Computational Biology and Bioinformatics
2021; 9(1): 1-14
http://www.sciencepublishinggroup.com/j/cbb
doi: 10.11648/j.cbb.20210901.11
ISSN: 2330-8265 (Print); ISSN: 2330-8281 (Online)

Using Evolutionary Trees for the Colorectal Cancer Prognosis Prediction

Hung-Yu Yan[1], Dun-Wei Cheng[2], Peng-Chan Lin[2, 3], Hsin-Hung Chou[4], Meng-Ru Shen[5], Sun-Yuan Hsieh[2, 6, *]

[1]Institute of Medical Informatics, National Cheng Kung University, Tainan, Taiwan

[2]Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan

[3]Department of Oncology, Department of Genomics Medicine, National Cheng Kung University Hospital, College of Medicine, National Cheng Kung University, Tainan, Taiwan

[4]Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan

[5]Department of Obstetrics and Gynecology, National Cheng Kung University, Department of Pharmacology, National Cheng Kung University Hospital, College of Medicine, National Cheng Kung University, Tainan, Taiwan

[6]Institute of Manufacturing Information Systems and Institute of Medical Informatics, National Cheng Kung University, Tainan, Taiwan

Email address:

Q56051079@gs.ncku.edu.tw (Hung-Yu Yan), dunwei.ncku@gmail.com (Dun-Wei Cheng), pengchanlin@gmail.com (Peng-Chan Lin), chouhh.g@gmail.com (Hsin-Hung Chou), mrshen@mail.ncku.edu.tw (Meng-Ru Shen), hsiehsy@mail.ncku.edu.tw (Sun-Yuan Hsieh)

[*]Corresponding author

To cite this article:

Hung-Yu Yan, Dun-Wei Cheng, Peng-Chan Lin, Hsin-Hung Chou, Meng-Ru Shen, Sun-Yuan Hsieh. Using Evolutionary Trees for the Colorectal Cancer Prognosis Prediction. *Computational Biology and Bioinformatics*. Vol. 9, No. 1, 2021, pp. 1-14.
doi: 10.11648/j.cbb.20210901.11

Received: January 2, 2021; **Accepted**: January 29, 2021; **Published**: February 10, 2021

Abstract: The measurement of tree similarity based on structure comparison has been long used in diverse fields. We applied the evolutionary tree method to study the cancer genome. Cancer evolutionary trees, representing cancer diversity, provide information on the clonal evolution and the clinical outcome of cancer patients. This study considered 107 colorectal cancer (CRC) patients who received deep targeted sequencing of cancer tissues. The evolutionary trees of individual cancer patients were reconstructed from genome sequencing data based on variant allele frequencies (VAFs) of point mutations and small insertions or deletions (indels). The main purpose of this study was to predict cancer recurrence. We mapped the structure of a cancer evolutionary tree to a rooted tree and developed a canonical-form transformation for solving tree isomorphism to ensure that each patient has a unique tree structure. We proposed an algorithm for comparing tree similarity in terms of cost calculation between evolutionary structure trees. The cost was calculated using the node position, tree size (or number of nodes), tree height, node depth, number of descendants of the node (the size of the subtree with the node as a root), and relationship of the node with other nodes. After tree similarity comparison, the cancer patients were clustered into two groups through *k*-means clustering. The clustering information indicated that the evolutionary structure trees were associated with gender and tumor invasion stage. Several machine-learning strategies including random forest, support vector machine (SVM), bagging, and boosting were used to predict cancer recurrence in these patients. Our results revealed that combining the clustering information of evolutionary structure trees increased the prediction performance compared with using clinical information alone, and tree similarity comparison can help in the prognostic analysis of cancer patients.

Keywords: Cancer Evolutionary Trees, Colorectal Cancer, Evolutionary Structure Trees, Canonical-form Transformation, Tree Simiarity Comparison

1. Introduction

Cancer is a disease caused by the accumulation of somatic mutation. The cancer clonal theory states that most tumors originate in a single cell and that tumor progression and clonal expansions are caused by genetic variability; thus, a tumor is the result of clonal evolution [1]. Understanding the evolutionary history of tumor growth may provide valuable insights into tumor cell proliferation and survival [2]. Many studies have recently integrated evolutionary theory and genomic data into modern techniques for studying tumor growth and progression [3]. In this study, we used the evolutionary tree method to understand tumor progression. The tumor composition and evolutionary structure can be determined through somatic variant calling by using variant allele frequencies (VAFs) according to the read counts of the tumor and matched normal cell sequences in each patient. The VAFs of somatic variants can be used to reconstruct the cancer evolutionary histories as a cancer evolutionary tree, which reflects the somatic variant accumulation in each patient [4, 5].

In recent years, many methods have been developed to construct cancer evolutionary trees from single-nucleotide variants in bulk sequencing data [6-8] and single-cell sequencing data [9-11]. Cancer evolutionary trees are of two types: sample tree and sub-clonal tree. Several methods are used for the construction of a sample tree [12-14], including maximum parsimony [15, 16], distance matrix method [17], maximum likelihood estimation (expectation-maximization (EM) algorithm) [18], and Bayesian sampling (Markov chain Monte Carlo (MCMC)) [19]. A sub-clonal tree clusters driver mutations into several subclones with a common frequency and reconstructs the lineage according to the following two assumptions [20-22]: (1) a mutation cannot recur during tumor evolution and (2) each mutation is acquired once and is never lost. In these trees, the root represents a normal cell and the subsequent node of the root represents a founder cell. The nodes below the founder cell represent descendant subclones, and the edge lengths indicate the number of driver mutations that are newly accumulated in the descendant nodes [23, 24]. In this study, rooted cancer evolutionary trees were constructed with a sub-clonal architecture.

Studies have indicated that the cancer evolutionary tree patterns differ between clear cell renal cell carcinomas and non-small-cell lung cancer [25, 26]. The branching patterns of cancer evolutionary trees and the somatic mutation fraction of subclones are crucial for identification of the type or even the prognosis of cancer.

Studies have measured the similarity between two trees by using edit cost, which refers to the number of insertions or deletions of nodes required for transforming one tree into the other [27-29].

Computational methods for quantifying the similarity between two cancer evolutionary trees based on tree structure have not attracted much research attention [24]. In this study, we mapped the cancer evolutionary tree structure to a rooted tree and developed an algorithm for effective comparison among trees through the exchange of operations between the subtrees. This procedure is referred to as canonical-form transformation and tree similarity comparison. After tree comparison for each patient, we clustered the patients into two groups and developed several machine learning models to predict cancer prognosis. Figure. 1 presents the entire experimental process.

2. Materials and Method

2.1. Patients and Samples

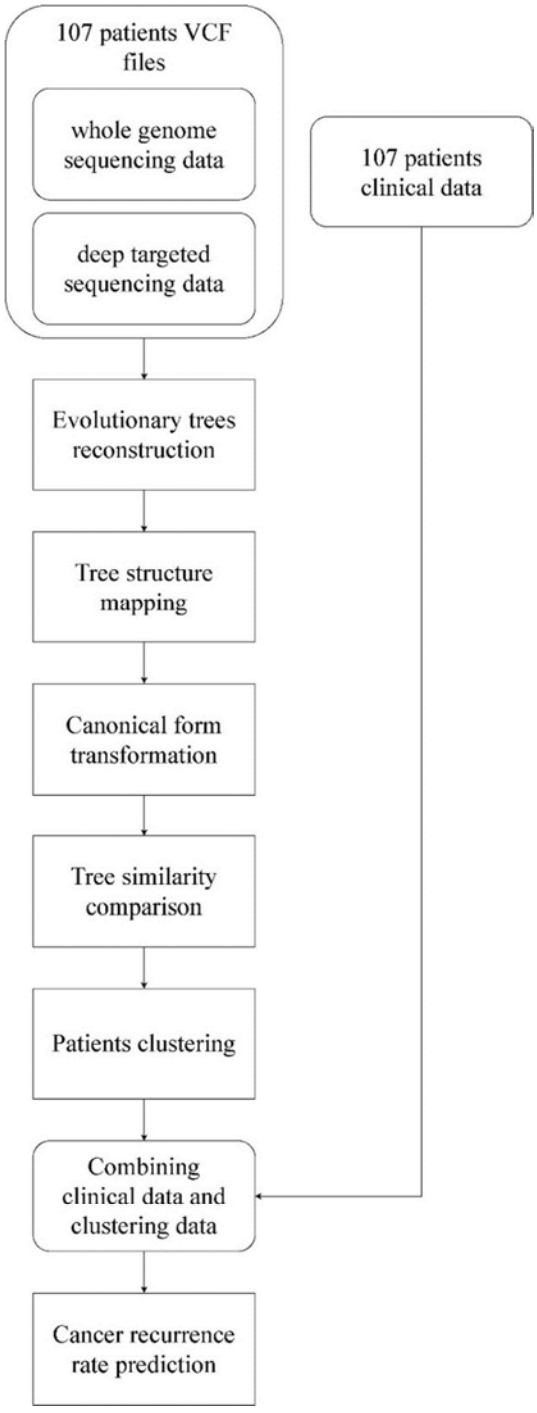


Figure 1. *Overall workflow of the experiment.*

In this study, 107 patients with stage III colorectal cancer (CRC) were recruited from National Cheng Kung University Hospital (NCKUH) between January 2014 and January 2019. All CRC patients received the standard surgical resection followed by adjuvant chemotherapy with the mFOLFOX6 (5-fluorouracil, leucovorin, and oxaliplatin) regimen. Clinical information of the patients was obtained from medical records. Tumor tissue and blood samples were collected at the time of enrollment. This study was approved by the Institutional Review Board of NCKUH (A-ER-103-395 and A-ER-104-153) and was performed in accordance with the guidelines of the Declaration of Helsinki. All participants provided written informed consent.

