A Method to Mine Workflows from Provenance for Assisting Scientific Workflow Composition

Reng Zeng and Xudong He
Computing and Information Sciences
Florida International University
Miami, Florida 33199, USA
Email: {rzeng001,hex}@cis.fiu.edu

W.M.P. van der Aalst
Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: w.m.p.v.daalst@tue.nl

Abstract

Scientific workflows have recently emerged as a new paradigm for representing and managing complex distributed scientific computations and are used to accelerate the pace of scientific discovery. In many disciplines, individual workflows are large and complicated due to the large quantities of data used. As such, the workflow construction is difficult or even impossible when relevant domain knowledge is missing or the workflows require collaboration within multiple domains. Recent efforts from scientific workflow community aiming at large-scale capturing of provenance present a new opportunity for using provenance to provide recommendation during building scientific workflows. This paper presents a method based on provenance to mine models for scientific workflows, including data and control dependency. The mining result can either suggest part of others’ workflows for consideration, or make familiar part of workflow easily accessible, thus provide recommendation support for scientific workflow composition.

1. Introduction

Scientific computing has entered a new era of large scaled sharing provided by the cyberinfrastructure. Scientific workflows have recently emerged as a new paradigm for declarative representation of scientific applications as complex compositions of software components and the dataflow among them [1]. Current scientific workflow management systems, such as Pegasus [2], Kepler [3], Taverna [4], VisTrails [5] facilitate the definition and execution of scientific workflows. However, it is often very hard to create and maintain scientific workflows. In many disciplines, individual workflows are large due to the large quantities of data used. In [6] three stages in the creation of workflows are suggested. The first stage is to create a workflow template, which specifies the high-level structure of a workflow including executables and dataflow. The second stage is to create workflow instances, which specify what concrete data are used in the computation. Workflow instances are independent of execution resources. The third stage is to create an executable workflow, which specifies the hosts where computation will occur, and the appropriate data movements across distributed locations. These three stages enable the management of the complexity of workflow creation by making the process more modular, but creating scientific workflows is a challenge to domain scientists, and updating scientific workflows is also a challenge as the workflow templates can keep evolving when they are used by domain scientists.

Recent efforts from scientific workflow community [5] [7] [8] [9] [10] [11] aiming at large-scale capturing of provenance present a new opportunity for using provenance to provide recommendation during creating or updating scientific workflows. Provenance, in scientific workflow community, refers to the sources of information, including entities and processes, involved in producing or delivering an artifact. Provenance is important for scientists to assess data quality, validate results, reproduce experiments, consequently provenance capture becomes an important scientific workflow research area. Many existing scientific workflow management systems, such as Taverna, Kepler, VisTrails and Pegasus, capture provenance information implicitly in an event log that records events related to the start and end of particular steps in the workflow.
execution and the corresponding data read and write events. Provenance is also argued in [10] as first class data in the cloud. Originally motivated by the needs in the scientific domain, the VisTrails provenance technology [5] and infrastructure are general and applicable to a wide range of applications that involve complex computational processes, including non-workflow systems that, called in the sequel system-level monitoring systems, allow users to leverage provenance using the same applications and environments that they are used to.

Based on provenance of a combination of system-level monitoring and workflow-based systems, this paper aims at providing a general method to mine workflows from provenance. Several researchers have investigated how to synthesize a process model from event logs [12] [13] [14]. The research area of process mining focuses on extracting information about processes by examining event logs. Practical experience has shown that typical information recorded in event logs includes information about which activities are performed, at what time, by whom and in the context of which case (i.e., process instance) [12]. By explicitly using the case context, process discovery algorithms are capable of constructing process models that accurately describe the process [13]. Since both event logs and provenance contain process information, and a given scientific workflow may be executed multiple times [15] thus creating multiple workflow execution instances, the process discovery algorithms can be used to mine a workflow from provenance. However, the process discovery algorithms ignore the data dependency while scientific workflows are data centric. This paper presents a method based on provenance to mine models for scientific workflows, including data and control dependency. The mining result can either suggest part of others’ workflows for consideration, or make familiar part of workflow easily accessible. Existing works [16] [17] [18] on providing recommendation support during scientific workflows composition use only data dependency. Our work provides a new direction in using captured provenance with a method using both control dependency and data dependency.

2. Process Mining of Provenance

The goal of this paper is to find a scientific workflow model from provenance and to provide recommendation support during scientific workflows composition based on the mined workflows, as shown in Figure 1.

