Similarity measure models and algorithms for hierarchical cases

Dianshuang Wu, Jie Lu, Guangquan Zhang

Decision Systems & e-Service Intelligence (DeSI) Lab
Centre for Quantum Computation & Intelligent Systems (QCIS)
Faculty of Engineering and Information Technology, University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia

Corresponding author: Jie Lu, Tel. : +61 02 95141838
E-mail: sd_wds@hotmail.com (D. Wu), jielu@it.uts.edu.au (J. Lu), zhangg@it.uts.edu.au (G. Zhang)

Abstract

Many business situations such as events, products and services, are often described in a hierarchical structure. When we use case-based reasoning (CBR) techniques to support business decision-making, we require a hierarchical-CBR technique which can effectively compare and measure similarity between two hierarchical cases. This study first defines hierarchical case trees (HC-trees) and discusses related features. It then develops a similarity evaluation model which takes into account all the information on nodes' structures, concepts, weights, and values in order to comprehensively compare two hierarchical case trees. A similarity measure algorithm is proposed which includes a node concept correspondence degree computation algorithm and a maximum correspondence tree mapping construction algorithm, for HC-trees. We provide two illustrative examples to demonstrate the effectiveness of the proposed hierarchical case similarity evaluation model and algorithms, and possible applications in CBR systems.

Keywords: Hierarchical similarity, Hierarchical cases, Tree similarity measuring, Case-based reasoning

1. Introduction

Case-based reasoning (CBR) is the process of solving new problems based on the solutions for similar past problems (Aamodt & Plaza, 1994). CBR provides a powerful learning ability to use past experiences as a basis for dealing with new problems, and facilitates the knowledge acquisition process by reducing the time required to elicit solutions from experts. It is represented by a four-step (4Rs) cycle: retrieve, reuse, revise and retain (Aamodt & Plaza, 1994). In the first 'R' stage, when a new problem is input, CBR retrieves the most similar case from the case base.

Obviously, designing an effective case similarity evaluation method to identify the most similar cases is a key issue in CBR. Many models and algorithms have been developed to measure similarity between two cases, which are described in a set of attributes (Falkman, 2000). However, in practice, some cases can only be described by hierarchical tree structures. Therefore, we need to explore effective similarity measures for hierarchical cases in order to apply CBR systems.

Fig. 1 shows an example of an avian flu case describing the infection situation of birds in an area at a given time (Zhang, Lu & Zhang 2009). Obviously, it is a hierarchical case and viewed as a tree structure. This tree case has seven nodes called “wild birds”, “farm poultry”,…, “water bird” and “no water bird”. The “water bird” node indicates that 40% of water birds were infected. From its edge, we can see 70% of farm poultry are water birds. Similarly, there are 60% of birds in the farm poultry area.

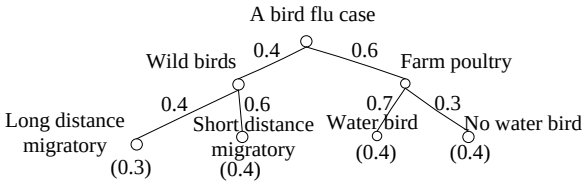


Fig.1. A hierarchical case: bird flu

From this example, we can summarize the following features of tree cases: (1) every node is associated with a concept; (2) all concepts represented by nodes of a tree case, construct a hierarchical structure. Nodes at different depths represent concepts with different abstraction levels. The child nodes can be viewed as a refinement of the concept expressed by their parent node; (3) all leaves of a tree case are assigned values. Other nodes’ values can be assessed by aggregating their children’s; (4) every node is assigned a “weight” to represent its importance relative to its parent node. As different cases may arise from different sources at different times, the tree structures, nodes’ concepts, weights and values in different trees are probably not all the same. To evaluate the similarity between these tree-structured hierarchical cases, all the information should be considered.

The research in this paper is related to work on tree similarity measure and structured case similarity measure. Tree structured data are used in many fields, such as e-business (Bhavsar, Boley, & Yang, 2004), bioinformatics (Tran, Nguyen, & Hoang, 2007), XML Schema matching (Jeong, Lee, Cho, & Lee, 2008), document classification and organization (Rahman & Chow,

2010) and case-based reasoning (Ricci & Senter, 1998). The similarity measure of tree structured data is essential for many applications. One widely used tree similarity measure is tree edit distance (Zhang, 1993; Kailing, Kriegel, Schonauer, & Seidl, 2004), in which edit operations including insertion, deletion and re-labeling with costs are defined, and the least cost of a sequence of edit operations needed to transform one tree to another is used as the similarity measure between the two trees (Bille, 2005). The main difference between various tree edit distance algorithms lies in the set of allowed edit operations and their related cost definitions (Yang, Kalnis, & Tung, 2005). In (Xue, Wang, Ghenniwa, & Shen, 2009), the conceptual similarity measure between labels was introduced in the cost of edit operations to compare concept trees of ontology. Another kind of tree similarity measure is based on a maximum common sub-tree (MCS) or sub-tree isomorphism between two trees (Akutsu & Halldorsson, 2000). This method uses the size of MCS between two trees, or metrics defined by MCS as the similarity measure. In (Torsello, Hidovic, & Pelillo, 2004), four novel distance measures for attributed trees based on the notion of a maximum similarity sub-tree isomorphism were proposed. In (Lin, Wang, McClean, & Liu, 2008), the number of all common embedded sub-trees between two trees was used as the measure of similarity. The methods mentioned above mostly deal with node-labeled trees. In (Bhavsar, Boley, & Yang, 2004), node-labeled, arc-labeled, arc-weighted trees were used as product/service descriptions to represent the hierarchical relationship between the attributes. To compare these trees, a recursive algorithm to perform a top-down traversal of trees and the bottom-up computation of similarity was designed. However, their trees had to conform to the same standard schema, i.e. the trees should have the same structure and use the same labels, though some sub-trees were allowed to be missing (Yang, Sarker, Bhavsar, & Boley, 2005). As trees for hierarchical cases in our research are different to previous ones, we need to develop a new similarity measure method for them.