2.2. Evolution from Cancer Genome

To elucidate tumor development and progression in each patient, we reconstructed the cancer evolutionary tree by using genome sequencing data based on VAFs of point mutations and small insertions or deletions (indels) obtained from variant call format (VCF) files. The VCF files contain information regarding the variants' chromosomes, positions, bases, reference genes and some data retrieved from the Single Nucleotide Polymorphism Database (dbSNP). In this study, we obtained tumor target sequencing and germline whole genome sequencing data of 107 CRC patients. Information regarding somatic mutations was obtained using the tumor-normal single-nucleotide variant (SNV) calling tool deepSNV [30]. We used the ANNOVAR [31] tool to annotate somatic mutations and filter out indel variants not reported in the 1000 Genomes Project, dbSNP, or Exome Aggregation Consortium (ExAc). After collecting the point mutations and indels of each patient, we reconstructed their cancer evolutionary tree (Figure. 2) by using PhyloWGS [14].

2.3. Cancer Evolutionary Tree

Figure. 2(a) presents an example of cancer evolution in our study. Tumor evolves from a normal cell (clone A) to a cancerous cell. A founder cell (founder clone B) is established after a normal cell acquires several passenger mutations and driver mutations (founder somatic mutations). Subclones (C and D) evolve by acquiring subsequent somatic mutations. A root (A) and its adjacent node (B) represent the normal and founder cell, respectively. The following nodes indicate subclones (C and D), and edge lengths indicate the number of somatic mutations acquired in the subclones (Figure. 2(b)).

2.4. Tree Structure Mapping for the Cancer Evolutionary Tree

To compare cancer evolutionary trees, the constructed cancer evolutionary tree is mapped to the field of use in graph theory.

The cancer evolutionary tree contains the cancer evolutionary history, which denotes the evolution of tumors over time, and the number of nodes in the tree indicates the mutation accumulation. Here we focus on the tree structure. The structure of the cancer evolutionary tree only refers to the height of the tree, depth of the nodes, and relationship among nodes and does not refer to the node size, nodeinformation, and edge lengths. We mapped the cancer evolutionary tree structure to a directed rooted tree, which we referred to as the evolutionary structure tree (Figure. 3). For example, Figure. 3 is an evolutionary structure tree, which is mapped from the structure of the cancer evolutionary tree in Figure. 2(b), with node A as the root.

In Figure. 3, node A is the root node as well as the parent node of node B, node B is the child node of node A, and nodes C and D are siblings or leaf nodes.

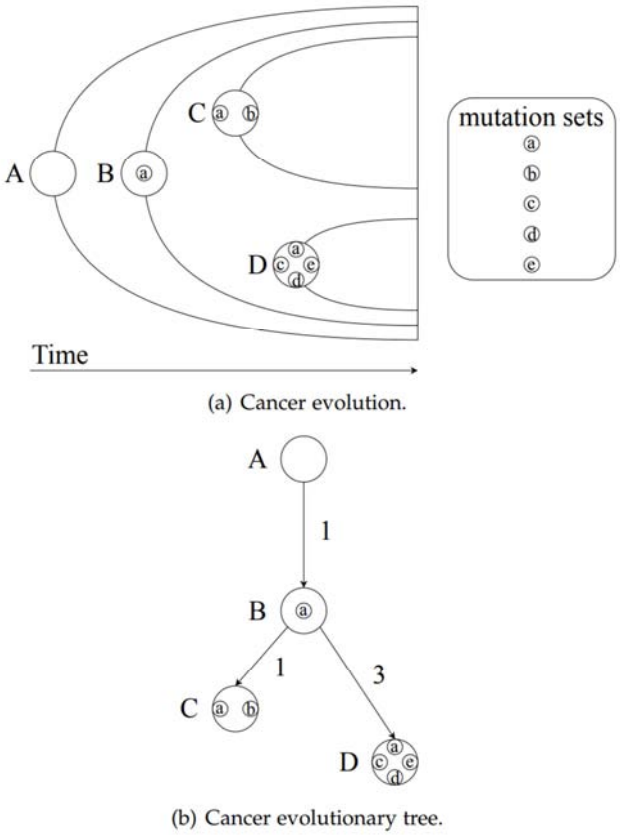


Figure 1. *Tumor initiation, tumor growth, and the corresponding cancer evolutionary tree. Characteristic driver mutations associated with cancer development and progression can be identified by reconstructing the evolutionary histories of cancerous cells. In Figure. 2(a), the evolutionary history of a tumor over time is obtained by replacing noncancerous cells (circle A or clone A) with one cancerous clone (circle B or founder clone B), which then develops into multiple cancerous subclones (circle C and D or subclones C and D). Tumor cells develop new clones by undergoing oncogenic mutations that have proliferative properties and allow the tumor descendants to expand relative to the other tumor clones. Each large circle (A, B, C, and D) refers to a clone. Each small circle (a, b, c, d, and e) refers to a mutation set providing selective advantages. Each clone inherits all of its parent's mutations. For example, subclones A, B, C, and D carry mutation set a.*

2.5. Evolutionary Structure Tree

An *evolutionary structure tree* (Figure. 4) is a three-tuple $T = (V, E, root(T))$, where V is a set of nodes including the root; E is a set of edges satisfying $E \in V \times V$, which denotes an orientation away from the root; and

$root(T)$ denotes the root of the tree T.

If $(u, v) \in E$, then a relationship is established such that node u points to node v. Therefore, v is referred to as a child node of node u, and u is referred to as a parent node of node v, respectively denoted as $v = child(u)$ and $u = parent(v)$.

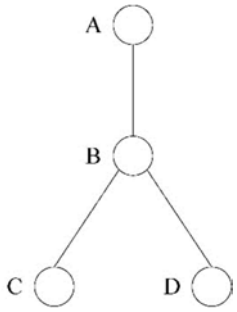


Figure 2. *Evolutionary structure tree obtained from the cancer evolutionary tree in Figure. 2(b). The cancer evolutionary tree contains information on cancer evolution and mutation accumulation.*

For each node v in tree T, the set of all children of node v is represented as $C(v)$, which also means that node v is linked to these children nodes; the set of all descendants of node v (including its direct children and indirect descendants) is represented as $D(v)$; the height of the subtree rooted at node v is represented as $h(v)$; and the number of nodes in the subtree rooted at node v is represented as n_v. We also define three attributes for each node $v = (h(v), n_v, C(v))$.

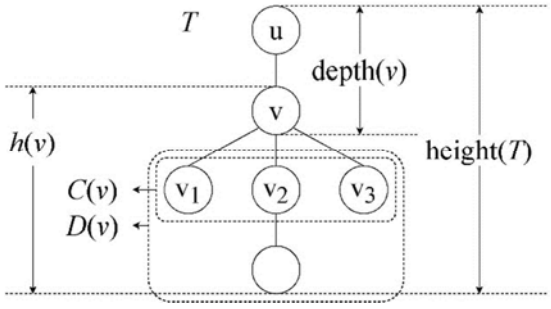


Figure 3. *An example of evolutionary structure tree.* $root(T)$ *is the node* u, $h(v) = 2$, $n_v = |D(v)| + 1 = 4 + 1 = 5$, $height(T) = 3$, *and* $depth(v) = 1$.

The *height* of tree T denoted as $height(T)$ is the number of edges between the tree's root node and its farthest leaf; the height of a tree with only root node is 0. The depth of node v denoted as $depth(v)$ refers to the number of edges from node v to the tree's root node, with the depth of the root node being 0.

The set of all children of node v is $\{v_1, v_2, \ldots, v_n\}$. A sequence of n child subtrees $\langle t_{v_1}, t_{v_2}, \ldots, t_{v_n} \rangle$ of node v in tree T and the n child subtrees are placed from left to right in the ascending order of the index.

2.6. Canonical-Form Transformation for Unique Tree Structure

Some evolutionary structure trees are similar in structureor isomorphic. We hope that these trees can have a unique tree structure. Isomorphic trees that do not have an identical structure will be clustered into different groups because of the differences, resulting in the subsequent clustering problem.

Two trees are considered isomorphic when each node in one tree has a corresponding node in the other tree and vice versa; that is, two evolutionary structure trees $T_1 = (V_1, E_1, root(T_1))$ and $T_2 = (V_2, E_2, root(T_2))$ have a bijection between the node sets $\varphi: V_1 \to V_2$ such that $\forall u, v \in V_1$, $(u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2$ and $\varphi(root(T_1)) = root(T_2)$. Obviously, both trees will have identical number of nodes.

We established a series of steps to solve the tree isomorphism problem and compare subtrees. The *canonical form* of an evolutionary structure tree can be obtained by performing a series of subtree exchanges, where the order of the siblings from left to right reflects the comparison results.

We also performed the canonical-form transformation for modularization to identify a unique canonical form of each evolutionary structure tree. The canonical-form transformation involves three steps by which layer-by-layer exchange operations are executed from root to leaf until all the nodes are searched. The layer-by-layer search helps avoid the inconsistency in the information of subtrees caused by an irregular search and exchange. In this process, the exchange operations are first executed between all sibling nodes in the identical layer, followed by a search of child nodes in the next layer until all leaf nodes have been it has searched. The output of the canonical-form transformation is a unique canonical form of an evolutionary structure tree.

