Extracting Workflow Structures through Bayesian Learning and Provenance Data

Mahsa Naseri
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
naseri@cs.usask.ca

Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, USA
simone.ludwig@ndsu.edu

***Abstract*—Mining workflow models has been a problem of interest for the past few years. Event logs have been the main source of data for the mining process. Previous workflow mining approaches mostly focused on mining control flows that were based on data mining methods, as well as exploited time constraints of events to discover the workflow models. In this work, we present a mining approach which not only takes the behaviourial aspect of workflows into account, but also takes advantage of their informational perspective. Provenance information is a source of reasoning, learning, and analysis since it provides information regarding the service inputs, outputs and quality of service values. Therefore, provenance information along with Bayesian structure-learning methods are exploited for this purpose. Two constraint-based Bayesian structure-learning algorithms are investigated and modified in order to make use of additional provenance information. We will show that this leads to better mining results based on three common mining scenarios.**

I. Introduction

Combining a set of tasks together in a specific order for the purpose of achieving a specific goal is a process taking place in all different areas of science, from business to chemistry, physics, math, etc. During such a process, referred to as a workflow process, tasks might have prerequisites and are run in sequential or parallel order. The most common form of representing a workflow model is the directed graph. Tasks or activities are usually enclosed in boxes or circles and are referred to as vertices of the graph while the arrows depict the edges, which represent the direction of the flow. As for some workflow systems, the workflow events and their timing information are recorded sequentially into logs. Keeping track of certain data attributes of a process being executed, and storing this information into an event log is a procedure taking place in certain systems. These event logs usually contain limited amount of information about the process and mostly include the process id, the name of the task, and the execution time of each task. Mining workflows and processes through event logs has been a problem of interest. Event logs have been analysed and searched in order to analyze the effectiveness of a workflow process, to discover previous workflow models, to find the hidden causal relationships existing among tasks, etc.

The process of workflow mining is referred to as the task of extracting process knowledge from the event logs. It discusses techniques for acquiring a workflow model from a workflow log. As mentioned in [1], the desire for companies to learn about their processes is the main reason behind exploitation and development of process mining techniques.

The basic idea behind workflow mining is to construct the workflow's directed graph from the information gathered from the execution of the workflow process. This process is usually done using an algorithmic technique or statistical analysis. Machine learning, data mining, genetic algorithms, and sequence mining are the main approaches of workflow mining applied in literature. Data mining methods for discovering sequential patterns, statistical analysis methods for building and extracting statistical dependencies, or a combination of both methods have been used.

In [2], the authors discuss the data mining algorithms used to discover sequential patterns. The algorithms include the Generalized Sequential Pattern (GSP), which has the advantage of taking the time constraints into account, and the Apriori algorithm [3]. Methods used for event-data analysis have also been used for workflow mining. These methods vary from purely algorithmic ones to purely statistical ones [4], or a combination of both techniques [3].

The research provided in [1] presents a comparison of the current workflow mining approaches and categorizes them based on the contribution of each work. Some works, such as the one presented in [5], are limited to sequential models. Other approaches such as [6] support more complex structures including concurrent processes, but are limited to workflow models without repetitive or duplicate tasks. In [7], [8], and [9], approaches are presented that allow the appearance of the same task in the workflow model. Some methods such as the Markovian approach do not target any of the major workflow mining issues. Current approaches do not mine process models from different perspectives. Also, there is no one single approach that targets many of the workflow mining issues. Thus, new approaches are required that address these issues.

We believe, these challenges can be resolved if the logs provide more information. As discussed in [10], a process data warehouse is required to apply workflow mining. In [11], we propose and discuss the benefits of an architecture that addresses workflow problems using provenance information along with statistical and mining methods. One of the compo-

nents of such an architecture is the workflow policy graph extractor, which learns and mines workflow patterns from provenance data. Following the proposed architecture, in order to be able to mine workflow models using various perspectives and to simplify the discovery of causal relations, we propose the exploitation of provenance information for the purpose of process mining.

In this paper, we exploit Bayesian structure learning methods along with provenance information. Two constraint-based algorithms of Parents and Children (PC) and Max-Min Parents and Children (MMPC) were selected for learning, and modified in order to improve the workflow mining task. Our approach is different from the previous approaches as it exploits both data and control aspects of workflows for mining, discovers concurrent processes, and supports structures with duplicate tasks.