Structured case similarity measure in the literature is usually based on the maximal common sub-graph or sub-graph isomorphism (Burke, MacCarthy, Petrovic, & Qu, 2000; Sanders, Kettler, & Hendler, 1997). In (Ricci & Senter, 1998), the similarity measure on tree structured cases, taking into account both the tree structures and node labels' semantics, was researched. A sub-tree isomorphism with the minimum semantic distance was constructed and the minimum semantic distance was used as the similarity measure. This research is closely related to ours. However, the

positions of corresponding nodes are not restricted in their sub-tree isomorphism, and this is not suitable for our hierarchical cases, because nodes at different depths represent concepts at different abstraction levels. Also, nodes' values are not involved in their similarity measure.

In this paper, we present a comprehensive similarity evaluation model considering all the information on nodes' structures, concepts, weights and values to compare tree structured hierarchical cases. To express the concept correspondence between nodes in different trees, the concept correspondence degree is defined. A maximum correspondence tree mapping based on nodes' structures and concepts is constructed to identify the corresponding nodes between two trees. Based on the mapping, the values of corresponding nodes are compared. Finally, the similarity measure of trees is evaluated by aggregating both the conceptual and value similarities.

This paper is organized as follows. In Section 2 we describe the features of hierarchical case trees by mathematical formulas. Section 3 presents a similarity evaluation model to compare any two hierarchical case trees. A set of algorithms to compute the similarity between hierarchical cases is provided in Section 4. Section 5 presents two examples to demonstrate the effectiveness of the proposed hierarchical case similarity evaluation model and algorithms, and possible applications in CBR systems. It also compares the proposed HC-tree similarity model with other approaches. Section 6 concludes the paper and discusses tasks for our further study.

2. Hierarchical case trees

A tree is defined as a directed graph $T = (V, E)$ where the underlying undirected graph has no cycles and there is a distinguished root node in V, denoted by $root(T)$, so that for all nodes $v \in V$, there is a path in T from $root(T)$ to node v (Valiente, 2002). In real applications, the definition can be extended to represent practical objects. To express the concepts, values and weights associated with the nodes of hierarchical cases and the hierarchical relationships between nodes, the original tree structure is enriched and a hierarchical case tree (HC-tree) is defined.

Definition 2.1: HC-tree. An HC-tree is a structure $T = (V, E, A, W, R)$, in which V is a finite set of nodes, E is a binary relation on V where each pair $(u, v) \in E$ represents the parent-child relationship between two nodes $u, v \in V$, A is a set of attributes assigned to each node in V, W is a function to assign each node a weight to represent its degree of importance

to its siblings, thereby satisfying the sum of the weights of all the children of one node is 1, and R is a function to assign a value to every leaf node to describe the degree of its relevant attribute.

Two features of the HC-tree should be highlighted. First, all nodes in the HC-tree represent concept meanings, which are obtained from the attributes. As a hierarchical structure, the concept of one node depends, not only on the attribute itself, but also its children's. Therefore, nodes at different depths represent concepts at different abstraction levels, and nodes at higher layers represent more significant concepts than lower nodes. Secondly, every node in the HC-tree has a value. The leaves' values are indicated by R, and the internal nodes' values can be computed by aggregating their children's.

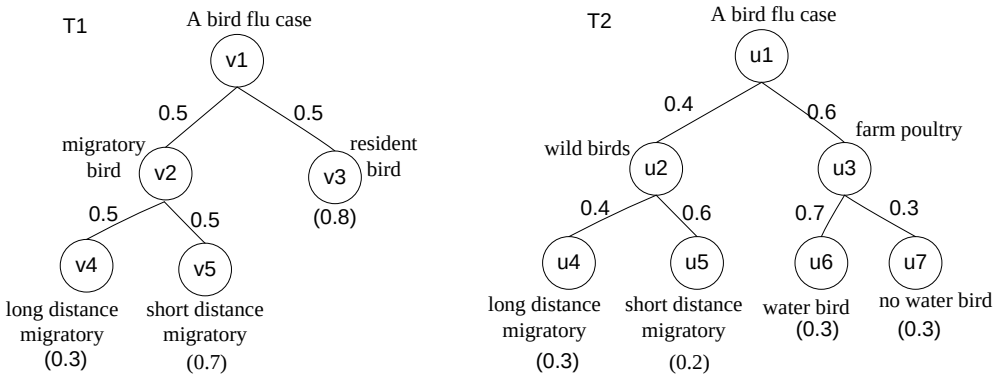


Fig.2. Two examples of HC-trees

Two examples of HC-trees, both describing the situation of bird flu, are illustrated in Fig. 2. The labels beside the nodes represent their attributes. The number beside the edge is the weight of the child. The number under each leaf represents its value. In T_1, "A bird flu case" is described by two aspects, "migratory bird" and "resident bird", with both taking the same weight. Similarly, "migratory bird" is described by two sub-aspects, "long distance migratory" and "short distance migratory". From "long distance migratory", we can see that 30% of birds were infected. As T_1 and T_2 are from different sources, their structures and nodes' weights are different. Their attribute terms are also not identical.

To evaluate the conceptual correspondence between attributes in different HC-trees, a conceptual similarity measure between attributes is introduced as in (Xue, Wang, Ghenniwa, & Shen, 2009).

Definition 2.2: Attribute Conceptual Similarity Measure. An attribute conceptual similarity

measure sc_{A_1,A_2} is a set of mappings from two attribute sets A_1, A_2 used in different HC-trees to the set [0, 1], $sc_{A_1,A_2} : A_1 \times A_2 \to [0,1]$, in which each mapping denotes the conceptual similarity between two attributes. For convenience, the sub-script A_1, A_2 is omitted so that there is no confusion. For $a_1 \in A_1$ and $a_2 \in A_2$, we say a_1 and a_2 are similar if $sc(a_1, a_2) > 0$, and the larger $sc(a_1, a_2)$ is, the more similar the two attributes are.