2.6.1. Exchange Operation

The exchange operation involves selection of two subtrees, t_{v_1} and t_{v_2}, under node v in the rooted tree T, execution of the following three steps, and placement of the larger subtree on the right.

2.6.2. Steps of the Canonical-Form Transformation

Step 1. All subtrees under the nodes are placed from left to right according to height, such that the larger subtree is placed on the right (Figure. 5). Subtrees with identical heights are sorted using Step 2.

As shown in Figure. 5, in Step 1, the height of the left and right subtrees is compared first, irrespective of the number of nodes.

Step 2. When subtrees with identical height are encountered under the nodes, they are placed from left to right based on the number of nodes, such that the larger subtree is placed on the right (Figure. 6). Step 2 involves counting the number of nodes in each subtree. Therefore, this step can be time consuming when the size of the evolutionary structure tree is large. In addition, some subtrees may have an identical number of nodes. This problem can be solved using Step 3.

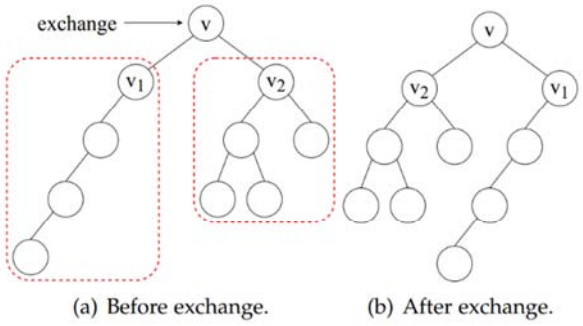


Figure 4. *Exchange of the child subtrees of the root in Step 1. Nodes v_1 and v_2 are children of node v, t_{v_1} is the subtree that has node v_1 as a root, and t_{v_2} is the subtree that has node v_2 as a root. Because the height of t_{v_1} is 3 and the height of t_{v_2} is 2, the larger subtree t_{v_1} is placed on the right.*

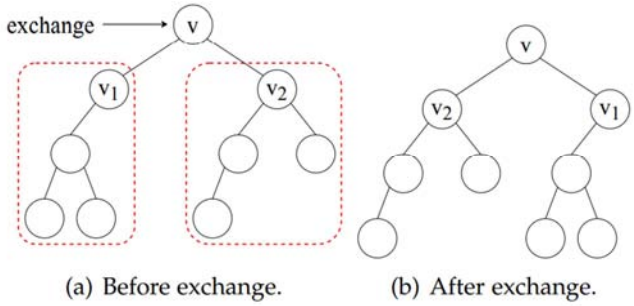


Figure 5. *Exchange of the child subtrees of the root in Step 2. Nodes v_1 and v_2 are children of node v, t_{v_1} is the subtree that has node v_1 as a root, and t_{v_2} is the subtree that has node v_2 as a root. Because the number of nodes in t_{v_1} is 3 and the number of nodes in t_{v_2} is 2, the larger subtree t_{v_1} is placed on the right.*

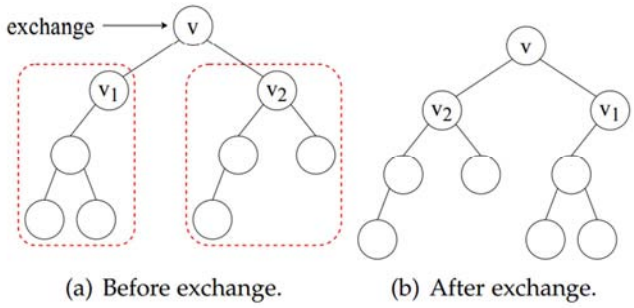


Figure 6. *Exchange of the child subtrees of the root in Step 3. Node v_1 and v_2 are children of node v, t_{v_1} is the subtree that has node v_1 as a root, and t_{v_2} is the subtree that has node v_2 as a root. After calculation of the number of nodes in the child subtrees t_{v_1} and t_{v_2}, the list for t_{v_1} is $[3]$ and that for t_{v_2} is $[2,1]$. Therefore, the larger subtree t_{v_1} is placed on the right.*

Step 3. When the subtrees under the nodes have identical heights and numbers of nodes, they are placed in lexicographical order from left to right (Figure. 7). The following is an example of the comparison method by lexicographical order. Two subtrees rooted at nodes u and v, respectively, have a sequence of m and n child subtrees $\langle t_{u_1}, t_{u_2}, \ldots, t_{u_m} \rangle$ and $\langle t_{v_1}, t_{v_2}, \ldots, t_{v_n} \rangle$. These two sequences are respectively used to calculate the number of nodes in the subtrees, which is stored as two lists $[n_{u_1}, n_{u_2}, \ldots, n_{u_m}]$ and $[n_{v_1}, n_{v_2}, \ldots, n_{v_n}]$. Two orderedlists $[n_{u_1}', n_{u_2}', \ldots, n_{u_m}']$ and $[n_{v_1}', n_{v_2}', \ldots, n_{v_n}']$ such that $n_{u_1}' \geq n_{u_2}' \geq \ldots \geq n_{u_m}'$ and $n_{v_1}' \geq n_{v_2}' \geq \ldots \geq n_{v_n}'$ are then used to compare the two items of order n_{u_i} and n_{v_i} according to the index i from 1 to m or n until the two items are no longer equal for comparison. Finally, the larger item determines that the subtree should be placed on the right.

Figure. 8-10 present a series of examples of the canonical-form transformation.

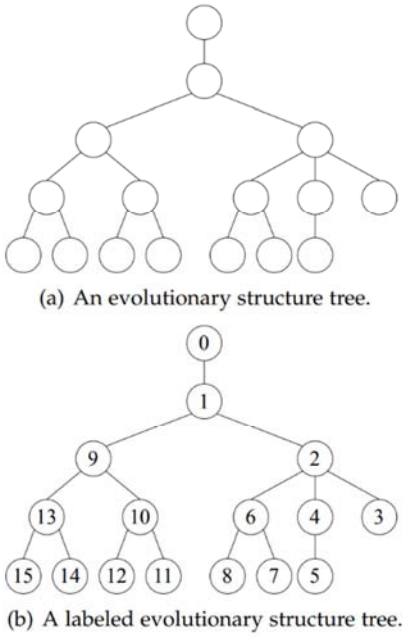


Figure 8. *Labeling of nodes helps to clearly indicate the exchange operations.*

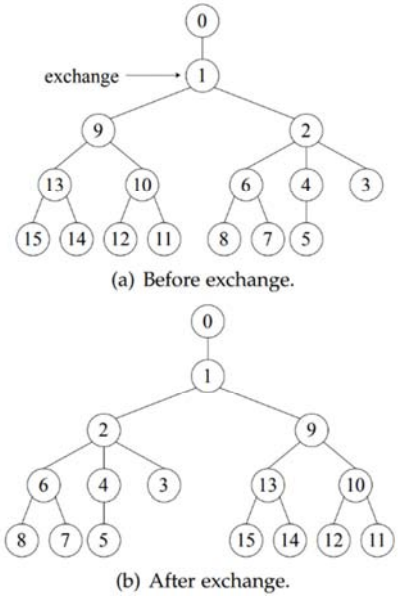


Figure 9. *Exchange of the child subtrees of node 1 in Step 3. Because the list for the left subtree is $[3,3]$ and that for the right subtree is $[3,2,1]$, the subtree order is exchanged.*

2.6.3. Canonical-Form Transformation Algorithm

The tree structures of two trees can be compared using the three steps described in Section 2.6.2 in the sequential order.

We assume that the structure of tree T_1 is greater than that of tree T_2 $(T_1 > T_2)$, the structure of tree T_1 is smaller than that of tree T_2 $(T_1 < T_2)$, and the structure of tree T_1 is equal to that of tree T_2 $(T_1 == T_2)$.

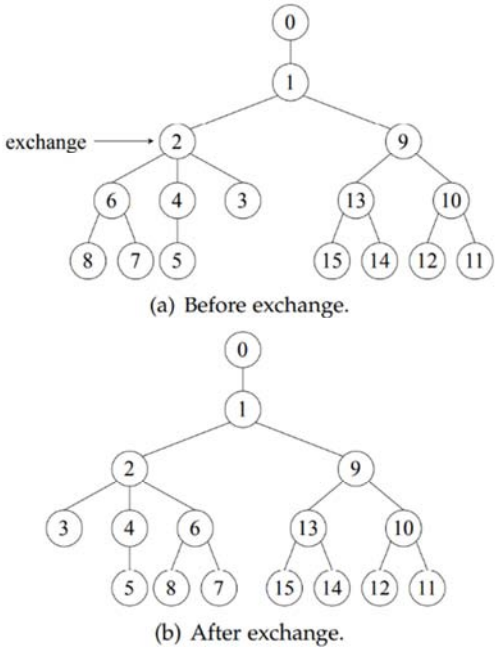


Figure 10. *Exchange the child subtrees of node 2 in Step 1. Because the height is 1, 1, 0 from left to right, the subtree order is exchanged.*

Algorithm 1 Canonical-Form(T, v_r)

Input: tree: T, the root node of T: v_r
1: Let C be Link-Child(T, v_r)
2: Let $T(C)$ be the subtree in the tree under the node C
3: **if** C is not empty **then**
4: **for each** c_1 in C **do** ▷ compare each node in the identical layer
5: Canonical-Form$(T(c_1))$
6: **for each** c_2 in C **do**
7: **if** $c_1 == c_2$ **then**
8: **continue**
9: **end if**
10: **if** HeightNode-Comparison$(T(c1), T(c2)) ==$ " $>$ " **then** ▷ exchange if comparison is not equal
11: exchange $T(c_1) \longleftrightarrow T(c_2)$
12: **else if** HeightNode-Comparison$(T(c1), T(c2)) ==$ " $==$ " **then** ▷ to compare by lexicographical order
13: **if** LGO-Comparison$(T(c_1), T(c_2)) ==$ " $>$ " **then** ▷ exchange if comparison is not equal
14: exchange $T(c_1) \longleftrightarrow T(c_2)$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: **end if**

Figure 11. *Algorithm 1 Canonical-Form.*

Some basic functions of the algorithm are as follows.

