

Utility-Based Control Flow Discovery from Business Process Event Logs



Kritika Anand

Computer Science

Indraprastha Institute of Information Technology, Delhi (IIIT-D), India

A Thesis Report submitted in partial fulfilment for the degree of

MTech Computer Science

6 June 2015

1.: Prof. Ashish Sureka (Thesis Adviser)

2. Prof. Sanjit Kaul (Internal Examiner)

2. Ms. Radhika Venkatasubramanyam (External Examiner)

Day of the defense: 6 June 2015

Signature from Post-Graduate Committee (PGC) Chair:

Abstract

Process Aware Information Systems (PAIS) are IT systems which support business processes and generate event-logs as a result of execution of the supported business processes. Fuzzy-Miner (FM) is a popular algorithm within Process Mining which consists of discovering a process model from the event-logs. In traditional FM algorithm, the extracted process model consists of nodes and edges of equal value. However, in real-world applications, the actors, activities and transition between activities may not be of equal value. In this paper, we propose a utility-based Fuzzy Miner (UBFM) algorithm to efficiently mine a process model driven by a utility threshold. The term utility can be measured in terms of profit, value, quantity or other expressions of user's preference. The focus of this paper is to take into consideration the statistical (based on frequency) and semantic (based on user's objective) aspect into account. We conduct experiments on real-world dataset and demonstrate the effectiveness of our approach.

This Thesis work is dedicated to my parents for their endless love and support. This work is also dedicated to SGI President Dr.Daisaku Ikeda whose writings are always a source of encouragement and fresh determination.

Acknowledgements

“Even if things don’t unfold the way you expected, don’t be disheartened or give up. One who continues to advance will win in the end.”

- Daisaku Ikeda

This is the motto of my advisor Prof. Dr. Ashish Sureka whose help, stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis. His excellent guidance and motivation has always been the source of inspiration whenever I got stuck in the work.

Besides my advisor, I would like to thank all of my thesis committee members : Prof. Sanjit Kaul and Ms. Radhika Venkatasubramanya for their valuable time.

Next, I would like to thank my fellow mate Nisha Gupta for her constant support, encouragement and help during the course of my thesis. I also want to thank my roommate Ruchita Bansal for her support, interest and valuable hints.

Last but not the least, I would like to extend special thanks to my family for their patient love which enabled me to complete this work.

Declaration

This is to certify that the MTech Thesis Report titled **Utility-Based Control Flow Discovery from Business Process Event Logs** submitted by **Kritika Anand** for the partial fulfillment of the requirements for the degree of *MTech* in *Computer Science* is a record of the bonafide work carried out by her under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Professor Ashish Sureka

Indraprastha Institute of Information Technology, New Delhi

Contents

List of Figures	ix
List of Tables	xi
1 Research Motivation and Aim	1
1.1 Process Mining: An Overview	2
1.1.1 Process Model	3
1.2 Fuzzy-Miner	4
1.3 Introduction to Utility	5
1.3.1 Importance of Utility	5
1.3.1.1 ITIL Service Change	5
1.3.1.2 Repair/Replacement Process	6
1.3.1.3 Purchasing Process Example from SAP	6
2 Related Work and Research Contributions	11
2.1 Related Work	11
2.1.1 Process Mining	11
2.1.2 High Utility Itemset Mining	12
2.2 Novel Research Contributions	13
3 Research Framework and Solution Approach	14
3.1 Fuzzy-Miner	14
3.2 Utility Based Fuzzy-Miner	21
4 Experimental Analysis and Results	28
4.1 Experimental Dataset	28
4.1.1 BPI 2014 Dataset	28

CONTENTS

4.1.2	Airport Dataset	30
4.2	Experimental Results	32
4.2.1	Airport Dataset Results	32
4.2.2	BPI 2014 Dataset Results	39
5	Limitations and Future Work	44
6	Conclusion	45
	References	46

List of Figures

1.1	Types of Process Mining Techniques [1]	3
1.2	Small Process Model	4
1.3	Subset of Information Technology Infrastructure Library (ITIL) service change flow	5
1.4	Subset of repair/replacement process flow	6
1.5	Subset of process model build using purchasing example from SAP	7
1.6	Subset of process model build using purchasing example from SAP	7
1.7	Subset of process model build using purchasing example from SAP	8
1.8	Subset of process model build using purchasing example from SAP	8
3.1	Significance and correlation metrics used in FM	14
4.1	Pareto chart showing the distribution of activities and their cumulative count. Y-axis is in logarithmic scale	30
4.2	Histogram for case variants	30
4.3	Activities ordered according to increasing order of datetime stamp for IncidentId <i>IM0000004</i>	31
4.4	Normalized unary utility significance for Airport dataset	33
4.5	Unary significance metric for Airport dataset	34
4.6	Binary significance metric for Airport dataset	35
4.7	Process model created using UBFM at edge filter = 0.0 and node filter = 0.0	36
4.8	Process model created using UBFM at edge filter = 0.25 and node filter = 0.25	37

LIST OF FIGURES

4.9	Process model created using UBFM at edge filter = 0.5 and node filter = 0.5	37
4.10	Process model created using UBFM at edge filter = 0.75 and node filter = 0.75	38
4.11	Process model created using FM at edge filter = 0.25 and node filter = 0.25	39
4.12	Process model derived from event-logs in BPI 2014 dataset discovered using UBFM at edge filter = 0.0 and node filter = 0.0	40
4.13	Process model derived from event-logs in BPI 2014 dataset discovered using UBFM at edge filter = 0.5 and node filter = 0.25	41
4.14	Snapshot of process model derived from event-logs in BPI 2014 dataset discovered using FM at edge filter = 0.5 and node filter = 0.25	42
4.15	A line chart shows decrease in number of edges at different steps in UBFM at increasing value of node and edge filter	43

List of Tables

4.1	BPI 2014 Dataset Detail of Table Incident Activity Detail	29
4.2	Airport Dataset Detail	30
4.3	Snapshot of Airport Dataset for Trace3.0	31
4.4	Labels Corresponding to Activity Name in Airport Dataset	32
4.5	External Utility of Activities and Precedence Relations	32
4.6	Binary Utility Significance Metric for Airport Dataset	33
4.7	Binary Correlation Metric for Airport Dataset	34
4.8	External Utility Provided by Business Process Analyst	40

Research Motivation and Aim

Process Mining consists of analyzing event-logs generated by Process Aware Information Systems (PAIS) for the purpose of discovering run-time process models, checking conformance between design-time and run-time process maps, analyzing the process from control flow and organizational perspective for the purpose of process improvement and enhancement [2]. FM algorithm [3] is one of the fundamental algorithms in Process Mining consisting of discovering a process model (reconstructing causality and work flow between activities) from an event-log consisting of process instances or traces. The amount of detail to be shown in the output of the traditional FM algorithm can be adjusted using node and edge filters. Several real-world run-time process models can be quite complex (spaghetti-like) due to the large number of variations and flexibility in process flow allowed by the PAIS. Filters are used by the process analyst or owner to focus on the important and interesting aspects of the process model and remove nodes and edges which are not important. Node and edge filters in traditional process discovery algorithms like FM are based on frequency of events and precedence relationships (captured as one aspect under metric called as significance). Increasing the value of node and edge filters removes events and relationships which have low significance and low correlation. However, the output display behaviour and the abstraction level of the process model in traditional process discovery algorithms are not driven by utility or economic objectives or value.

The concept of economic utility of events, activities and relationships is not captured in traditional FM algorithm. Utility-based data mining is an area that has attracted several researchers attention dealing with cost-sensitive learning, economic factors and

utility considerations. A popular example in Utility-based Association Rule or Frequent Pattern mining is of lipstick and perfume which falls into the class of high-utility rare itemset rather than bread and butter which can be categorized as high-frequency but low utility itemset. Utility based process mining is a relatively unexplored area and the work presented in this paper is motivated by the need to investigate efficient utility-based process mining (in particular process discovery which is the focus of this paper) approaches which does not capture only statistical correlations but also semantic significance of events, activities and relations.

1.1 Process Mining: An Overview

A Process-Aware Information System (PAIS) is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models [4]. Example PAIS's are workflow management systems, case-handling systems, enterprise information systems, etc. PAIS log events and activities during the execution of a process. Process Mining is a relatively young and emerging discipline consisting of analyzing the event logs from such systems for extracting knowledge such as the discovery of runtime process model (discovery), checking and verification of the design time process model with the runtime process model (conformance analysis) and improving the business process (recommendation and extension) [5] [6]. Process mining uses data mining techniques in the context of business process management and enables the application of innovative approaches for improving the management of business processes. As it can be seen from Figure 1.1, there are three types of process mining techniques:

1. **Process Discovery** - It takes input as event log and produces a process model.
2. **Process Conformance** - This technique takes an existing process model as a reference and compares it with the given event log, to check if the given event log conforms to the process model and vice-versa.
3. **Process Enhancement** - This technique extends or improves the existing process model. It takes the existing process model as input and tries to extract new information from it.