Figure 1. Background of the method described in this paper (denoted by solid arrows)

2.1. What are Scientific Workflows and Provenance?

There are many works on scientific workflows and provenance, that use different terminology for scientific workflows and different ways to organize provenance [19][20][7][4]. The common basics of scientific workflow and provenance this paper relies on are given as follows.

A task is a procedure or a group of procedures to execute computational activities. A data product can be a single data object or a collection of data objects. A scientific workflow is a directed graph where nodes are tasks and edges between nodes represent either data dependency or control dependency. Provenance records the task invocations and data products used or generated by each invocation. Formally, Provenance ⊆ P(Data × Task × Data). Data dependency is the relationship between two tasks $t_1$ and $t_2$ when $t_2$ needs $t_1$’s output as input, denoted as $t_1 \prec_d t_2$. Formally, $t_1 \prec_d t_2$ iff $\exists d_1, d_2, d_3 \in Data$ . $(d_1, t_1, d_2) \in$ Provenance $\land (d_2, t_2, d_3) \in$ Provenance. Data dependency can be derived from provenance as causality relation pairs, such as $t_1 \prec_c t_2$. Control dependency is the relationship between two tasks $t_1$ and $t_2$ when a task $t_1$ is required to be invoked before invoking another task $t_2$, it is denoted as a causality relation pair $t_1 \prec_c t_2$. A causality relation pair infers data dependency or control dependency, denoted as $t_1 \prec t_2$. A task trace, corresponds to a run of a scientific workflow, is a sequence of task invocations. Formally, let $\Sigma$ be the set of all tasks that appear in task traces, a task trace is a sequence of task invocations, denoted as $t_1, t_2, t_3, \ldots, t_m$ where $t_i \in \Sigma$ for $1 \leq i \leq m$.

2.2. Scientific Workflow Models in Petri Nets

The algorithm in this paper mines a Petri net as a model to represent a scientific workflow. Tasks
are modeled by transitions and causal relations are modeled by places and arcs. A place corresponds to a condition which can be used as pre-condition and/or post-condition for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ingoing arcs.

This paper uses WF-nets [12] that is based on Place/Transition nets, a variant of the classic Petri net model.

**Definition 1:** Place/Transition nets
A Place/Transition net, or simply a P/T-net, is a tuple \((P, T, F, M)\) where
1) \(P\) is a finite set of places,
2) \(T\) is a finite set of transitions such that \(P \cap T = \emptyset\), and
3) \(F \subseteq (P \times T) \cup (T \times P)\) is a set of directed arcs.
4) \(M : P \rightarrow \mathbb{N}\) is a function that associates each place with a natural number.

A place \(p\) is an input place of a transition \(t\), also called pre-condition, if there is a directed arc from the place \(p\) to the transition \(t\), i.e. \((p, t) \in F\). Similarly a place \(p\) is an output place of a transition \(t\), also called post-condition, if \((t, p) \in F\).

**Definition 2:** Workflow nets
Let \(N = (P, T, F, M)\) be a P/T-net and \(t'\) be a transition such that \(t' \notin P \cup T\), \(N\) is a workflow net (WF-net) iff:
1) Object creation: \(P\) contains an input place \(i\) such that \(i \in P\), and
2) Object completion: \(P\) contains an output place \(o\) such that \(o \in P\),
3) Connectedness: \(N' = (P, T, \{t'\}, F \cup \{(o, t'), (t', i)\})\) is strongly connected.

### 2.3. A Simple Example

To illustrate the principle of the algorithm in this paper, we consider the task trace extracted from provenance shown in Table 1. Suppose the workflow that generated the provenance is given in Figure 2. Consider the fact that scientific workflows evolve quickly thus the change is implicitly recorded in provenance, and provenance capturing systems support non-workflow environment, the changing workflows behind the provenance are often unknown before mining. Since the control flow does not generate any data product, we cannot get the control flow from provenance. This paper aims at mining the control dependency from provenance to provide recommendation support during workflow composition.

![Figure 2. A sample workflow (the circle arrow denotes control dependency, and the other arrows denote data dependency)](image)

<table>
<thead>
<tr>
<th>Workflow Running Identifier</th>
<th>Task Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>1</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>e</td>
</tr>
</tbody>
</table>

#### 2.4. Construction of a Causality Table

**Definition 3:** Direct precedence table
For \(n \) tasks, the direct precedence table is a \(n \times n\) matrix \(P, P = [p_{ij}]\) where \(1 \leq i, j \leq n\) and \(p_{ij}\) is the number of times that task \(t_i\) directly precede task \(t_j\).

