Semantic Similarity of Workflow Traces with Various Granularities

Qing Liu[1], Quan Bai[2], Yi Yang[2]

[1] Software & Computational Systems, Data61, CSIRO, Australia
Q.Liu@csiro.au
[2] Auckland University of Technology, New Zealand
Quan.Bai, Yi.Yang@aut.edu.nz

Abstract A workflow trace describes provenance information of a particular workflow execution. Understanding workflow traces and their similarity have many applications in both scientific research and business world. Given workflow traces generated by heterogeneous systems with difference granularities, it is a challenge for users to understand their similarities. In this work, we investigate workflow traces' granularity problem and their similarity method. Algorithms are developed to transform a trace into its multi-granularity forms assisting by a workflow trace ontology; A novel generic semantic similarity algorithm is proposed that not only considers the structural similarity but also the semantics coverage embedded in traces during transformation. Furthermore, theoretical analysis is presented to compute the maximum and minimum semantic similarity; Our approach enables the two workflow traces can be compared with any granularity. The experiment using real world workflow traces demonstrates the effectiveness of the proposed methods.

Keywords: Proveance, Workflow Trace, Granularity, Workflow Trace Similarity

1 Introduction

Workflow provenance records the processes, process dependencies and data input and output of processes during a workflow execution. The provenance of a particular workflow execution is often referred as *workflow trace* [4]. Workflow trace can be seen as the meta-data of a particular workflow. It provides information on how a workflow is executed and how its results are derived from original data. Workflow provenance helps scientists/business to validate the processes involved in a workflow and examine its data quality.

Different users may have different needs on what to analyze and how detailed the provenance should be [2]. This refers to the research question of provenance granularity. Provenance granularity provides an abstraction of processes on different levels of details involved in a workflow. For example, a data scientist may want to view a detailed workflow provenance to analyze the performance of the workflow and improve the existing workflow. People at the managerial level

may want to view a workflow provenance from a coarse level to check if certain components are included in the workflow. Furthermore, it is often required by users to compare workflow traces to understand why one trace generates "better" results than another for advanced scientific discovery. Workflow trace similarity also has many other applications, such as clone detection, trustworthiness measurement and storage size reduction etc. Since workflow traces may be generated by heterogeneous systems and may have different abstraction level, the challenge lies in knowing how to compute the similarity of two traces if they are with different granularities.

There are some existing solutions to workflow trace multi-granularity representation and comparison. In these works, the granularities are determined by user-specific heuristics. In other words, different users could define different granularities given the same process trace. This leads to the problem that a specific multi-granularity workflow trace may not be understood by others and cannot be generalized for use with other applications. Most of the existing trace comparison methods model traces using graphs and their similarities are computed only based on their graph structural similarity. [8] propose a granularity transformation method using a workflow trace ontology and also developed similarity algorithms with semantics in mind. However, the semantic coverage during trace transformation is not considered.

In this paper, we investigate workflow traces' granularity problem and their similarity method. Our approach enables two workflow traces can be compared with any granularities. Specifically, the contributions of the paper are: (1) given a workflow trace, *Disperse* algorithm is developed to transform a trace with more abstract concepts into its detailed forms assisting by a workflow trace ontology; (2) a novel generic semantic similarity algorithm is proposed that not only considers the structural similarity but also the semantics coverage similarity embedded in traces; and (3) theoretical analysis is presented to compute the maximum and minimum semantic similarity. The rest of the paper is organized as follows. In Section 2, some existing works in the field of provenance modelling and provenance similarity analysis are reviewed. Section 3 introduces the definition of concept trace and its transformation methods. In Section 4, we propose the concept of semantic similarity of two workflow traces. A computation method and the analysis of semantic similarity are also presented. Section 5 evaluates the proposed techniques using some real workflow traces. Finally, the paper is concluded in Section 6.

2 Related Work

A provenance model is a representation of artefacts, processes and their relations involved in the information life cycle of data [8]. In recent years, with the maturity of Semantic Web technologies, several Semantic Web-based provenance models have proposed. Examples include Provenance Data Model (PROV) [7] and Open Provenance Model (OPM) [1]. Benefited from Semantic Web, these models can support data linkage and multi-granularity provenance generated

in heterogeneous environments. Provenance granularity has been studied in [1, 6, 8–10, 12]. Granularities are constructed from users' perspectives, and limited to specified application domains. Liu et al. modeled workflow trace granularities using *Workflow Trace Ontology* (WTO) [8]. WTO extends the *opmo:Process* class in OPMO [1] by defining sub-classes to describe different levels of semantics of executed processes [1]. In addition, an annotation property, *hasDepth*, is defined for all the classes in WTO to describe abstraction levels (depth) across granularities. Similar to the four-tier model in [6], the classes with small value of depth in WTO carry coarse semantic information. In this paper, we will use WTO to describe the multi-granularity workflow trace.

Provenance similarity analysis is another important problem in provenance research. Xie et al. presented a provenance compression algorithm to compress provenance graphs. It builds the provenance data into a name-identified reference list and the similarity of processes in the provenance is measured by the process's successors' similarity [11]. Chapman et al. developed a set of provenance factorization algorithms to reduce the provenance storage. The factorization algorithms are based on the pre-defined provenance node similarity functions, and two nodes in the provenance data are considered to be similar if they are specifically similar under the similarity function [5]. In [3], the authors used *graph edit distance* method to define the similarity between provenance graphs. One major limitation from edit distance is that it requires pre-defined cost functions for each elementary operations to calculate similarity, which is not generic and flexible. Liu et al. defined a similarity on the provenance data across different granularities [8]. They used the Maximum Common Subgraph (MCS) to compare two traces with the same depth. However, the semantic coverage among concept traces with different granularities are not captured using the traditional MCS algorithm.

