FlowRecommender: A Workflow Recommendation Technique for Process Provenance

Ji Zhang[1], Qing Liu[2] and Kai Xu[2]
[1] Department of Mathematics and Computing,
University of Southern Queensland, Australia
Email: *ji.zhang@usq.edu.au*
[2]CSIRO ICT Centre, Hobart, TAS, Australia
Email: {*q.liu, kai.xu@csiro.au*}

Abstract

The increasingly complicated workflow systems necessitates the development of automated workflow recommendation techniques, which are able to not only speed up the workflow construction process, but also reduce the errors that are possibly made. The existing workflow recommendation systems are quite limited in that they cannot produce a correct recommendation of the next node if the upstream nodes/sub-paths that determine the occurrence of this node are not immediately connected with it. To solve this drawback, we propose in this paper a new workflow recommendation technique, called *FlowRecommender*. FlowRecommender features a more robust exploration capability to identify the upstream dependency patterns that are essential to the accuracy of workflow recommendation. These patterns are properly register offline to ensure a highly efficient online workflow recommendation. The experimental results confirm the promising effectiveness and efficiency of FlowRecommender.

1 Introduction

In recent years, workflow systems are becoming more and more complicated as a result of a fast growing number of scientific processes available. Scientific workflows are based on the automation of scientific processes in which scientific programs are associated, based on data and control dependencies [1]. These scientific processes, mostly taking the form of Web services, could either be local or remote scientific tools or programs that can be shared by scientists from a common domain. However, the construction of most workflows are based on some pre-determined templates and relevant domain knowledge plays a crucial role in creating these templates. As such, the workflow construction is difficult or even impossible when domain knowledge is missing or the workflows are to be constructed by amateurs in the field. Workflow recommendation based on provenance turns out to be a possible and promising approach in the case when no templates are available.

Provenance of workflows is a practice to archive historical workflows that have been executed and, sometimes, also the intermediate and final results generated by the workflow processes. A number of provenance systems and techniques have been proposed [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. The provenance of workflows is of considerable value to scientists. From it, one can ascertain the quality of the data based on its ancestral data and derivations, track back sources of errors, allow automated reenactment of derivations to update a data, and provide attribution of data sources [4]. Recent work has also shown that provenance information (the metadata required for reproducibility) can be used to simplify the process of pipeline creation [5]. Tools for assisting automatic construction of workflows are increasingly desirable to facilitate the construction of complicated workflows. An effective and efficient workflow recommendation technique is useful in these tools. First, it can speed up the workflow construction process by reducing the development time. Second, it can provide a guidance for choosing the mostly likely node and, therefore, minimize the errors that are possibly made in the workflow construction.

An important observation in workflow construction practice is that, in most cases, the prediction of a downstream node is only dependent on one of its adjacent upstream sub-paths in the workflow. Here, the adjacent subpath is not necessarily continuous nor immediately connected the node. The influence exerted by the remote upstream sub-paths becomes negligible when the distance between the node and the upstream sub-path increases. The existing work perform recommendation either based on the last node only [1] or the continuous paths that ends in the last node in the current workflow [2]. They cannot perform recommendation based on the paths that are not continuous nor does not terminate in the last node of the current workflow. For example, suppose we need to produce the recommendation of the next node for a partial workflow $c \rightarrow a \rightarrow b$. The method in [1] provides prediction based on node b only while the method in [2] provides prediction based on one of the two continuous sub-paths that end at node b: $c \rightarrow a \rightarrow b$ and $a \rightarrow b$. These two methods will fail to provide correct recommendation if the next node is actually decided by other sub-paths such as c, a, $c \rightarrow a$ or $c \rightarrow \ldots \rightarrow b$.

To solve the drawbacks of the existing work, we propose a new workflow recommendation technique based on workflow provenance, called *FlowRecommender*. FlowRecommender provides a more effective yet efficient means for producing the prediction by investigating the correlation of each possible workflow node with respect to its adjacent upstream paths. More specifically, FlowRecommender takes two main stages for performing workflow recommendation: the offline stage and the online stage. In the offline stage, FlowRecommender extracts the patterns of nodes that will appear in the workflows. The patterns are called the *influencing upstream sub-paths* of the nodes that determine the occurrence of these nodes in the workflows. The extracted patterns are registered into the so-called *pattern table* for the subsequent recommendation. In the online stage (when recommendation is required), the pattern table is scanned to match the patterns with the current partial workflow under construction. The node is recommended if its influencing upstream sub-path matches the partial workflow. Compared with the existing methods, FlowRecommender is advantageous in that it features a stronger capability to identify the influencing upstream sub-paths, leading to a better recommendation performance.