1) Append(): appends an element to the end of the list
2) TreeHeight(): obtains the height of the tree
3) TreeNodeCount(): obtains the number of nodes in the tree
4) Length(): counts the number of characters in a list

The Canonical-Form algorithm (Algorithm 1) can be used to achieve a unique canonical form of the tree.

The Link-Child algorithm (Algorithm 2) provides all the child nodes of the node (not including indirect offspring) and an iterative search idea to the Canonical-Form algorithm to enable it to search nodes layer by layer from root to leaf until all the nodes are searched.

Algorithm 2 Link-Child(T, v)

Input: tree: T, node: v
Output: list: $childList$
1: Let $childList$ be empty list
2: **for each** n in T **do**
3: **if** v is the parent node of node n **then**
4: $childList$.Append(n) ▷ append the node n to the list
5: **end if**
6: **end for**
7: **return** $childList$

Figure 12. *Algorithm 2 Link-Child.*

The HeightNode-Comparison algorithm (Algorithm 3) is used for comparing the tree structure of two subtrees. This algorithm is only run in Steps 1 and 2.

Algorithm 3 HeightNode-Comparison$(T1, T2)$

Input: tree: $T1$, tree: $T2$
Output: a string representing the size relationship: {" $>$ ", " $<$ ", " $==$ "}
1: **if** TreeHeight$(T1) >$ TreeHeight$(T2)$ **then** ▷ compare the height
2: **return** " $>$ "
3: **else if** TreeHeight$(T1) <$ TreeHeight$(T2)$ **then**
4: **return** " $<$ "
5: **else**
6: **if** TreeNodeCount$(T1) >$ TreeNodeCount$(T2)$ **then** ▷ compare the number of nodes
7: **return** " $>$ "
8: **else if** TreeNodeCount$(T1) <$ TreeNodeCount$(T2)$ **then**
9: **return** " $<$ "
10: **else**
11: **return** " $==$ "
12: **end if**
13: **end if**

Figure 13. *Algorithm 3 HeightNode-Comparison.*

The LGO-Comparison algorithm (Algorithm 4) is used when two subtrees have identical heights and numbers of nodes. The comparison is performed in lexicographical order (Step 3).

The LGO algorithm (Algorithm 5) is used by the Trees-Sort algorithm to output a list for all subtrees in

lexicographical order sorted from large to small.

Because all child subtrees of each node are placed from left to right, we propose the Trees-Sort algorithm. The Trees-Sort algorithm (Algorithm 6) compares the subtree structures from small to large, and the output of the Trees-Sort algorithm is used to sort and order the subtrees.

The Compare-Merge algorithm (Algorithm 7) used merge sort to sort the subtrees for comparison in the Trees-Sort algorithm.

Algorithm 4 LGO-Comparison($T1, T2$)

Input: tree: $T1$, tree: $T2$
Output: a string representing the size relationship: {" < ", " > ", " == "}
Let S_1 be LGO($T1$)
Let S_2 be LGO($T2$)
if Length(S_1) ≤ Length($S2$) **then** ▷ determine the length of the list we want to compare
  Let L be Length(S_1)
else
  Let L be Length(S_2)
end if
for $i \leftarrow 1$ to L **do**
  if HeightNode-Comparison($S_1[i], S_2[i]$) is not " == " **then** ▷ two subtrees can get the results about tree similarity comparison
    return HeightNode-Comparison($S_1[i], S_2[i]$)
  end if
end for
return " == "

Figure 14. *Algorithm 4 LGO-Comparison.*

Algorithm 5 LGO(T)

Input: tree: T
Output: list of subtrees: S'
Let S be subtrees $\{T_1', T_2', ..., T_n'\}$ of T
Let S' be empty list
$S' \leftarrow$ Trees-Sort($T, 1, n$) ▷ sort subtrees in list
$S' \leftarrow S'$.Reverse() ▷ reverse the list
return S'

Figure 15. *Algorithm 5 LGO.*

Algorithm 6 Trees-Sort(T, p, r)

Input: A sequence of n subtrees $\langle t_1, t_2, ..., t_n \rangle$ of the tree T, indices p, r such that $p \leq r$
Output: A permutation $\langle t_p', t_{p+1}', ..., t_r' \rangle$ of the input sequence such that $t_p' \leq t_{p+1}' \leq ... \leq t_r'$
if $p < r$ **then**
  $q \leftarrow \lfloor (p + r)/2 \rfloor$ ▷ q is the index representing the median of the sequence
  Trees-Sort(T, p, q) ▷ divide the first half of the sequence
  Trees-Sort($T, q + 1, r$) ▷ divide the second half of the sequence
  Compare-Merge(T, p, q, r) ▷ merge the sorted sequence
end if

Figure 16. *Algorithm 6 Trees-Sort.*

2.7. Tree Similarity Comparison and Cost of Tree Similarity Comparison

Here we present the three operations involved in the tree similarity comparison method and the calculation method for comparing the differences in tree structure costs [32].

2.7.1. Operations in Tree Similarity Comparison

According to the previous canonical-form transformation, each evolutionary structure tree has a unique structure. The degree of structural similarity between two evolutionary structure trees can be compared by performing three operations, following which a cost is determined, which is considered the difference in similarity between the two evolutionary structure trees. The cost is a real number.

Algorithm 7 Compare-Merge(T, p, q, r)

Input: A sequence of n subtrees of the tree T, indices p, q, r such that $p \leq q \leq r$ and subsequence $T[p, ..., q]$ is sorted and subsequence $T[q + 1, ..., r]$ is sorted
Output: The two subsequences are merged into a single sorted subsequence in $T[p, ..., r]$
$n_1 \leftarrow q - p + 1$ ▷ give the length of the first half of the sequence
$n_2 \leftarrow r - q$ ▷ give the length of the second half of the sequence
Create sequences $L[1, ..., n_1]$ and $R[1, ..., n_2]$
for $i \leftarrow 1$ to n_1 **do** ▷ move the sequence we want to compare
  $L[i] \leftarrow T[p + i - 1]$
end for
for $j \leftarrow 1$ to $n2$ **do** ▷ move the sequence we want to compare
  $R[j] \leftarrow T[q + j]$
end for
$i \leftarrow 1$
$j \leftarrow 1$
for $k \leftarrow p$ to r **do**
  if HeightNode-Comparison($L[i], R[j]$) == " > " **then** ▷ compare the subtrees in the sequence
    $T[k] \leftarrow R[j]$
    $j \leftarrow j + 1$
  else
    $T[k] \leftarrow L[i]$
    $i \leftarrow i + 1$
  end if
end for

Figure 17. *Algorithm 7 Compare-Merge.*

The main idea of tree similarity comparison is transforming one tree T into another tree T' for comparison by performing the three operations.

Operation 1. Deleting node v (denoted as $delete(v)$). In this operation, the position of the deleted node v is identified in the original tree and then the position of the parent node of the original node v is changed to point to the new child nodes.

If $v \neq root(T)$, then $V' = V - \{v\}$, $E' = E - \{(v, w) | w \in C(v)\} - \{(parent(v), v)\} + \{(parent(v), w) | w \in C(v)\}$.

In Figure 18, node v is the node to be deleted. Therefore, node v is first deleted, and then, the position of node a (the parent node of the original node v), which originally pointed to node v, is changed to point to nodes b, c, and d.

Operation 2. Inserting node v under node u (denoted as $insertu(v)$).

Before performing this operation, the following actions must be performed: compare whether the children of node u in the original tree T_1 intersect with the children of node v in tree T_2. If an intersection exists, node v points directly to this set. This method can help determine the relationship of the node to other nodes and reduce redundant operations, thereby reducing costs (the insertion operation without and with the aforementioned method is presented in Figure 19 and 20, respectively).

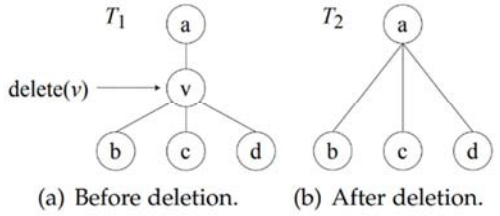


Figure 18. *Deletion operation.*

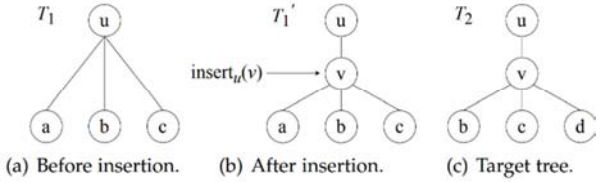


Figure 19. *Insertion operation.*

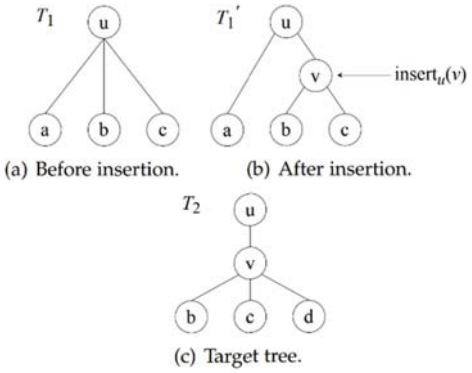


Figure. 20. *Insertion operation (cost reduction).*

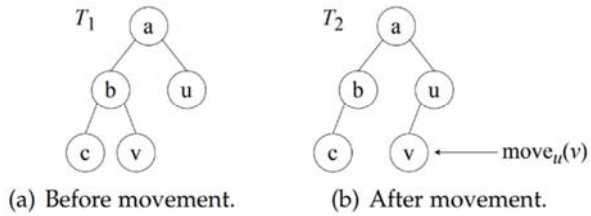


Figure 21. *Moving operation.*

We have $V' = V + \{v\}$, $E' = E - \{(u, w)|w \in D(u) \cap D'(v)\} + \{(u, v)\} + \{(v, w)|w \in D(u) \cap D'(v)\}$.