3 Concept Trace Transformation

Workflow trace ontology is domain specific and normally defined by domain experts. In this paper, we construct a WTO for Montage[3] dataset as an example to explain the idea. In Figure 1, all vertices in ellipses represent executed processes (ontology instances) in a workflow; and rectangles represent high level abstractions of executed processes (ontology classes). For example, *mjpeg45* is an executed process which generates JPEG images from FITS files. Each class in WTO model has a property *hasDepth*, and the value of *hasDepth* denotes the actual level of a class in WTO. We use both *depth* and *conceptual abstraction level* interchangeably to describe a class level in WTO. The larger a depth is, the closer a class approaching to an instance level. On the other hand, the coarser a class's conceptual abstraction level is, the closer a class to the *Process*. The rest of this section presents the definition of concept trace and the algorithms for transforming a workflow trace into a concept trace with a required granularity.

[3] https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

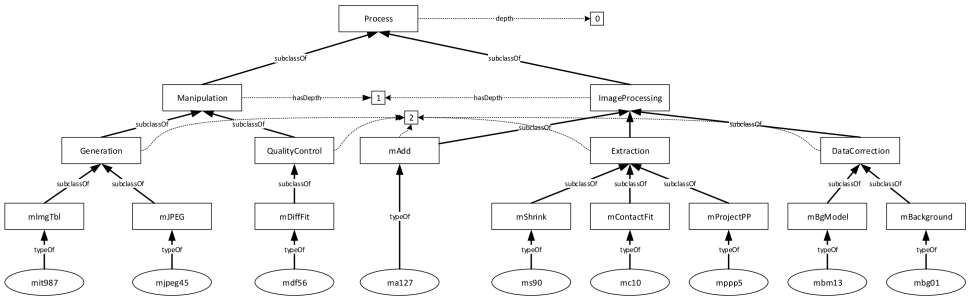

Figure 1: Montage Workflow Trace Ontology

3.1 Concept Trace Definition

As discussed in Section 2, a workflow trace can be modeled as a directed graph. Each vertex in a graph represents an executed process and each edge in a graph represents its data flow. Given a WTO, processes in a workflow trace can be mapped into different classes to present different levels of conceptual abstractions.

Definition 1 (Strict Concept Trace at Depth n). *Given a workflow trace p, a strict concept trace at depth n is a directed labelled graph $C_p^n = (V, E, L)$ where V is a set of vertices, each vertex representing a conceptual process v_i; $E \subseteq V \times V$ is a set of directed edges, each edge (v_i, v_j) representing data flow from v_i to v_j; L is a labelling function, each v_i has a label $L(v_i) \in \{WTO_{class}^n\}$, where WTO_{class}^n are the classes in WTO whose property "*`hasDepth`*" is $\leq n$.*

In some cases, an executed process can only be mapped to a class with depth $n' < n$. By the above definition, a concept trace with depth n represents a conceptual abstraction of a workflow trace using a defined granularity n. Then two concept traces having the same level of conceptual abstraction are comparable.

It is possible that an execution process could be mapped to a class in WTO with any depth. When mapping a workflow trace into a concept trace with depth n, if the depth of a mapped class is larger than n, it means that the current class is too detailed and a coarse abstraction is required. Therefore, we need to transform the class into its super-class with depth n. We call this procedure as a converge procedure. On the other hand, if the depth of a mapped class is smaller than n, it means that the current class is too abstract. We need to transform the class into its subclass in WTO with depth n. We call this procedure as a disperse procedure. If the depth of a mapped class is equal to n, no further operation is required. After all the labels have class labels with depth n, the last step is to merge adjacent vertices with the same labels into one vertex. This is because a concept trace represents a workflow trace on a conceptual abstraction view. Adjacent vertices having the same labels mean that the vertices carry the same conceptual function.

Algorithm 1: *Converge* (p,n,WTO)

Input : p is a workflow trace; n is the depth of concept trace which will be generated; WTO is the workflow trace ontology
Output : p's partial concept trace ${C'}_p^n$

```
1 while traversing p do
2 |   if the label of a visiting vertex is an instance in WTO then
3 |   |   replace the vertex label by its corresponding super-class at depth n in WTO;
4 |   end
5 end
6 C'^n_p ← p;
```

In summary, to transform a workflow trace into its concept trace with depth n, there are three steps involved: *Converge*, *Disperse* and *Merge*. Since the merge procedure is straight forward, next we present two algorithms that provide converge and disperse functions.

Converge Given a workflow trace and a target depth n, for every vertex with its label's depth is larger than n, we need to change its label using its super-class in WTO with depth n. The converge algorithm used in this paper is similar to that in [8]. For self contained purpose, it is presented in Algorithm 1. An example will be given at the end of this section.

Disperse *Disperse* procedure can be understood as the reverse of *converge* procedure. During *converge*, vertices with detailed conceptual abstraction are converted to the vertices with coarse conceptual abstraction. In other words, the vertices' labels are changed from sub-classes to their corresponding super-classes. Since every class has one and only one super-class with depth n, every p can be transformed to one and only one concept trace, ${C'}_p^n$, after converging process. On the contrast, *disperse* procedure is to disperse a class in WTO to its sub-classes. As one class may have more than one sub-class in WTO, a dispersed trace is not unique given a converged trace.

