. For all the individuals in this sub-pedigree F
3. If u has out degree $= \{w_i\}$ and in-degree $= \emptyset$ then

   {

   If $w_i \in X$ //(marks the presence of labelled leaves) then

   Store $m \leftarrow u$ and $n \leftarrow l_{wi}$;

   Elseif $w_{i-t} \in X$ and $w_t \in X$ // (marks the presence of unlabelled and labelled leaves) then

   Store $m \leftarrow u$ and $n \leftarrow l_{wi}$ and n_t null

   Else Store $m \leftarrow u$ and $n \leftarrow g(null)$; // marks the presence of unlabelled leaves only

   }

http://dx.doi.org/10.21622/ace.2022.02.1.043

4. Else Delete $m+n$ from F // here u is a leaf and its parent is v
5. If $F \neq \ominus$ then return to step 2;
6. Else Stop

3.1.2. Pairing Sub-Families in F^0 and F^*

F^* has a connected set of sub families as a bipartite graph. If F_0 is a sub-pedigree of F^*, then a sub family of F_0 can be directly mapped to a sub family of F^*. This is done by Pair Bipartite algorithm. An example is shown in Figure 3.

Select the largest sub-family $K_{p,d}$ that belongs to F^* and map with a sub family from F_0. In case of labelled leaves, leaves with similar labels are retained while those that are unlabelled have edges removed to transform the sub family of F_0 into a sub family of F^*. $K_{p,d}$ is transformed into $K_{v,w}$ as shown in the example Figure 4.

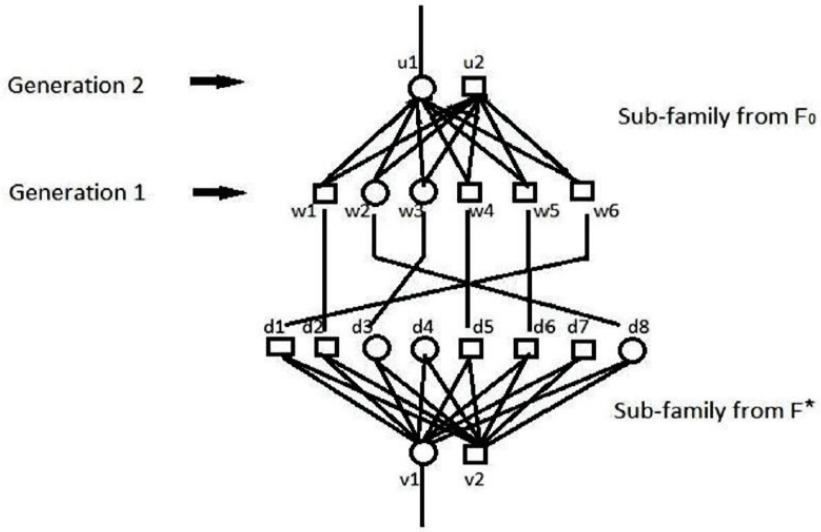

Figure 3: In the case of a labeled leaves: Determine the similar leaves between two sub-families from F_0 and F^* and remove the leaves with different labels.

PairBipartite(F_0,F^*,$K_{p,d}$)
Input : F_0, F^* are sub families and $K_{p,d}$ is the sub family
// t = the number of labelled leaves in the sub-family in F^*
// b = the set of sub families in F_0
b = Sub Family Algorithm(F_0, $K_{v,w}$)
Suppose b is the set of sub families in F_0 and the sub families are smaller than $K_{p,d}$;
If b = null then F_0 is not a sub pedigree of F^*
Else the largest sub family $K_{v,w}$ in F_0 is selected
{ If there exists labelled leaves in the sub family then
Remove $d-w$ leaves (i.e. leaves that are in d and not in w) and have different labels such that
$K_{p,w}$ becomes a sub family of F^*
Else if there exists labelled and unlabeled leaves in a sub family present in set b then