The reminder of this paper is organized as follows.

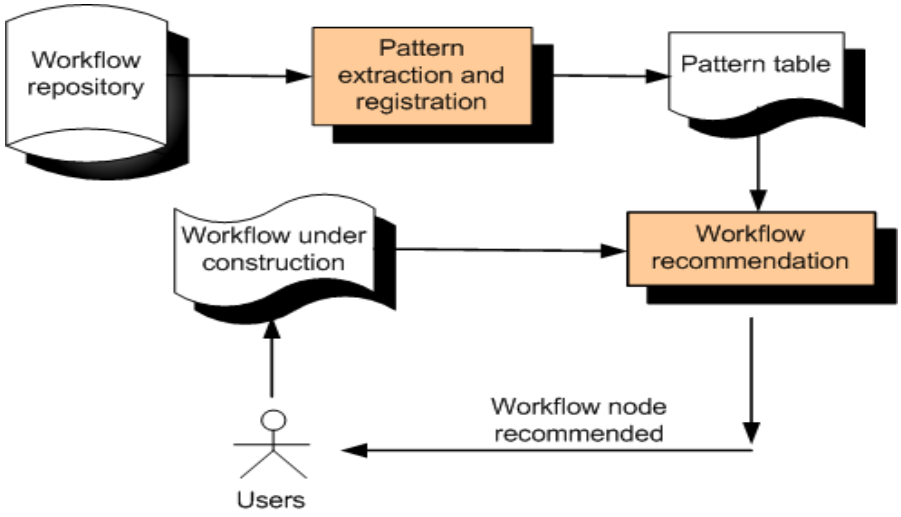

Figure 1: The system architecture of FlowRecommender

Section 2 presents an overview of FlowRecommender and its system architecture. In Section 3, greater details are given on the two major modules of FlowRecommender, i.e. pattern extraction and registration, and the workflow recommendation. The experimental results are reported in Section 4. The final section concludes the whole paper and presents some future research directions.

2 An Overview of FlowRecommender

In this section, we will present an overview of FlowRecommender for workflow recommendation based on workflow provenance. The system architecture of FlowRecommender is presented in Figure 1. Generally, there are two modules involving in FlowRecommender, i.e., pattern extraction and registration and workflow recommendation. The first two modules are performed offline while the last module is conducted online during the construction of workflows.

- **Pattern extraction and registration**. The patterns of the candidate nodes are extracted from the workflows in the provenance. Here, the candidate nodes are those tools/programs that can be utilized to extend/complete the workflow in the recommendation, and the pattern for each candidate node refers to its influencing upstream sub-path that determines the occurrence of this node. Such pattern is identified when the correlation (measured by *confidence*) between the sub-path and the node is sufficiently strong. The discovered patterns are registered into the *pattern table*, making FlowRecommder ready for the subsequent workflow recommendation module;
- **Workflow recommendation**. During workflow construction, workflow recommendation module tries to match the influencing upstream sub-paths of the candidate nodes against the current workflow under construction. The nodes are recommended to users once its influencing upstream sub-path matches the current workflow.

From the system architecture of FlowRecommender as shown in Figure 1, we can see that, in the workflow construction process, a close cycle is formed among the following components: the workflow currently under construction, the workflow recommendation generation module and the end human users. The workflow currently under construction serves the input to the recommendation generation module. Based on the current status of the workflow, the recommendation generation module tries to provide recommendation as to which node should be selected to extend/complete the workflow. The recommendation results are fed to the users who will decide whether the recommendation is followed. The users' decision will lead to the extension of the worklow. This cycle continues until the workflow has been constructed to the point such that the desired task has been fulfilled.

The major constituting modules of FlowRecommnder are discussed in details in the following subsections.

2.1 Pattern Extraction and Registration

In this section, we will discuss in details how to extract patterns from provenance that are useful to the workflow recommendation. These patterns serves as a sort of signatures to activate the recommendation of certain nodes to extend/complete the partial workflows.

2.1.1 Pattern Extraction

The patterns of the candidate nodes are extracted from the provenance. The candidate nodes are those nodes that can be potentially used to extend/complete the partial workflows under construction. The patterns are the influencing upstream sub-paths that determine the occurrence of nodes in the workflows.