Algorithm 2 describes how to disperse a converged concept trace to its concept trace at depth n. Given a converged trace ${C'}_p^n$, a target depth n and a workflow trace ontology WTO, the algorithm produces a random number of concept traces of p at depth n. While traversing ${C'}_p^n$, if the depth of a vertex label v is less than the target depth n, it creates an empty set S_{source} and puts all the in-neighbors of v into it. It also creates an empty set S_{dest} to store all the out-neighbors of v (Lines 3-6). Then v is removed from ${C'}_p^n$ (Line 7) because it needs to be dispersed. To do that, the algorithm creates a temporal graph G' to maintain the dispersed vertex's structure. It randomly creates x new vertices V' with labels assigned by L' where L' can assign a subclasses with depth n of v's label to a vertex (Line 8-15). The algorithm partitions V' into l disjoint sets and connects the vertices in between each disjoint set (Lines 16-19). It ensures that G' is fully connected and acyclic. Next the elements in G' is treated as the dispersed structure of v and linked to the rest of V (Lines 20-26). As the last step of transformation, if adjacent vertices have the same labels, they are merged as one vertex. Combined with Algorithms 1 and 2, we can generate a strict concept trace with depth

n given a workflow trace with any granularity. Figure 2 shows an example of Converge and Disperse procedure.

Algorithm 2: *Disperse* (C'^{n}_{p}, n, WTO)

Input : a converged concept trace C'^{n}_{p}; n is the depth required; WTO is the workflow trace ontology
Output: p's concept trace C^{n}_{p}

while *traversing* C'^{n}_{p} **do**
  if *the label of a visiting vertex* v $hasDepth < n$ *in* WTO **then**
    Create empty sets S_{source} and S_{dest} ;
    foreach $(u,v) \in E$, $(v,w) \in E$ **do**
      $S_{source} \leftarrow u$; $S_{dest} \leftarrow w$;
    end
    Remove v from V;
    $x \leftarrow$ a random positive integer;
    $y = 0$;
    while $y \leq x$ **do**
      Create a new vertex v';
      Assign $L'(v')$ that $L'(v').hasDepth = n$ and v' is a sub-class of v;
      $V' \leftarrow v'$;
      $y \leftarrow y + 1$;
    end
    Randomly partition V' into l disjoint sets $V'_1, V'_2 ... V'_l$;
    for $i = 1..l-1$ **do**
      Add edges (v_1, v_2) where $v_1 \in V'_i$ and $v_2 \in V'_{i+1}$ to make a directed connected graph;
    end
    for $i = 1...S_{source}.size()$ **do**
      Add an edge (v_1, v_2) where $v_1 \in S_{source}$ and $v_2 \in V'$;
    end
    for $i = 1..S_{dest}.size()$ **do**
      Add an edge (v_1, v_2) where $v_1 \in V'$ and $v_2 \in S_{dest}$;
    end
    Update V and E;
  end
end

Example 1. Given $n = 2$ and a workflow trace in Figure 2a with depth listed on top of each vertex, *mShrink* and *mConcatFit* are converged and merged as *Extraction* in C^2_p by WTO in Figure 2b. Semantically the two processes provdes the same conceptual function *Extraction* at depth 2. In WTO, class *Manipulation* has two sub-classes *Generation* and *QualityControl* at depth 2. Therefore, *Manipulation* can be dispersed into different graphs, such as *Genation* $\rightarrow$ *QualitycControl* in Figure 2b or *Gneration* in Figure 2c. Both C^2_ps are valid.

4 Semantic Similarity of Workflow Traces

In this section, first, we define the similarity of two workflow traces. Then the concept of semantic coverage that captures the semantic movement during trace

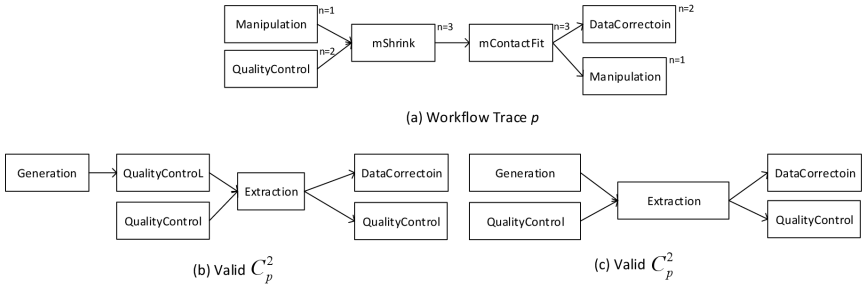

Figure 2: An Example of Concept Traces' Converge and Disperse Procedure

transformation as well as the concept of semantic similarity and its computation are introduced. This is followed by the analysis on how to compute the maximum / minimum semantic similarity.

4.1 Structure Similarity

As we mentioned before, workflow traces generated by heterogeneous systems may be represented using various granularities. By transforming two workflow traces into their strict concept traces with depth n. They have the same conceptual abstraction level. Therefore, their similarity at depth n can be computed by applying traditional graph similarity methods. In this paper, we use the *Maximum Common Sub-graph* (MCS) method to calculate the similarity (see Equations 1).

$$s(C_{p_1}^n, C_{p_2}^n) = \frac{|MCS(C_{p_1}^n, C_{p_2}^n)|}{|C_{p_1}^n| + |C_{p_2}^n| - |MCS(C_{p_1}^n, C_{p_2}^n)|} \tag{1}$$

In Equations 1, $|C_{p_1}^n|$ and $|C_{p_2}^n|$ are the sizes of two strict concept traces with depth n respectively and $|MCS(C_{p_1}^n, C_{p_2}^n)|$ is the size of maximal common subgraph between $C_{p_1}^n$ and $C_{p_2}^n$.

The larger the s is, the more similar the two traces are. Since we are more interested in the similarity of two workflow traces as a whole but not with a particular depth that is proposed in [8], the similarity of two workflow traces can be calculated by averaging the similarity of concept traces with all depths:

$$Similarity(p_1, p_2) = \frac{\sum_{n=1}^{depth(WTO)} s(C_{p_1}^n, C_{p_2}^n)}{depth(WTO)} \tag{2}$$

where $depth(WTO)$ is the maximum depth of WTO.