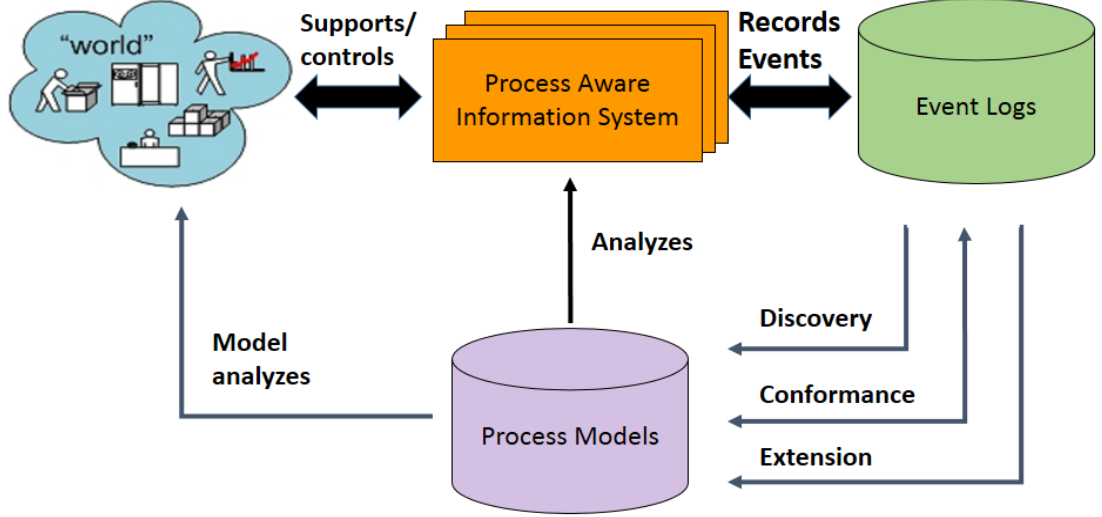


Figure 1.1: Types of Process Mining Techniques [1]

Process mining techniques attempt to extract non-trivial and useful information from event logs. The event logs obtained can be analyzed in three different perspectives: (1) the process perspective, it focuses on the control flow i.e. ordering of activities, (2) the organizational perspective, it focuses on the actors performing activities and (3) the case perspective, it focuses on the properties of cases [7]. The knowledge obtained this way can increase understanding of the processes within an organization. The focus of this thesis is to demonstrate the utility based control flow discovery from the event logs.

1.1.1 Process Model

Figure 1.2, shows the process model. Process models are processes of the same nature that are classified together into a model. It has a starting point and an ending point. All the activities lie between these two points. In the event log, set of activities refer to a particular process instance (cases). In all cases, there is an initial and a final activity. All initial activities are connected to the starting point and all the final activities are connected to the ending point. "Open" is an initial activity and "Closed" is a final activity, as shown in Figure 1.2. Arrow shows transitions and rectangular box represents a state. Frequencies specified on arrows, represent transitions and activities inside rectangular box represent a state. It basically shows how many times one state (ac-

tivity) goes to another state (activity) like "Open" activity goes 8501 times to "Status Change" activity.

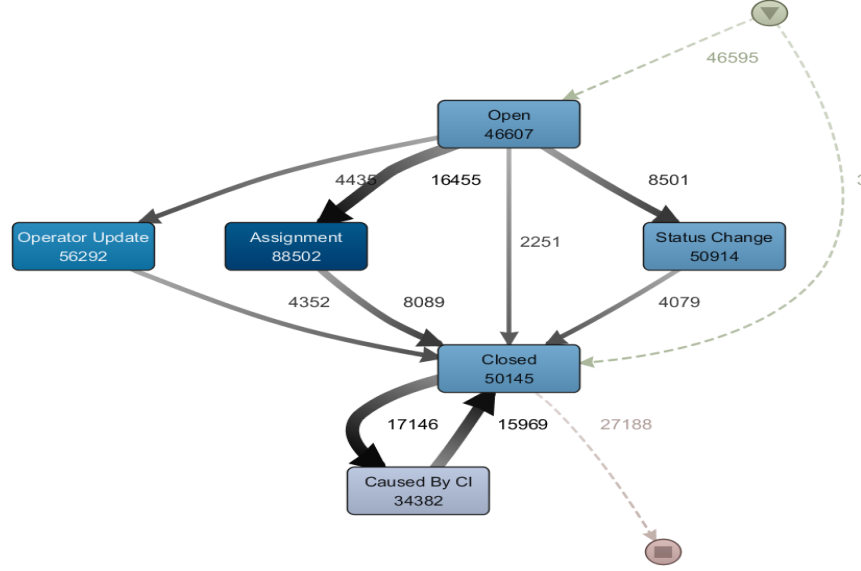


Figure 1.2: Small Process Model

1.2 Fuzzy-Miner

The process mining algorithms prior to the FM produce “spaghetti models” for the less structured processes. Such models produce correct and accurate results but they are difficult to analyze and interpret by the business process analyst. To abstract meaningful information from the spaghetti models FM came into existence. FM uses an analogy of road map [3]. Activities in a process is compared with locations and precedence relations with connections between those locations in a map. FM employs the concept of filters to remove nodes and edges which are not important. Node and edge filter in FM are based upon the two metrics: (1) significance and (2) correlation of events and precedence relations [3]. Increasing the value of node and edge filter removes the events and relationships which have low significance and low correlation. Significance uses the concept of frequency and eliminates the exceptional cases. Correlation measures how

two events following one another are closely related.

1.3 Introduction to Utility

The drawback of the FM is that it treats all the activities and edges with the same importance and weights. Real life event logs often contain activities and precedence relations which are:

1. Very frequent but of low utility to the organization.
2. Less frequent but of high utility to the organization.

Considering the above two cases frequency alone should not be the driving factor for process simplification. Therefore, concept of utility of events and precedence relations should also be taken into account while simplifying the process models. The term utility can be measured in terms of profit, value, quantity or other expressions of user's preference. The focus of this thesis is to take into consideration the statistical (based on frequency) and semantic (based on user's objective) aspect into account.

1.3.1 Importance of Utility

We came up with the following 3 real life scenarios that demonstrates the importance of utility based control flow discovery.

1.3.1.1 ITIL Service Change

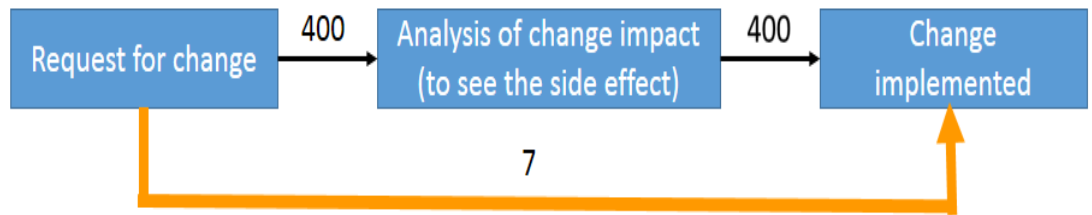


Figure 1.3: Subset of Information Technology Infrastructure Library (ITIL) service change flow

Figure 1.3 represents a small subset of activities and edges in ITIL service change. The number above the edges represents the frequency and the thickness of the edge

represents the utility of transition. In 7 cases, change is implemented without analyzing its impact on the system. The transition from ‘Request for Change’ and ‘Change Implemented’ is of high utility and low frequency (HULF). The change may lead to implementation of an unfavourable modification. The FM process discovery may miss this flow at high value of edge and node filter, but utility based process discovery will detect the edge between the ‘Request for Change’ and ‘Change Implemented’ as high utility.