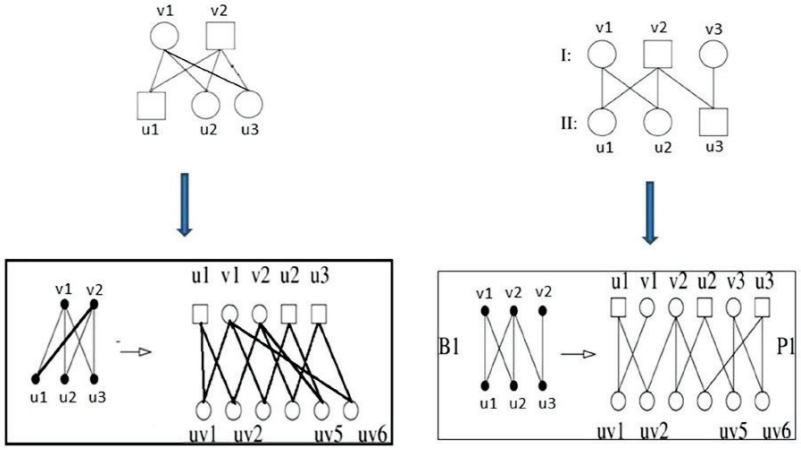

Figure 4: Mapping two sub-families with unlabeled leaves by using the MCIP algorithm in Amar et al. [14]

Remove $d-w$ leaves in $K_{p,d}$ those whose names are null or have different names
Else if unlabelled leaves are present in the sub family then
Remove $d-w$ edges;
Convert $K_{v,w}$ in F_0 and $K_{p,d}$ in F^* that forms the sub families of F_0' and F^*
Initialize empty set $E(f_1)$, $I(f_1)$
Suppose v is marked as the male vertex where $v \geq 2$;
Suppose w is marked as the female vertex where $w \geq 0$;
F_1 the new sub-family is designated as $(I(f_1), E(f_1))$
Generation 1: $I(f_1) \leftarrow v \cup w$
Generation 2: replace each edge between v and w by a new leaf vertex which is marked as female
$\mathrm{E}(F_1) \leftarrow E(f_1) \cup \{(v, vw), (w, vw)\}$
Repeat the steps to convert the sub tree $K_{p,w}$ in F^* into f_2 denoted by $(I(f_2), E(f_2))$, Match f_1 and f_2 using the MCIP algorithm as discussed by Jiang [13] to exhibit isomorphism across pedigrees and levels. Justifying that evaluation of the reconstructed pedigree exuding the isomorphic behaviour to the original can be used by geneticists accordingly.

The validity of the Algorithm Bipartite is embedded in Lemma 2, which presents an assumption and substantiates it with a proof:

Lemma 2 *(F_0,F^*,$K_{p,d}$) outputs two sub-families $F_{0'}$ and $F^{*'}$ which are sub pedigrees of F_0 and F^*, respectively. Then F_0 is a sub pedigree of F^* if and only if $F_{0'}$ is a sub pedigree of $F^{*'}$.*

Proof. Initially the sub family algorithm (F_0, $K_{m,n}$) is executed to determine all the sub families in F_0 and store the information of the sub families.

"If b = null, then F_0 is not a sub pedigree of F^*" can be proved through contradiction. Consider that σ denotes isomorphic mapping from F_0 to a sub pedigree of F^*. Then σ would chart a vertex or an individual in F_0 to the parent of sub family $K_{v,w}$ in F^*. The possible cases that v can be either parent or leaf of the sub family $K_{p,d}$ proves that F_0 has a sub family not larger than $K_{p,d}$. This contradicts the proposition that b is null. Let there be two sub families that are denoted as outputs $F_{0'}$ and $F^{*'}$. Execution of the Step 4 of the Pair Bipartite Algorithm removes some edges from F^* and then removes two identical families, one from F_0 and F^*. So $F_{0'}$ and $F^{*'}$ can be written as $F_{0'} = F_0/K_{v.,w}$ and $F^{*'} = F^*/K_{v,w}$ and in the case of $F^{*'}$ all the edges incident to

http://dx.doi.org/10.21622/ace.2022.02.1.043

d in $K_{p,d}$ but not to w in $K_{v,w}$ are removed. This proves that $F_{0'}$ and $F^{*'}$ are sub pedigree of F_0 and F^*. Since the $F_{0'}$ and $F^{*'}$ are the sub pedigrees of F_0 and F^*, if an identical sub family is removed then if F_0 is not a sub pedigree of F^*, then $F_{0'}$ cannot be a sub pedigree of $F^{*'}$.

Now, if F_0 is a sub-pedigree of F^* and σ is an isomorphic mapping from F_0 to a sub-pedigree of F^*, the mapping σ must map a sub-family in F_0 to the sub-family in F^*. Select the largest sub family $K_{v,w}$ from the sub pedigree F_0 and perform the following operations :

- In case of the labelled leaves, the number of edges that have to be removed is $d-w$ edges, where edges starting from d and connecting w leaves should be of the same gender, otherwise they have to be removed.
- In case of unlabeled leaves, remove $d-w$ edges
- In case of labelled and unlabelled leaves in the sub family, save the edges with common gender and remove the edges which traverse to unlabelled leaves.