Strict concept traces at depth 0 does not contribute to the similarity because all workflow traces can be converged to *Process* at depth 0 that does not express any differences. The benefit of computing an overall similarity between two

workflow traces is that it enables users to understand the overall relationship between the two traces better. However, general graph similarity methods only evaluate graphs' structure similarity without considering the semantic abstraction that graphs represent. For concept traces, each vertex represents a high level conceptual abstraction of an executed processes. If we apply general graph similarity methods directly, it would ignore the semantic information involved in its concept trace. Next, we will explain what is the problem and how we approach it.

4.2 Semantic Similarity

In this sub-section, first we introduce the concept of Semantic Coverage. Then the Semantic Similarity (SS) is defined and its computation method are presented.

Semantic Coverage

As discussed, disperse procedure substitutes a vertex representing a class to a graph which is composed by its sub-classes defined in WTO. Since a vertex can be dispersed into many different graphs, we have shown that a strict concept trace with depth n may not be unique if a workflow trace contains vertices with depth smaller than n ($1 \leq n \leq depth(WTO)$).

A vertex with depth $< n$ can be dispersed into a graph with a large number ($\rightarrow \infty$) of vertices at depth n. While this is true as a graph, it is not practical in real world data and the structure of a workflow trace will also be destructed. On the other hand, disperse is a procedure of concretization. The semantic coverage of a vertex with depth n is not as complete as that of its ancestors. Therefore, we need to capture the movement of semantic coverage during converge and disperse procedure while not concerning the number of new vertices generated by disperse procedure. Formally, we define the semantic coverage as follows:

$$SC(g(v^n)) = \begin{cases} 1 & \text{if } depth(v) \geq n+1 \\ \frac{|L(g(v^n))|}{|WTO_v^n|} & otherwise \end{cases} \tag{3}$$

, where $g(v^n)$ is a graph generated by dispersing/coverging v to depth n, $|L(g(v^n))|$ is the number of concepts involved in graph $g(v^n)$ and $|WTO_v^n|$ is v's number of descendent concepts at depth n defined in WTO.

Since we do not care the actual graph structure, $g(v^n)$, dispersed by v, but the semantic coverage as discussed before, we use a virtual v to represent $g(v^n)$. At this point, a concept trace can be represented using a vertex-weighted graph in which a vertex weight describes $v's$ semantic coverage at depth n compared with that in its original workflow trace.

Example 2. Figure 3(a) is the concept trace with depth 2 by transforming Figure 2(a). The first *Manipulation* in its original workflow trace is dispersed into a graph *Generation* which cannot fully represent the concept *Manipulation*. Since in WTO, *Manipulation* has two children, the weight of virtual vertex *Manipulation* in $C_{p_1}^2$ is 1/2. However, in Figure 3(b), *Manipulation* is dispersed into a graph $Generation \rightarrow QualityControl$ and the number of concepts involved

is 2. Therefore, the weight for $Manipulation$ in $C^2_{p_2}$, representing subgraph $Generation \rightarrow QualityControl$, is $2/2 = 1$. Since $Extraction$ is generated by converging processes $mShink \rightarrow mContactFit$ with depth 3 which is ≥ 2, the weight of $Extraction$ after transformation is 1 as defined in Equation (3). Furthermore, $C^2_{p_1}$ and $C^2_{p_2}$ can be described as a vertex-weighted graphs that contain virtual vertex $Manipulation$ respectively but not including the actual structure graph such as $Generation \rightarrow QualityControl$ in $C^2_{p_2}$.

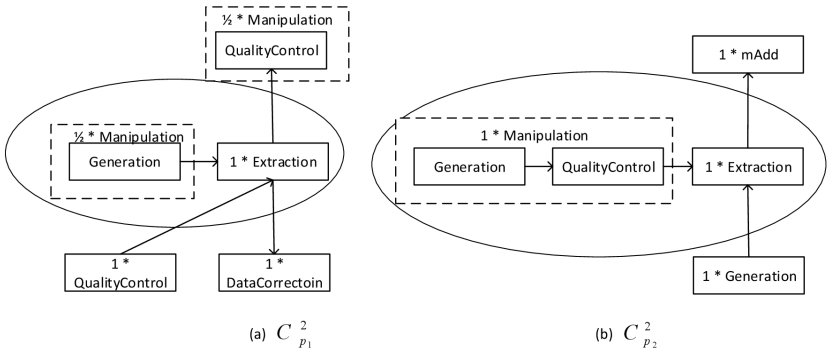

Figure 3: An Example of Weighed Concept Trace and Semantic Similarity

Semantic Similarity Definition

Based on Equation 3, the Semantic Similarity (SS) of two strict concept traces at depth n can be computed by computing the similarity of their corresponding vertex-weighted graphs. Formally, Equation 4 re-defines $|MCS|$ in Equation 1 by incorporating the concept of semantic coverage into it:

$$|MCS^s(C^n_{p_1}, C^n_{p_2})| = \sum_{v \in MCS(C^n_{p_1}, C^n_{p_2})} min(SC(g(v^n_{p_1})), SC(g(v^n_{p_2}))) \quad (4)$$

, where $min(SC(g(v^n_{p_1})), SC(g(v^n_{p_2})))$ returns $v's$ possible maximum common semantic coverage of two concept traces with depth n. Then the SS of two workflow traces can be defined as follows:

$$s^s(C^n_{p_1}, C^n_{p_2}) = \frac{|MCS^s(C^n_{p_1}, C^n_{p_2})|}{|C^n_{p_1}| + |C^n_{p_2}| - |MCS^s(C^n_{p_1}, C^n_{p_2})|} \quad (5)$$

From the above definition, it can be seen that the traditional MCS similarity, Equation (1), is a special case of SS in which the weight of every vertex always equals to 1. This is because that the semantic coverage is always assumed as 1 which may not be true for workflow traces with various granularities involved. The SS definition is able to capture the semantic similarity between two concept traces to be compared.