The rest of this paper is organized as follows: our methodology is described in Section 2 along with the constraints, conditions, and modifications applied to the constraint-based algorithms. Section 3 provides the implementation details. In Section 4, a case study is conducted showing three different cases, as well as the performance evaluation results are presented. Section 5 presents the conclusion of the research provided.

II. Methodology

A. Bayesian Structure Learning Methods

Bayesian structure learning methods can be categorized into two groups of constraint- and score-based algorithms. Constraint-based algorithms perform structure learning in two steps. The first step discovers the skeleton using conditional hypothesis tests. The skeleton is the undirected structure in which only the location of edges are determined with no directions. The second step finds the orientation of the edges in the skeleton. The scored-based methods address structure learning as a model selection problem. After having defined a scoring function that evaluates how well a structure matches the data, these methods search through all possible network structures for the highest scored network, and thus is NP-hard.

Compared to score-based methods, constraint-based approaches are more suitable for the purpose of knowledge discovery as they produce more accurate results. These methods, which are also referred to as conditional independence learners, use conditional independence tests to detect the Markov blankets of the variables in order to compute the structure of the Bayesian network.

Conditional independence tests in structure learning are concerned with nodes/variables that are necessarily independent given the structure of the underlying Directed Acyclic Graph (DAG). The independence assertions are learnt from data and are used in both steps of these algorithms. The first phase exploits the conditional independence test to determine whether an edge should exist between two nodes, and represents the result as an misdirected skeleton. In order to learn the structure of DAGs, a sufficient condition is "faithfulness". The faithfulness assumption asserts that the conditional independencies observed in the distribution of a network are due to the structure of the network. It allows us to move from a probability distribution to a DAG. The later step of the constraint-based algorithms discovers the separating sets, which indicate the set of nodes that are sufficient to d-separate two nodes from each other.

B. Modeling the Workflow Model Extraction Problem as Bayesian Structure Learning

Constraint-based approaches in literature are different based on the type of independence test or ordering heuristics. Among the possible algorithms, we have selected the PC [12], as well as, the MMPC [13] algorithms. Using statistical or information-theoretic tests, these algorithms estimate based on the data whether certain conditional independencies between the variables hold. They start from a complete, undirected graph and delete edges recursively based on conditional independence decisions. This yields an undirected graph, which can then be partially directed and further extended to represent the underlying DAG. The PC algorithm has an intuitive basis and guarantees the recovery of the original causal structure under ideal conditions [14]. It is faster than similar approaches such as SGS [15] and produces better results. The other algorithm, MMPC, outperforms on average several constraint-based algorithms such as PC, Sparse Candidate [16], etc. [13].

As both of these algorithms take a similar approach towards structure learning, i.e. using conditional independence tests, they were selected as the main methods applied to the workflow data for the aim of mining.

Constraint-based structure learning methods are based on the assumption of having a very large database. This condition is satisfied since provenance information is being used for this purpose. The other condition that should be satisfied is that the independence relationships have to have a perfect representation of a DAG. In case of workflow mining, these relationships represent the data flow connections that exist between services (tasks). This condition can not be assumed to be true since these relationships in a workflow structure might not necessarily represent a DAG depending on the degree of "faithfulness" of the data. Thus, we will be presenting modified versions of these algorithms which provide better results in case of unfaithful workflow models.

In order to model the problem workflow mining as a Bayesian structure discovery, services serve as the nodes of the Bayesian graph, each having values representing different states a service provider provides. The links in the Bayesian graph represent the causal relationships that exist among the services. Therefore, the graph extracted from the provenance data depicts the workflow policy graph.

To evaluate the degree of dependence between services using conditional independence tests, the output parameters and values of services are taken into account. The values are matched against each other to assess the mutual information the services provide, given the current discovered structure, and investigate if dependencies can be found.

We can not assume to have a Bayesian structure in the data if the workflows do not follow the conditions underlying such a structure. Thus, the faithfulness of the data can not be guaranteed as it depends on the workflow models. Therefore, the Bayesian learning algorithms need to be modified in order to be able to discover the structure as accurate as possible even in the absence of the faithfulness condition.