Definition 2.1: Candidate Node Set for all the workflows. The Candidate Node Set with respect to all the workflows in the provenance, denoted as $CNS(D)$, is the set of nodes that can be potentially recommended in various locations of workflows. It is defined as the set of nodes that have appeared in the workflows but do not only appear in the start position of the workflows.

Definition 2.2: Upstream sub-paths. The upstream sub-paths of a node v in a workflow w is defined as the sequences of ordered nodes that appear before v in w. For example, in a workflow $b \rightarrow a \rightarrow c$, the set of upstream sub-paths for node c are $\{b, a, b \rightarrow a\}$.

We evaluate the correlation of a node and its upstream sub-paths through the measure of *confidence*. Confidence of a node v given a upstream sub-path p is the probability that v appears given that p has already appeared in the workflow. It is defined as

$$Conf(v, p) = \frac{freq(v, p)}{freq(p)}$$

where $freq(v, p)$ and $freq(p)$ correspond to the frequency/count that v and p occur together and p occurs alone in the workflow, respectively.

Unlike association rules, only confidence is leveraged in our work to measure the significance of the patterns extracted, instead of using both support and confidence. In workflow domain, it is likely that some workflows are executed in a quite low frequency, but their constituting nodes and/or paths feature strong correlations with other. If support measure is used, then it may lead to many low-frequency workflows being screened out and the recommendations based on these workflows becomes impossible.

Definition 2.3: Influencing upstream sub-path. For a node v, the influencing upstream sub-path p in a workflow is defined as the sub-path that satisfies that the confidence of v given p is no less than a given confidence threshold σ_{conf}, as follows:

$$Conf(v, p) \geq \sigma_{conf}$$

The technical challenge lies in extracting the influencing upstream sub-path for a node v is that we are not able to accurately pinpoint the location and order of the influencing upstream sub-path of v. The concept of the location of an influencing upstream sub-path p in a workflow is relative to the end (last node) of this workflow. This is what we call the backward location of a sub-path within a workflow, which is defined as follows:

Definition 2.4: Backward location of an influencing upstream sub-path. The backward location of an influencing upstream sub-path p within a workflow is defined as the distance (i.e., the number of edges) between the first node of p and the last node of w, i.e.

$$Location(p) = Dist(start(p), end(w)), p \in w$$

To solve the difficulty in pinpointing the location and order of the influencing upstream path of given nodes, we devise a technique to do this in a *progressive* fashion. For a given node, the technique first evaluates its confidence with respect to its upstream sub-paths consisting of nodes with the smallest overall distance (based on the location of the sub-path within the workflow as defined in Definition 2.4) to ensure that the more adjacent upstream paths are evaluated first, followed by the more remote ones. In other words, we evaluate the sub-paths with a distance of 1, 2, ... For the sub-paths with the same distance from the end of the workflows, we will first evaluate those with a smaller order. In the case of a tie of the order of constituting nodes in the paths, the algorithm will randomly choose a path for evaluation. This design is consistent with the rationale that the influencing upstream path of a candidate node is relatively close to the location of the candidate node in the workflow.

As the algorithm will potentially evaluate all the possible upstream sub-paths, thus the total number of such sub-paths could be large especially when the sub-path is far from the node in the workflow. To prevent an explosion of the possible upstream paths, a parameter k ($k \geq 1$) will be used to specify the maximum backward location for the sub-paths to be evaluated. In other words, k will determine the extent to which the upstream back-track will be performed to find the influencing upstream sub-paths for the given candidate node. For a candidate node, the upstream sub-path exploration is continued until either of the following conditions is met:

1. The influencing upstream sub-path of this node is found; or
2. All the sub-paths with a backward location not exceeding k have been evaluated.

The algorithm for finding the influencing upstream sub-paths for all the candidate nodes is presented in Figure 2. To speed up the pattern extraction, particularly the calculation of confidence, we leverage *inverted indexing* of workflows, which speeds up the search of the workflows where a given node appears. This can significantly reduce the number of workflows to be evaluated for calculating confidence. The inverted indexing based on the candidate nodes are first performed in order to streamline the subsequent confidence calculation. $CNS(D)$ is cloned to $SetOfPendingNodes$ which will dynamically updated in the algorithm to track the set of nodes whose patterns have not yet been found. The FOR loop in Line 3 controls the order of the sub-paths the algorithm will evaluate, increasing from 1 through k. The algorithm will continue when not all the candidate nodes have been evaluated. Once the influencing upstream sub-path for the node has been identified, the node and its pattern will be registered into the pattern able and the algorithm will start to process the next candidate node. Pattern registration in pattern table will be discussed in the next subsection. The BREAK clauses in Line 10 enables the algorithm to be terminated early the moment when the influencing upstream path has been identified with respect to each candidate node.