1.3.1.2 Repair/Replacement Process

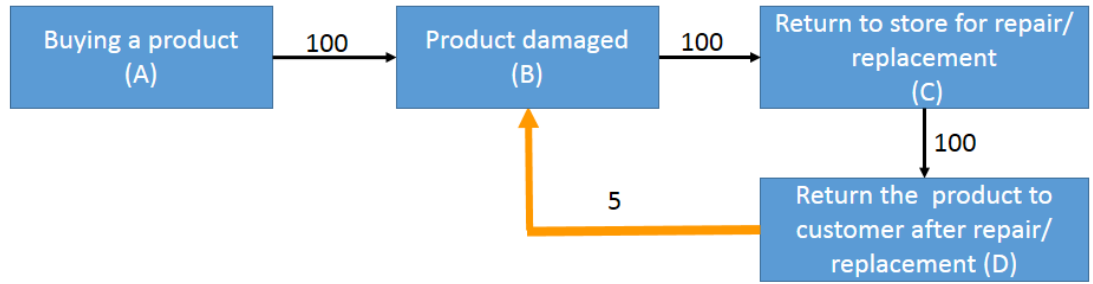


Figure 1.4: Subset of repair/replacement process flow

Figure 1.4 shows subset of return/replacement process flow. The number above the edges represents the frequency and the thickness of the edge represents the utility of transition. The edge from ‘D’ to ‘B’ represents out of 100 products which are returned to the customer after repair/replacement, 5 are still damaged. The flow is of high importance from the organizational perspective due to several reasons like defect in a particular product, fraudulent dealer, etc. The flow comes under the category of high utility and low frequency (HULF). Due to the low frequency this undesirable flow will be pruned in FM at high value of edge and node filter.

1.3.1.3 Purchasing Process Example from SAP

The example dataset of SAP has been taken from link <https://fluxicon.com/disco/>.