Assume a workflow consisting of 3 sequential services of A, B, and C for finding an Internet Provider (IP) address, searching for the city name based on the IP, and finding the weather forecast for that city. This workflow structure example will not satisfy the faithfullness condition as both "IP" and "city" services' output values provide the same mutual information for the "weather" service. Thus, "weather" and "city" are assessed as independent given "IP". Such issues prevent the Bayesian structure learning algorithms to discover the whole model of a workflow. Given the described example, the learnt model will only include arcs from A to B and to C, and services B and C are assumed to be independent. This will result in a workflow structure represented as B <- A ->C while the original model suggests A ->B ->C.

In order to overcome these issues, the original PC and MMPC algorithms are augmented with certain heuristics that make their results more accurate in case of the unfaithfulness condition. These heuristics exploit timing information provided in the provenance data, gathered during workflow executions, to identify the ordering of the variables for variable selection, to discover the mutually exclusive or parallel services, and to find services that provide the same information.

C. Parents and Children (PC) Algorithm

The first phase of the PC algorithm will be used for learning the structure of the graph. As discussed in [12], the performance of this phase of the algorithm can be improved by knowing the ordering of the edges. Since the provenance information are used for learning, the timestamps including the starting time of services and duration can be exploited for the purpose of ordering. This results not only in a better performance of the algorithm with less time complexity, but also saves execution time of the V-structure phase of the algorithm that is aimed at discovering edge directions. The Modified_PC algorithm is shown in Algorithm 1. Apart from using a timely ordered set of nodes, the two functions of "Check_Splits" and "Check_Same_Info" are the main modifications applied to the original PC algorithm. The first function, i.e. Check_Splits, checks if the two variables being checked for independency are parallel-splits (and-splits or or-splits). A parallel split creates a split in a workflow model. In case of and-split all the branches will be active, as for an or-split only one branch is active at a time. In order to check this, the starting time of the two variables, i.e. services, are taken into account. If the time difference between the two starting times is less than a threshold, the two services are considered parallel and based on their data values are added to the split-and or split-or lists. The other function, "Check_Same_Info", checks if the two variables x and y being checked for independence given variable z provide the same information, i.e. fall into the same information provider category. It performs the assessment by first checking if x belongs to y's conditional set. If this is not the case, y and z are tested for independence conditioned on x. If either of these tests are true, then x and y and z provide the same information, and thus, the scenarios such as the one presented in the previous subsection are discovered correctly.

Algorithm 1 Modified_PC Algorithm

```
function MODIFIED_PC(G, O)
    Input: Fully connected graph G, Timely Ordered Variables O
    i = 0
    repeat
        for all x ∈ V do
            for all y ∈ Adj_x do
                if Check_Splits(x, y, G, O) then
                    Continue
                end if
                Determine if S ⊂ |Aj_x|{y} | with | |S| = iandI(x, y|S)
                if this set exists then
                    if Check_Same_Info(x, y, G, O, S) then
                        if time_difference(x,y) < time_difference(y,S) then
                            Remove S-y link from G
                            break
                        else
                            Remove x-y link from G
                            Add y-S link to G
                            break
                        end if
                    else
                        Make S_xy = S
                        Remove x-y link from G
                    end if
                end if
            end for
        end for
    until |Adj_x|<i∀x
end function
function CHECK_SAME_INFO(x, y, cond)
    if !check_mutual(x, cond) | && !!check_mutual(y, cond) then
        if x ∈ Conditional_Set(y, cond) then
            return true
        end if
        if I(y, cond|x) then
            Add x to Conditional_Set(y, cond)
            return true
        end if
    end if
    return false
end function
```

D. Max-Min Parents and Children (MMPC) Algorithm

This algorithm is based on the local discovery algorithm called Max-Min Parents and Children (MMPC). The Max-Min part of the algorithm name refers to the heuristic the algorithm uses, while the parents and children part refers to its output [13].