2.1.2 Pattern Registration

When they have been extracted, the influencing upstream sub-paths of the nodes in Candidate Node Set for the provenance will be registered into the pattern table. Next, we will present the definition of pattern table.

Definition 2.4. Pattern Table. The pattern table is an $n \times 2$ table, where x_{i1} is the node that possibly appears in the workflows (i.e., $x_{i1} \in CNS(D)$) and x_{i2} is the corresponding influencing upstream sub-path of the node given in x_{i1}, where $1 \leq i \leq n$.

An influencing upstream sub-path is represented as an ordered sequence of nodes in the pattern table. Each node in the sequence is associated with the location information represented by its distance to the candidate node given in the filed of x_{i1}. The order of this sequence of nodes is consistent with the order they appear in the workflows from where they are extracted, but they do not necessarily appear consecutively. The pattern table is pre-constructed before recommendation is performed.

An example of pattern table is given in Table 1. Suppose that this table is derived from a repository of workflows involving a total of 7 nodes (labeled as a, b, c, d, e, f and g) and there are, however, only 3 nodes (i.e., c, d and e) whose influencing upstream sub-paths are identified given a certain confidence threshold level: the 2-order sub-path $a(3) \rightarrow b(1)$ is identified for node c and 1-order sub-path $c(2)$ and $g(1)$ are found for nodes d and e, respectively. The influencing upstream sub-path of node c, i.e., $a(3) \rightarrow b(1)$, means that node c is recommended when the current workflow under construction contains a path that takes the form of $* \rightarrow a \rightarrow ? \rightarrow b$, where the wildcard symbol asterisk(*) represents a sub-path with any possible sequence of nodes while the question-mark(?) represents a single node.

2.2 Workflow Recommendation

The workflow recommendation that our method provides is offered in a stepwise fashion; the systems automatically recommends the next most likely node to choose in order to extend/complete the current workflow that is under construction. The users can exert to activate/inactivate workflow recommendation anytime in the construction process, providing users with a great flexibility to choose construction with or without automatic workflow recommendation.

Definition 2.5: Candidate Node Set for a workflow.

Algorithm: findInfluencingSubPath(D, k)
Input: The whole workflow repository D and the limit k for upstream back-tracking for identifying patterns.
Output: The influencing upstream sub-paths of nodes in $CNS(D)$.
1. Perform inverted indexing based on the nodes in $CNS(D)$;
2. $SetOfPendingNodes \leftarrow CNS(D)$;
3. FOR $i = 0$ to $k - 1$ DO
4. FOR each node v in $SetOfPendingNodes$ DO
5. FOR each sub-path p of backward location of i in the workflows w of $index(v)$, starting from the smallest order, DO {
6. $Conf(v, p) \leftarrow computeConfidence(v, p, D)$;
7. IF $Conf(v, p) \geq \sigma$ THEN {
8. Register p for v in the pattern table;
9. Remove v from $SetOfPendingNodes$;
10. BREAK; }}

Figure 2: The algorithm for finding the influencing upstream sub-paths for candidate nodes

Candidate node label	Influencing Upstream Sub-path
c	a(3)→ b(1)
d	c(2)
e	g(1)

Table 1: A sample pattern table

The Candidate Node Set for a workflow w, denoted as $CNS(w)$, is the set of nodes that can be potentially recommended to extend/complete an incomplete workflow w that has been constructed. It is defined as the set of nodes that satisfy the I/O constraints w.r.t w. That is, the input data type of the node in $CNS(w)$ matches the output data type of the last node of w. Obviously, we have $CNS(w) \subseteq CNS(D)$, and $CNS(w)$ may change when w is constructed at different stages.