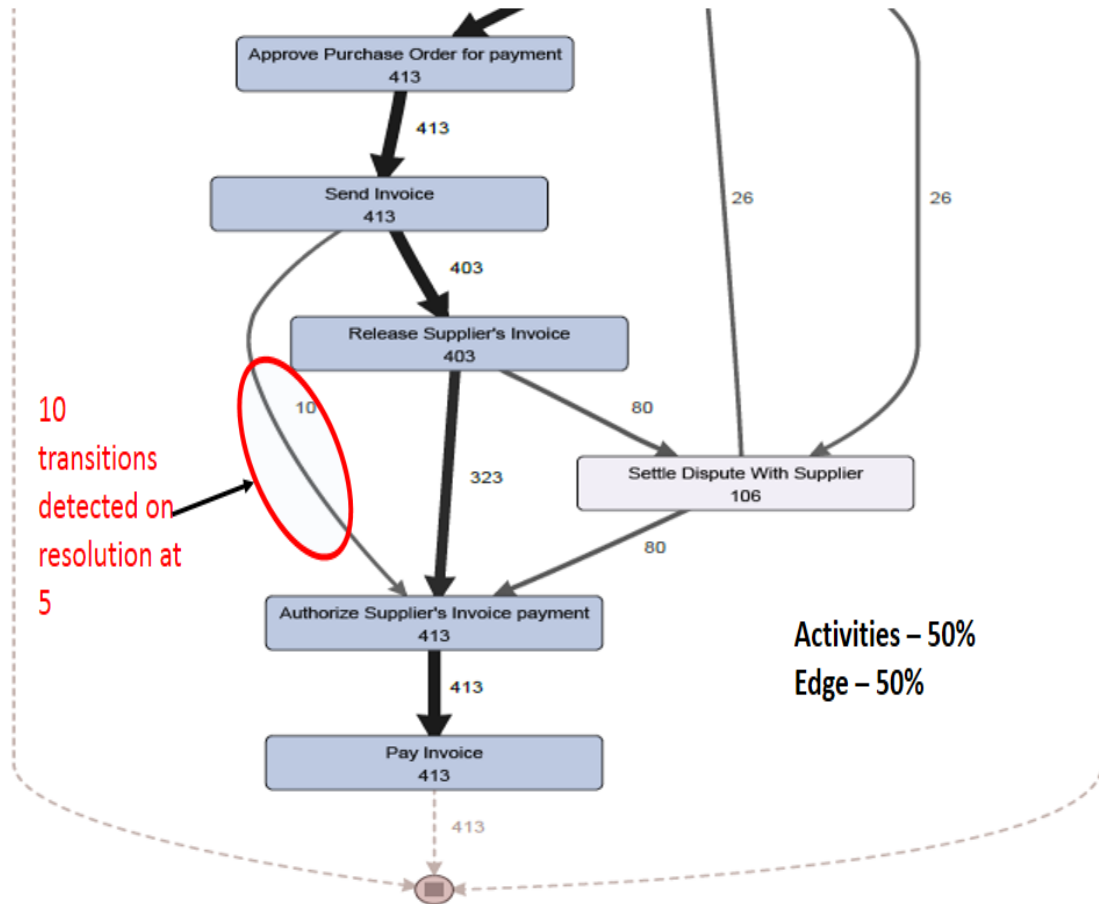


Figure 1.5: Subset of process model build using purchasing example from SAP

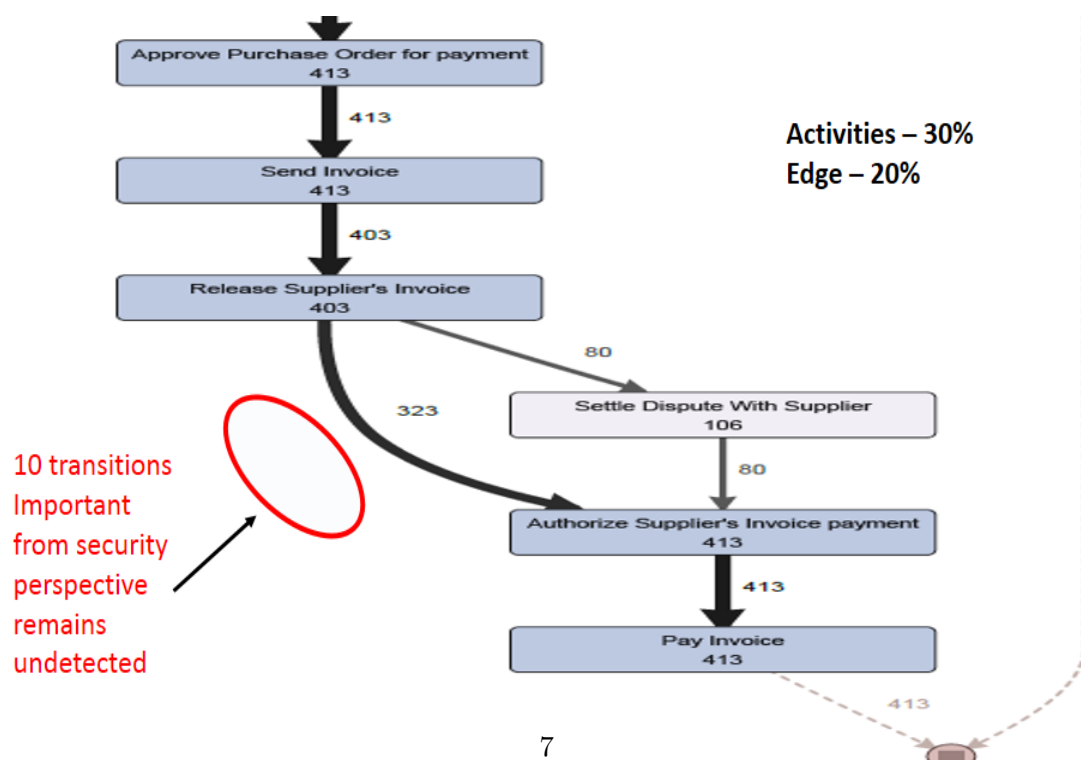


Figure 1.6: Subset of process model build using purchasing example from SAP

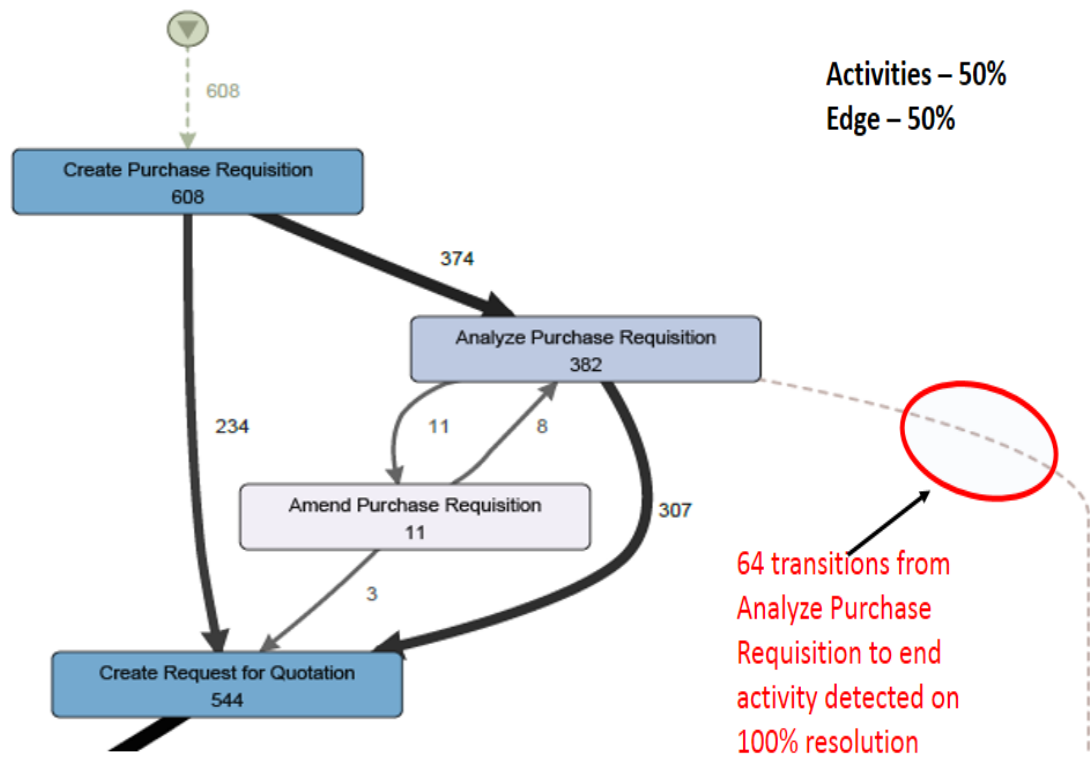


Figure 1.7: Subset of process model build using purchasing example from SAP

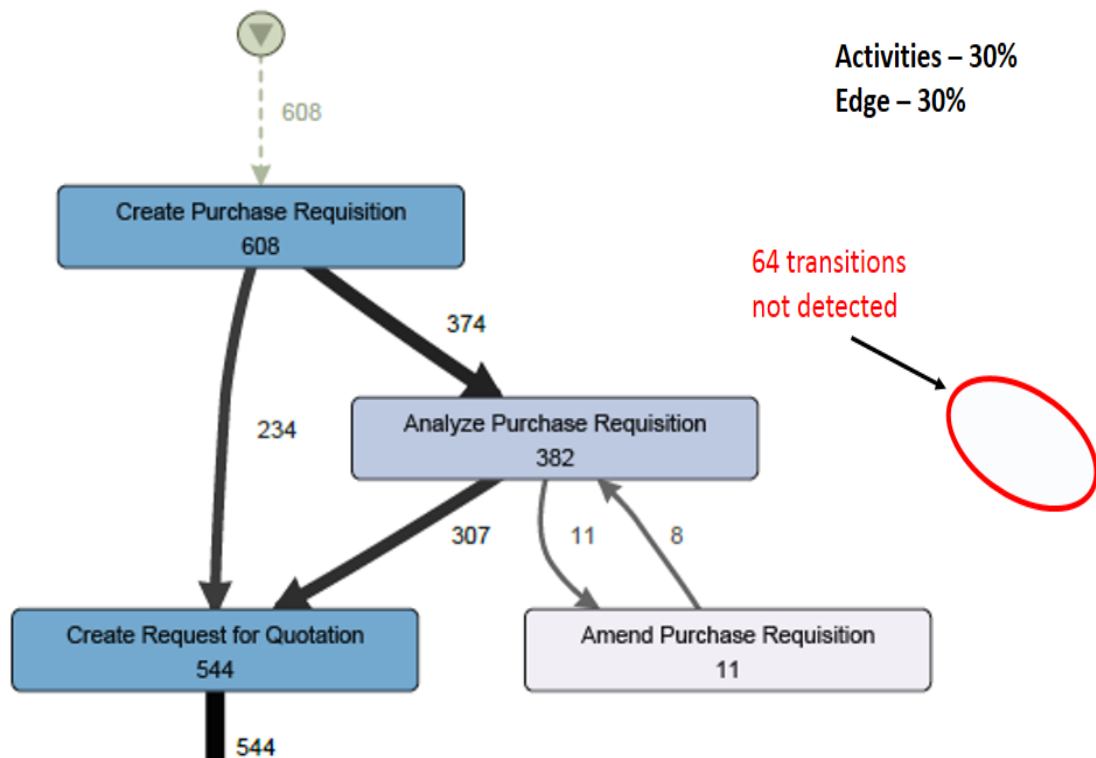


Figure 1.8: Subset of process model build using purchasing example from SAP

1. Figure 1.5 shows the subset of process model constructed using DISCO tool¹ at activities resolution set to 50% and path resolution set to 50%. Low percentage value of resolution in DISCO corresponds to the high value of edge and node filter. Increase in the resolution value on slider in DISCO lowers the value of edge and node filter. 100% resolution represents that all of the activities and paths are included in the process model. The red oval depicts 10 transitions from activity ‘Send Invoice’ to activity ‘Authorize Supplier Invoice Payment’. Figure 1.6 depicts that at activity resolution 30% and path resolution 20% these transitions which are important from security perspective remains undetected.
2. Figure 1.7 shows that there are 64 transitions from ‘Analyze Purchase Requisition’ to ‘End’ activity which are detected at 100% resolution. The flow is of high utility for the organization as it helps in detection of reasons for the termination of purchase requests before even request for quotation is initiated. But, when the activity and edge resolution is set to 30% these transitions get pruned as shown in Figure 1.8 .

The above examples shows the high utility and low frequency (HULF) cases. Real life event logs can contain the nodes and the edges which are of low utility to the organization are highly frequent. The low utility and high frequency (LUHF) cases will make the process model more complex and hard to analyze. For example, an activity which is responsible for saving the process state after every five activities [3]. Therefore, it is important to take into account both HULF and LUHF cases while simplifying the process model. This motivated us to switch from FM to UBFM.

The research aim of the work presented in this thesis is the following:

1. To investigate and analyze the existing FM algorithm for creating process models.
2. To demonstrate the importance of utility based process model in process mining.
3. To incorporate the utility concept in traditional FM algorithm, so as to come up with a new UBFM algorithm.

¹<https://fluxicon.com/disco/>

4. To demonstrate the effectiveness of our proposed approach, we conduct experiment on synthetic dataset and a large and real-world incident management data of an enterprise.

2

Related Work and Research Contributions

In this chapter we discuss previous work closely related to our research and list the novel research contributions of our work in context to already existing work.

2.1 Related Work

We divide related work into following two lines of research:

2.1.1 Process Mining

The first papers on process mining appeared in 1995, when Cook et al. [8][9][10] started to mine process models from event logs in the context of software engineering. They called it process discovery. Process mining in the business sense was first introduced 1998 by Agrawal et al. [11]. They called it workflow mining. Since then, many groups have focussed on mining process models.

Cook et al. [8][9][10] were the first ones to work on process mining. They aim at retrieving a complete and correct model, but a model that express the most frequent patterns in the log. Agrawal et al. [11] were the first ones to apply process discovery (or process mining, as we name it) in a business setting. The algorithm does not handle duplicate tasks and assumes that no task appears more than once in a process instance.

Van der Aalst et al. [6] prove that α -algorithm can learn an important class of workflow nets, called structured workflow nets, from complete event logs. The α -algorithm is

sensitive to noise and incompleteness of event logs. Moreover, the original α -algorithm was incapable of discovering short loops or non-local, non-free choice constructs. The heuristic miner by Weijters et al. [12] can be seen as an extension of the α -algorithm [6]. The heuristic miner can deal with noise and can be used to express the main behavior (i.e., not all the details and exceptions) registered in an event log. It supports the mining of all common constructs in process models (i.e., sequence, choice, parallelism, loops, invisible tasks and some kinds of non-freechoice), except for duplicate tasks. Therefore, the heuristic miner is a more robust algorithm. Finally, Günther and Van der Aalst [3] propose FM, an adaptive simplification and visualization technique based on significance and correlation measures to visualize the behavior in event logs at various levels of abstraction. The main contribution of FM is that it can also be applied to less structured, or unstructured processes of which the event logs cannot easily be summarized in concise, structured process models.