MMPC focuses on learning substructures around each variable. It is invoked by each variable of the network, referred to as t, in order to identify the existence of edges to and from that variable, and to discover the structure of the network. Similarly to PC, this algorithm starts with a fully connected graph and exploits two heuristics to discover the dependencies. The first phase, which is referred to as the forward phase, incrementally discovers edges using the Max-Min heuristic. The Max-Min heuristic selects the variable that maximizes the minimum association with a selected variable relative to the so far learned graph. It uses the function Assoc$(x, t|Z)$ which

measures the strength of dependency between x and t given a set of variables Z. As mentioned in [12], the justification for the Max-Min heuristic is to select the variable that remains dependent even after conditioning all the subsets of the so far discovered network. The second phase removes the false positives that might have been entered in the first phase by running conditional independence tests on x and t given any subset of the learned graph.

The mutual information existing between the output values of the two services is used as the criteria evaluating the strength of the association. The MinAssoc function determines the minimum dependency achieved between x and t over all the subsets of the variables discovered.

The modified MMPC algorithm is presented in Algorithm 2. The function Ind(X;T—Z) return true if x and t are conditionally independent given Z. As for the modified PC algorithm, the Check_Splits function is used to discover the split-ands or split-ors of the workflow graph. The MaxMinHeuristic function is modified so that if two x variables are equally mutually informative, the one which is closer in time to the selected t is chosen as the variable representing the maximum association.

Algorithm 2 Modified_MMPC Algorithm

function MODIFIED_MMPC(t,D)
  Input: target variable t, data D
  Output: parents and children of t
  $G = 0$
  repeat
    $<F, assoF> = MaxMinHeuristic(t, G)$
    if $assocF<>0$ **then**
      $G = G \cup F$
    end if
  until G has not change
  for all $x \in G$ **do**
    if $\exists S \subset Gs.t. Ind(x, t \mid S)$ **then**
      if $Check\_Same\_Info(x, t, s)$ **then**
        if $time\_difference(x, t) > time\_difference(t, S)$ **then**
          $G = G \setminus \{s\}$
        else
          $G = G \setminus \{x\}$
          $G = G \cup \{s\}$
        end if
      end if
    end if
  end for
  return G
end function
function MAXMINHEURISTIC(t,G)
  Input: variable t, subset of variables G
  Output: maximum over all variables of the min association with t relative to G, and variable that achieves the maximum
  $assocf = max\ x\ \in\ v\ MinAssoc(x, t|G)\ if\ !Check\_Splits(x, t)$
  $f = arg\ max\ x\ \in\ v\ MinAssoc(x, t|G)$
  return $<f, assocf>$
end function

III. IMPLEMENTATION

In order to perform real world and valuable experiments, Taverna (version 2.1) [17] was selected as a practical provenance system and was expanded to incorporate the additional features required for our experiments. Taverna does not record timing information such as start or execution time of services during workflow runs. Since Taverna does not record non-functional specifications of web services, Taverna's provenance data model was changed to allow the storage of the QoS values of services. A QoS tracker was added to Taverna to record the QoS specifications of the WSDL services imported by Taverna. The QoS recorder exploits WebInject [18], a tool for automated testing of web applications and web services, to monitor the services. Transaction monitors are set up for service-level monitoring of response time and availability of web services. Apart from these parameters, the QoS tracker also keeps track of the execution time and status of execution of services. The conditional independence test from the "bnlearn" library of the R [19] package was exploited to discover the causal dependencies between services. The "Rserve" [20] server creates the facility to connect to R libraries through our application.

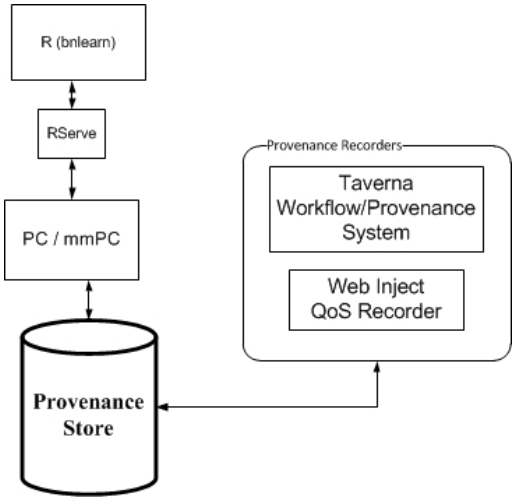

Fig. 1: Implementation Model

An overview of the implementation model is shown in Figure 1. At the beginning, a workflow is created by Taverna. The related services are added to WebInject to record QoS parameters such as timing information. Every time a workflow instance is run, the data regarding the functional aspects of the workflow is generated by the Taverna system while the QoS values are produced by WebInject. All this information is stored into the provenance store and is exploited later by the modified PC and MMPC algorithms for the learning of the workflow structure. These algorithms use R libraries through Rserve to assess the conditional independence tests and discover the causal relationships.