Using the example above, a direct precedence table is shown in Table 2.

**Definition 4:** Indirect precedence table
For \(n\) tasks, the indirect precedence table is a \(n \times n\) matrix \(S, S = [s_{ij}]\) where \(1 \leq i, j \leq n\) and \(s_{ij}\) is calculated as follows. For task \(t_i\) and \(t_j\), in each workflow running, if there is a sequence \(t_i, t_k, \ldots, t_m, t_j\), suppose the number of tasks from \(t_k\) to \(t_m\) is \(m - k + 1\), add \(\delta^{m-k+1}\) (\(\delta = 0.8\)) to \(s_{ij}\). 0.8 is chosen after experimentation which satisfies two requirements: 1) for direct precedence, \(\delta^{m-k+1} = 1\); 2) The longer distance, the smaller addition.

Using the example above, an indirect precedence table is shown in Table 3.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
According to the definition, data dependency can be derived from provenance as causality relation pairs. For the example above, it is $a \prec_d e, a \prec_d c, a \prec_d b, c \prec_d d$. Following is the construction of weight table combining both precedence tables and data dependency.

**Definition 5:** Weight Table
For $n$ tasks, the weight table is a $n \times n$ matrix $W$, $W = [w_{ij}]$ where $1 \leq i, j \leq n$ and $w_{ij}$ is calculated as follows. First, $w_{ij} = p_{ij} + s_{ij}$; second, if $i \prec_d j$ is present in causality relation pairs derived from provenance as data dependencies, add $\sigma$ to $w_{ij}$ where $\sigma$ is the number of workflow running.

Using the example above, $\sigma = 2$, the weight table is shown in Table 4.

**Definition 6:** Confidence Table
For $n$ tasks, the confidence table is a $n \times n$ matrix $C$, $C = [c_{ij}]$ where $1 \leq i, j \leq n$ and $c_{ij}$ is calculated as follows: $c_{ij} = w_{ij} - w_{ji}$.

Using the example above, the confidence table is shown in Table 5.

The causality table is shown in Table 6. For each pair $t_1 \prec t_2$, if it is not data dependency $t_1 \prec_d t_2$, then it is control dependency $t_1 \prec_c t_2$.

Rules are designed as below to update causality table:

1) For tasks $t_k, t_m$ and its causality $t_k \prec t_m$, if its weight is lower than 1, remove it.
2) For tasks $t_k, t_m$ and its causality $t_k \prec t_m$, if there is $t_k \prec \ldots \prec t_m$, in which each pair has higher confidence than the one of $t_k \prec t_m$, remove $t_k \prec t_m$ from causality table.

Firstly, for each valid causality, there are many chances to get higher than 1, such as direct precedence, data dependency, or indirect precedence (e.g. $0.8^2 + 0.8^4 = 1.0496 > 1$). For two independent tasks ($i$ and $j$) executing concurrently, it does not result in a causality dependency, because chances are one starts before the other in a random fashion, thus $w_{ij}$ and $w_{ji}$ are close enough so that $c_{ij} = w_{ij} - w_{ji}$ should be less than 1. Secondly, for $t_k \prec \ldots \prec t_m$, it is highly possible that $t_k \prec t_m$ get higher than 1 for multiple indirect precedences, but actually there is no direct causality between $t_k$ and $t_m$.

Using the steps and rules above for the example, valid causality pairs are derived: $a \prec b, b \prec c, c \prec d, a \prec c, a \prec c$, a Petri net can be constructed as Figure 3, which matches exactly the original workflow.

The construction of a causality table is summarized in Algorithm 1.

---

**Table 3. Indirect precedence table**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0.8</td>
<td>1.44</td>
<td>1.152</td>
<td>0.512</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.6</td>
<td>0.64</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.64</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 4. Weight table**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>3.8</td>
<td>3.44</td>
<td>1.152</td>
<td>3.512</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1.6</td>
<td>0.64</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>1</td>
<td>0.8</td>
<td>0.64</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 5. Confidence table**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>5.8</td>
<td>3.44</td>
<td>1.152</td>
<td>3.512</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1.6</td>
<td>-0.36</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.36</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0.36</td>
<td>0</td>
<td>-0.36</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 6. Causality table**