The moment when the recommendation is required to extend or complete a workflow w, we need to go through evaluating the influencing upstream sub-path of each node in $CNS(w)$, which have been stored in the pattern table, to see whether they match the current workflow under construction. To perform pattern matching, we need to first define the distance between a partial workflow w and an influencing upstream sub-path p of a candidate node. Specifically, such distance, denoted as $Dist(w, p)$, is defined as the normalized sum of the location difference between the same pair of nodes in w and p as

$$Dist(w, p) = \frac{\sum Dist(n_i^w, n_j^p)}{Order(p) \cdot Order(w)}, n_i^w \in w, n_j^p \in p$$

where n_i^w and n_j^p represent the same node in w and p with (probably) different locations within w and p, $1 \leq i \leq |w|$ and $1 \leq j \leq |p|$.

Based on the above definition, we know that $0 \leq Dist(w, p) < 1$. We have $Dist(w, p) = +\infty$ if w does not have the same sequence of nodes appearing in p for a candidate node. This is to ensure that the partial workflow w and its matched influencing upstream sub-path p have the same sequence of nodes, though these nodes may have (slightly) different locations within the workflow and sub-path.

A distance threshold, denoted as σ_d, needs to be specified to determine whether the partial workflow w matches the influencing upstream sub-path p of a candidate node. That is, if $Dist(w, p) \leq \sigma_d$ then we say that w matches p and does not otherwise. σ_d is a parameter providing flexibility for controlling the accuracy/fuzziness in pattern matching. The larger σ_d is, the less accurate (more fuzzy) the matching will be, and vice versa.

If the patterns are matched for more than one candidate downstream nodes, then the recommendation can be presented in a *probabilistic* manner. Specifically, suppose $matchedCNS(w)$ is the set of matched candidate nodes that satisfies that

$$matchedCNS(w) \subseteq CNS(w)$$

and

$$\forall v \in matchedCNS(w), Dist(w, p) \leq \sigma_d$$

where p is the influencing upstream sub-path of node v. Each node in $matchedCNS(w)$ will be recommended with a probability to indicate the strength that this node is recommended. The probability is quantified proportionally based on the confidence level, i.e.,

$$Strength(v, w) = \frac{Conf(v, w)}{\sum_i Conf(v_i, w)}$$

where $v_i \in matchedCNS(w)$. $strength(v_i, w)$ satisfies that $0 < strength(v_i, w) \leq 1$ and $\sum_i strength(v_i, w) = 1$.

If no influencing upstream sub-path can be matched against the partial workflow under construction for any candidate node, then only the nodes that satisfy the I/O interface of the workflow will be recommended (i.e., the output datatype of the last node of the workflow matches the input datatype of the node to be recommended), each with the same strength of $\frac{1}{|CSN(w)|}$, where $|CNS(w)|$ denotes the number of nodes in $CNS(w)$.

The algorithm for the recommendation generation is presented in Figure 3. Two sets, $SetOfRecommendedNodes$ and $SetOfStength$, are used to record the set of nodes whose patterns matched the workflow and their respective recommendation strength, respectively. These two sets are initialized as empty sets at the beginning (Step 1 and 2). The pattern table is then scanned to identify those nodes whose patterns match the partial workflow and these nodes are stored in $SetOfRecommendedNodes$ (Step 3-5). Their strength in the recommendation is calculated and stored in $SetOfStength$ based on their confidence level. Finally, the recommendation is presented by returning $SetOfRecommendedNodes$ and $SetOfStength$ to users.

3 Experimental Evaluation

In this section, we present experimental evaluation of the our workflow recommendation technique. Three major

Algorithm: RecommendationGeneration(w)
Input: A partial workflow w.
Output: The set of nodes recommended for w and their respective strength.
1. $SetOfRecommendedNodes \leftarrow \emptyset$;
2. $SetOfStength \leftarrow \emptyset$;
3. FOR each node v registered in the pattern table DO
4. IF $Dist(w, p) \leq \sigma_d$, where p is the pattern of v, THEN
5. $SetOfRecommendedNodes \leftarrow \cup v$;
6. FOR each node $v \in SetOfRecommendedNodes$ DO
7. $SetOfStength \leftarrow \cup \frac{Conf(v,w)}{\sum_i Conf(v_i,w)}$, where $v_i \in SetOfRecommendedNodes$;
8. Output $SetOfRecommendedNodes$ and $SetOfStength$;

Figure 3: The algorithm for producing workflow recommendation

sets of experiments are carried out, evaluating the accuracy of recommendation, scalability towards large provenance, and sensitivity to the major parameters. The program is developed in C++ and all the experiments are conducted in Windows Vista 2.26GHz system with a main memory of 2G.