2.1.2 High Utility Itemset Mining

Agrawal et al. [13] propose apriori algorithm, it is used to obtain frequent itemsets from the database. This relies on the basic properties that all subsets of a frequent itemset are frequent and that all supersets of an infrequent itemset are infrequent. Wang et al. propose that items or transactions may be of varying importance to the user. For example, the itemset (Perfume, Diamond) may suggest higher potential profits to a sales manager than the itemset (Perfume, Lipstick) [14]. A pattern is of utility if its use by a person contributes to reaching a goal. Different people may have divergent goals concerning the knowledge that can be extracted from a dataset. For example, one person may be interested in finding all sales with high profit in a transaction dataset, while another may be interested in finding all transactions with large increases in gross sales. This kind of interestingness is based on user-defined utility functions in addition to the raw data [15]. Liu et al in [16] proposes a Two-phase algorithm for finding high utility itemsets. In Phase I, only the combinations of high transaction weighted utilization itemsets are added into the candidate set at each level during the level-wise search. In phase II, only one extra database scan is performed to filter the overestimated itemsets [17]. Shankar et al. [18] presents a novel algorithm Fast Utility Mining (FUM) which finds all high utility itemsets within the given utility constraint threshold. To generate different types of itemsets the authors also suggest a technique

such as Low Utility and High Frequency (LUHF) and Low Utility and Low Frequency (LULF), High Utility and High Frequency (HUHF), High Utility and Low Frequency (HULF) [17].

2.2 Novel Research Contributions

In context to existing work and to the best of our knowledge, the study presented in this thesis makes the following novel contributions:

1. While there has been work done on creating process models, we are the first to introduce the utility concept in process mining.
2. While the statistical aspect has been taken into account in FM, we transform the spaghetti like process model to comprehensible process model taking into account both the statistical (based on frequency) and semantic (based on user's goal) aspect into account.
3. An in-depth and focused empirical analysis on a real-world dataset (Rabobank Group ¹: Activity log for incidents) and synthetic dataset (Airport data ²) demonstrating the effectiveness of the proposed approach.

¹<http://data.3tu.nl/repository/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35>

²<https://dl.dropboxusercontent.com/u/48972351/AFDATASET.csv>

3

Research Framework and Solution Approach

3.1 Fuzzy-Miner

FM is a popular process mining algorithm for event logs which are less structured. FM provides high level view of process and hides undesired details.

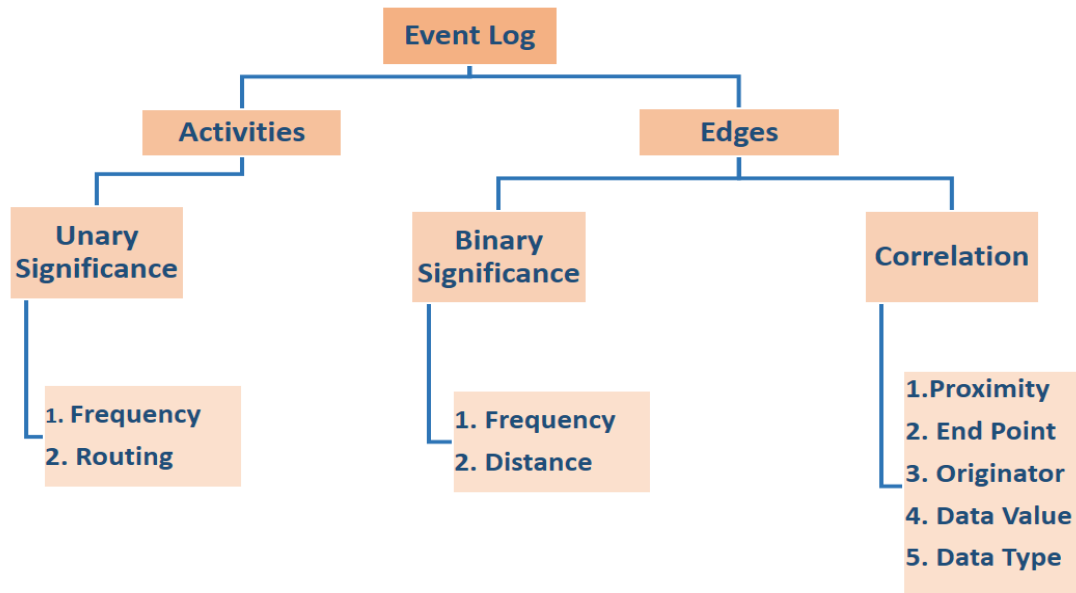


Figure 3.1: Significance and correlation metrics used in FM

FM uses following fundamental metrics [3] as shown in Figure 3.1.

1. **Significance** - It measures the relative importance of behaviour of activities (events) and edges. Higher the frequency of events and edges, higher is their significance.
2. **Correlation** - It measures how much two events following one another are closely related.

FM uses following initial parameters [3]:

1. **Preserve Threshold** (λ_p) - It specifies how significant two conflicting relations need to be in order to both be preserved. Two relations are said to be in conflict when two events in the process model are connected by the edges in both directions.
2. **Ratio Threshold** (λ_r) - If set to a high value, conflicting relations will rather be resolved by removing both relations from the graph. If set to a low value, conflicting relations will rather be preserved by removing only the weaker relation.
3. **Edge Filter** (ef) - Higher the ef value, lesser edges will be included in the process model.
4. **Node Filter** (nf) - All event classes with significance measure lower than nf value will be subjected to filtering.
5. **Utility Ratio** (ur) - ur specifies the weightage given to significance and correlation. ur value of 0.75 specifies that 75% significance and 25% correlation is taken into account.
6. **Attenuation** ($attenuation$) - It ensures that longer distance relationship affect the measurement less than the direct following relationship.
7. **Maximum Distance** ($distance$) - It measures the number of hops considered while detecting the edges within each trace. Each trace is ordered according to increasing order of timestamp. For example, corresponding to TraceID 'IM0000012', the sequence of activities are 'A;B;C;D;E;F;'. For distance equal to

4, the pair of activities considered for calculating binary significance and correlation are (A,B), (B,C), (A,C), (C,D), (B,D), (A,D), (D,E), (C,E), (B,E), (A,E), (E,F), (D,F), (C,F) and (B,F).

The pseudocode for FM algorithm is shown in Algorithm 1. It consists of 3 phases: Phase 1 involves calculation of unary significance for activities and binary significance and correlation for precedence relations. Phase 2 involves edge removal and node aggregation and abstraction. Phase 3 involves the construction of process model using the GraphViz ¹ API. The input to the algorithm is data comprising of Event Log, ef , nf , λ_p , λ_r , ur , $attenuation$ and $distance$. The algorithm returns process model as output. We choose $\lambda_p = 0.6$, $\lambda_r = 0.7$, $ur = 0.75$, $attenuation = \text{linear}$ and $distance = 4$ for our experiment. Functions *calculateRoutingSignificance()*, *calculateDistanceSignificance()*, *EdgeFiltering()* and *makeInitialClusters()* are borrowed from ProM ².

Steps 1-17 represent Phase 1. We calculate unaryFrequencySignificance, binaryFrequencySignificance and binaryCorrelation in Steps 4-8 for each trace within Event Log. binaryCorrelation is sum of Proximity, Originator, Data Value, Data Type and End Point correlations. We calculate routingSignificance and distanceSignificance by calling the Functions *calculateRoutingSignificance()* and *calculateDistanceSignificance()* respectively. Step 15 calculates binarySignificance metric as sum of binaryFrequencySignificance metric and distanceSignificance metric. Step 17 calculates unarySignificance metric as sum of unaryFrequencySignificance metric and routingSignificance metric.

Steps 18-31 represent Phase 2. Steps 19-22 perform conflict resolution for each precedence relation. *conflictResolution()* calculates the relative importance of edge in both the directions based on their binarySignificance values. If importance of both edges are greater than λ_p then both the edges are categorized as length-2 loop and signifies that both relations are important and thus need to be preserved, else they are categorized as exception or concurrency based on λ_r . Step 24 calculates utility of each edge using binarySignificance and binaryCorrelation based on ur . We identify nodes for aggregation or abstraction by calling Function *makeInitialClusters()* in Step 25. The nodes whose unarySignificance is less than nf are either aggregated or abstracted. Nodes who are less significant but highly correlated are the candidates for clustering. Edge filtering is done in Step 27 for each node. For every incoming edge to that node, we

¹<http://graphviz.org/>

²<https://svn.win.tue.nl/repos/prom/Packages/Fuzzy/>

normalize edge *util* value. If the normalized util value of edge is greater than *inLimit* then that precedence relation is preserved, else the relation is pruned. Same procedure is repeated for every outgoing edge from that node. Node aggregation occurs in Step 28 by calling the function *merge()*. The function call *merge()* checks all the predecessors of candidate cluster generated in Step 25. If all of the predecessors are clusters then the candidate cluster is merged with the most correlated one. Similarly, we check for all the successors. Node abstraction involves removal of isolated and singular clusters in Steps 29-30. Significance of clusters is average of significance of nodes contained in it.