In Figure 19, the parent node of nodes a, b, and c is directly changed. However, in the method shown in Figure 20, tree T_2 is compared with tree T_1 to verify whether an intersection exists. It is found that node a is only present in tree T_1 and not in tree T_2, indicating no intersection. Therefore, during the insertion operation, the state of node a need not be changed. This method helps reduce the cost.

Operation 3. Moving node v to be positioned under node u (denoted as $move_u(v)$).

To move the position of a node, the node is first deleted and then inserted at the required position. Thus, both delete and insert operations are used (Figure 21).

Moreover, details such as the position of nodes in the tree and the relationship of the node with other nodes should be considered in the operations.

2.7.2. Cost of Tree Similarity Comparison

A similarity comparison between two trees, T_1 and T_2, yields a cost, which is a real number, after every operation, Op. The cost of each operation is denoted as $K(Op)$.

If $OP = \{Op_1, Op_2, \ldots, Op_n\}$ is an operation sequence, then the cost of a sequence of operations can be represented as $K(OP)$.

If OP is an operation sequence used for transforming tree T_1 into tree T_2, then the cost from T_1 to T_2 is defined as $K(T_1 \rightarrow T_2)$. Moreover, the cost of tree similarity comparison between two trees, T_1 and T_2, can be defined as $K(T_1, T_2) = \min\{K(T_1 \rightarrow T_2), K(T_2 \rightarrow T_1)\}$.

2.7.3. Computing the Cost of Tree Similarity Comparison

The cost of tree similarity comparison is influenced by the position of the node, size of the tree (or the number of nodes), height of the tree, depth of the node, number of descendants of the node (the size of the subtree having the node as a root), and relationship of the node to other nodes. These factors should be considered when performing tree similarity comparison between two trees. For example, first, an operation performed on a node at a shallow depth will incur a higher cost than that performed on a node at a deeper region; moreover, the cost will be lower when the node is near the bottom of the tree. Second, a node that has more descendants will incur a higher cost than a node with fewer descendants. In general, the node with more descendants will be located at a shallow depth, with some exceptions. The tree structure should be evaluated before conducting the operation. Third, performing operations on a larger tree will incur a lower cost than performing operations on a smaller tree. Finally, two trees may differ only in the position of some nodes, and the operation of moving the node will incur a lower cost than the operation of deleting or inserting the node.

The cost of the three operations can be calculated using the following formulas.

1) Cost of deletion operation (Figure 22)

$$K(delete) = height(T) - depth(v) + 1 + |D(v)|/|V| \quad (1)$$

where v is a nonroot node, $height(T)$ is the height of tree T,

$depth(v)$ is the depth of node v, $|D(v)|$ is the number of descendants of node v (direct and indirect offspring), and V is the original node set before the deletion. In general, $depth(root(T)) = 1$, and $depth(v) > 1$ if v is a nonroot node. If node v to be deleted is a leaf node, $D(v) = \emptyset$ and $|D(v)| = 0$. This means that no extra cost is incurred in the deletion of a leaf node. When v is the lowest leaf node in the tree $(height(T) = depth(v))$, deleting v will incur a minimal where v is a nonroot node, $height(T)$ is the height of tree T, $depth(v)$ is the depth of node v, $|D(v)|$ is the number of descendants of node v (direct and indirect offspring), and V is the original node set before the deletion. In general, $depth(root(T)) = 1$, and $depth(v) > 1$ if v is a nonroot node. If node v to be deleted is a leaf node, $D(v) = \emptyset$ and $|D(v)| = 0$. This means that no extra cost is incurred in the deletion of a leaf node. When v is the lowest leaf node in the tree $(height(T) = depth(v))$, deleting v will incur a minimal cost. The final cost is $K(delete(v)) = 1/|V|$.

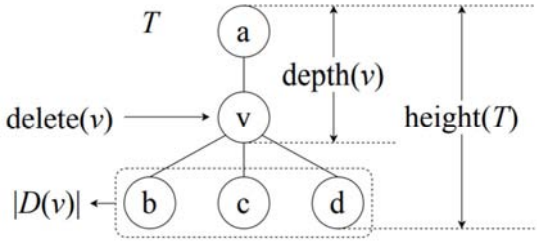


Figure 22. *An example of cost of deletion operation:* $height(T) = 2$, $depth(v) = 1$, $|D(v)| = 3$, $|V| = 5$. *Therefore, the deletion cost is given by* $(2 - 1 + 1 + 3)/5 = 1$.

2) Cost of insertion operation (Figure 23)

$$K(insert_u(v)) = height(T) - depth(u) + 1 + |D(v)|/|V| \quad (2)$$

where $height(T)$ is the height of tree T, $depth(u)$ is the depth of node u, $|D(v)|$ is the number of descendants that v obtains after it is inserted (direct and indirect offspring) and V is the original node set before the insertion. The insertion operation differs from the deletion operation in some aspect. Node v cannot be inserted at the root position because the root cannot be changed arbitrarily. Therefore, the node can only be inserted under the root. The calculation of the depth of inserting node v also differs from that in the deletion operation. Because before doing the operation, we only know to insert the node v under the node u, so the $depth(u)$ will be used.

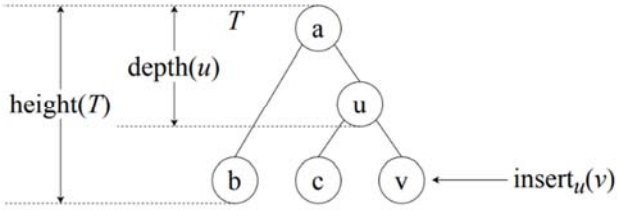


Figure 23. *An example of cost of insertion:* $height(T) = 2$, $depth(u) = 1$, $|D(v)| = 0$, $|V| = 4$ *(before inserting node v, the size of the tree T is 4). Therefore, the insertion cost is given by* $(2 - 1 + 1 + 0)/4 = 0.5$.

3) Cost of moving operation (Figure 24)

$$K(move_u(v)) = 1/2[K(delete(v)) + K(insert_u(v))] \times (|V| - 2)/|V| \quad (3)$$

where $|V| > 2$ (the tree has a root and at least two nonroot nodes) and $u \neq parent(v)$. Note that $insert_u(v)$ will be performed on a tree without node v. In this formula, both the deletion and insertion operations are considered because the operation of moving involves deleting the node first and then inserting it. Some factors are used for balancing this operation. The factor 1/2 adjusts the cost of operation because the node is not actually deleted from and inserted back to the tree. Another factor $(|V| - 2)/|V|$ adjusts the cost to ensure that in an extreme case where v is the only node other than the root, its moving cost is zero because it will obtain the identical result after doing the operation of moving the node and actually it can't be moved. Moreover, as the number of nodes in the tree increases, the effect of the factor $(|V| - 2)/|V|$ on the operation of moving the node becomes weaker.

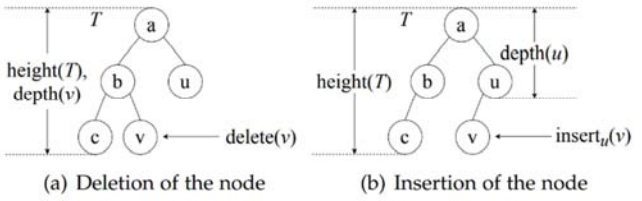


Figure 24. *An example for cost incurred while moving a node by first deleting it (*$height(T) = 2$, $depth(v) = 2$, $|D(v)| = 0$, $|V| = 5$*; therefore, the deleting cost becomes* $(2 - 2 + 1 + 0)/5 = 0.2$*) and then inserting it (* $height(T) = 2$, $depth(u) = 1$, $|D(v)| = 0$, $|V| = 4$ *(before insertion of node v, the size of the tree T is 4); therefore, the inserting cost becomes* $(2 - 1 + 1 + 0)/4 = 0.5$*). The final moving cost is given by* $(1/2) \times (0.2 + 0.5) \times (5 - 2)/5 = 0.21$.

Figure 25 presents an example of calculating the cost of tree similarity comparison between two trees.