If the two sub families are similar, then it is validated that a sub family in F_0 can be mapped onto sub family of F^*. To validate isomorphism, it is required to prove that $K_{v,w1}$ and $K_{p,w1}$ are isomorphic if two levels of sub families are isomorphic. The sub family $K_{v,w}$ converts into a new form by assigning males to all individuals v and females to individuals w. The new sub family is formed by creating new individuals vw_1, vw_2 that are marked as female and the set of edges denoted as
$E(f_1) \leftarrow E(f_1) \cup \{(v, vw), (w, vw)\}$ in $k_{v,w1}$ and the set of edges in new sub family sub(f_2) takes the form
$E(f_2) \leftarrow E(f_2) \cup \{(v, vw), (w, vw)$ in $K_{p,w1}$.

Mapping is applied by using (MCIP) algorithm between f_1 and f_2 to prove the isomorphism as described by Jiang [13]. For all other sub families in F_0 mappings are similar to that of σ.

Let the mapping be σ and let it map the sub families in F_0 to their matching families in F^*. The mapping σ and σ are the same on all other individuals in F_0. The mapping induced by σ on individuals of $F_{0'} = F_0/K_{p,w}$ is an isomorphic mapping from $F_{0'}$ to a $F^{*'} = F^*/K_{p,w}$. Hence, $F_{0'}$ is a sub pedigree of $F^{*'}$ which proves the lemma.

Algorithm PairBipartite and Lemma 2 have helped prove that if there exists a connected component in F^* which is a sub family then a sub family also exists in F_0 and directly determine its mirror image and all other sub family's mirror image in F^*.

3.1.3. Parameterized Algorithms for the BCP Problem

The algorithm Pair BiPartite can be repeatedly applied as long there are sub families in F_0 for which images can be created in F^*, while removing sub families from F_0 and $F^{*'}$ till there exists no more sub families. If F^* is not empty, then F^* constitutes of a path $sp = [u_0, u_1, u_2]$ of length equal to 2. If the collection of sub families exist in F_0 and if F_0 is a sub pedigree of F^*, then there will exist at least one edge among the two edges of sp that will be absent from the isomorphic image from F_0 to F^*. The edge can be removed and F_0 continues to be a sub pedigree of F^*. Having removed the edge in F^*, a branch and search algorithm of time complexity $2^k.n^{O(1)}$ is used, where k is the parameter of the instance (F_0,F^*), since the number of edges decreases by 1. The value k here is equal to number of edges in F^* minus the number of edges in F_0.

http://dx.doi.org/10.21622/ace.2022.02.1.043

To summarize and simplify the discussion, an isomorphism, σ, from F_0 to a sub-pedigree of $F*$, will narrow down to the fact that an edge e in F^* is in σ if the edge e is present in the image of σ. This can further be validated in the consecutive sections.

Lemma 3 *Suppose that (F_0,$F*$) is the instance of (BCP) containing labeled and unlabeled leaves, and the edges be e_1 = [a,b] and e_2 = [b, c], be the two different edges in F^* and they share the common vertex b. If F_0 is a sub pedigree of F^*, then for every isomorphism, σ, from F_0 onto F^*, there exists at least one edge e_1 or e_2 that is not isomorphic or does not have an image in F^*.*

Proof. Let F_0 be a sub pedigree of F^* and two edges e_1 and e_2 be the edges that are in isomorphic mapping from F_0 to a sub pedigree of F^*.

Algorithm Branch-path (F_0,F^*)
Input: (F_0,F^*) is an instance from path covered sub-pedigree
Let u be an individual node that has an out-degree
If $u \leq 1$ in F^* then return;
Elseif $u > 1$;
For all distinct edges e1,e2 in u do
Return Branch-path (F_0,F^*/(E_u/e1,e2)) ;
If u does not exist then stop.