Example 3. Figure 3 shows two concept traces with depth 2. Their MCS (circled) is $1/2 * Manipulation \rightarrow 1 * Extraction$ by Equation (4). The semantic meaning behind this is that *Manipulation* is half semantically covered in MCS but *Extraction* is fully presented. Therefore, its $|MCS^s(C^2_{p_1}, C^2_{p_2})| = 1/2 + 1 = 1.5$.

There are two properties held by Equation 5:

Property 1. If $s^s(C^n_{p_1}, C^n_{p_2}) \neq 0$, where $1 \leq n \leq depth(WTO)$, then $s^s(C^{n'}_{p_1}, C^{n'}_{p_2}) \neq 0$ where $1 \leq n' < n$.

Property 2. If $s^s(C^n_{p_1}, C^n_{p_2}) = 0$, where $1 \leq n \leq depth(WTO)$, then $s^s(C^{n''}_{p_1}, C^{n''}_{p_2}) = 0$ where $n < n'' \leq depth(WTO)$.

By converge procedure, we can see that if two vertices $v_1^{n''}$ and $v_2^{n''}$ are isomorphically mapped to each other in $MCS^s(C^{n''}_{p_1}, C^{n''}_{p_2})$, their corresponding semantic-parent vertices v_1^n and v_2^n $(n < n'')$ must also be isomorphically mapped to each other in $MCS^s(C^n_{p_1}, C^n_{p_2})$ because their converged vertices' labels are the same. Therefore, the following proposition is held (the proof for the proposition is omitted due to space limitation).

Proposition 1. *If $MCS^s(C^{n''}_{p_1}, C^{n''}_{p_2}) \neq \varnothing$, for any vertex $v_1^{n''} \in C^{n''}_{p_1}$, $v_2^{n''} \in C^{n'}_{p_2}$ and $v_1^{n'}/v_2^{n'} \in MCS^s(C^{n''}_{p_1}, C^{n''}_{p_2})$, if $f(v_1^{n''}) \rightarrow v_2^{n'}$, there must have $f(v_1^n) \rightarrow v_2^n$ where v_1^n / $v_2^n \in C^n_{p_1}$ / $C^n_{p_2}$ respectively, v_1^n / v_2^n is the super-class of $v_1^{n''}$ / $v_2^{n''}$ respectively, $1 \leq n < n''$. that is isomorphically mapped to $v_2^n \in C^n_{p_2}$, the converged vertex at depth n' of v_1^n where $n' < n$ must also isomorphically map to the converged vertex of v_2^n in $MCS(C^{n'}_{p_1}, C^{n'}_{p_2})$.*

Here $f(x) \rightarrow y$ represents the corresponding mapping between x and y.

Semantic Similarity Computation

To compute the semantic similarity of two workflow traces p_1 and p_2, first, p_1 and p_2 are transformed to $C^1_{p_1}$ and $C^1_{p_2}$, respectively. The reason we do depth 1 transformation is for concept trace with depth 1, every vertex has the minimum depth. It implies that there is no disperse procedure required during transformation. Therefore, p_1 and p_2 can be transformed to one and only one $C^1_{p_1}$ and $C^1_{P_2}$ respectively. Then $MCS^s(C^1_{p_1}, C^1_{p_2})$ is computed. If $MCS^s(C^1_{p_1}, C^1_{p_2}) = \varnothing$, $s^s(C^n_{p_1}, C^n_{p_2}) = 0$. By Property 2, we can conclude that $Similarity^s(p_1, p_2) = 0$. If $MCS^s(C^1_{p_1}, C^1_{p_2}) \neq \varnothing$, mappings $f(v^1_{p_1}) \rightarrow v^1_{p_2}$ $(v^1_{p_1} \in C^1_{p_1},\ v^1_{p_2} \in C^1_{p_2},\ v^1_{p_1}$ / $v^1_{p_2} \in MCS^s(C^1_{p_1}, C^1_{p_2}))$ can be obtained.

The above step is important because it provides mapping $f()$ between vertices of p_1 and p_2 that contribute to MCS with no ambiguity. The mapping will guide the MCS computation at depth 2. By Propostion 1, only vertices in p_1 and p_2 that contribute to $MCS^s(C^1_{p_1}, C^1_{p_2})$ may contribute to $MCS^s(C^2_{p_1}, C^2_{p_2})$. Therefore, we only need to check those vertices' transformation to compute $MCS^s(C^2_{p_1}, C^2_{p_2})$. We will show an example later on to explain the idea. Similarly, the semantic similarity at depth n can be computed by using the MCS mapping at $n-1$ (see Equation 6).

$$Similarity^s(p_1,p_2) = \frac{\sum_{n=1}^{depth(WTO)} s^s(C^n_{p_1},C^n_{p_2})}{depth(WTO)} \quad (6)$$

By identifying vertices' semantic coverages of two strict concept traces with the same depth, we are able to compare two workflow traces semantically with any granularities. Since there is a lot of possible strict concept traces with depth n that can be generated given a workflow trace, if we just compare two random generated concept traces with depth n, their similarity, $s^s(C^n_{p_1},C^n_{p_2})$, is arbitrary and therefore, the overall similarity, $Similarity^s(p_1,p_2)$, is also arbitrary that may not make sense for users. In the next sub-section, we will analyse SS and discuss how to compute the maximum and minimum SS of two workflow traces with any granularities that may provide users a better understanding of the relationship between two workflow traces.