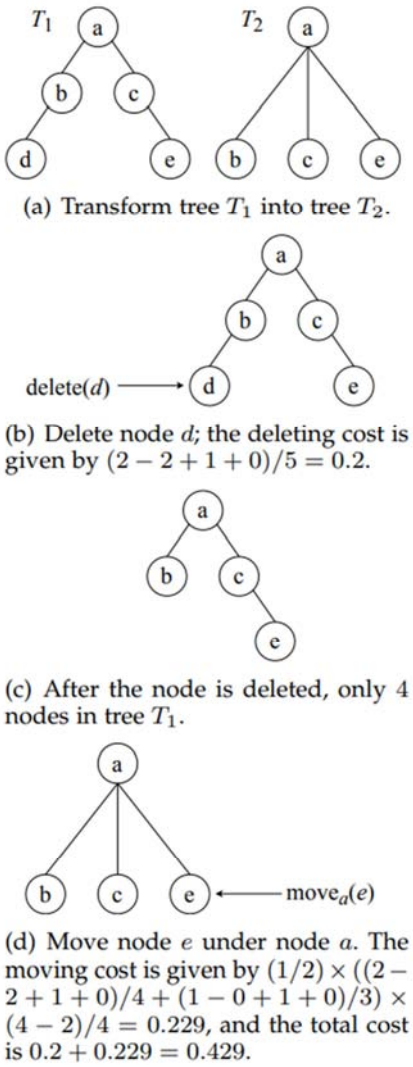
2.8. Patient Clustering

We used tree similarity comparison (mentioned in Section 2.7) to compare the evolutionary structure trees to obtain 107 × 107 matrix data. Because patients exhibit two cancer types, nonrecurrent and recurrent, we divided the obtained matrix data into two clusters through *k*-means clustering. We also used the average silhouette method to measure the quality of the clusters (Figure 26) [33]. The average silhouette method combines two factors, cohesion and separation. It can be used to evaluate the impact of different algorithms or different operating modes of algorithms on the clustering results based on the same original data. The optimal number of clusters is two (Figure 26). Now, each patient obtains new information, which is an attribute classified as cluster 1 or cluster 2.

2.9. Cancer Recurrence Rate Prediction

We collected clinical data of each patient, including age, gender, tumor site (site), tumor staging (stage), tumor invasion stage (T), and tumor nodal stage (N). Age comprises two attributes, Young and Old, where Young refers to the age

group 0-65 years and Old refers to the age group 66 years and above. Gender comprises two attributes, Female and Male. Site comprises four attributes, Left, Right, Right and Left, and Unknown. Stage comprises two attributes, 2 and 3, where 2 indicates cancer stage 1 and cancer stage 2, and 3 indicates cancer stage 3 and above. The feature of T comprises two attributes, Early and Late, where Early means T1 and T2 and Late means T3 and T4 (in TNM staging system). The feature of N comprises two attributes, 1 and 2, where 1 means N0 and N1 and 2 means N2 and N3 (in TNM staging system). We demonstrated the importance of tree similarity comparison in recurrent prediction. We established two groups, a control group and a test group, with one more clustering information, for final prediction and comparison. We used these two groups as the input to the Cancer Recurrence Rate Prediction (as shown in Figure 27.) and assessed whether the prediction of the recurrence rate improved on addition of one clustering data set (presented in Section 3). We then combined the clinical data of each patient and their respective clustering data. Figure 28 presents the combined distribution of all patient data.



(a) Transform tree T_1 into tree T_2.

(b) Delete node d; the deleting cost is given by $(2-2+1+0)/5 = 0.2$.

(c) After the node is deleted, only 4 nodes in tree T_1.

(d) Move node e under node a. The moving cost is given by $(1/2) \times ((2-2+1+0)/4 + (1-0+1+0)/3) \times (4-2)/4 = 0.229$, and the total cost is $0.2 + 0.229 = 0.429$.

Figure 25. *An example of tree similarity comparison between two trees, T_1 and T_2, and the cost of transforming tree T_1 into tree T_2.*

In the Cancer Recurrence Rate Prediction, we first divided the data into two types: training set and testing set. Of the data for 107 patients, data of 77 patients were included in the training set (80%) and those of 30 patients were included in the testing set (20%). Now, the training set contained 62 nonrecurrent types and 15 recurrent types, and the testing set contained 15 nonrecurrent types and 15 recurrent types. To ensure the training set is balanced (1:1), we performed data augmentation. We used Synthetic Minority Over-sampling Technique (SMOTE) to generate new samples [34]. The basic principle of SMOTE is to analyze for minority samples and artificially synthesize new samples based on the minority samples and add them to the data set; thus, the recurrent type in the training set was increased to 62. Then, we used random forest to select important features and the first five features (Figure 29.) [35]. Finally, we use random forest, support vector machine (SVM), bagging [36], and boosting [37] to train the model and used the testing set to predict the results.

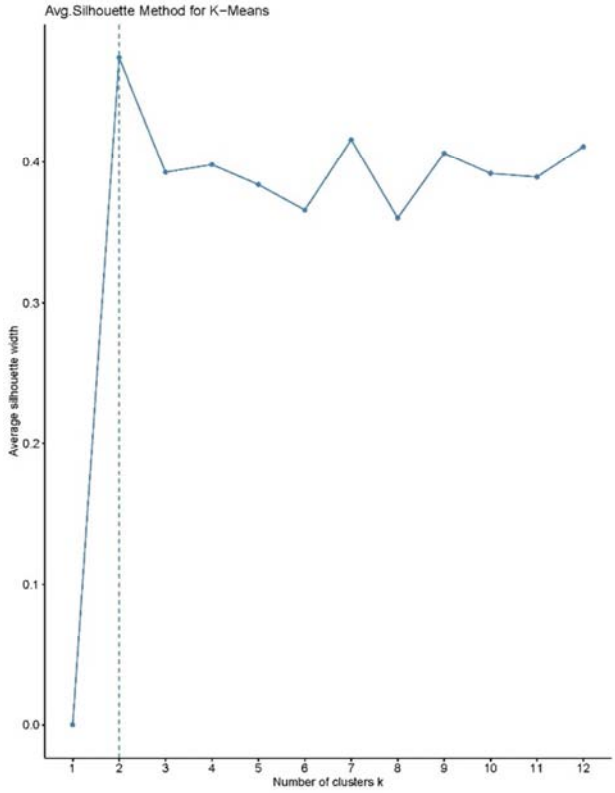


Figure 26. *Measurement of the quality of the cluster and determination of the optimal number of clusters.*

3. Experimental Results and Discussion

In Figure 28, the left half presents the heatmap distribution of all 107 patients. This denotes the score distribution obtained by comparing tree structures with each other. The upper left corner is cluster 1, and the other is cluster 2. During comparison with other groups based on cluster analysis, a set of objects is clustered into the group with certain defined identical properties. We compared the differences in the tree structures using *k*-means clustering.

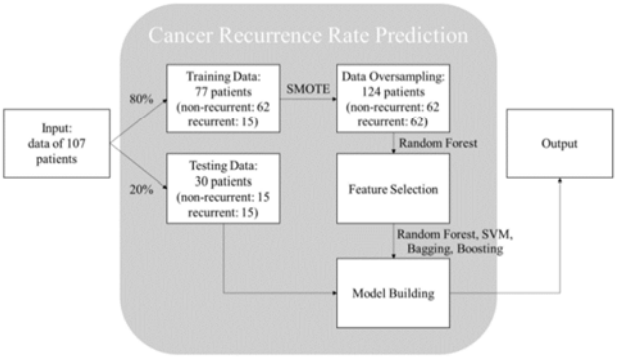


Figure 27. *Cancer Recurrence Rate Prediction.*

Therefore, when the evolutionary structure trees in one group are compared with those in another group, the color distribution is significantly different. This means that the evolutionary structure trees that are not in the identical group do not have close scores, so the evolutionary structure trees in these two groups exhibit huge differences in the tree structure, such as tree height and number of nodes. In the right half of Figure 28, the clinical data are presented by a clustered sorting. This means that the clinical data are distributed according to the clustering data, which makes it convenient to observe the status distribution of each group. These clinical data are recurrent status, survival status, age, gender, tumor invasion stage (T), tumor nodal stage (N), tumor site (site), and tumor staging (stage).

Table 1. *Relationship between clustering information and clinical data.*

Clinical Data	No. of the Patients in the Cluster		*p*-value
	Cluster 1	Cluster	
Age			1
Young (0 ~ 65)	53	28	
Old (66 up)	17	9	
Gender			0.02449
Female	41	13	
Male	29	24	
Site			0.3888
Left	55	26	
Right	14	10	
Right and Left	1	0	
Unknown	0	1	
Stage			0.4493
2 (Cancer stage 1, 2)	4	4	
3 (Cancer stage 3 up)	66	33	
T (Primary Tumor)			0.02749
Early (T1, T2)	13	1	
Late (T3, T4)	57	36	
N (Regional Lymph Nodes)			1
1 (N0, N1)	50	26	
2 (N2, N3)	20	11	

We also used the clustering information and clinical data for analysis. TABLE 1 indicates that the structure of the cancer evolutionary tree was associated with gender and tumor invasion stage. Early-stage (T1 and T2) patients tend to have a simpler structure of the cancer evolutionary tree than late-stage (T3 and T4) patients. For early-stage patients, the height of the cancer evolutionary tree is relatively small, the depth of the node in the tree is relatively low, and the number of nodes in the tree is relatively few. Cluster 2 patients tend to have a more complicated structure of the cancer evolutionary tree than cluster 1 patients. For cluster 2 patients, the height of the cancer evolutionary tree is relatively large, the depth of the node in the tree is relatively large, and the number of nodes in the tree is relatively high. Among our 107 CRC patients, the ratio of male to female is approximately 1:1. CRC with obvious gender differences affects males more than females, and males not only develop cancer more often but are also more likely to die from their disease [38-40]. Male patients tend to have a more complicated structure of the cancer evolutionary tree than female patients.