Conceptual similarity between two attributes can be given by domain experts or calculated based on linguistic analysis methods. As an example, we define the conceptual similarity between the attributes of T_1 and T_2 in Fig. 2 as follows:

sc(migratory bird, wild birds)=0.7, *sc(migratory bird, farm poultry)=0.1,*

sc(resident bird, wild birds)=0.6, *sc(resident bird, farm poultry)=0.8,*

sc(resident bird, water bird)=0.4, *sc(resident bird, no water bird)=0.4,*

sc(long distance migratory, water bird)=0.1, *sc(long distance migratory, no water bird)=0.2,*

sc(short distance migratory, water bird)=0.2, *sc(short distance migratory, no water bird)=0.2.*

3. A similarity evaluation model for HC-trees

A similarity evaluation model for HC-trees is proposed in this section. In the model, maximum correspondence tree mapping is constructed to identify the corresponding node pairs of two HC-trees based on nodes' structures and concepts, and the conceptual similarity between two HC-trees is evaluated. Based on the mapping, the value similarity between two HC-trees is evaluated, and the final similarity measure between two HC-trees is assessed as a weighted sum of their conceptual and value similarities.

3.1 Maximum correspondence tree mapping

To identify two corresponding nodes in different HC-trees, both their structures and concepts should be considered.

There are two structural restrictions. First, as nodes at different depths represent concepts at different abstraction levels, it is reasonable to assume that the corresponding nodes in the mapping should be at the same depth. Therefore, the roots of two HC-trees should be in the mapping. Secondly, as the children nodes can be viewed as a refinement of the concept expressed by the

parent node, two separate sub-trees in one tree should be mapped to two separate sub-trees in another.

In addition to satisfying structural restrictions, it is important that the corresponding nodes have a high conceptual similarity degree. To express the concept correspondence between two nodes in two HC-trees respectively, the following definition is introduced:

Definition 3.1: Node Concept Correspondence Degree. Let V_1 and V_2 be node sets of T_1 and T_2 respectively. A node concept correspondence degree $cord$ is a set of mappings from V_1 and V_2 to the set [0, 1], $cord : V_1 \times V_2 \to [0,1]$, in which each mapping denotes the concept correspondence between two nodes of two HC-trees.

$cord$ is symmetric, i.e. for $v \in V_1$ and $u \in V_2$ we have $cord(v,u) = cord(u,v)$.

Let v and u be two nodes of T_1 and T_2 respectively. There are three cases: (1) both v and u are leaves, (2) v is a leaf and u is an internal node, (3) both v and u are internal nodes. In the first case, as nodes' concepts are derived from the attributes assigned to them, the concept correspondence degree between v and u can be defined as the conceptual similarity of their attributes. In the other two cases, as the internal node's concept is also affected by its children, the children's concepts should be considered in the definition. Thus, they should be defined recursively. The definitions of $cord$ for the three cases are presented respectively as follows:

Definition 3.2: Concept Correspondence Degree between Two Leaves. Let v and u be two leaves of T_1 and T_2 respectively. The concept correspondence degree between v and u, $cord(v,u)$ is defined as:

$$cord(v,u) = sc(v.a, u.a) \tag{3.1}$$

where $v.a$ and $u.a$ represent attributes of v and u respectively.

For example, v_4 and u_6 are two leaves of T_1 and T_2 respectively in Fig. 2. The attribute of v_4 is "long distance migratory" and of u_6 is "water bird". The concept correspondence degree between v_4 and u_6 is defined by *sc(long distance migratory, water bird)*, and $cord(v_4, u_6)$ =0.1.

Definition 3.3: Concept Correspondence Degree between a Leaf and an Internal Node. Let v be a leaf of T_1, u be an internal node of T_2, and $C(u)=\{u_1,u_2,...,u_q\}$ be u's children set. The concept correspondence degree between v and u, $cord(v,u)$ is defined as:

$$cord(v,u)=\alpha\cdot sc(v.a,u.a)+(1-\alpha)\cdot\sum_{i=1}^{q}w_{2i}\cdot cord(v,u_i) \tag{3.2}$$

where α is the influence factor of the parent node and w_{2i} is the weight of u_i.

For example, v_3 is a leaf node of T_1 and u_3 is an internal node of T_2 in Fig. 2. The concept correspondence degree between v_3 and u_3 is computed by the formula $cord(v_3,u_3)=\alpha\cdot sc(v_3.a,u_3.a)+(1-\alpha)\cdot(0.7\cdot cord(v_3,u_6)+0.3\cdot cord(v_3,u_7))$. In the formula, $sc(v_3.a,u_3.a)$ is 0.8. $cord(v_3,u_6)$ and $cord(v_3,u_7)$ can be computed by Definition 3.2, and $cord(v_3,u_6)$ =0.4 and $cord(v_3,u_7)$ =0.4. If α is 0.5, we can achieve $cord(v_3,u_3)$ =0.6.

Definition 3.4: Concept Correspondence Degree between Two Internal Nodes. Let v and u be two internal nodes of T_1 and T_2 respectively, and $C(v)=\{v_1,v_2,...,v_p\}$ and $C(u)=\{u_1,u_2,...,u_q\}$ be v and u's children sets, respectively. Let $G_{v,u}=(V,E)$ denote the bipartite graph induced by v and u, which is constructed as follows: $V=C(v)\cup C(u)$, $E=\{(s,t):s\in C(v),t\in C(u)\}$. The weights of edges are defined as $weight_{s,t}=cord(s,t)$. $MWM_{v,u}$ is the maximum weighted bipartite matching of $G_{v,u}$. Then, the correspondence degree between v and u, $cord(v,u)$ is defined as:

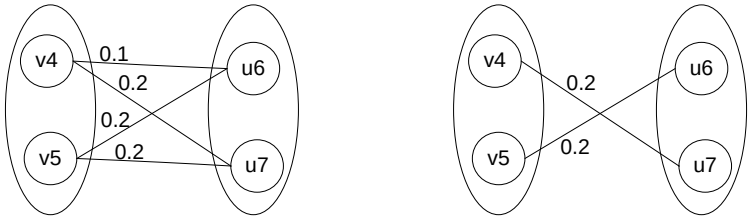
$$cord(v,u)=\alpha\cdot sc(v.a,u.a)+(1-\alpha)\cdot\sum_{(v_i,u_j)\in MWM_{v,u}}\frac{1}{2}(w_{1i}+w_{2j})\cdot cord(v_i,u_j) \tag{3.3}$$

where w_{1i} is the weight of v_i in T_1 and w_{2j} is the weight of u_j in T_2.