Lemma 4 *Let (F_0,F^*) be an instance of BCP. Let there be a simple path $sp = [u_0, u_1, u_2]$ in F^*, where u_0 is a leaf and u_2 is an individual of degree 2. If F_0, a sub pedigree of F^*, constitutes of labelled and unlabelled leaves, then there exists an isomorphism σ from F_0 onto F^* such that at least one of the edges in the path sp is not mapped by σ*

Proof. Assume that edges $[u_0, u_1], [u_1, u_2]$ are in σ and exhibit isomorphism from F_0 to a sub pedigree of F^*. This is with reference to the proof of lemma 3, which states that no edge of the form $[u_0, z], [z, u_2]$ can be in σ, where z $\neq$ u_2. Since u_2 has an in-degree 2 in F^*, the path $[u_1, u_2]$ is the image of the sub family of the type $K_{p,d}$ under σ in F_0. u_0 is the leaf in F^* and image of the type $K_{1,0}$ in F_0. The isomorphic mapping σ can be modified, such that the sub family of the type $K_{2,2}$ in F_0 be mapped onto $[u_1, u_2]$ in F^* and the sub family of the type $K_{1,0}$ in F_0 be mapped to the leaf u_0 in F^*. It then becomes easy to validate that the final mapping σ' is a σ mapping from F_0 to F^* with the edge $[u_0, u_1]$ not in σ'.

The incorporation of the above lemma has been done using the BCP-I and BCP- II algorithms. The algorithms are described as follows:

In the BCP-I algorithm, an instance of the problem is considered in which F_0 is a set of sub families and F^* is a family. The objective is to find out whether F_0 is a sub pedigree of F^*.

If F^* is null, then F_0 is a sub pedigree of F^*, else if K $<$ 0, then F_0 is not a sub pedigree of F^*. If it contains a single sub family then PairBipartite algorithm will be called to reduce the number of individuals in that sub family, which has same number of individuals in F^* and F_0. Then both families F_0 and F^* are tested for similarities or isomorphism. The other criterion within the PairBipartite can be that $F^{*'}$ contains a simple path of length equal to 2 and $F^{*'}$ can contain labelled and unlabelled leaves where each of the branches will be processed.

http://dx.doi.org/10.21622/ace.2022.02.1.043

Algorithm BCP-I(F_0,F^*,k)
Input : F_0, F^* is an illustration of the BCP problem where F_0 is a set of sub families and F^* is a family.
Output : F_0 is a sub pedigree of F^*

1. If $F_0 = \emptyset$ then Stop //(F^* constitutes the sub pedigree F_0)
2. Else if K < 0 then Stop //(F^* does not constitute the sub pedigree F_0)
3. If F^* contains sub family $K_{p,d}$ then
4. $(F_0',F^{*\prime})$ = PairBipartite(F_0,F^*,$K_{p,d}$); //(k = the difference in the number of edges between the two)
5. Return to BCP-1($F_{0'}$, $F^{*'}$);
6. If F^* is not empty then a simple path $sp = [u_0, u_1, u_2]$ is chosen which is of length 2 then
   a. If u_2 has an out-degree =2 in F^* then
      Branch-Path(F_0,$'$ $F^{*'}$);
      Return BCP-I(F_0,$F^*\backslash$u$_1$,u_2,$k-1$);
   b. Else
      Branch-Path (F_0',$F^{*\prime}$);
      Return BCP-I(F_0,$F^*\backslash$(u$_0$,u_1),[$(k-1)$ or (u_1,u_2)];

Theorem 2 *The algorithm BCP-I solves the problem BCP, by determining F_0 is a sub-pedigree of F^*. The parameter 'k' is the difference between the number of edges in the two sub-pedigrees F_0 and F^* with labelled and unlabeled leaves.*

Proof
Input: (F_0, F^*) as an instance of the BCP problem. Families F_0 and F^* have equal number of individuals. The algorithm terminates execution in case of no individuals are in F^* ($F^* = \emptyset$, for k < 0. Hence, F_0 is not a sub pedigree of F^*.

If F^* contains a sub family $K_{p,d}$ within its family, then according to lemma 3 (F_0' $F^{*'}$) is returned by the algorithm Pair BiPartite(F_0' F^*;$K_{p,d}$) and is solved by recursively calling BCP-I (F_0,F^*,k).

Now, if $F^{*'}$ contains a path $sp = [u_0, u_1, u_2]$ and u_0 is the leaf, then if u_2 in $F^{*'}$ is an individual of degree 2, by lemma 4, if F_0 is a sub pedigree of F^*, then there exists an isomorphism from F_0 to F^* where edge (u_1,u_2) is not in σ.

Therefore if F_0 is a sub pedigree of F^* then F_0 is also a sub pedigree of $F^*\backslash$ (u_1,u_2). If the degree of the individual, say u_2, is larger than 2 then lemma 3 can prove that with every isomorphism mapping σ from F_0 onto F^* either (u_0,u_1) or (u_1,u_2) is not in σ or no edges in the set of edges belonging to individual u_2 is in σ. Thus, if F_0 is a sub pedigree of F^* then the algorithm will return a YES else will return a NO for the instance F_0, F^*.