The workflow provenance that we will use in the experiments are generated synthetically. To render the workflow repository generated as being close to the real-life application scenarios as possible, four major aspects of design are carefully considered in the designing process: a) What is the total number of workflows in the provenance (denoted as $N_{provenance}$)? b) What are the nodes that will appear in the workflow provenance (here, the total number of nodes that will appear in the provenance is denoted as N_{nodes})? c) What is the length of each workflow? and d) What is the order of nodes appearing in each workflow?

Both $N_{provenance}$ and N_{node} can be easily specified as positive integers. Once N_{node} is specified, the generator automatically generates a set of nodes as $P_1, P_2, \ldots, P_{N_{node}}$. A maximum length of workflows, denoted as l_{max}, is specified and the length of each workflow is a random integer variable in the range of $[1, l_{max}]$.

To decide the order of nodes appearing in workflows, a set of matrices are constructed to decide the transitional probability for each pair of nodes. Specifically, each entry $x_{a,b}$ in the matrix $\mathcal{M}(i)$ corresponds to the transitional probability of node b given node a that is of a distance of i before b. Here, i is an integer and, without losing generality, we set it in the range of $[1, 3]$, meaning that the occurrence of a particular node in the workflows we generate is depended on a preceding node that is of a distance not exceeding than 3. Certainly, one may specify i as another valid value. Each workflow is initialized using a node randomly chosen from the set of nodes. When each subsequent node needs to be generated in the workflow, a matrix is randomly chosen from the matrix set (which contains three matrices) and the new node is generated based on the transitional probability presented in this matrix. This design ensures that occurrence of a node within workflows is not only determined only by its immediately preceding node, but also some more remote non-connected nodes. The workflow grows in this way until its specified length is reached.

3.1 Effectiveness Study

In order to carry out effectiveness study, we need to have a mechanism to validate the accuracy of the recommendation provided by FlowRecommender. To this end, we sample a small fraction of the workflows from the provenance (e.g., 10%) to evaluate the effectiveness of recommendation of FlowRecommender. This set of workflows is called the *test set*. For each workflow in this test set, a so-called *test node* is randomly chosen. The test node, which is the node that has really been executed, will be compared with the recommendation produced using FlowRecommender.

The effectiveness of recommendation is measured by the accuracy of recommendation. Because the nodes in the workflows are generated stochastically based on the transition probability matrices, thus we will consider the top m recommended nodes when we evaluate the recommendation accuracy. A hit (i.e., accurate recommendation) is counted as long as the test node is one of the top m recommended nodes by FlowRecommender. We compare recommendation accuracy of FlowRecommender with that of other two competitive recommendation methods. The first method recommends the next likely node based on its immediately preceding single node, and the other one performs recommendation based upon the immediately preceding continuous upstream sub-paths. The comparison result is shown in 4. The value of m is set as 3 in this experiment. This figure shows that FlowRecommender performs much better than the method recommends node based on immediately preceding node. This is because that there are quite a few nodes in the workflows whose occurrence is not based on its immediately preceding node. FlowRecommender is also superior to the method that performs recommendation using the immediately preceding continuous upstream sub-paths. After a closer examination, we find there are some nodes that correlates with the upstream sub-paths that are not connected with itself.

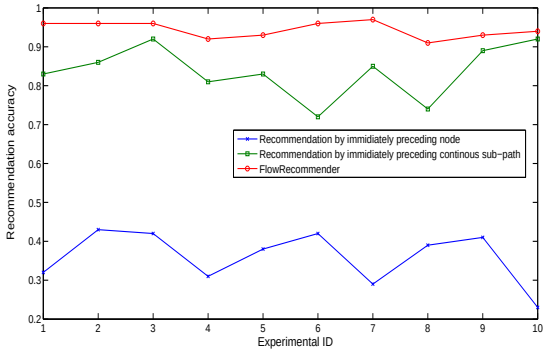

Figure 4: The accuracy of FlowRecommender

3.2 Efficiency Study

We also want to evaluate the efficiency of recommendation. We mainly investigate the execution time of pattern extraction and workflow recommendation which are performed in the offline and online fashion, respectively. Figure 5 and 6 report the execution time of these two steps under varying number of workflows in the provenance. First, from Figure 5, we can see that the extraction of patterns from the provenance scale in an approximately linear manner with respect to the size of the provenance. This is because that the complexity of constructing the inverted

indexing of workflows dominates that the step of pattern extraction and, when constructing the indexing, the whole workflow provenance needs to be scanned. Interestingly, we do not observe such a linear scalability behavior for the workflow recommendation step. As Figure 6 reveals, the execution time of the workflow recommendation step is independent of the provenance size. This is because that it is the pattern table, instead of the original workflow provenance, that needs to be scanned to find the matched patterns for the current workflow under construction. The size of the pattern table is determined by the number of candidate nodes for the whole provenance, i.e., $CNS(D)$. $CNS(D)$ will remain unchanged if the workflows in the provenance are constructed using only a fixed set of nodes. The fluctuation of the execution time results from the different size of $CNS(w)$ at different recommendation locations within the workflow.