<table>
<thead>
<tr>
<th>Causality Relation Pair</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \prec b$</td>
<td>3.8</td>
</tr>
<tr>
<td>$b \prec c$</td>
<td>2</td>
</tr>
<tr>
<td>$c \prec d$</td>
<td>4</td>
</tr>
<tr>
<td>$d \prec e$</td>
<td>0.36</td>
</tr>
<tr>
<td>$a \prec e$</td>
<td>3.512</td>
</tr>
<tr>
<td>$e \prec b$</td>
<td>0.36</td>
</tr>
<tr>
<td>$a \prec c$</td>
<td>3.44</td>
</tr>
<tr>
<td>$a \prec d$</td>
<td>1.152</td>
</tr>
<tr>
<td>$b \prec d$</td>
<td>1.6</td>
</tr>
<tr>
<td>$c \prec d$</td>
<td>-0.36</td>
</tr>
</tbody>
</table>

![Figure 3. A resulting Petri net (all causality pairs are included, and an AND-split is used for task a)](image-url)
Algorithm 1 Construction of a Causality Table

Input: Provenance for σ running of a workflow that contains n tasks
Output: A causality table T

1: for all workflow running instances Run do
2: for all (t_i, t_j) t_i directly precedes t_j do
3: p_{ij}++
4: end for
5: for all (t_i, t_j) t_i indirectly precedes t_j do
6: Assume the sequence as t_i, t_k, ..., t_m, t_j
7: δ = 0.8
8: s_{ij}+= δ^{n-k+1}
9: end for
10: end for
11: for i = 1 to n do
12: for j = 1 to n do
13: w_{ij} = p_{ij} + s_{ij}
14: if ∃d_1, d_2, d_3 ∈ Data_4(d_1, t_i, d_2) ∈ Run \cap (d_2, t_j, d_3) \in Run then
15: w_{ij}+==σ
16: end if
17: end for
18: end for
19: for i = 1 to n do
20: for j = 1 to n do
21: c_{ij} = w_{ij} - w_{ji}
22: if c_{ij} ≥ 1 then
23: T = T \cup \{(t_i \prec t_j, c_{ij})\}
24: end if
25: end for
26: end for
27: for all (t_i \prec t_j, c_{ij}) ∈ T do
28: if \exists t_k < ... < t_m \cdot (t_i \prec t_k) \cap (t_m \prec t_j) \cap (c_{ik} > c_{ij}) \cap (\ldots) \cap (c_{mj} > c_{ij}) then
29: T = T \backslash \{(t_i \prec t_j, c_{ij})\}
30: end if
31: end for

2.5. Generating a Petri Net from a Causality Table

It is straightforward to derive a causality graph from a causality table, but it requires additional information to generate a Petri net from a causality table, including parallelism and choice of tasks. In Petri nets, parallelism can be represented with an AND-split, and choice can be represented with a OR-split. For instance, if there are causality pairs t_i < t_j and t_i < t_k, the type of a split from t_j to t_j AND/OR t_k has to be detected, to generate a Petri net. The principle of detection is to employ the weight table above to check the pattern of t_j and t_k: 1) w_{jk} = 0 and w_{kj} = 0, that shows the pattern t_jt_k or t_kt_j cannot appear, it is a OR-split; 2) Otherwise, that shows the pattern t_jt_k or t_kt_j can appear, it is an AND-split. The algorithm to detect the type of a split is given in Algorithm 2. It is assumed that a OR-split is placed after an AND-split, i.e. it is conjunctions of clauses, and each clause is a disjunction of tasks.

Algorithm 2 Detection of the type of the splits

Input: Weight table w, task t_0 and tasks t_1, ...t_i, ...,t_n in which t_0 ≺ t_i (1 ≤ i ≤ n) is a causality pair
Output: A set of clauses Disj in which each is a set of tasks that are in the OR-relation

A set of tasks Conj in which each is in the AND-relation

1: Derive W′ from W with the rows and columns related with t_1, ...t_i, ...,t_n
2: Let W′ = [w_{ij}]; Disj = ∅; Conj = ∅
3: Create n empty sets: Set_i where 1 ≤ i ≤ n
4: for i = 1 to n do
5: for j = 1 to n do
6: if w_{ij} = 0 & w_{ji} = 0 & i ≠ j then
7: Set_i = Set_i ∪ \{i, j\}
8: end if
9: end for
10: end for
11: for i = 1 to n do
12: for j = 1 to n do
13: if Set_i is empty then
14: Conj = Conj ∪ \{t_i\}
15: else
16: Disj = Disj ∪ \{< Set_i >\}
17: end if
18: end for

2.6. Providing Recommendation for Scientific Workflow Composition

Given a partial workflow, based on a set of causality tables and a set of Petri nets, this paper provides a method to recommend a next most likely task and related part of a Petri net.