In Definition 3.4, the maximum weighted bipartite matching $MWM_{v,u}$ identifies the most correspondence node pairs amongst v and u's children. The contribution of their children can

therefore be fully considered when evaluating their concept correspondence degree.

For example, v_2 and u_3 are two internal nodes of T_1 and T_2 respectively in Fig. 2. To compute their concept correspondence degree, a bipartite graph G_{v_2,u_3} is constructed as Fig. 3 (a), in which the numbers beside the edges represent their weights. The maximum weighted bipartite matching of G_{v_2,u_3} is illustrated in Fig. 3 (b). Then, The concept correspondence degree between v_2 and u_3 is computed by the formula $cord(v_2,u_3) = \alpha \cdot sc(v_2.a,u_3.a) + (1-\alpha)\cdot(((0.5+0.3)/2)\cdot cord(v_4,u_7)+((0.5+0.7)/2)\cdot cord(v_5,u_6))$. In the formula, $sc(v_2.a,u_3.a)$ is 0.1. $cord(v_4,u_7)$ and $cord(v_5,u_6)$ are computed by Definition 3.2, and $cord(v_4,u_7)$ =0.2 and $cord(v_5,u_6)$ =0.2. Let α be 0.5, so that we achieve $cord(v_3,u_3)$ =0.15.



(a) (b)

Fig.3. A bipartite graph G_{v_2,u_3} and its maximum weighted bipartite matching

With the above definitions, the concept correspondence degree of any node pair between two HC-trees can be evaluated. The maximum correspondence tree mapping considering both the structural restrictions and nodes' concept correspondence is defined as follows.

Definition 3.5: Maximum Correspondence Tree Mapping. Let V_1 and V_2 be node sets of HC-trees T_1 and T_2, respectively. A mapping $M \subseteq V_1 \times V_2$ is a maximum correspondence tree mapping if it satisfies the following conditions:

1. $v_1 = v_2 \Leftrightarrow u_1 = u_2$ for any pair $(v_1,u_1),(v_2,u_2) \in M$
2. $(root(T_1), root(T_2)) \in M$
3. $(parent(v), parent(u)) \in M$ for all non-root nodes $v \in V_1$ and $u \in V_2$ with

$(v,u) \in M$

4. $cord(v,u) > 0$ for all nodes $v \in V_1$ and $u \in V_2$ with $(v,u) \in M$

5. $MWM_{v,u} \subset M$ for all nodes $v \in V_1$ and $u \in V_2$ with $(v,u) \in M$, where $MWM_{v,u}$ is the maximum weighted bipartite matching of bipartite graph $G_{v,u}$ constructed of v and u's children with edges weighted by their children's concept correspondence degree.

In the above Definition 3.5, the first condition ensures that the mapping is a one-to-one mapping. Conditions 2 and 3 ensure the mapping satisfies the structural restrictions. The last two conditions represent the conceptual restrictions. Condition 5 ensures that most correspondence node pairs are in the mapping. As an example, the maximum correspondence tree mapping between T_1 and T_2 in Fig. 2 is illustrated in Fig. 4, in which corresponding nodes are connected by dashes. The construction process of the mapping will be described in Section 5.1.

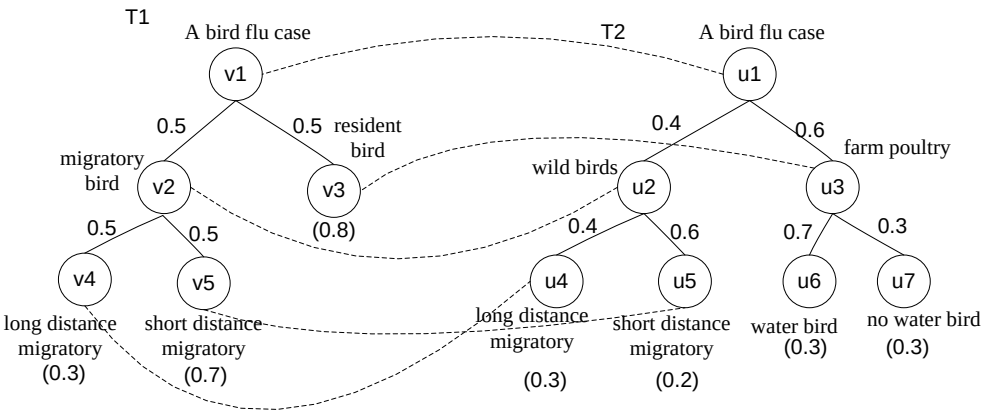


Fig.4. Maximum correspondence tree mapping between T_1 and T_2

From the recursive definitions of node concept correspondence degree, it is obvious that $cord(root(T_1), root(T_2))$ is computed by aggregating the $cord$ of all corresponding node pairs, which reflects the conceptual similarity between two HC-trees. We can define the conceptual similarity between two HC-trees as follows:

Definition 3.6: Conceptual Similarity between HC-trees. Let T_1 and T_2 be two HC-trees. The conceptual similarity between T_1 and T_2, $sct(T_1, T_2)$ is defined as

$sct(T_1,T_2) = cord(root(T_1), root(T_2))$.

Taking T_1 and T_2 in Fig. 2 as an example, their conceptual similarity, $sct(T_1,T_2)$ is computed as $cord(v_1,u_1)$.

3.2 Value similarity between HC-trees

Based on the maximum correspondence tree mapping M, the values of two HC-trees can be compared.

The value similarity between two corresponding nodes in M is evaluated first. As only leaf nodes are assigned values in HC-trees initially, for any $(v,u) \in M$, there are two cases: (1) v is a leaf node, or none of v's children are in M, (2) some of v's children are in M. We provide the computation formulas of the value similarity between v and u, $sv_M(v,u)$ for the two cases respectively.

For case 1, $sv_M(v,u)$ is computed as:

$$sv_M(v,u) = s(value(v), value(u)) \tag{3.4}$$

where $value(v)$ denotes v's value and $s(\cdot)$ denotes a value similarity measure.