Steps 32-35 represent Phase 3. We save the edges to be included in the process model in 'Graph.dot' file. Then, we construct the process model using GraphViz API.

FM can construct different process model at distinct values of edge and node filter. Business analyst can study the process model at desired level of abstraction.

Algorithm 1: Fuzzy-Miner Algorithm(Event Log, ef , nf , λ_p , λ_r , ur , $attenuation$, $distance$)

Data: Event Log

Result: A Process Model

```
1 create an empty 2-D arrayList binarySignificance, binaryCorrelation
2 create an empty 1-D arrayList unarySignificance
3 foreach Trace  $t$  in Event Log do
4   foreach Activity  $A$  in  $t$  do
5     unaryFrequencySignificance( $A$ ) += 1
6   foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  &  $\text{noOfHops}(A, B) < 4$  do
7     binaryFrequencySignificance( $A, B$ ) +=  $1 * \text{attenuation}$ 
8     binaryCorrelation( $A, B$ ) = Proximity( $A, B$ ) +
      Endpoint( $A, B$ ) + Datavalue( $A, B$ ) + Datatype( $A, B$ ) + Originator( $A, B$ )
9 Normalize the metrics unaryFrequencySignificance, binaryCorrelation,
  binaryFrequencySignificance
10 foreach distinct Activity  $A$  in Event Log do
11   calculateRoutingSignificance( $A$ , binaryFrequencySignificance,
    binaryCorrelation)
12 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
13   calculateDistanceSignificance( $A, B$ , binaryFrequencySignificance,
    unaryFrequencySignificance)
14 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
15   binarySignificance( $A, B$ ) = binaryFrequencySignificance( $A, B$ ) +
    DistanceSignificance( $A, B$ )
16 foreach Activity  $A$  in Event Log do
17   unarySignificance( $A$ ) = unaryFrequencySignificance( $A$ ) +
    routingSignificance( $A$ )
18 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
19   sigFwd = binarySignificance( $A, B$ )
20   sigRwd = binarySignificance( $B, A$ )
21   if sigFwd > 0 & sigRwd > 0 then
22     conflictResolution( $A, B$ ,  $\lambda_r$ ,  $\lambda_p$ , binarySignificance, binaryCorrelation)
23 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
24   util( $A, B$ ) =  $ur * \text{binarySignificance}(A, B) + (1 - ur) * \text{binaryCorrelation}(A, B)$ 
25 clusters = makeInitialClusters(Node  $A$ ,  $nf$ , unarySignificance, binaryCorrelation)
26 foreach Activity  $A$  in EventLog do
27   edgeFiltering( $A$ ,  $ef$ , binarySignificance, binaryCorrelation)
28 clusters = merge(clusters)
29 clusters = removeIsolatedClusters(clusters)
30 clusters = removeSingularClusters(clusters)
31 setSignificance(clusters)
32 Construct a "Graph.dot" file
33 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  & normalizedBinarySignificance( $A, B$ ) != 0 & normalizedBinaryCorrelation( $A, B$ ) != 0 do
34   Add the edge  $A \rightarrow B$  in Dot file
35 Make process model from "Graph.dot" using the GraphViz API
```

Function calculateRoutingSignificance(Activity i , binaryFrequencySignificance, binaryCorrelation)

```

1 inValue = 0
2 outValue = 0
3 foreach Activity  $x$  in Event Log do
4     sig = binaryFrequencySignificance( $x$ ,  $i$ )
5     cor = binaryCorrelation( $x$ ,  $i$ )
6     inValue += sig*cor
7     sig = binaryFrequencySignificance( $i$ ,  $x$ )
8     cor = binaryCorrelation( $i$ ,  $x$ )
9     outValue += sig*cor
10    if invalue == 0 & outvalue == 0 then
11        quotient = 0
12    else
13        quotient =  $\frac{inValue - outValue}{inValue + outValue}$ 
14        if quotient < 0 then
15            quotient = -quotient
16    RoutingSignificance[ $i$ ] = quotient

```

Function calculateDistanceSignificance(Activity x , Activity y , binaryFrequencySignificance, unaryFrequencySignificance)

```

1 sigSource = unaryFrequencySignificance[ $x$ ]
2 sigTarget = unaryFrequencySignificance[ $y$ ]
3 sigLink = binaryFrequencySignificance[ $x$ ][ $y$ ]
4 distanceSignificance[ $x$ ][ $y$ ] =  $1 - \frac{(sigSource - sigLink) + (sigTarget - sigLink)}{(sigSource + sigTarget)}$ 

```

Function EdgeFiltering(Node A , float ef , binarySignificance, binaryCorrelation)

```

1 Normalize the  $util(B, A)$  for every incoming edge to  $A$ 
2 if  $util(B, A) > ef$  then
3   | preserve the link
4 else
5   | remove the link
6 Repeat the same for every outgoing edge from  $A$ 

```

Function makeInitialClusters(Node A , nf , unarySignificance, binaryCorrelation)

```

1 if unarySignificance( $A$ )  $< nf$  then
2   | Add node  $A$  to victims
3   | neighbor = find most correlated neighbour
4   | if neighbor  $\in$  clusterNode then
5     | | Add  $A$  to clusterNode
6 else
7   | Create a new clusterNode
8   | Add  $A$  to clusterNode
9   | Add clusterNode to clusters
10  | Remove  $A$  from the graph

```

3.2 Utility Based Fuzzy-Miner

In traditional FM algorithm, the extracted process model consists of nodes and edges of equal value. However, in real-world applications, the actors, activities and transition between activities may not be of equal value. Therefore, we propose a UBFM algorithm to efficiently mine a process model driven by utility threshold. The utility is introduced in FM to mine for high utility process model by considering profit, value, quantity of activities and nodes.

UBFM uses following additional parameters:

1. **Node Utility** (nu) - Business process analyst/expert provides external values to activities in event log. The external value is given on the basis of benefit or usefulness to the organization in terms of profit, time, cost, security, etc.
2. **Edge Utility** (eu) - Business process analyst/expert provides external values to edges in event log. The value of a precedence relation signifies its importance in the process flow within an organization.

UBFM along with Node Utility (nu) and Edge Utility (eu) uses same set of initial parameters as traditional FM but with different semantic meaning. Node Filter (nf) and Edge Filter (ef) remove activities and edges according to their utility values. Higher the filter values, lesser activities and edges will be included in the process model.

We introduce two terminologies:

1. **Utility value of a single activity** - It is the product of frequency of activity and its nu value.
2. **Utility value of a single edge** - It is the product of frequency of an edge and its eu value.

The pseudocode for UBFM is shown in Algorithm 2. It consists of 3 phases: Phase 1 involves calculation of unary significance for activities and binary significance and correlation for precedence relations taking into consideration the statistical (based on frequency) and semantic (based on user's objective) aspects. Phase 2 involves edge removal and node aggregation and abstraction. Phase 3 involves the construction of process model using the GraphViz API. The input to the algorithm is data comprising of Event Log, ef , nf , eu , nu , λ_p , λ_r , ur , *attenuation* and *distance*. The algorithm

returns process model as output. We choose $\lambda_p = 0.6$, $\lambda_r = 0.7$, $ur = 0.75$, *attenuation* = linear and *distance* = 4 for our experiment.

Steps 1-17 represent Phase 1. We calculate unaryUtilitySignificance, binaryUtilitySignificance and binaryCorrelation in Steps 4-8 for each trace within Event Log. binaryCorrelation is sum of Proximity, Originator, Data Value, Data Type and End Point correlations. We calculate routingSignificance and distanceSignificance in Step 11 and 13. Step 15 calculates binarySignificance metric as sum of binaryUtilitySignificance metric and distanceSignificance metric. Step 17 calculates unarySignificance metric as sum of unaryUtilitySignificance metric and routingSignificance metric.

Steps 18-32 represent Phase 2. Steps 19-22 perform conflict resolution for each precedence relation using λ_p and λ_r . Step 24 calculates utility of each edge using binarySignificance and binaryCorrelation based on ur . We identify nodes for aggregation or abstraction by calling Function *findClusters* in Step 25. *edgefiltering()*, *merge()*, *removeIsolatedClusters()* and *removeSingularClusters()* work in similar manner as in FM taking into account the utility values instead of frequency. We remove the low utility clusters by calling the Function *removeLowUtilityClusters* in Step 31.