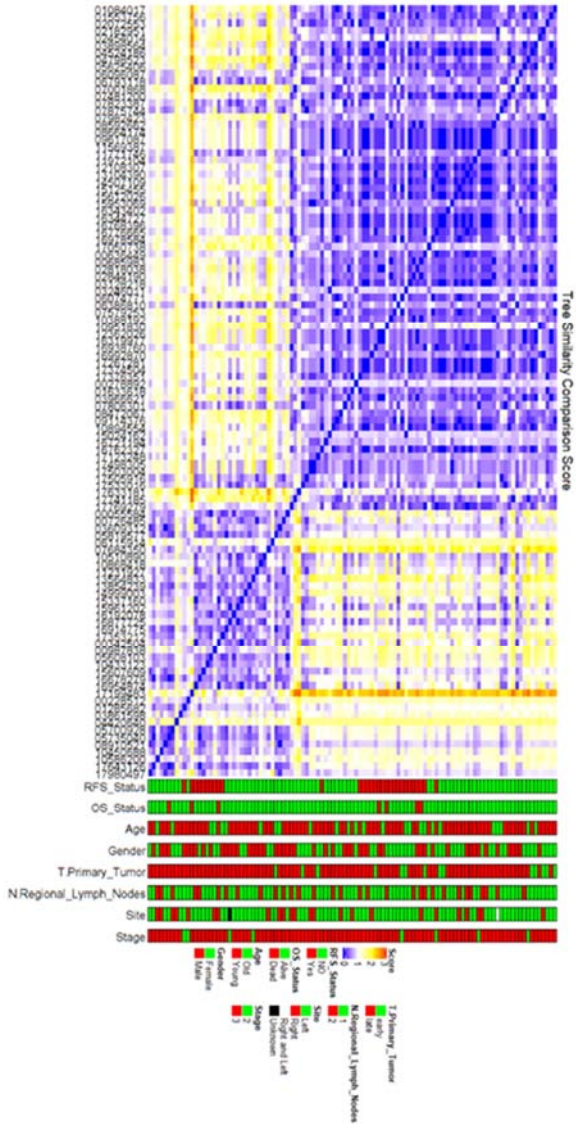


Figure 28. *Heatmap. According to the clustering information, 107 patients were divided into two clusters. The left half presents the score distribution obtained by comparing the patients' tree structures. The right half presents the clinical data distribution when patients were divided into two clusters.*

We divided the patients into two clusters according to the clustering information. Figure 30 presents the distribution of the number of SNVs between cluster 1 and cluster 2. The mean of cluster 1 is 162, and the mean of cluster 2 is 135.

Finally, we used random forest, SVM, bagging, and boosting to predict the recurrence of these CRC patients. The optimal accuracy of 0.667 was obtained using the boosting model with one more clustering information (as shown in TABLE 2).

Table 2. *Improvements in recurrence rate prediction.*

Method	Recurrence Rate Prediction		Change in Prediction Rate (C - O)
	Clinical Data (O)	Clinical Data + Cluster (C)	
Random Forest	0.533	0.567	0.034
SVM	0.5	0.5	0
Bagging	0.533	0.533	0
Boosting	0.6	0.667	0.067

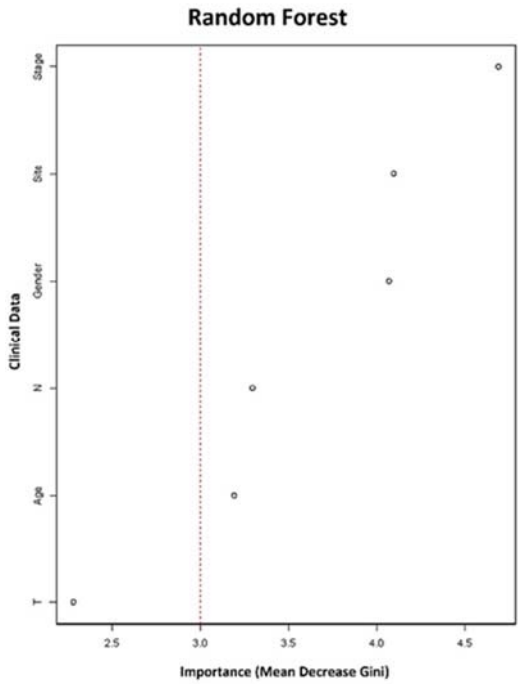


Figure 29. *Variable importance and variable selection*

4. Conclusion and Future Work

In this study, we combined genetic data with clinical data and integrated them into cancer evolutionary trees for cancer recurrence prediction. Through tree similarity comparison, we obtained clustering information of all patients. Our results revealed that integrating the clustering information can improve the performance compared with using only clinical information. An accuracy rate of 0.667 was obtained for our model. If patients with new diagnosis of CRC receive therapy in NCKUH in the future, our model can be used for prediction according to their reconstructed cancer evolutionary tree and their clinical data. Finally, we can improve the model prediction accuracy based on whether recurrence is noted in these CRC patients in the future.

Although the results indicate that the prediction of the cancer recurrence rate can be improved, many areas for improvement remain. In our study, we only collected data of 107 patients, and the data types were unbalanced. Consequently, the amount of data as input for the machine learning model was insufficient. In the future, the amount of patient data and other information in the clinical data should be increased to help train machine-learning models. Another research direction is to apply the developed models for prediction in patients with other cancer types.

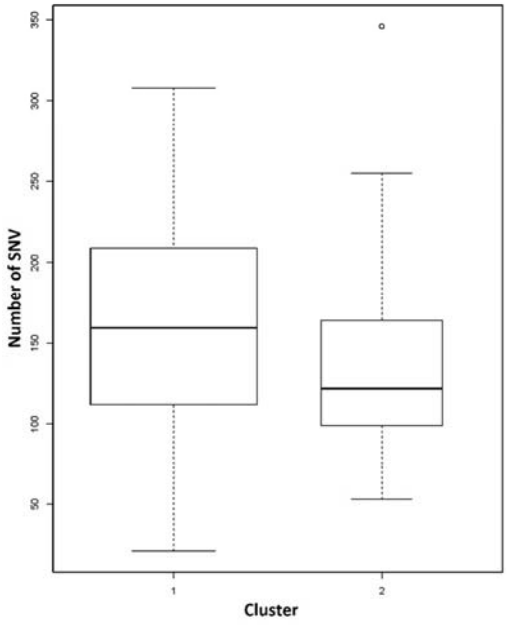


Figure 30. *Difference between cluster 1 and cluster 2.*

Acknowledgements

We thank Chien-Chang Pan for assistance with bioinformatics of the study and comments that greatly improved the manuscript.

References

[1] P. Nowell, “The clonal evolution of tumor cell populations,” *Science*, vol. 194, no. 4260, pp. 23–28, Jan. 1976.

[2] C. Swanton, “Cancer evolution: the final frontier of precision medicine?,” *Annals of Oncology*, vol. 25, no. 3, pp. 549–551, 2014.

[3] R. Schwartz and A. A. Schaffer, “The evolution of tumour phylogenetics: principles and practice,” *Nature Reviews Genetics*, vol. 18, no. 4, pp. 213–229, 2017.

[4] P. J. Campbell, E. D. Pleasance, P. J. Stephens, E. Dicks, R. Rance, I. Goodhead, G. A. Follows, A. R. Green, P. A. Futreal, and M. R. Stratton, “Subclonal phylogenetic structures in cancer revealed by ultra-deep sequencing,” *Proceedings of the National Academy of Sciences*, vol. 105, no. 35, pp. 13081–13086, 2008.

[5] A. Schuh, J. Becq, S. Humphray, A. Alexa, A. Burns, R. Clifford, S. M. Feller, R. Grocock, S. Henderson, I. Khrebtukova, Z. Kingsbury, S. Luo, D. Mcbride, L. Murray, T. Menju, A. Timbs, M. Ross, J. Taylor, and D. Bentley, "Monitoring chronic lymphocytic leukemia progression by whole genome sequencing reveals heterogeneous clonal evolution patterns," *Blood*, vol. 120, no. 20, pp. 4191–4196, 2012.

[6] M. El-Kebir, L. Oesper, H. Acheson-Field, and B. J. Raphael, "Reconstruction of clonal trees and tumor composition from multi-sample sequencing data," *Bioinformatics*, vol. 31, no. 12, pp. i62–i70, 2015.

[7] S. Malikic, A. W. Mcpherson, N. Donmez, and C. S. Sahinalp, "Clonality inference in multiple tumor samples using phylogeny," *Bioinformatics*, vol. 31, no. 9, pp. 1349–1356, 2015.

[8] W. Jiao, S. Vembu, A. G. Deshwar, L. Stein, and Q. Morris, "Inferring clonal evolution of tumors from single nucleotide somatic mutations," *BMC Bioinformatics*, vol. 15, no. 1, p. 35, 2014.

[9] K. Jahn, J. Kuipers, and N. Beerenwinkel, "Tree inference for singlecell data," 2016.

[10] M. El-Kebir, "SPhyR: tumor phylogeny estimation from single-cell sequencing data under loss and error," *Bioinformatics*, vol. 34, no. 17, pp. i671–i679, 2018.

[11] S. Miura, L. A. Huuki, T. Buturla, T. Vu, K. Gomez, and S. Kumar, "Computational enhancement of single-cell sequences for inferring tumor evolution," *Bioinformatics*, vol. 34, no. 17, pp. i917–i926, 2018.

[12] E. Letouze, Y. Allory, M. A. Bollet, F. Radvanyi, and F. Guyon, "Analysis of the copy number profiles of several tumor samples from the same patient reveals the successive steps in tumorigenesis," *Genome Biology*, vol. 11, no. Suppl 1, 2010.