If v is a leaf node, $value(v)$ is assigned initially. Otherwise, it is computed by aggregating its children's values. $s(\cdot)$ can be defined according to the specific applications. For example, let two attributes' values be a_1 and a_2, and their value range be r; then their similarity measure can be defined as $s(a_1,a_2) = 1 - |a_1 - a_2|/r$. In the example in Fig. 4, as values of nodes are all within [0, 1], the similarity between two values is calculated as one minus the distance between them. For v_3 and u_3 in Fig. 4, $value(v_3)$ is assigned initially as 0.8, and $value(u_3)$ can be computed as 0.3. The value similarity between v_3 and u_3 is then computed as 0.5.

In case 2, let $v_1, v_2, ..., v_p$ be v's children and $u_1, u_2, ..., u_q$ be u's children. $sv_M(v,u)$ is computed as:

$$sv_M(v,u) = \sum_{(v_i,u_j)\in M} \frac{1}{2}(w_{1i} + w_{2j}) \cdot sv_M(v_i,u_j) \quad (3.5)$$

where w_{1i} is the weight of v_i in T_1 and w_{2j} is the weight of u_j in T_2.

Take v_2 and u_2 in Fig. 4 as an example. Their value similarity is computed as $sv_M(v_2,u_2) = ((0.5+0.4)/2) \cdot sv_M(v_4,u_4) + ((0.5+0.6)/2) \cdot sv_M(v_5,u_5)$. In the formula, $sv_M(v_4,u_4)$ and $sv_M(v_5,u_5)$ can be calculated by Formula (3.4), and $sv_M(v_2,u_2)$=0.725.

With Formula (3.4) and (3.5), the value similarity between any corresponding nodes in M can be computed. As the recursive characteristic of Formula (3.5), the value similarity between the roots of two HC-trees is computed by aggregating the value similarity of all corresponding node pairs, which represents the value similarity of the two HC-trees. Therefore, we define the value similarity between two HC-trees as follows.

Definition 3.7: Value Similarity between HC-trees. Let T_1 and T_2 be two HC-trees, and M be their maximum correspondence tree mapping. The value similarity between T_1 and T_2, $svt(T_1,T_2)$ is defined as $svt(T_1,T_2) = sv_M(root(T_1),root(T_2))$.

Taking T_1 and T_2 in Fig. 4 as an example, their value similarity, $svt(T_1,T_2)$ is computed as $sv_M(v_1,u_1)$.

3.3 Similarity measure of HC-trees

Based on the conceptual similarity and value similarity of two HC-trees, the similarity measure of HC-trees is defined as follows.

Definition 3.8: Similarity Measure of HC-trees. The similarity between T_1 and T_2 is defined as:

$$sim(T_1,T_2) = \alpha_1 \cdot sct(T_1,T_2) + \alpha_2 \cdot svt(T_1,T_2) \quad (3.6)$$

where $\alpha_1 + \alpha_2$=1.

In this definition, both the concepts and values of two HC-trees are comprehensively considered. α_1 and α_2 are weights of the two parts, which can be defined according to the

specific applications.

4. Similarity measurement algorithms for HC-trees

Algorithms to compute the similarity between two HC-trees are presented in this section. The flowchart in Fig. 5 shows the entire process.

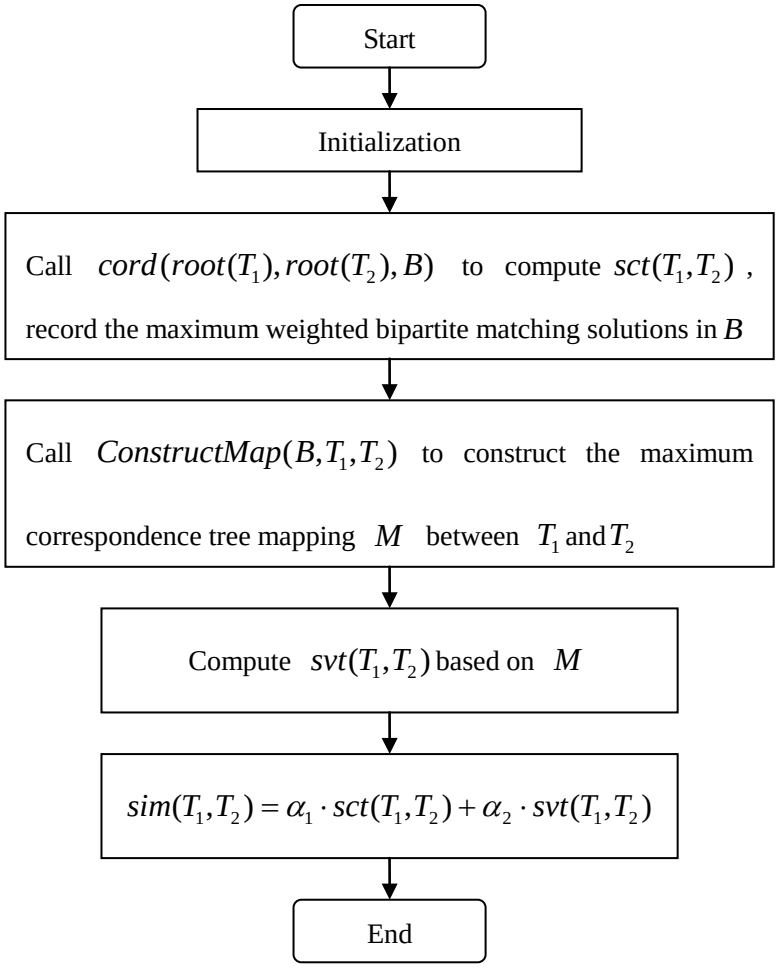


Fig.5. Flowchart to compute the similarity between two HC-trees

We can see from Fig.5 that $sct(T_1, T_2)$ is firstly computed by calling $cord(root(T_1), root(T_2), B)$, where B is a node set list which is indexed by the nodes in T_1. All the maximum weighted bipartite matching solutions during computing $cord(root(T_1), root(T_2))$ are recorded in B. The maximum correspondence tree mapping M is then constructed based on B by calling $ConstructMap(B, T_1, T_2)$. $svt(T_1, T_2)$ is computed based on M. Finally, the similarity of T_1 and T_2 is returned by aggregating their conceptual and value similarities. The algorithm is illustrated as follows.