Steps 33-36 represent Phase 3. We save the edges to be included in the process model in 'Graph.dot' file. Then, we construct the process model using GraphViz API.

The UBFM differs from traditional FM in following aspects:

1. UBFM uses two additional input parameters: a) node utility and b) edge utility.
2. UBFM uses the utility values of activities and edges instead of frequency to calculate the metrics in Phase 1.
3. We add an additional function *removeLowUtilityClusters()* as shown in Function *removeLowUtilityClusters*. Low utility cluster means the cluster in which the sum of significance of nodes present in cluster is less than the node filter. *removeLowUtilityClusters()* removes low utility clusters from the process model by making a connection between predecessors and successors. The low utility clusters are highly correlated but unimportant from organizational perspective.
4. To reduce complexity of algorithm, we change the function *makeInitialClusters()* in FM to *findClusters()* in UBFM. *findClusters()* will find nodes which are

less significant but highly correlated. First, we generate candidate clusters during Steps 1-10 in findClusters. Second, we calculate the sum of significance of all nodes contained in all the candidate clusters. Third, if the sum is less than the nf value then there is no need to perform the Steps 29-31 in Algorithm 2 . This will reduce the computational overhead of UBFM.

5. UBFM will take into account the HULF and LUHF cases if the utility values of edges and activities are above threshold values while constructing the process model.

Algorithm 2: Utility Based Fuzzy Miner Algorithm(Event Log, ef , nf , eu , nu , λ_p , λ_r , ur , $attenuation$, $distance$)

Data: Event Log

Result: A Process Model

```
1 create an empty 2-D arrayList binarySignificance, binaryCorrelation
2 create an empty 1-D arrayList unarySignificance
3 foreach Trace  $t$  in Event Log do
4   foreach Activity  $A$  in  $t$  do
5     unaryUtilitySignificance( $A$ ) +=  $nu[A]$ 
6   foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  &  $noOfHops(A, B) < 4$  do
7     binaryUtilitySignificance( $A, B$ ) +=  $eu[A][B] * attenuation$ 
8     binaryCorrelation( $A, B$ ) = Proximity( $A, B$ ) +
      Endpoint( $A, B$ ) + Datavalue( $A, B$ ) + Datatype( $A, B$ ) + Originator( $A, B$ )
9 Normalize the metrics unaryUtilitySignificance, binaryCorrelation,
  binaryUtilitySignificance
10 foreach distinct Activity  $A$  in Event Log do
11   calculateRoutingSignificance( $A$ , binaryUtilitySignificance, binaryCorrelation)
12 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
13   calculateDistanceSignificance( $A, B$ , binaryUtilitySignificance,
    unaryUtilitySignificance)
14 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
15   binarySignificance( $A, B$ ) = binaryUtilitySignificance( $A, B$ ) +
    DistanceSignificance( $A, B$ )
16 foreach Activity  $A$  in Event Log do
17   unarySignificance( $A$ ) = unaryUtilitySignificance( $A$ ) ++
    routingSignificance( $A$ )
18 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
19   sigFwd = binarySignificance( $A, B$ )
20   sigRwd = binarySignificance( $B, A$ )
21   if sigFwd > 0 & sigRwd > 0 then
22     conflictResolution( $A, B$ ,  $\lambda_r$ ,  $\lambda_p$ , binarySignificance, binaryCorrelation)
23 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  do
24   util( $A, B$ ) =  $ur * binarySignificance(A, B) + (1-ur) * binaryCorrelation(A, B)$ 
25 clusters = findClusters(Node  $A$ ,  $nf$ , unarySignificance, binaryCorrelation)
26 foreach Activity  $A$  in EventLog do
27   edgeFiltering( $A$ ,  $ef$ , binarySignificance, binaryCorrelation)
28 clusters = merge(clusters)
29 clusters = removeIsolatedClusters(clusters)
30 clusters = removeSingularClusters(clusters)
31 clusters = removeLowUtilityClusters(clusters)
32 setSignificance(clusters)
33 Construct a "Graph.dot" file
34 foreach pair  $(A, B)$  where  $(A, B) \in \text{Activity}$  & normalizedBinarySignificance( $A, B$ ) != 0 & normalizedBinaryCorrelation( $A, B$ ) != 0 do
35   Add the edge  $A \rightarrow B$  in Dot file
36 Make process model from "Graph.dot" using the GraphViz API
```

Function findClusters(Node A , nf, unarySignificance, binaryCorrelation)

```

1 if unarySignificance( $A$ ) < nf then
2   | Add node  $A$  to victims
3   | neighbor = find most correlated neighbour
4   | if neighbor  $\in$  clusterNode then
5   |   | Add  $A$  to clusterNode
6 else
7   | Create a new clusterNode
8   | Add  $A$  to clusterNode
9   | Add clusterNode to clusters
10  | Remove  $A$  from the graph
11 sum = 0
12 foreach clusterNode  $c$  in clusters do
13   | foreach Node  $node$  in  $c$  do
14   |   | sum = sum + node.significance
15 if sum < nf then
16   | Remove all the clusters

```

Function calculateDistanceSignificance(Activity x , Activity y , binaryUtilitySignificance, unaryUtilitySignificance)

```

1 sigSource = unaryUtilitySignificance[ $x$ ]
2 sigTarget = unaryUtilitySignificance[ $y$ ]
3 sigLink = binaryUtilitySignificance[ $x$ ][ $y$ ]
4 distanceSignificance[ $x$ ][ $y$ ] = 1 -  $\frac{(sigSource - sigLink) + (sigTarget - sigLink)}{(sigSource + sigTarget)}$ 

```

Function removeLowUtilityClusters(Node A , nf , unarySignificance, binaryCorrelation)

```

1 foreach clusterNode  $c$  in clusters do
2   sum=0
3   foreach Node  $node$  in  $c$  do
4     sum = sum + node.significance
5   if sum <  $nf$  then
6     Remove cluster  $c$  from graph

```

Function calculateRoutingSignificance(Activity i , binaryUtilitySignificance, binaryCorrelation)

```

1 inValue = 0
2 outValue = 0
3 foreach Activity  $x$  in Event Log do
4   sig = binaryUtilitySignificance( $x$ ,  $i$ )
5   cor = binaryCorrelation( $x$ ,  $i$ )
6   inValue += sig*cor
7   sig = binaryUtilitySignificance( $i$ ,  $x$ )
8   cor = binaryCorrelation( $i$ ,  $x$ )
9   outValue += sig*cor
10  if invalue == 0 & outvalue == 0 then
11    quotient = 0
12  else
13    quotient =  $\frac{inValue - outValue}{inValue + outValue}$ 
14    if quotient < 0 then
15      quotient = -quotient
16  RoutingSignificance[ $i$ ] = quotient

```

Function EdgeFiltering(Node A , float ef , binarySignificance, binaryCorrelation)

- 1 Normalize the $\text{util}(B, A)$ for every incoming edge to A
 - 2 **if** $\text{util}(B, A) > ef$ **then**
 - 3 └ preserve the link
 - 4 **else**
 - 5 └ remove the link
 - 6 Repeat the same for every outgoing edge from A
-

4

Experimental Analysis and Results

4.1 Experimental Dataset

We conduct our experiment on 2 datasets:

- (a) Large real world data from Business Process Intelligence 2014 (BPI 2014).
- (b) Small synthetic Airport dataset depicting the behavior of passenger at airport.

4.1.1 BPI 2014 Dataset

BPI 2014 dataset consists of large real world data from Rabobank Group Information and Communication Technology (ICT). The data is related to the Information Technology Infrastructure Library (ITIL) process implemented in the Bank. The dataset is provided in the CSV format. It contains the event logs from interactions records, incidents records, incident activities and change records. The provided dataset is of six month duration from January, 2013 - March, 2014. The attributes of original .CSV files are converted to appropriate data types, such as standardized timestamp formats for analysis. After loading the data on to MySQL database, we build four tables: Interaction detail, Incident detail, Incident activity detail and Change detail. We analyze all the tables and amongst

them we choose Incident activity detail table to build process models. The main reason for choosing this table is because it has information regarding the type of activities performed on a particular incident id and also the timestamp when this incident activity type started.