IV. CASE STUDY AND PERFORMANCE EVALUATION

A. Three Scenarios

In order to observe the performance of the modified algorithms, 3 different workflow scenarios were considered and tested. The first one consists of a sequential workflow scenario that does not satisfy the faithfulness condition, the second one contains a parallel workflow structure, and the third includes a complex scenario composing of two split constructs.

The web services used by the three scenarios were selected randomly from different service providers and were placed

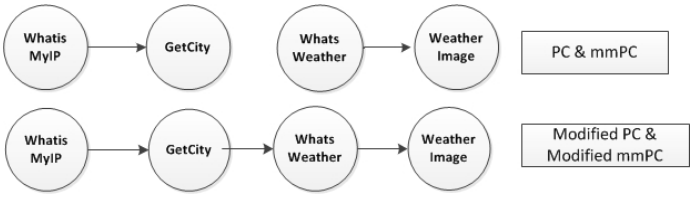

Fig. 2: Case Study 1: a sequential workflow scenario

together as a workflow in Taverna. Having executed the workflows for multiple times, a dataset of 10,000 rows was created for the experiments.

1) Case 1: The first scenario is a completely sequential workflow consisting of 4 web services of "WhatisMyIP", "GetCity", "WhatsWeather", "GetWeather". The "WhatisMyIP" service finds the IP of the customer, sends it to the "GetCity" service, which finds the city based on the IP address. The city name is passed to the "GetWeather" service and the weather forecast of the city is predicted. Having the forecast, the forecast image is shown to the workflow user presenting the weather condition for his location. Figure 2 shows the structure of the workflow as well as the experimental results. As can be seen from the figure, the original "PC" and "MMPC" algorithms discover some of the edges, while the modified PC and MMPC algorithms discover the complete workflow graph even though the data for this scenario does not satisfy the faithfulness condition.

2) Case 2: The next workflow scenario includes four services, two being run in parallel. Figure 3 shows the scenario along with the results of the Bayesian structure learning algorithms. As can be seen from the figure, the original PC and MMPC algorithms can not find the complete parallel structure. This is due to the lack of the faithfulness condition in the structure of the workflow scenario graph. The "productID", "salePrice", and "AveragePrice" mutually provide the same information, thus the original PC and MMPC algorithms find the two services of "SalePrice" and "AveragePrice" independent given the "ProductID". These issues have been resolved with the modified algorithms by the two added functions. The modified algorithms discover the two services of "SalePrice" and "StockPrice" as parallel, and thus, they are removed from the each other's conditional sets. Thus, the complete structure is discovered via these modified approaches.

3) Case 3: As for the third scenario, three separate relevant workflows were considered and the structure learning algorithms were used to discover the graph policy from these workflows. The scenario involves services for receiving and delivering an order in case of a valid credit card payment as well as the availability of the product. Figure 4 shows the three paths that can be taken based on the service outputs. In this scenario, the distinguishable difference between the performances of the modified algorithms versus the original algorithms shows the effects the modifications have had on the extraction of more accurate graphs. This scenario includes a split-or, a split-and and a join. The original algorithms can