Algorithm 1. Similarity measure algorithm for HC-trees

similarity(T_1 , T_2)

input: two trees T_1 and T_2

output: similarity between T_1 and T_2

1 for all $v \in V$

$B(v) \leftarrow \Phi$

2 $sct \leftarrow cord(root(T_1), root(T_2), B)$

3 $M \leftarrow ConstructMap(B, T_1, T_2)$

4 $svt \leftarrow sv_M(root(T_1), root(T_2))$

5 return $\alpha_1 \cdot sct + \alpha_2 \cdot svt$

The algorithm of concept correspondence degree computation function $cord(v,u,B)$ is illustrated by algorithm 2 as follows.

Algorithm 2. Node concept correspondence degree computation algorithm

$cord(v,u,B)$

input: two nodes v and u

output: concept correspondence degree between v and u

1 if both v and u are leaves

2 return $sc(v.a, u.a)$

3 else if u is an internal node, and $u_1, u_2, ..., u_q$ be u 's children,

4 return $\alpha \cdot sc(v.a, u.a) + (1-\alpha) \cdot \sum_{i=1}^{q} w_{2i} \cdot cord(v, u_i, B)$

5 else $C(v) \leftarrow v$'s children $v_1, v_2, ..., v_p$

6 $C(u) \leftarrow u$'s children $u_1, u_2, ..., u_q$

7 for i=1 to p

8 for j=1 to q

9 $c_{ij} \leftarrow cord(v_i, u_j, B)$

10 $m \leftarrow ComputeMatching(C(v) \cup C(u), c)$

11 for each $(v_k, u_l) \in m$, if $c_{kl} > 0$

12 $B(v_k) \leftarrow B(v_k) \cup \{u_l\}$

13 return $\alpha \cdot sc(v.a, u.a) + (1-\alpha) \cdot \sum_{(v_k, u_l) \in m} ((w_{1k} + w_{2l})/2) \cdot c_{kl}$

A recursive process follows from the definition of concept correspondence degree. The most important part in the procedure are lines 5-13, where both v and u are internal nodes. A bipartite graph is constructed, taking their children as nodes, and the correspondence degrees between their children as the weights of edges. Function $ComputeMatching(\cdot)$ (Jungnickel, 2008) returns the maximum weighted bipartite matching, which identifies most correspondence node pairs among v and u's children. The matches are recorded in B, which are local maximum correspondence matches. For one node in T_1, there may be more than one node matching it during the computation process. However, as proved in (Valiente, 2002), there is a unique maximum correspondence tree mapping $M \subseteq V_1 \times V_2$ so that $M \subseteq B$. Given B, the corresponding maximum correspondence tree mapping M can be reconstructed as follows: Set $M(root(T_1))$ to $root(T_2)$ and, for all nodes $v \in V_1$ in pre-order, set $M(v)$ to the unique node u with $(v,u) \in B$ and $(parent(v), parent(u)) \in B$. The reconstruction procedure is illustrated by algorithm 3 (Valiente, 2002).

Algorithm 3. Maximum correspondence tree mapping construction algorithm

$ConstructMap(B, T_1, T_2)$

input: node set list B, two HC-trees T_1 and T_2

output: maximum correspondence tree mapping M from T_1 to T_2

1 $M(root(T_1)) \leftarrow root(T_2)$

2 *list* $L \leftarrow preorder\_traversal(T_1)$

3 for all $v \in L$

4 if v is nonroot and $B(v) \neq \Phi$

5 for all $u \in B(v)$

6 if $M(parent(v)) == parent(u)$

7 $M(v) \leftarrow u$

8 break

9 return M

5. Two illustrative examples and comparison with other approaches

The proposed HC-tree similarity model and algorithms are to be used in CBR systems, such as CBR-based warning systems (Zhang, Lu & Zhang 2009), CBR-based recommender systems (Lu et al 2010) and web mining systems (Wang, Lu & Zhang, 2007). To show the effectiveness of our model, two examples are provided in this section. In the first example, the process of computing the similarity between two HC-trees in Fig. 2 is presented to show the behavior of the proposed algorithms in Section 4. In the second example, our similarity model is used in the retrieve stage of a simple CBR system to demonstrate the effectiveness of the model. The proposed model is then compared with other tree similarity evaluation methods.

5.1 Similarity measure computation between two HC-trees

The similarity between T_1 and T_2 in Fig. 2 is computed by the proposed similarity measurement algorithms as follows.

First, the conceptual similarity between T_1 and T_2, $sct(T_1, T_2)$ is computed by calling $cord(v_1, u_1, B)$. Let the coefficient α be 0.5, $sct(T_1, T_2)$ is computed as 0.856. During the recursive computation process, many maximum weighted bipartite matching problems are resolved, and the solutions are recorded in B: $B(v_2) = \{u_2\}$, $B(v_3) = \{u_3\}$, $B(v_4) = \{u_4, u_7\}$, $B(v_5) = \{u_5, u_6\}$.

Secondly, given B, the maximum correspondence tree mapping M between T_1 and T_2 is constructed by calling $ConstructMap(B, T_1, T_2)$: $M(v_1) = \{u_1\}$, $M(v_2) = \{u_2\}$, $M(v_3) = \{u_3\}$, $M(v_4) = \{u_4\}$, $M(v_5) = \{u_5\}$. The mapping is illustrated in Fig. 4.

Based on the mapping M, the value similarity between T_1 and T_2, $svt(T_1,T_2)$ is evaluated by computing $sv_M(v_1,u_1)$. The computation uses Formulas 3.4 and 3.5 to achieve $svt(T_1,T_2)$ as 0.6.

Finally, let the weights α_1 and α_2 be both 0.5; the final similarity measurement between T_1 and T_2, $sim(T_1,T_2)$ is computed by $0.5 \cdot sct(T_1,T_2) + 0.5 \cdot svt(T_1,T_2)$ =0.73.

5.2 Similar cases retrieval

The proposed similarity model is used to retrieve similar cases in a CBR system in the following example.