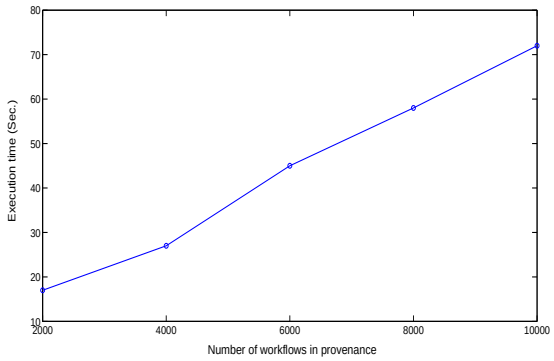

Figure 5: The efficiency of pattern extraction

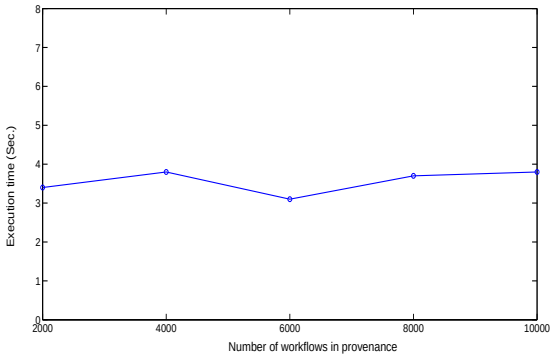

Figure 6: The efficiency of recommendation

4 Related Work

There are abundant research work carried out on service discovery and optimization for composition (which can be considered workflows) in the Service Oriented Computing (SOC) domain. The nodes (in this case, Web services) that are appropriate to the task that are to be fulfilled are first discovered and the best Web service(s) is then identified for execution in the workflow through some service optimization process [3]. In a sense, this resembles to a workflow recommendation problem where the service discovery and optimization help recommend to users the best service that needs to be execute in each step. Nevertheless, there are some fundamental difference between these work and the problem that we are trying to address in this paper. The Web services are recommended based on some pre-defined template where the high-level abstract Web services are well-specified by users when the workflow is constructed. Based upon those abstract Web services, the so-called concrete Web services (namely the instances of abstract Web services) are discovered and the best one is recommended to construct the workflow. In contrast, the construction of workflows in our problem is purely driven by the workflows themselves archived in the provenance without leveraging any pre-defined templates.

A recommendation service that aims at suggesting frequent combinations of scientific programs for reuse is proposed in [1]. This is an early effort to provide workflow recommendation using provenance. This recommendation service is designed to work over repository of workflow execution logs. It allows users discover useful workflow components and how they can be combined, and collected provenance histories are used to recommend a set of candidate services that may be useful to individual scientists. The drawbacks of this method, however, are as follows:

1. The recommendation of a node in the workflow is only depended on its immediately upstream node. For example, node b is recommended for execution after node a (i.e., $a \rightarrow b$) as long as there exists a high correlation between a and b. However, in many cases, correlation exists for non-consecutive nodes and, therefore, this method is not able to identify these patterns for recommendation;

2. This method performs this on-the-fly in the workflow construction process. As such, this method is not efficient to generate recommendation as the computation of confidence typically involves costly workflow scans.

A more sophisticated workflow recommendation technique is propose in [2]. This technique is designed only as a module in a workflow visualization system, called *vis-Complete*. This method decomposes the original workflow (pipeline) into a number of linear paths with varying number of orders, and the confidence of each possible continuous sub-path in the workflow are quantified in order to provide recommendation. All the possible sub-paths (with varying orders) that terminates at the same node in the workflow are ranked based on the their corresponding confidence score. The downstream node/path that features the highest confidence amongst all candidates is picked up for completing the current workflow. The major drawbacks of this method are summarized as follows:

1. In this method, the recommendation of a node for completing the sub-workflow under construction is dependent on the confidence level of only the paths are immediately connected with this node. For example, given a sub-workflow in the provenance $a \rightarrow b \rightarrow c$, this method will, for node c, evaluate the confidence of c given both $a \rightarrow b$ and b i.e., $Conf(c|a \rightarrow b)$ and $Conf(c|b)$. However, if the actual influencing upstream sub-path structure of node c is node a, then this method is not able to find this pattern for recommendation purpose;

2. This method evaluates the confidence of *all* possible paths which involving calculating the support (i.e., frequency) of both upstream and downstream sub-paths. Given the potentially large number of workflows accumulated in the repository, such calculation is rather expensive.