[13] H. Zare, J. Wang, A. Hu, K. Weber, J. Smith, D. Nickerson, C. Song, D. Witten, C. A. Blau, and W. S. Noble, "Inferring Clonal Composition from Multiple Sections of a Breast Cancer," *PLoS Computational Biology*, vol. 10, no. 7, 2014.

[14] A. G. Deshwar, S. Vembu, C. K. Yung, G. H. Jang, L. Stein, and Q. Morris, "PhyloWGS: Reconstructing subclonal composition and evolution from whole-genome sequencing of tumors," *Genome Biology*, vol. 16, no. 1, 2015.

[15] J. S. Farris, "Methods for Computing Wagner Trees," *Systematic Biology*, vol. 19, no. 1, pp. 83–92, 1970.

[16] W. M. Fitch, "Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology," *Systematic Zoology*, vol. 20, no. 4, p. 406, 1971.

[17] D. Penny, "Inferring Phylogenies.—Joseph Felsenstein. 2003. Sinauer Associates, Sunderland, Massachusetts.," *Systematic Biology*, vol. 53, no. 4, pp. 669–670, 2004.

[18] T. Stijnen, "Maximum Likelihood Estimation Methods," *Encyclopedia of Medical Decision Making*.

[19] N. T. Hobbs and M. B. Hooten, "Markov Chain Monte Carlo," *Bayesian Models*, 2015.

[20] I. Hajirasouliha, A. Mahmoody, and B. J. Raphael, "A combinatorial approach for analyzing intra-tumor heterogeneity from highthroughput sequencing data," *Bioinformatics*, vol. 30, no. 12, pp. i78–i86, 2014.

[21] C. A. Miller, B. S. White, N. D. Dees, M. Griffith, J. S. Welch, O. L. Griffith, R. Vij, M. H. Tomasson, T. A. Graubert, M. J. Walter, M. J. Ellis, W. Schierding, J. F. Dipersio, T. J. Ley, E. R. Mardis, R. K. Wilson, and L. Ding, "SciClone: Inferring Clonal Architecture and Tracking the Spatial and Temporal Patterns of Tumor Evolution," *PLoS Computational Biology*, vol. 10, no. 8, 2014.

[22] V. Popic, R. Salari, I. Hajirasouliha, D. Kashef-Haghighi, R. B. West, and S. Batzoglou, "Fast and scalable inference of multi-sample cancer lineages," *Genome Biology*, vol. 16, no. 1, 2015.

[23] N. Beerenwinkel, R. F. Schwarz, M. Gerstung, and F. Markowetz, "Cancer Evolution: Mathematical Models and Computational Inference," *Systematic Biology*, vol. 64, no. 1, 2014.

[24] Y. Matsui, A. Niida, R. Uchi, K. Mimori, S. Miyano, and T. Shimamura, "phyC: Clustering cancer evolutionary trees," *PLOS Computational Biology*, vol. 13, no. 5, 2017.

[25] M. Gerlinger, S. Horswell, J. Larkin, A. J. Rowan, M. P. Salm, I. Varela, R. Fisher, N. Mcgranahan, N. Matthews, C. R. Santos, P. Martinez, B. Phillimore, S. Begum, A. Rabinowitz, B. Spencer-Dene, S. Gulati, P. A. Bates, G. Stamp, L. Pickering, M. Gore, D. L. Nicol, S. Hazell, P. A. Futreal, A. Stewart, and C. Swanton, "Genomic architecture and evolution of clear cell renal cell carcinomas defined by multiregion sequencing," *Nature Genetics*, vol. 46, no. 3, pp. 225–233, 2014.

[26] J. Zhang, J. Fujimoto, J. Zhang, D. C. Wedge, X. Song, J. Zhang, S. Seth, C.-W. Chow, Y. Cao, C. Gumbs, K. A. Gold, N. Kalhor, L. Little, H. Mahadeshwar, C. Moran, A. Protopopov, H. Sun, J. Tang, X. Wu, Y. Ye, W. N. William, J. J. Lee, J. V. Heymach, W. K. Hong, S. Swisher, I. I. Wistuba, and P. A. Futreal, "Intratumor heterogeneity in localized lung adenocarcinomas delineated by multiregion sequencing," *Science*, vol. 346, no. 6206, pp. 256–259, 2014.

[27] S. Cohen, "Indexing for subtree similarity-search using edit distance," *Proceedings of the 2013 international conference on Management of data - SIGMOD 13*, 2013.

[28] J. Allali and M.-F. Sagot, "Novel Tree Edit Operations for RNA Secondary Structure Comparison," *Lecture Notes in Computer Science Algorithms in Bioinformatics*, pp. 412–425, 2004.

[29] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava, and T. Yu, "Approximate XML joins," *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD 02*, 2002.

[30] M. Gerstung, C. Beisel, M. Rechsteiner, P. Wild, P. Schraml, H. Moch, and N. Beerenwinkel, "Reliable detection of subclonal singlenucleotide variants in tumour cell populations," *Nature Communications*, vol. 3, no. 1, 2012.

[31] K. Wang, M. Li, and H. Hakonarson, "ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data," *Nucleic Acids Research*, vol. 38, no. 16, Mar. 2010.

[32] Y. Xue, C. Wang, H. H. Ghenniwa, and W. Shen, "A new tree similarity measuring method and its application to ontology comparison," *2008 12th International Conference on Computer Supported Cooperative Work in Design*, 2008.

[33] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jan. 2002.

[35] T. K. Ho, "Random decision forests," *Proceedings of 3rd International Conference on Document Analysis and Recognition.*

[36] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[37] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[38] F. Conforti, L. Pala, V. Bagnardi, T. D. Pas, M. Martinetti, G. Viale, R. D. Gelber, and A. Goldhirsch, "Cancer immunotherapy efficacy and patients sex: a systematic review and meta-analysis," *The Lancet Oncology*, vol. 19, no. 6, pp. 737–746, 2018.

[39] F. Rampen, "Malignant melanoma: Sex differences in response to chemotherapy?," *European Journal of Cancer and Clinical Oncology*, vol. 18, no. 1, pp. 107–110, 1982.

[40] R. L. Siegel, K. D. Miller, and A. Jemal, "Cancer statistics, 2016," CA: A Cancer Journal for Clinicians, vol. 66, no. 1, pp. 7–30, 2016.

Biography

Hung-Yu Yan received the BS degree from the Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan, in 2016. He is now a Master student in the Institute of Medical Informatics, National Cheng Kung University, Taiwan.

Dun-Wei Cheng is currently pursuing the Ph.D. degree in the Department of Computer Science and Information Engineering at National Cheng Kung University in Tainan City, Taiwan. He received the M.S. degree from the Department of Computer Science and Information Engineering at National University of Kaohsiung, in 2011. His current research interests include financial computing, artificial intelligence, system-level fault diagnosis and hub location problem.

Peng-Chan Lin is the Director of the Center for Hospice and Palliative Shared Care at National Cheng Kung University Hospital (NCKUH). He graduated from the school of medicine, National Yang-Ming University, and received his Ph.D. degree in the Institute of clinical medicine, college of medicine, National Cheng Kung University (NCKU), Tainan, Taiwan. Dr. Lin is an attending physician in the oncology department and a principal investigator in the clinical trial center of NCKUH. He is an assistant professor in college of medicine, NCKU and is appointed as the deputy secretary of Taiwan Oncology Society.

Hsin-Hung Chou is an associate professor at the Department of Information Management, Chang Jung Christian University. He received the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University at 2004. His research interests include Computer Game, Machine Learning, Algorithm Design, Cloud Computing, and so on. In recent decade, Dr. Chou dedicates in developing the machine learning techniques for computer games. He has received the reward of the special outstanding researcher at 2015 by Ministry of Science and Technology. His computer Othello program Mothello has ever won the silver metal at Computer Olympia 2015 held by ICGA (International Computer Game Association). And his 9X9 computer Go program Wingo has ever won the bronze metal several times at the Computer Game contests held by TAAI (Association of Taiwan Artificial Intelligent) and TCGA (Taiwan Computer Game Association).

Meng-Ru Shen is the superintendent at National Cheng Kung University Hospital (NCKUH), and the distinguished professor of pharmacology, obstetrics, and gynecology at National Cheng Kung University (NCKU). He received the Ph.D. degree from the University of Oxford in 2002. In addition to molecular biology research, Dr. Shen dedicates himself to carrying out clinical research about chemotherapy-induced peripheral neuropathy (CIPN), drug development, and artificial intelligence in daily clinical care in recent years.

Sun-Yuan Hsieh received the PhD degree in computer science from National Taiwan University, Taipei, Taiwan, in June 1998. He then served the compulsory two-year military service. From August 2000 to January 2002, he was an assistant professor at the Department of Computer Science and Information Engineering, National Chi Nan University. In February 2002, he joined the Department of Computer Science and Information Engineering, National Cheng Kung University, and now he is a distinguished professor. He received the 2007 K. T. Lee Research Award, President Citation Award (American Biographical Institute) in 2007, the Engineering Professor Award of Chinese Institute of Engineers (Kaohsiung Branch) in 2008, the National Science Council Outstanding Research Award in 2009, and IEEE Outstanding Technical Achievement Award (IEEE Tainan Section) in 2011. He is Fellow of the British Computer Society (BCS). His current research interests include design and analysis of algorithms, fault-tolerant computing, bioinformatics, parallel and distributed computing, and algorithmic graph theory.