Theorem 3 *The algorithm BCP-I(F_0, F^*) solves the BCP problem in time O(2.42Kn(k+logn)) where n is the size of an instance which contains labelled and unlabelled leaves. The parameter k represents the difference between the number of edges in the sub pedigree F_0 and F^*.*

http://dx.doi.org/10.21622/ace.2022.02.1.043

Proof

The sub families in F_0 can be organized based on the size of the sub families using appropriate data structures. The largest sub family $K_{v,w}$ in F_0 for a given sub family $K_{p,d}$ in F^* can be found in O(log n) time. If F^* is not empty then from u_0, which is the leaf node in F^*, a path of the form $[u_0,u_1,u_2]$ can be constructed in F^* of length equal to 2.

The Branch and Search Algorithm in BCP-I can be described as a bounded search in which each internal node having more than one child maps to a branch in the algorithm, and the leaf vertex matches to a decision formulated by the algorithm. By removing sub families from F_0 and F^*, a path of length 2 is constructed in F^*. To find out the sub families by removing edges at each step from F^*, the algorithm takes K branching steps, and a time of O(n) to identify the sub families. The algorithm takes time of O(nlogn) for removing the matching sub families.

Assume that, u_2 has a degree ≥ 2 so that, the subset edges E_{u2} contain at least 2 edges, then a computational path can be traced to each leaf. This is achieved using the result of the recurrence relation in You et al. [19], this search results in tracing the path to 2.42K leaves.

Finally, the run time of the algorithm BCP-I is O(2.42K n(k + log n)). This is the time taken to identify sub families and remove them, in addition to the time to search for the computational paths to the leaves when edges are more than 2.

A second version of the algorithm which diminishes the time complexity as per the results reported by [20] is presented as BCP-II. This algorithm executes in $O(bK.n^{O(1)})$ where b is less than 2.42, and n is superimposed by a higher degree polynomial. The BCP-II algorithm is presented in below.

Algorithm BCP-II(F_0, F^*, k)
Input : F_0, F^* is an illustration of the BCP problem where F_0 is a set of sub families and F^* is a family.
Output : F_0 is a sub pedigree of F^*
Parameter: K = $K_0 - K^*$, the difference between the number of edges in F_0 and the number of edges in F^*
If F^* = null then Stop // F^* constitutes the sub pedigree F_0
If k < 0 then Stop// F^* does not constitute the sub pedigree F_0
Select an individual u_1 with the largest out-degree from F^*
If degree(u_1) $\leq c$ then solve by using [20]
If u_1 is a parent P and is in an isolated sub family $K_{p,d}$ then
$(F_0',F^{*'})$ = PairBipartite (F_0, F^*, $K_{u,d}$)
Return BCP-II(F_0', F^*, c, k)
Else select a simple path $sp = [u_0, u_1, u_2]$ of length 2 in F^* then
BranchPath(F_0',F^*)
Return BCP (F_0, $F^*\backslash(u_0,u_1)$, c, $k-1$)
OR BCP (F_0, $F^*\backslash(u_1,u_2)$, c, $k-1$)
OR BCP (F_0, $F^*\backslash Eu_1$,c,k-1,Eu_1);

Associated with the above algorithm is the theorem and the subsequent proof that algorithm BCP-II can be executed in a time that is upper bounded by $O(bK.n^{O(1)})$ where c is a constant ≥ 1, n is the size of the sub pedigree instance (F_0,F^*) constituting of labelled and unlabelled leaves, and k denotes the difference of number of edges.

http://dx.doi.org/10.21622/ace.2022.02.1.043

Theorem 4 *The algorithm BCP-II (F_0,F^*,c,k) solves the problem BCP in time O(bK $n^{O(1)}$), when there exists a constant c ≥ 1, and n is the size of the instance (F_0,F^*) with labelled and unlabeled leaves, and k denotes difference in the number of edges between sub pedigrees F_0,F^*.*

Proof

The algorithm BCP-I, undergoes changes with the inclusion of step 4, as denoted in the preceding BCP-II algorithm, which requires validation and justification. In a study conducted by Baumbach et al. [20], they discussed (Star Cover Tree) SCT problem which constitutes of a tree and a set of stars. These stars can be connected through edges between them in such a way that the resulting tree is isomorphic to T. Baumbach et al. [20] have proved that when the count of unique stars is bounded by a constant k, CTS can be solved in polynomial time of order $O(n^{2c+3})$. In this work, simple undirected graphs are considered. The star constitutes of a set of vertices connected by edges that define the size with at least one internal vertex.