A causality path is a sequence of tasks t_1, ...t_i, ...t_n in which t_i < t_{i+1} (1 ≤ i < n) are causality pairs. The length of causality path is n.

For the current task t_0 selected in the partial workflow, a set of possible next tasks can be easily found by looking up the set of causality tables as \{p_i \mid t_0 ≺ p_i\} and 1 ≤ i ≤ m, where m is the number of found tasks. A method is given in Algorithm 3 to provide recommendation. Firstly, the method gives an indicator
on each causality table how it matches the given partial workflow. Secondly, for each possible next task, the method gets a recommendation rate by two factors: the weight of the corresponding causality pairs and the indicator of match level. Finally, the method gives recommendation confidence \( Conf_i \) for each possible next task:

\[
Conf_i = \frac{rate_i}{\sum_{j=1}^{m} rate_j}
\]

where \( rate_i \) is the recommendation rate for each possible next task.

**Algorithm 3 Providing Recommendation**

**Input:** A causality path that end at current task \( t_0 \) in the partial workflow \( t_n, ..., t_1, t_0 \);

a set of causality tables \( \{T_j \mid 1 \leq j \leq k \} \)

**Output:** A set of possible next tasks with recommendation rates \( R \)

1. Let the set of possible next tasks be \( \{p_i \mid t_0 \prec t_i \text{ and } 1 \leq i \leq m \} \)
2. for \( j = 1 \) to \( k \) do
3. \( \text{match}_j = 1 \)
4. for \( i = 1 \) to \( n \) do
5. if \( \exists (t_x \prec t_y, w_{xy}) \in T_j, t_x = t_{i-1} \land t_y = t_i \text{ then} \)
6. \( \text{match}_j++ \)
7. end if
8. end for
9. end for
10. for \( i = 1 \) to \( m \) do
11. \( rate_i = 0 \)
12. for \( j = 1 \) to \( k \) do
13. if \( \exists (t_0 \prec t_i, w_i) \in T_j \text{ then} \)
14. \( \text{rate}_i+=w_i \times \text{match}_j \)
15. end if
16. end for
17. end for
18. for \( i = 1 \) to \( m \) do
19. \( R = R \cup \{(p_i, \frac{\text{rate}_i}{\sum_{j=0}^{m} \text{rate}_j})\} \)
20. end for

**3. Evaluation**

The method described in this paper is evaluated using a Java program for the accuracy of recommendation. The provenance being used in this section are generated with a real scientific workflow from the open scientific workflow repository myExperiment, that is the challenge workflow from the third Provenance Challenge (http://www.myexperiment.org/workflows/750).

To evaluate the accuracy of recommendation, the method is applied to each task of workflows. For each task \( t_i \), there are \( n \) dependent tasks, and there are \( p \) possible next tasks with recommendation confidence. \( n \) tasks are picked up from the set of possible next tasks with highest confidence if available, in which there are \( m \) tasks matched with one of \( n \) dependent tasks, that is \( n \) hits out of \( n \) real ones. And, it is also \( m \) hits out of \( p \) recommendations. The accuracy of recommendation for each task is defined as:

\[
\text{accuracy}_i = \frac{m^2}{n \times p}
\]

Figure 4 compares the method described in this paper to the methods using only control dependency or only data dependency for recommendation. The \( \alpha \) algorithm only mine control dependency while most recommendation algorithms uses only data dependency, this paper combines both control dependency and data dependency to improve the recommendation accuracy. As shown in Figure 4, our method performs better than the method that mines only control dependency, because the data dependency is utilized in the algorithm to assist mining control dependency; and our method performs better than the method that uses only data dependency except in task 3, because for some tasks that only have data dependencies, our method may give false control dependency, thus lower down the accuracy.