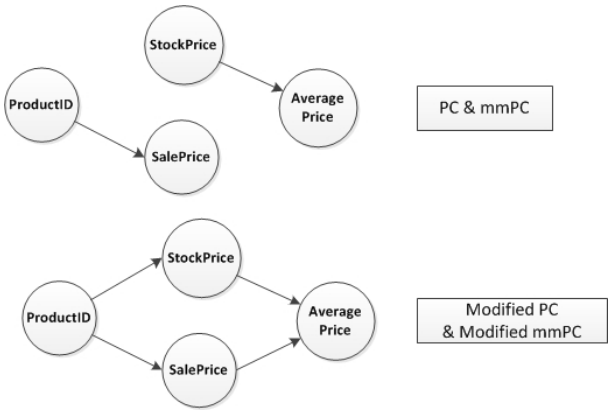

Fig. 3: Case Study 2: workflow with parallel parts

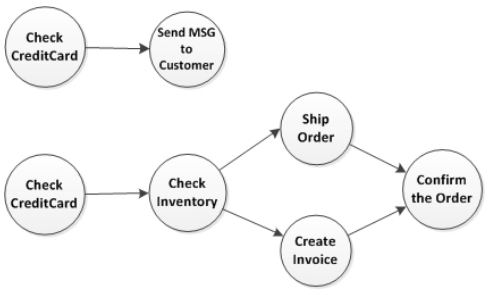

Fig. 4: Case Study 3: more complicated scenario

not extract many of the edges due to the large dependencies that exist within the service data. Figures 5 and 6 display the results of the modified algorithms as well as the original algorithms.

B. Experimental Results

We evaluated the performance of the modified algorithms with regards to execution time. The original algorithms of PC and MMPC were compared with the modified ones in terms of workflow sizes and execution time. For the experiments, data values of sequential workflows of sizes 5 to 25, with increments of 5, were randomly generated. The experiments were done 10 times on different datasets. As can be seen from the graph shown in Figures 7 and 8, the modified algorithms have a steeper slope with the PC algorithm having

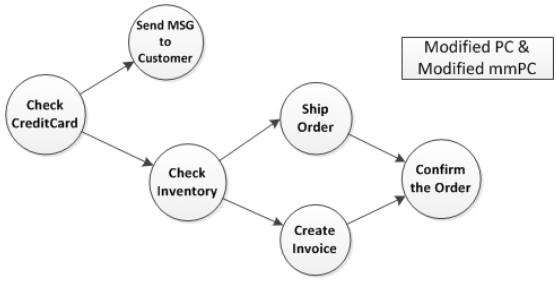

Fig. 5: Case Study 3: workflow of Modified PC and MMPC

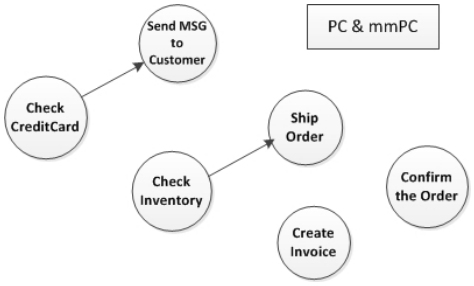

Fig. 6: Case Study 3: workflow of PC and MMPC

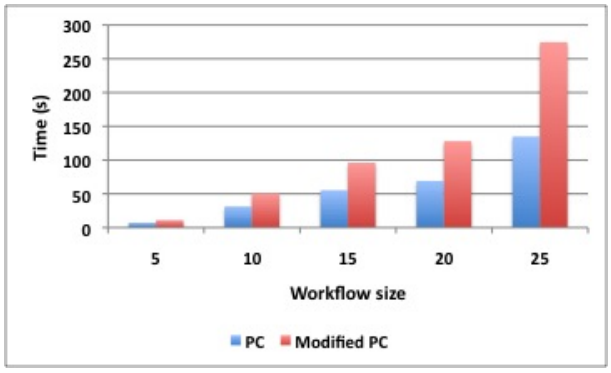

Fig. 7: Performance of MMPC and modified MMPC

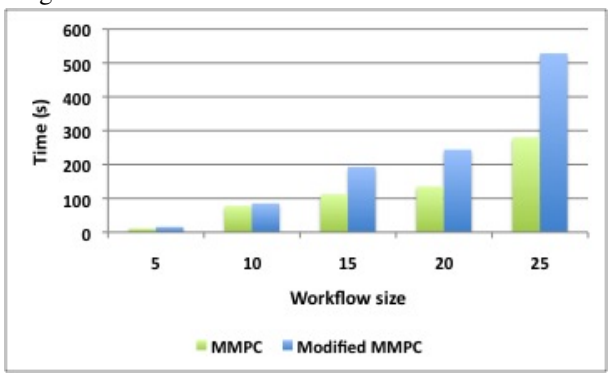

Fig. 8: Performance of PC and modified PC

the better performance overall. The modified PC and MMPC algorithms consume more time since the Check_Same_Info and Check_Mutual functions increase the number of conditional independence tests the original algorithms use. The MMPC algorithm uses more conditional tests compared to PC since both the forward and backward phases, perform tests of independency.