The following is built on Theorem 2 in [14], and the fact that the bipartite graph consists of two equal sized stars, such that one of the two stars is present on the path. Any sub-pedigree with labelled and unlabeled leaves represents a branch of the sub-pedigree which has a connected component of distinct sub-families, so that the Tree Edit Distance with insertion and deletion of edges can be used to solve the BCP problem, where c represents various sizes of sub-families in F_0.

Moreover, if the sub-families in F_0 have size bounded by c, then F_0 has at most (c + 1) sub-families of various types inclusive of sub-families without children having a size of 0. Therefore, the BCP problem can be solved in time of $O(n^{2c+5})$. When all the connected components of sub families of size (c + 2) in F^* are linked, such that the sub-families form a new sub pedigree graph with a definite number of sub-families, and have the same size in F_0, are added on to F^*, then the algorithm executes in time of $O(n^{2c+9})$, where the sub-families in F_0 have their size bounded by c.

If an individual has a degree ofr edge $(u_1) \leq c$, in order to let F_0 be a sub-pedigree of F^*, no sub-family in F_0 can have size exceeding c because u_1 is the individual with the maximum out degree in F^*. Therefore, from the discussion above this step is solved in time $O(n^{2c+9}) = n^{O(1)}$, where the polynomial component is the complexity of higher degree. If u_1 is not an isolated sub-family in F^*, then there exists a simple path of length 2.

Applying lemma 4, the complexity for each computational path in the search takes polynomial time, again the branching of the algorithm gives the recurrence relation P(X) = 2P(X − 1) + P(X–c), where P(X) defines the number of leaves.

Using the standard method shown by Chen et al. [21], the above recurrence relation is solved to obtain $P(x) = x^c - 2x^{c-1} - 1$, with root b > 2, and the total number of leaves in the search is bK. Thus, the algorithm BCP-II is executed in $b^k n^{O(1)}$.

Next the concept of Kernelization is introduced and it is proved that linear time kernelization algorithm yields a kernel of size $O(K^3)$ for the BCP problem. The theorem to prove the performance of the algorithm follows.

Theorem 5 *The problem "A set of Bipartite graphs Covering the Pedigree graphs (BCP)" is executed in time $O((2 + EPSILON)^k n^{O(1)})$, where the constant EPSILON ¿ 0.*

http://dx.doi.org/10.21622/ace.2022.02.1.043

Proof

The proof depends on theorems 3 and 4 for solving the algorithms BCP-I and BCP-II, when EPSILON ¿ 0. Given an instance (F_0, P) of the BCP problem, a reduction rule is applied by the Kernelization algorithm which removes certain leaves from F_0 and P such that there exists a sub family in F_0 whose size is bounded by O(K^2). It is said that a K vertex cover has a kernel of size O(K^2).

K vertex cover algorithm tries to identify whether a set of vertices in a graph covers all the edges of the graph or not. As a subsequence, the Kernelization algorithm takes a space which is upper bounded by O(K^3), that is the size of an instance (F_0', P′) of BCP, where the total number of individuals are bounded by O(K^3), that is the time taken to identify and remove the sub families and to identify the branch path till the leaf node. Proof of the above algorithm is done using the method of construction in lemma 5.

Lemma 5 *Suppose that (F_0, P) is an instance of BCP. Let z be any individual with an out-degree larger than k + 1 in the sub-pedigree F^*. Then there exists an isomorphism from F_0 to a sub-pedigree of P, where the individual z is the image of the parent of a sub-family in F_0. Moreover, at least degree(z)–k + 1 neighbours of z are leaves in F^*.*

Proof

Let σ be an isomorphism from F_0 to a sub pedigree P, such that that there exists a set E_σ in F^* that cannot be in the set E_σ, but if z is the image of a leaf then it is bounded by 1 only. Also z has a degree $\geq K+1$ in F^* which contradicts the assumption that E_σ has only k edges. Hence, z has to be the image of the internal parent of a sub family in F_0.