**4. Related Works**

The \( \alpha \) algorithm [12] assumes completeness of event logs, this paper proposes an algorithm based on the \( \alpha \) algorithm to use data dependency improving the mining result. There are also a number of process mining algorithms implemented to mine incomplete event logs, such as fuzzy miner, heuristic miner. The fuzzy miner [14] assumes that problems in mining large scale processes are caused by mismatch between fundamental assumptions of traditional process mining, and the characteristics of real-life processes. Fuzzy miner developed an adaptive simplification and visualization technique for process models, which is based on two metrics, significance and correlation. The two metrics are similar to the concept of data clustering domain where a binary distance metric is inferred to find related subsets of attributes. In the context of scientific workflows, significance, which can be determined both for tasks and precedent relations over them, measures the relative importance of behavior. As such, it specifies the level of interest we have
in tasks and their control dependency. Correlation is only relevant for precedence relations over tasks, which measures how closely related two events following one another are. The heuristics miner is a practical applicable mining algorithm that can deal with noise, and can be used to express the main behavior (i.e. not all details and exceptions) registered in an event log [21]. It includes three steps: (1) the construction of the dependency graph, (2) for each activity, the construction of the input and output expressions and (3) the search for long distance dependency relations. All those miners do not utilize data dependency, the algorithm of this paper can be applied to any of them enhancing the mining result.

This paper is related to workflow recommendation papers based on provenance. Besides the difference in recommendation technique with other papers, this paper has a unique advantage that it can build a whole workflow model for general reference. The work in [16], based on large scale databases of workflow execution traces, proposes exploiting these databases with a “knowledge light” approach to reuse, applying case based reasoning (CBR) methods to those traces to support scientists’ workflow generation process in two phases. The first phase is retrieving from a database the entries for all workflows containing any one of the current tasks, the second phase is similarity assessment based on the ranking by the size of the largest mapping produced between current tasks and retrieved cases.

This paper uses a different approach to do recommendation for workflow generation, which has two advantages compared with [16]: this paper does not use expensive graph matching algorithms, thus is more efficient; and this paper can make recommendation on both data dependency and control dependency while [16] only considers data dependency in their analysis. FlowRecommender [17] makes recommendation based on the path in partial workflow, instead of last node in partial workflow. Provenance are synthetically generated from a set of nodes, as a set of node sequences. If there is a path, that has 5 possible following nodes, each of 5 nodes then has 20% confidence, it would be difficult to determine the threshold. [22] also investigate on the method to suggest next possible tasks, however, they use stored provenance information recorded during past design sessions, and their method is based on static attribute such as data input of a task. This paper uses runtime provenance which enable utilizing existing workflow repositories, and computes causality between tasks including control dependency. The work in [18] uses a framework for service oriented scientific workflow reuse, its recommendation is based on searching a collection of workflows with the help of annotation. They first collect scientific workflows from centralized repositories such as myExperiment, then integrate annotations generated from various heterogeneous data sources such as author annotations at different levels (for example, workflow, service, or data channels), user comments at runtime, best practices, and statistical data of existing scientific workflows and services, including popularity and usage patterns. They also support manual annotation. With the collected workflows and integrated annotations, they use Apache Lucene, an open source search engine, to index the information in collection and associated annotations. Their method can provide relevant information, but cannot suggest a confidence level of each recommendation. The work in [23] shows that clustering can be effective to organize large collections of workflows, but it cannot make recommendation for an individual task. However, it can be applied to address an issue of the method in this paper by clustering a collection of workflows. The tasks having the same task identifier may belong to different scientific domains, but they are processed as the same task in the method of this paper. Clustering can help to restrict the recommendation in a single scientific domain.

There is also a related work in data mining area that focuses on pairwise temporal patterns [24]. They state the problem of mining event relationships as: given event sequence, finding all pairwise statistically dependent patterns that can be characterized as tem-
poral patterns, that assert dependency between events and specify the timing information, such as “event a happens after event b, say, about 5 minutes”. Their result is in fact the precedence table in this paper. Since this paper focuses on scientific workflow area, provenance provides workflow running identifier for each event so that it is obvious to get the precedence table, which is the pairwise event dependency in [24]. Combined with algorithms in [24], the method in this paper can be applied to unstructured data, or semi-structured data, such as computer system log files.

5. Conclusion

This paper presents a method based on provenance to mine models for scientific workflows, including data and control dependency. The mining result can either suggest part of others’ workflows for consideration, or make familiar part of workflow easily accessible, thus provide recommendation support for scientific workflows composition, which offers a new approach to build workflows in the context of scientific workflows. Given the fact that provenance captured in any scientific workflow based systems or system level monitoring systems contains information about tasks and their temporal order, the proposed algorithm can give both control and data dependency for recommendation during scientific workflows composition. The method provided in this paper can be applied to any scientific workflow management systems.

Acknowledgments

This work was partially supported by the NSF of U.S. under award OISE-0730065, HRD-0833093, and Presidential Fellowship of Florida International University.

References