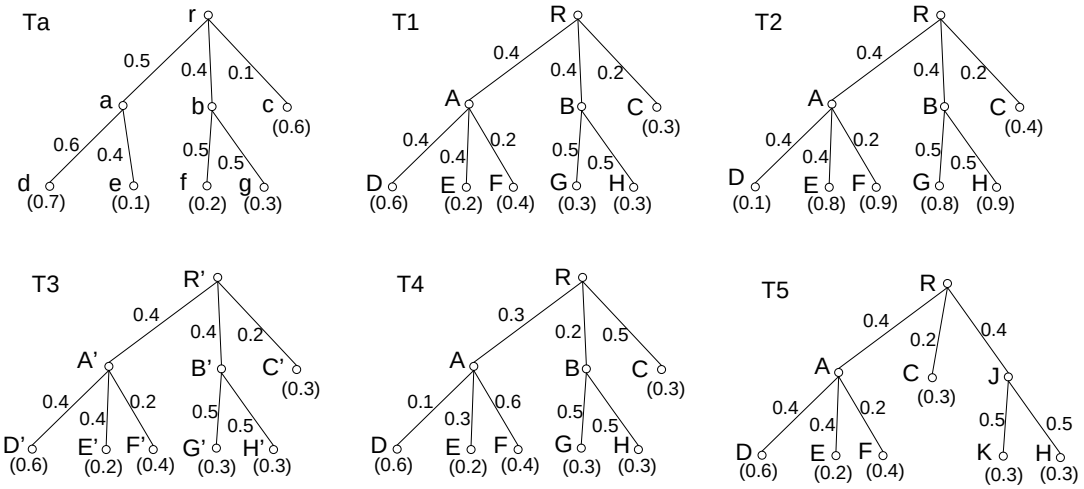


Fig.6. A new case T_a and five existing cases in a case base

As illustrated in Fig. 6, HC-tree T_a represents a new problem to be resolved and T_1,…, T_5 represent five solved problems in a case base. The conceptual similarity between their attributes is defined as follows: *sc(r,R)=0.7, sc(a,A)=0.9, sc(a,B)=0.6, sc(b,A)=0.5, sc(b,B)=0.8, sc(d,D)=1, sc(d,E)=0.5, sc(d,G)=0.4, sc(e,D)=0.5, sc(e,E)=0.9, sc(e,H)=0.4, sc(f,F)=1, sc(f,H)=0.6, sc(f,G)=0.7, sc(g,G)=0.9, sc(g,H)=0.7, sc(r,R')=0.6, sc(a,A')=0.7, sc(a,B')=0.6, sc(b,A')=0.5, sc(b,B')=0.7, sc(d,D')=0.9, sc(d,E')=0.4, sc(d,G')=0.3, sc(e,D')=0.4, sc(e,E')=0.8, sc(e,H')=0.3, sc(f,F')=0.7, sc(f,H')=0.6, sc(f,G')=0.6, sc(g,G')=0.7, sc(g,H')=0.6.*

To retrieve the most similar cases to T_a, the similarities between T_a and cases in the case base are evaluated using the similarity model proposed in this paper. Let the coefficients α, α_1

and α_2 be all 0.5 in the model, and the similarity between two values be calculated as one minus the distance between them. The results are illustrated in Table 1. As can be seen in Table 1, T_1 is most similar to T_a, so T_1 is retrieved.

Table 1 Similarity between T_a and cases in the case base

	T_1	T_2	T_3	T_4	T_5
$sct(T_a,T_i)$	0.703	0.703	0.600	0.623	0.548
$svt(T_a,T_i)$	0.745	0.304	0.745	0.537	0.365
$sim(T_a,T_i)$	0.724	0.504	0.672	0.580	0.456

As seen from Fig. 6, T_2 and T_1 are the same except for their values. Therefore, the conceptual similarity $sct(T_a,T_1)$ and $sct(T_a,T_2)$ are equal. However, as T_1's values are much closer to T_a's than T_2's, $svt(T_a,T_1)$ is larger than $svt(T_a,T_2)$, which makes T_1 more similar to T_a than T_2. T_3 and T_1 are different in terms of attributes. The concepts of T_1's attributes are more similar to T_a's than T_3's, which makes T_1 more similar to T_a than T_3. T_4 and T_1 have different attribute weights. The weights of nodes corresponding to T_a in T_4 are smaller than those in T_1, which makes T_4 less similar to T_a than T_1.

The example shows that our similarity model takes into account all the information on nodes' structures, concepts, weights and values and it can be used to retrieve the most similar cases effectively in CBR systems.

5.3 Comparison with other approaches

From the above examples, it can be seen that the proposed similarity evaluation model for HC-trees has five features: (1) it considers nodes' conceptual similarities; (2) it considers the hierarchical relations between concepts; (3) it compares corresponding nodes' values; (4) it considers the influence of nodes' weights; (5) it considers the semantics of nodes' structures. We compare our method with other tree similarity evaluation methods for these five aspects. We take

into account the methods of Ricci, & Senter's (1998), Xue, Wang, Ghenniwa, & Shen's (2009) and Bhavsar, Boley, & Yang's (2004), as they can represent different types of methods, respectively. The comparison results are illustrated in Table 2, where " √ " represents that the method has the related feature. Table 2 demonstrates that none of the earlier methods can compare the tree structured data as comprehensively as our method.

However, these features are essential to evaluate the similarity between complex tree structured hierarchical cases. As different HC-trees usually have different structures and attribute terms, the corresponding nodes between two HC-trees must be identified by evaluating their conceptual similarity. As attributes in hierarchical cases construct a hierarchical structure, the hierarchical relations between concepts and the semantics of nodes' structures must be considered. Nodes' values and relevant weights are essential to describe the case, so they must be compared when comparing two cases. With the above five features, the HC-trees can be compared comprehensively and accurately, and the most similar cases can be retrieved. Therefore, the proposed HC-tree similarity evaluation model is extremely suitable for retrieval of similar cases in CBR systems.