4.3 Semantic Similarity Analysis

The maximum/minimum $Similarity^s(p_1,p_2)$ can be achieved if $s^s(C^n_{p_1},C^n_{p_2})$ is maximized/minimized for $\forall n \in [1..depth(WTO)]$. It is clear that to find the maximum/minimum $s^s(C^n_{p_1},C^n_{p_2})$, we need to maximize/minimize $|MCS^s(C^n_{p_1},C^n_{p_2})|$ and minimize/maximize $|C^n_{p_1}|$ and $|C^n_{p_2}|$ respectively.

Given two workflow traces p_1, p_2 and a depth n, our goal is to generate their corresponding concept traces $C^n_{p_1}$ and $C^n_{p_2}$ that $s(C^n_{p_1},C^n_{p_2})$ is maximized. By Propsition 1, in converge procedure, a vertex $v^{n''}$ $(n'' \geq n)$ can only be transformed to one and only one vertex v^n. However, a vertex $v^{n'}$ $(n' \leq n)$ may have many representations after disperse procedure. Therefore, the key is to control the disperse procedure to achieve the maximum similarity. Next we analyze all the possible cases of how vertices v $(v \in p_k,\ k \in [1,2])$ can be transformed and demonstrate how to control the transformation procedure to reach the maximum similarity.

In the SS computation algorithm, if $MCS^s(C^1_{p_1},C^1_{p_2}) \neq \varnothing$, mappings $f(v^1_{p_1}) \rightarrow v^1_{p_2}$ $(v^1_{p_1} \in C^1_{p_1},\ v^1_{p_2} \in C^1_{p_2},\ v^1_{p_1}\ /\ v^1_{p_2} \in MCS^s(C^1_{p_1},C^1_{p_2}))$ can be obtained. For each $v^1_{p_k}$ $(k \in [1,2])$, we use $ref(v^1_{p_k})$ to represent the corresponding vertices in original workflow trace p_k that contribute to $v^1_{p_k}$. At this point, we can generalize the problem as: Given p_1, p_2, $C^{n-1}_{p_1}$, $C^{n-1}_{p_2}$, $ref(v^{n-1}_{p_k})$ and the mapping $f(v^{n-1}_{p_1}) \rightarrow v^{n-1}_{p_2}$ $(n \geq 2)$, how to generate $C^n_{p_1}$ and $C^n_{p_2}$ so that $s^s(C^n_{p_1},C^n_{p_2})$ is maximized. To transform p_k to $C^n_{p_k}$, each $v_k \in p_k$ must sit in one of the following cases and for each case, a rule is defined with principles of maximizing the $MCS^s(C^n_{p_1},C^n_{p_2})$ and minimizing their concept trace sizes.

- Case 1: for any vertex $v_k \in p_k$ not contributing to $MCS^s(C^{n-1}_{p_1},C^{n-1}_{p_2})$.
  - Case 1.1: If $depth(v_k) = n$, v_k is not changed;
  - Case 1.2: If $depth(v_k) > n$, by Proposition (1), we can use v_k's super-class with depth n to replace v_k;

- Case 1.3: If $depth(v_k) < n$, by Equation 5, we need to disperse v_k by any one of its sub-classes with depth n in WTO to minimize the size of $C^n_{p_k}$ that leads to maximize $s^s(C^n_{p_1}, C^n_{p_2})$;

– Case 2: vertices in p_k that contribute to $MCS(C^{n-1}_{p_1}, C^{n-1}_{p_2})$. This case is the most complex case since sometimes we need to consider both p_1 and p_2 during transformation to generate their concept traces with depth n for the maximum similarity. Specifically, given $f(v^{n-1}_{p_1}) \rightarrow v^{n-1}_{p_2}$ ($v^{n-1}_{p_1} \in C^{n-1}_{p_1}$, $v^{n-1}_{p_2} \in C^{n-1}_{p_2}$, $v^{n-1}_{p_1}$ / $v^{n-1}_{p_2} \in MCS(C^{n-1}_{p_1}, C^{n-1}_{p_2})$), we need to study the relationship between $ref(v^{n-1}_{p_1})$ and $ref(v^{n-1}_{p_2})$. There are 4 sub-cases involved.
  - Case 2.1: If $\forall v_i \in ref(v^{n-1}_{p_1})$, $\forall v_j \in ref(v^{n-1}_{p_2})$ and $depth(v_i/v_j) = n$, nothing needs to be changed since all vertices are proper for $C^n_{p_k}$;
  - Case 2.2: If $\forall v_i \in ref(v^{n-1}_{p_1})$, $\forall v_j \in ref(v^{n-1}_{p_2})$ and $depth(v_i/v_j) > n$, converge procedure applied and use v_k's super-class with depth n to replace v_k;
  - Case 2.3: If $\forall v_i \in ref(v^{n-1}_{p_1})$, $\forall v_j \in ref(v^{n-1}_{p_2})$, $depth(v_i) < n$ and $depth(v_j) > n$, it means v_i needs to be dispersed. To reach the maximum similarity, the way $g(v_i)$ dispersed by v_i must be the same as $g(v_j)$ converged by v_j. By this means, they can contribute to $MCS^s(C^n_{p_1}, C^n_{p_2})$ as they did for $MCS^s(C^{n-1}_{p_1}, C^{n-1}_{p_2})$;
  - Case 2.4: If $\forall v_i \in ref(v^{n-1}_{p_1})$, $\forall v_j \in ref(v^{n-1}_{p_2})$ and $depth(v_i/v_j) < n$, then both v_i and v_j have to be dispersed. In Equation 4, if both the semantic coverage $SC(g(v^n_i)$ and $SC(g(v^n_j)$ are maximized as 1, their contribution to $|MCS^s(C^n_{p_1}, C^n_{p_2})|$ is maximized.