Kernel BCP (F_0,P,$K_{m,n}$)

Input: F_0,P is an instance of BCP
// b is the set of sub families in F_0
Output: (F_0',P')
If the number of individuals in P = 0;
Then stop NO-instance of BCP;
b = Sub-family Algorithm(F_0,$K_{p,w}$);
T = Sub-family Algorithm (F^*,$K_{u,x}$);
If | b | ≠ | T | then stop;
F_0 is not a sub-pedigree of P
Order the sub-families in F_0 by ascending order that makes a gap;
For each two adjacent sub-families $K_{p,d}$ and $K_{p,w}$ that d−w > k + 1 do
Arbitrarily remove $d-w+K+1$) leaf individuals from each sub-family in b and T;
If the degree of the smallest sub-family in F_0 is greater than K+2 then
Return Kernel-BCP (F_0', P',$K_{m,n}$)
Return (F_0,P)

The kernelization algorithm for BCP is applied repeatedly, whenever there is a sub-family in F_0 that can be removed. At the end of the process, there are subfamilies that cannot be removed and the researchers obtain equal instances (F_0', P′) of BCP.

Lemma 6 *Suppose that the algorithm Kernel-BCP returns an instance (F_0',P') which are sub pedigrees of F_0 and P. F_0 is a sub pedigree if and only if F_0' is a sub pedigree of P'.*

http://dx.doi.org/10.21622/ace.2022.02.1.043

Proof

The families F_0 and P must have the same number of individuals. If P = Øor the number of sub families in $F_0 \neq$ P then there is no instance of the BCP problem. Step 1 and Step 4 of the algorithm handle these cases appropriately.

Assume that sub families in F_0 are ordered by the ascending order and there are two adjacent sub families $K_{p,d}$ and $K_{v,w}$ (w < d) in F_0 where d – w > k + 1, respectively. Images of the sub family in F_0 are found in P directly and some being larger than K_p, d are redundant and their images are removed. This process is used to reduce the scale of F_0 and P for an isomorphism from F_0 to P.

Theorem 6 *The Kernel-BCP algorithm is a linear time algorithm that returns an instance (F_0', P') for the BCP problem, and the number of individuals in a new instance is bounded by O (K^3), where k represents the count of sub-families in F_0.*

Proof

Since the Kernel-BCP algorithm is repeatedly applied whenever there is a sub-family in an ascending order in F_0 that creates a gap when removed. Therefore, all sub-families that can be removed from F_0 are removed in a single scanning in an ascending order in F_0. Once a sub-family which creates a gap is identified, a set of leaves in F_0 and P can be removed in linear time. Then, size of each sub-family in F_0 is not larger than n (n = the size of the sub-families) and k represents the number of sub-families. The sorting of the sub-families can be done in O(kn) time, which can be reduced to O(n) by using Counting-Sort. From the above discussion and lemma (6) and (5), an equivalent instance (F_0',P′), the total count of individuals in F_0' and P′ is bounded by O(K^3).

4. CONCLUSIONS AND FUTURE WORK

The study can extend towards the evaluation of reconstructed pedigree graphs until the third generation limited to the families and sub-families and also as a generalized algorithm until the seventh generation.

The non-availability of the requisite datasets acted as a limitation to the practical verification of the study. Although, from the theoretical viewpoint the algorithms showed promising results.

The evaluation of a large pedigree graph with labeled and unlabeled leaves, and proving isomorphism with the real pedigree is a problem that has been addressed critically and the solution has been presented to the geneticist for use. The paper introduces the problem of comparing labeled and unlabeled pedigree graphs. It also presents the parameterized and kernelization algorithms. The parameterized algorithms are supported by the standard branch-and-search process. Kernelization algorithms have a complexity of $O(k^3)$ for BCP. For further progress in research, the model presented can be generalized to allow the evaluation of unlabeled sub-pedigree with a large number of individuals and sub-families up to seven generations. Furthermore, experimental results are needed to test the applicability of the proposed algorithms. The evaluation of a largely reconstructed pedigree with labeled and unlabeled leaves and establishing isomorphism with the original shall help in unearthing many genetic facts with regards to the diseases.

http://dx.doi.org/10.21622/ace.2022.02.1.043

The research compares the labeled and unlabeled pedigrees and sub pedigrees with the original and uses the parameterized and kernelization algorithms to generate promising results as compared to theoretical study. Here, the kernelization algorithm utilizes the concept of reducing the size of the input as a function of the parameter, thereby removing the sub-pedigrees and sub-families to prove the isomorphic attribute with the original. The future scope of work is to incorporate the generalization of algorithms to accommodate multiple generations of a pedigree.

REFERENCES

[1] D. He, Z. Wang, E. Eskin, "Iped2: Inheritance path based pedigree reconstruction algorithm for complicated pedigrees," IEEE/ACM Trans. Comput. Biol. Bioinformatics, Vol.14, issue no 5, pp.1094-1103, September 2017.