V. Conclusion

In this paper, we exploited constraint-based Bayesian structure learning algorithms to extract the structure of workflows from provenance data. The output values of services were used to discover the data flow between services along with timing information to provide control flow information. Provenance information was used since it provides the appropriate amount of data gathered over time, and therefore, makes it suitable for learning. The two algorithms of PC and MMPC were modified in order to better discover the workflow models of the scenarios which do not support the faithfulness condition. PC, MMPC and the modified algorithms were assessed using 3 different scenarios. As the results presented, despite of the fact that some workflow scenarios might not follow the faithfulness condition, the changes applied to both algorithms (PC and MMPC) provide complete and robust structures. The benefit of mining accurate workflow structures comes at the price of higher execution times. Both modified algorithms take longer to find the correct workflow structure.

References

[1] A. Tiwari, C. J. Turner, B. Majeed, "A review of business process mining: state-of-the-art and future trends", Business Process Management Journal, Vol. 14 Iss: 1, pp.5 22, 2008.
[2] I. Altintas, "Lifecycle of Scientific Workflows and Their Provenance: A Usage Perspective", Proceedings of IEEE Congress on Service, 2008.
[3] W. Gaaloul, S. Alaoui, K. Baina, and C. Godart, "Mining Workflow Patterns through Event-Data Analysis". In Proceedings of the 2005 Symposium on Applications and the internet
[4] S. Hwang and W. Yang, "On the discovery of process models from their instances". Decis. Support Syst. 34, 1 , Dec. 2002, 41-57.
[5] J. Herbst, D. Karagiannis, "Integrating machine learning and workflow management to support acquisition and adaptation of workflow models", International Journal of Intelligent Systems in Accounting, Finance and Management, 9, pp. 6792, 2000.
[6] G. Schimm, "Mining exact models of concurrent workflows", Computers in Industry, Vol. 53, pp. 265-81, 2004.
[7] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining process models from workflow logs", in Schek, H.J. (Ed.), Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology, Springer Verlag, Heidelberg, 1998.
[8] J. Herbst, and D. Karagiannis, "Workflow mining with InWoLvE", Computers in Industry, Vol. 53, pp. 245-64, 2004.
[9] W. M. P. van der Aalst, and A. K. Alves de Medeiros, "Process mining and security: detecting process executions and checking process conformance", Electronic Notes in Theoretical Computer Science, Vol. 121, pp. 3-21, 2005.
[10] W. M. P. van der Aalst, and A. J. M. M. Weijters, "Process Mining: A Research Agenda", Computers in Industry, Vol. 53, No. 3, pp. 231-244, April 2004.
[11] M. Naseri and S. A. Ludwig, "A Multi-Functional Architecture Addressing Workflow and Service Challenges Using Provenance Data", Proceedings of Workshop for Ph.D. Students in Information and Knowledge Management (PIKM) at 19th ACM Conference on Information and Knowledge Management (CIKM), Toronto, Canada, October 2010.
[12] P. Spirtes, C. N. Glymour, R. Scheines, "Causation, Prediction, and Search", MIT Press, Cambridge, MA, No. of pages: 543. ISBN 0-262-19440-6, 2000.
[13] I. Tsamardinos, L. E. Brown, "The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm", Journal of Machine Learning, 0885-6125, vol. 65, pp. 31-78, 2006.
[14] J. Abellan, M. Goomez-Olmedo, and S. Moral, "Some Variations on the PC Algorithm", Proceedings of the third European workshop on probabilistic graphical models (PGM06) Prague, pp. 18, 2006.
[15] R. Daly, Q. Shen and S. Aitken, "Learning Bayesian networks: approaches and issues", Knowledge Engineering Review, vol. 26, pp. 99-157, 2001.
[16] N. Friedman, I. Nachman, and D. Peer, "Learning Bayesian network structure from massive datasets: the Sparse Candidate algorithm". In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99), 1999.
[17] Taverna, Available at: http://www.taverna.org.uk/.
[18] WebInject, Web Application and Web Services Test Tool, last retrieved from http://webinject.org/, 2012.
[19] The R Project for Statistical Computing last retrieved from http://www.r-project.org/, 2012.
[20] Rserve, last retrieved from http://cran.rproject.org/web/packages/Rserve/index.html,2012.