By the above transformation rules, the maximum SS can be achieved. Similarly, the minimum SS can be computed and will not be discussed due to space limitation. Figure 4 shows an example.

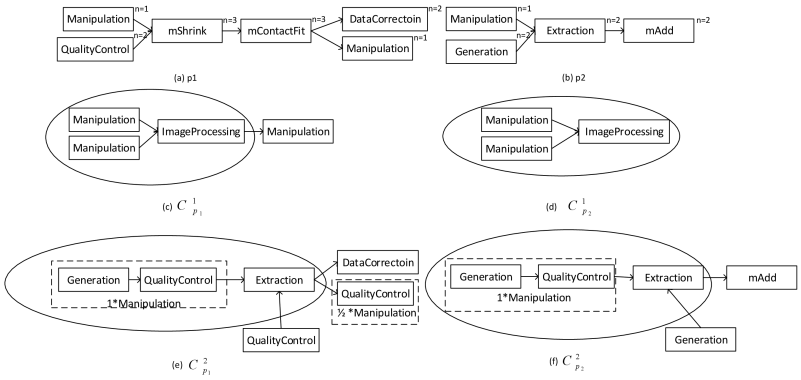

Figure 4: An example of Maximum Similarity

Example 4. Figure 4a and 4b show two workflow traces in which vertices presented have various granularities. Figure 4c and 4d are their corresponding concept

traces at depth 1. Since there is no disperse procedure involved, its maximum similarity is computed by applying Equation (1) directly. The $MCS^s(C^1_{p_1}, C^1_{p_2})$ is circled. To transform two traces into depth 2, the transformation case 1 is applied to all vertices in p_1 and p_2 that are not contributing to in the circle of Figure 4c and 4d. For $ImageProcessing$ in the circle, since the $ref(ImageProcessing^1_{p_1})$ is $mShrink \rightarrow mContactFit$ and $ref(ImageProcessing^1_{p_2})$ is $Extraction$, Case 2.2 and Case 2.1 are applied respectively. This generates the $Extraction$ vertex in Figure 4e and 4f that maximizes their contribution to $MCS^s(C^2_{p_1}, C^2_{p_2})$ (circled). But for the $Manipulation$ vertex in the Figure 4c and 4d circle, $ref(Manipulation^1_{p_1})$ is $Manipulation$ which is the same as $ref(Manipulation^1_{p_2})$. Therefore, we have to reach the maximum semantic coverage by applying case 2.4. That is why the weights of both virtual vertices in Figure 4e and 4f are 1. Through this transformation, the maximum $S^s(C^2_{p_1}, C^2_{p_2})$ can be achieved.

5 Experiments

Some experiments have been conducted to evaluated the proposed method. In the experiments, we compare the proposed method with the traditional structure similarity method using Equation (2) and demonstrate the effectiveness of our approach.

5.1 Datasets

The Montage dataset created by NASA/IPAC stitches together multiple input images to create custom mosaics of the sky. We take the workflow in the Montage dataset as our base workflow traces and design the corresponding WTO. To imitate heterogenous situation, the base workflow traces are transformed into the traces that have various granularities using the following steps: 1. Transform base workflow traces to strict concept traces with depth $depth(WTO)$, $C_p^{depth(WTO)}$; 2. Generate x induced sub-graphs of $C_p^{depth(WTO)}$, and for each induced sub-graph G including (a) Randomly select x vertices in G where $|x| < |G|$; (b) For each selected vertex v, randomly generate a number d where $d < depth(WTO)$; and (c) Replace v's label to v's super-class at depth d, then merge v's neighbors if necessary.

5.2 Maximum Semantic Similarity of Workflow Traces

In the experiments, we apply the proposed workflow similarity method to analyse the maximum semantic similarity of randomly selected workflow traces. Figure 5a and 5b show two workflow traces random selected. Note that in Figure 5, we use dotted rectangle to present virtual vertices and for vertices contributing to MCS are underlined.