[2] M. Ghimiray, R . Vernooy."The importance and challenges of crop germplasm interdependence: the case of Bhutan," Food Secur. Vol. 9, pp.301-310. April 2017.

[3] TY. Berger-Wolf, SI. Sheikh, et al., "Reconstructing sibling relationships in wild populations," Bioinformatics, Vol. 23, pp.49-56. 2007.

[4] Brown, DG.; Berger-Wolf, T. Discovering kinship through small subsets, In Algorithms in Bioinformatics; Moulton, V., Singh, M., Eds.; Springer:Berlin, Heidelberg, 2010; pp. 111-123.

[5] Thompson, EA. In Pedigree Analysis in Human Genetics ;Baltimore: Johns Hop-Kins University Press, 1986.

[6] B. Kirkpatrick, SC. Li, RM. Karp, E. Halperin. "Pedigree reconstruction using identity by descent," J. Comput. Biol. Vol.18, issue no 11, pp.136-152. November 2011.

[7] L. Sun, K. Wilder, MS. McPeek. "Enhanced pedigree error detection," Hum Hered. Vol.54, issue no 2, pp. 99-110. October 2002.

[8] MS. McPeek, L. Sun. "Statistical tests for detection of misspecied relationships by use of genome screen data," Am. J. Hum. Genet. Vol.66, issue no 3, pp.1076-1094. March 2000.

[9] M. Boehnk, NJ. Cox. "Accurate inference of relationships in sib-pair linkage studies," Am. J. Hum. Genet. Vol.61, issue no 2, pp. 423-429. August 1997.

[10] B. Kirkpatrick, Y. Reshef, H. Finucane, H. Jiang, B. Zhu, RM. Karp. "Comparing pedigree graphs," J. Comput. Biol. Vol.19, issue no 9, pp. 998-1014. October 2012.

[11] Z. Chen, Q. Feng, C. Shen, J. Wang, L. Wang. "Algorithms for pedigree comparison," IEEE/ACM Trans. Comput. Biology Bioinform, Vol.15, issue no 2, pp. 422-431. April 2018.

http://dx.doi.org/10.21622/ace.2022.02.1.043

[12] CM. Herbinger, P. TO'Reilly, RW. Doyle, JM. Wright, F. O'Flynn. "Early growth performance of Atlantic salmon full-sib families reared in single family tanks versus in mixed family tanks," Aquaculture, Vol.173, issue no 14, pp. 105-116. March 1999.

[13] H. Jiang, G. Lin, W. Tong, D. Zhu, B.Zhu. "Isomorphism and similarity for 2-generation pedigrees," BMC Bioinformatics, Vol.16, issue no Suppl(5), pp. 1-8. March 2015.

[14] LA. Amar, NA. Belal, S. Rashwan. "Comparing unlabeled pedigree graphs via covering with bipartite and pat," J. Comput. Biol. Vol.23, issue no 11, pp. 912-922. November 2016.

[15] Braun, Bremen L and Schott, David A and Portwood, John L and Andorf, Carson M and Sen, Taner Z. "PedigreeNet: a web-based pedigree viewer for biological databases," Bioinformatics, 35(20), pp. 4184-4186 October 2019.

[16] N. Xie, Q. Sun, J. Yang, Y. Zhou, H. Xu, L. Zhou, Y. Zhou. "High clinical heterogeneity in a Chinese pedigree of retinal vasculopathy with cerebral leukoencephalopathy and systemic manifestations (RVCL-S)," Orphanet Journal of Rare Diseases,16(1), pp.1-15, Jan 2021.

[17] H. Yu, G.Morota. "GCA: an R package for genetic connectedness analysis using pedigree and genomic data," BMC Genomics, 22(119), pp.1-8, February 2021.

[18] Z.J. Rawandoozi, T.P. Hartmann, S. Carpenedo, K. Gasic, et al., "Mapping and characterization QTLs for phenological traits in seven pedigree-connected peach families," BMC Genomics, 22(187), pp.1-16, March 2021.

[19] J. You, J. Wang, Q. Feng, F. Shi. "Kernelization and parameterized algorithms for covering a tree by a set of stars or paths," Theor. Comput. Sci. Vol 607, part 2, pp. 257-270. November 2015.

[20] J. Baumbach, J.Guo, R. Ibragimov. "Covering tree with stars," In: Computing and Combinatorics.;19th International Conference; COCOON; Springer: Hangzhou, China, 2013.

[21] J. Chen, IA. Kanj, W. Jia. "Vertex cover: Further observations and further improvements," J. Algorithms, Vol.41, issue no 2, pp. 280-301. November 2001.