5 Conclusions and Future Research Directions

In this paper, we propose a new workflow recommendation technique, called FlowRecommender, that leverages provenance of workflows to provide recommendation for the best node (e.g., tool/service/program) that needs

to be chosen to complete the workflow. FlowRecommender is able to find the influencing upstream sub-paths of nodes that are not necessarily immediately adjacent to them. FlowRecommender performs offline pattern extraction step which are maintained in the compact pattern table. This contributes to a highly efficient online recommendation when it is required.

There are some further research directions we are interested in exploring, including

1. First, FlowRecommender is only able to find the most adjacent influencing upstream sub-paths for candidate nodes in its current implementation. It is possible that, however, there exists multiple influencing upstream sub-paths for the same candidate node. The recommendation will fail if the influencing upstream sub-paths other than the one registered in the pattern table are present in the worklflow;

2. Second, there have been a plenty of techniques on indexing the sequence patterns. We would like to investigate how these techniques can be used in FlowRecommender to index the influencing upstream subpaths and to what extent the performance boost can be therefore achieved;

3. Finally, the premilinary experimental evaluation that we have performed is only based upon a synthetic workflow provenance. We plan to utilize the Web service workflow construction system we have developed for biological *in-silico* experiments to collect real-life workflows for a further performance validation of FlowRecommender.

References

[1] Frederico T. de Oliveira, Leonardo Gresta Paulino Murta, Claudia Werner, Marta Mattoso. Using Provenance to Improve Workflow Design. *2008 International Provenance and Annotation Workshop (IPAW)*, 136-143, 2008.

[2] David Koop, Carlos Eduardo Scheidegger, Steven P. Callahan, Juliana Freire, Claudio T. Silva. VisComplete: Automating Suggestions for Visualization Pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6): 1691-1698, 2008.

[3] U. S. Manikrao, T.V. Prabhakar, Dynamic Selection of Web Services with Recommendation System, *International Conference on Next Generation Web Services Practices*, 2005.

[4] Yogesh L. Simmhan, Beth Plale, Dennis Gannon. A Survey of Data Provenance Techniques. *Technical Report*, Computer Science Department, Indiana University, IUB-CS-TR618, 2005.

[5] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of Visualization)*, 13(6):1560-1567, 2007.

[6] D. P. Lanter. Design Of A Lineage-Based Meta-Data Base For GIS. *Cartography and Geographic Information Systems*, vol. 18, pp. 255-261, 1991.

[7] S. Miles, P. Groth, M. Branco, and L. Moreau. The requirements of recording and using provenance in e-Science experiments. *Technical Report*, Electronics and Computer Science, University of Southampton, 2005.

[8] I. T. Foster, J.-S. Vockler, M. Wilde, and Y. Zhao. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. *SSDBM*, pp. 37-46, 2002.

[9] D. P. Lanter. Lineage in GIS: The Problem and a Solution. *Technical Report*, National Center for Geographic Information and Analysis, 1990.

[10] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, and B. Didier. Multi-Scale Science, Supporting Emerging Practice with Semantically Derived Provenance. *Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[11] P. Groth, M. Luck, and L. Moreau. A protocol for recording provenance in service-oriented Grids. *OPODIS*, Grenoble, France, 2004.

[12] R. D. Stevens, A. J. Robinson, and C. A. Goble. myGrid: personalised Bioinformatics on the information grid. Bioinformatics, vol. 19, pp. 302i-304, 2003.

[13] C. Pancerella, etc. Metadata in the collaboratory for multi-scale chemical science. *Dublin Core Conference*, 2003.

[14] J. Frew and R. Bose. Earth System Science Workbench: A Data Management Infrastructure for EarthnScience Products. SSDBM, pp. 180-189, 2001.

[15] A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff. Tioga-2: A Direct Manipulation Database Visualization Environment. *ICDE*, 1996.