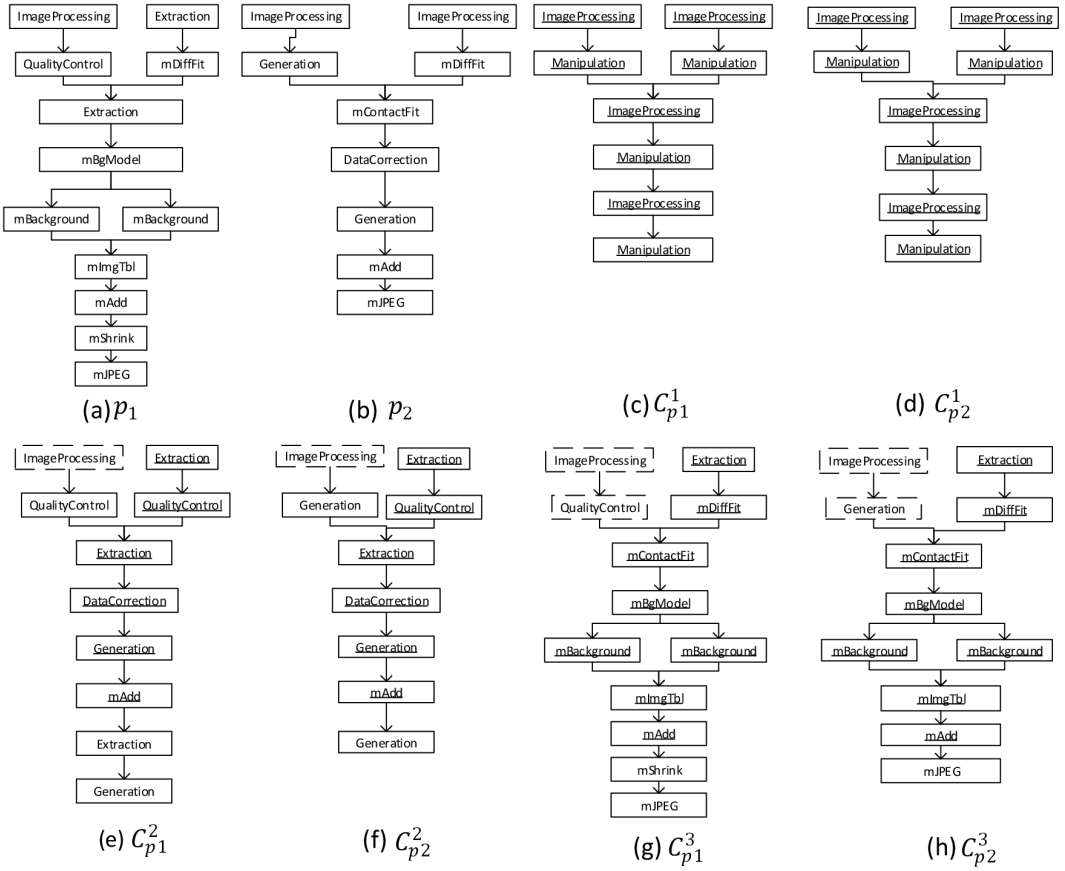

Figure 5: Concept Traces Evaluation

To calculate the similarity of workflow traces in Figure 5(a-b), the first step is to transform p_1 and p_2 to their strict concept traces at depth 1. $C^1_{p_1}$ and $C^1_{p_2}$ (Figure 5(c-d)) are isomorphic, so the semantic similarity of $C^1_{p_1}$ and $C^1_{p_2}$ is 1.

At depth 2, in order to maximize the similarity, *ImageProcessing* in p_2 is dispersed to *Extraction* to match the *Extraction* in p_1, *mBgModel* and *mBackground* in p_1 are converged to *DataCorrection*, the other vertices are converged to their depth 2 concepts respectively. Note that *ImageProcessing* in p_1 does not contribute to MCS at depth 2, so it is kept as a virtual vertex. The maximum semantic similarity of $C^2_{p_1}$ and $C^2_{p_2}$ is $\frac{6}{10+9-6} = \frac{6}{13} \approx 0.465$. Similarly, the concept trace of $C^3_{p_1}$ and $C^3_{p_2}$ which can produce the maximum semantic similarity are shown in Figure 5 (g-h). The similarity of $C^3_{p_1}$ and $C^3_{p_2}$ is $\frac{8}{12+11-8} = \frac{8}{15} \approx 0.533$. Therefore, the similarity of workflow trace p_1 and p_2 is $\frac{1+0.465+0.533}{3} \approx 0.667$.

As comparison, if it uses the original graph similarity method (Equation 2) to measure the similarity of p_1 and p_2, the similarity is $\frac{1}{12+9-1} = 0.05$ because

the MCS of p_1 and p_2 has only one vertex. The experiment shows that the proposed semantic similarity concept and its computation methods are able to identify the semantic similarity embeded in traces to be compared even they may have very different structures. This approach enable users to have a much better understanding on the conceptual level among various workflow traces.

6 Conclusion

In this paper, we propose the *Disperse* algorithmto that is able to transform a workflow trace with any granularity into a concept trace with required depth. To capture the similarity of conceptual abstraction between two workflow traces, the semantic similarity concept that not only considers the structure similarity but also the semantic coverage during transformation is proposed. The maximum / minimum semantic similarity is analysed and its computation method is also presented. Our similarity method is able to capture the semantic information embedded in the workflow traces and it provides a better solution on how to evaluate workflow traces with various granularities.

References

1. The open provenance model core specification (v1. 1)
2. Allen, M.D., Seligman, L., Blaustein, B., Chapman, A.: Provenance capture and use: A practocal guide. MITRE Corporation (2010)
3. Bao, Z., Cohen-Boulakia, S., Davidson, S.B., Eyal, A., Khanna, S.: Differencing provenance in scientific workflows. In: IEEE 25th International Conference on Data Engineering. pp. 808–819 (2009)
4. Bowers, S.: Scientific workflow, provenance, and data modeling challenges and approaches. Journal on Data Semantics 1(1), 19–30 (2012)
5. Chapman, A.P., Jagadish, H.V., Ramanan, P.: Efficient provenance storage. In: the 2008 ACM SIGMOD international conference on Management of data. pp. 993–1006 (2008)
6. Gotz, D., Zhou, M.X.: Characterizing users' visual analytic activity for insight provenance. Information Visualization 8(1), 42–55 (2009)
7. Groth, P., Moreau, L.: Prov overview. W3C Working Draft, 11th December (2012)
8. Liu, Q., et al.: Towards semantic comparison of multi-granularity process traces. Knowledge-Based Systems 52, 91–106 (2013)
9. Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., Silva, C.: Tackling the provenance challenge one layer at a time. Concurrency and Computation: Practice and Experience 20(5), 473–483 (2008)
10. Stephan, E.G., Halter, T.D., Ermold, B.D.: Leveraging the open provenance model as a multi-tier model for global climate research. In: Provenance and Annotation of Data and Processes, pp. 34–41. Springer (2010)
11. Xie, Y., et al.: Compressing provenance graphs. In: TaPP (2011)
12. Zhao, J., Wroe, C., Goble, C., Stevens, R., Quan, D., Greenwood, M.: Using semantic web technologies for representing e-science provenance. In: The Semantic Web–ISWC 2004, pp. 92–106. Springer (2004)