Table 2 Comparison between our proposed method and other methods

Method	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5
Our method	√	√	√	√	√
Ricci's method	√			√	
Xue's method	√				√
Bhavsar's method	√			√	√

6. Conclusion and future work

This paper defines the hierarchical case trees (HC-trees) to represent hierarchical cases. A similarity evaluation model to compare HC-trees is proposed and the related algorithms are presented. The concept correspondence degree between nodes is defined in the model and the conceptual similarity between trees is evaluated; a maximum correspondence tree mapping based on nodes' structures and concepts is proposed to identify the corresponding nodes of two trees; the value similarity between two trees is computed based on the mapping; the final similarity measure

between two trees is evaluated by aggregating their conceptual and value similarities. Two illustrative examples show that our method is highly effective for use in CBR systems.

Our future research includes (1) to define fuzzy-HC-trees and a fuzzy similarity evaluation model based on our previous study (Lu, Zhang, Ruan & Wu, 2007) and propose related algorithms in order to improve inference accuracy in CBR systems; (2) to develop software based on the proposed similarity evaluation model for integration into real CBR systems, such as our BizSeeker recommender system (Lu et al. 2010) helping measuring similarity between two business on their product trees and our CBR-based avian influenza risk early warning (Zhang, Lu & Zhang, 2009).

Acknowledgements

The work presented in this paper was supported by the Australian Research Council (ARC) under Discovery Project DP0880739 and the China Scholarship Council.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communication*, 7(1), 39-59.

Akutsu, T. & Halldorsson, M.M. (2000). On the approximation of largest common subtrees and largest common point sets. *Theoretical Computer Science*, 233, 33-50.

Bhavsar, V.C. Boley, H. & Yang, L. (2004). A weighted-tree similarity algorithm for multi-agent systems in e-business environments. *Computational Intelligence*, 20(4), 584-602.

Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3), 217-239.

Burke, E. MacCarthy, B. Petrovic, S. & Qu, R. (2000). Structured cases in case-based reasoning--re-using and adapting cases for time-tabling problems. *Knowledge-Based Systems*, 13(2-3), 159-165.

Falkman, G. (2000). Similarity measures for structured representations: a definitional approach. In E. Blanzieri, & L. Portinale (Eds.), *Advances in Case-Based Reasoning: 5th European Workshop, EWCBR 2000, Trento, Italy, September 6-9, 2000 Proceedings. Lecture Notes in Artificial Intelligence*, 1898, 380-392. Springer-Verlag Berlin Heidelberg.

Jeong, B. Lee, D. Cho, H. & Lee, J. (2008). A novel method for measuring semantic similarity for XML schema matching. *Expert Systems with Applications*, 34(3), 1651-1658.

Jungnickel, D. (2008). *Graphs, networks, and algorithms*, Berlin: Springer, 419-430.

Kailing, K. Kriegel, H.P. Schonauer, S. & Seidl, T. (2004). Efficient similarity search for hierarchical data in large databases. In E. Bertino et al. (Eds.), *Advances in Database Technology -EDBT 2004: 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004 Proceedings. Lecture Notes in Computer Science*, 2992, 676-693. Springer-Verlag Berlin Heidelberg.

Lin, Z. Wang, H. McClean, S. & Liu, C. (2008). All common embedded subtrees for measuring

tree similarity. *2008 International Symposium on Computational Intelligence and Design*, 1, 29-32.

Lu, J., Zhang, G., Ruan, D. and Wu, F. (2007). Multi-objective group decision making: methods, software and applications with fuzzy set techniques, Imperial College Press, London.

Lu J., Shambour, Q., Xu, Y., Lin, Q. and Zhang, G. (2010). A hybrid semantic recommendation system for personalized government-to-business e-services, Internet Research (Acceptance date: 30 January 2010).

Rahman, M. & Chow, T.W. (2010). Content-based hierarchical document organization using multi-layer hybrid network and tree-structured features. *Expert Systems with Applications*, 37(4), 2874-2881.

Ricci, F., & Senter, L. (1998). Structured cases, trees and efficient retrieval. In B. Smyth, & P. Cunningham (Eds.), *Advances in Case-Based Reasoning: 4th European Workshop, EWCBR-98, Dublin, Ireland, September 23-25, 1998 Proceedings. Lecture Notes in Artificial Intelligence*, 1488, 88-99. Springer-Verlag, Berlin Heidelberg New York.

Sanders, K.E. Kettler, B.P. & Hendler, J.A. (1997). The case for graph-structured representations. In D.B. Leake, & E. Plaza (Eds.), *Case-based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, ICCBR-97 Providence, RI, USA, July 25–27, 1997 Proceedings. Lecture Notes in Artificial Intelligence*, 1266, 245-254. Springer Berlin Heidelberg.

Torsello, A. Hidovic, D. & Pelillo, M. (2004). Four metrics for efficiently comparing attributed trees. *17th International Conference on Pattern Recognition (ICPR'04)*, 2, 467-470.

Tran, T. Nguyen, C.C. & Hoang, N.M. (2007). Management and analysis of DNA microarray data by using weighted trees. *Journal of Global Optimization*, 39(4), 623-645.

Valiente, G. (2002). *Algorithms on trees and graphs*, New York: Springer, 16-22, 206-224.

Wang, C., Lu, J. and Zhang, G. (2007), Mining key information of web pages: a method and its application, Expert Systems with Applications. Vol. 33, 425-433

Xue, Y. Wang, C. Ghenniwa, H.H. & Shen, W. (2009). A new tree similarity measuring method and its application to ontology comparison. *Journal of Universal Computer Science*, 15(9), 1766-1781.

Yang, L. Sarker, B.K. Bhavsar, V.C. & Boley, H. (2005). A weighted-tree simplicity algorithm for similarity matching of partial product descriptions. *Proceedings of ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering*, 55-60.

Yang, R. Kalnis, P. & Tung, A. (2005). Similarity evaluation on tree-structured data. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 754-765.

Zhang, J., Lu, J. and Zhang, G. (2009), Case-based reasoning in avian influenza risk early warning, The Second Conference on Risk Analysis and Crisis Response (RACR), October, 2009, Beijing, China. Atlantis Press, scientific publishing Paris France, 246-251

Zhang, K. (1993). A new editing based distance between unordered labeled trees. In A. Apostolico et al. (Eds.), *Combinatorial Pattern Matching: 4th Annual Symposium, CPM 93, Padova, Italy, June 2-4, 1993 Proceedings. Lecture Notes in Computer Science*, 684, 254-265. Springer-Verlag London, UK.