Table 4.1: BPI 2014 Dataset Detail of Table Incident Activity Detail

	Incident activity detail
Traces	46,606
Events	466,737
Start Timestamp	07.01.2013
End TimeStamp	02.04.2014
Incident Activity Type	39
Assignment Group	242

Table 4.1 indicates the statistics of Incident activity detail table. Each record contains an TraceID with the activities performed on it. It also contains information about the Assignment Group that is responsible for a particular activity. The attribute IncidentActivity Type represents the type of activity performed on the incident. There are 39 unique activities. Some of the examples are: Assignment (ASG), Status Change (STC), Update (UPD), Referred (REF), Problem Closure (PC), OOResponse (OOR), Dial-In (DI) and Contact Change (CC). Figure 4.1 represents the Pareto chart showing the distribution of activities and their cumulative count. The Y-axis is in logarithmic scale. Figure 4.2 shows the distribution of the case invariants. Figure 4.2 indicates that the dataset consists of a long of small number of case invariants. For a particular incident we order the activities according to increasing order of DateTime Stamp to make the event log more structured. This is done for all the unique incidents in the Incident activity detail table. Figure 4.3 shows the screenshot from MySQL of activities performed during one of the incidents ‘IM0000004’.

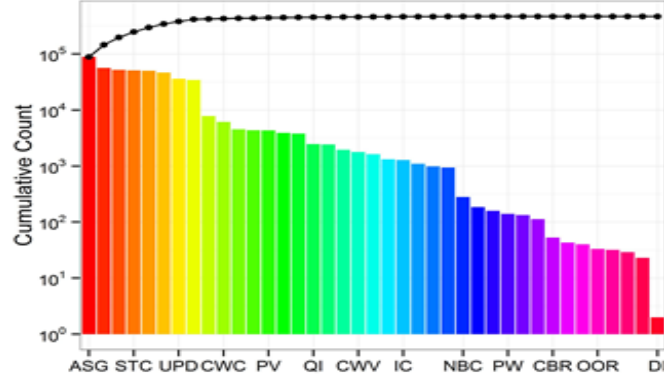


Figure 4.1: Pareto chart showing the distribution of activities and their cumulative count. Y-axis is in logarithmic scale

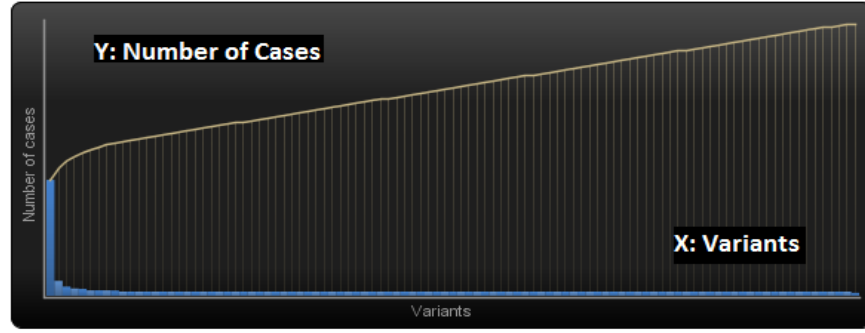


Figure 4.2: Histogram for case variants

4.1.2 Airport Dataset

We perform experiment on small synthetic dataset shown in Table 4.2 to illustrate the working of UBFM. Table 4.3 shows the sequence of activities performed during one of the traces ‘Trace3.0’.

Table 4.2: Airport Dataset Detail

	Airport Dataset Detail
Traces	8
Events	47
Activity Type	7
# of Originators	7

4.1 Experimental Dataset

IncidentID	DateStamp	IncidentActivity_Number	IncidentActivity_Type	AssignmentGroup	KMnumber	InteractionID	IncidentActivity_Type_Number
IM0000004	2013-01-07 08:17:17	001A3689763	Reassignment	TEAM0001	KM0000553	SD0000007	27
IM0000004	2013-11-04 13:41:30	001A5852941	Reassignment	TEAM0002	KM0000553	SD0000007	27
IM0000004	2013-11-04 13:41:30	001A5852943	Update from customer	TEAM0002	KM0000553	SD0000007	34
IM0000004	2013-11-04 12:09:37	001A5849980	Operator Update	TEAM0003	KM0000553	SD0000007	20
IM0000004	2013-11-04 12:09:37	001A5849979	Assignment	TEAM0003	KM0000553	SD0000007	2
IM0000004	2013-11-04 13:41:30	001A5852942	Assignment	TEAM0002	KM0000553	SD0000007	2
IM0000004	2013-11-04 13:51:18	001A5852172	Closed	TEAM0003	KM0000553	SD0000007	5
IM0000004	2013-11-04 13:51:18	001A5852173	Caused By CI	TEAM0003	KM0000553	SD0000007	4
IM0000004	2013-11-04 12:09:37	001A5849978	Reassignment	TEAM0003	KM0000553	SD0000007	27
IM0000004	2013-09-25 08:27:40	001A5544096	Operator Update	TEAM0003	KM0000553	SD0000007	20

↓
Order activities according to date timestamp
↓

IncidentID	DateStamp	IncidentActivity_Number	IncidentActivity_Type	AssignmentGroup	KMnumber	InteractionID	IncidentActivity_Type_Number
IM0000004	2013-01-07 08:17:17	001A3689763	Reassignment	TEAM0001	KM0000553	SD0000007	27
IM0000004	2013-09-25 08:27:40	001A5544096	Operator Update	TEAM0003	KM0000553	SD0000007	20
IM0000004	2013-11-04 12:09:37	001A5849980	Operator Update	TEAM0003	KM0000553	SD0000007	20
IM0000004	2013-11-04 12:09:37	001A5849979	Assignment	TEAM0003	KM0000553	SD0000007	2
IM0000004	2013-11-04 12:09:37	001A5849978	Reassignment	TEAM0003	KM0000553	SD0000007	27
IM0000004	2013-11-04 13:41:30	001A5852941	Reassignment	TEAM0002	KM0000553	SD0000007	27
IM0000004	2013-11-04 13:41:30	001A5852943	Update from customer	TEAM0002	KM0000553	SD0000007	34
IM0000004	2013-11-04 13:41:30	001A5852942	Assignment	TEAM0002	KM0000553	SD0000007	2
IM0000004	2013-11-04 13:51:18	001A5852172	Closed	TEAM0003	KM0000553	SD0000007	5
IM0000004	2013-11-04 13:51:18	001A5852173	Caused By CI	TEAM0003	KM0000553	SD0000007	4

Figure 4.3: Activities ordered according to increasing order of datetime stamp for IncidentId *IM0000004*

Table 4.3: Snapshot of Airport Dataset for Trace3.0

TraceID	DateStamp	ActivityName	Originator
Trace3.0	04-06-2009 17:21:03	Show Identification Proof	John
Trace3.0	04-06-2009 17:22:03	Luggage Check	Peter
Trace3.0	04-06-2009 17:23:03	Collect Boarding Pass	Katty
Trace3.0	04-06-2009 17:24:03	Security Check	Radha
Trace3.0	04-06-2009 17:25:03	Wait	Adolf
Trace3.0	04-06-2009 17:25:03	Board Flight	Om

4.2 Experimental Results

4.2.1 Airport Dataset Results

We perform experiment on Airport dataset to show working of UBFM. We assign labels to activities in dataset as shown in Table 4.4. Two of the inputs to UBFM is edge utility and node utility. We give default utility for all activities and precedence relation between activities as 5. We assign external utility to some of activities and relations as shown in Table 4.5.

Table 4.4: Labels Corresponding to Activity Name in Airport Dataset

Activity Name	Label
Show Identification Proof	0
Luggage Check	1
Collect Boarding Pass	2
Security Check	3
Wait	4
Board Flight	5
Enquiry	6

Table 4.5: External Utility of Activities and Precedence Realties

Activity / Precedence Relation	Utility
Enquiry	20
Wait	0
Luggage Check ->Enquiry	20
Luggage Check ->Board Flight	19
Show Identification Proof ->Security Check	1
Collect Boarding Pass ->Board Flight	20

Phase 1 involves calculation of unarySignificance, binarySignificance and binaryCorrelation as explained in Algorithm 2. First, we compute unaryUtilitySignificance metric by using frequency and node utility. Figure 4.4 depicts normalized unaryUtilitySignificance metric. Second, we calculate binaryUtilitySignificance of a precedence relation. The utility of a precedence relation is equal to sum of product of utility and attenuation factor. It takes into account frequency of


Activity Name	Frequency * /unit Utility		Activity Name	Utility
Show Identification Proof	$8*5 = 40$	Normalization And Assigning the weight 	Show Identification Proof	1
Luggage Check	$7*5 = 35$		Luggage Check	0.875
Collect Boarding Pass	$8*5 = 40$		Collect Boarding Pass	1
Security Check	$7*5 = 35$		Security Check	0.875
Wait	$8*0.01 = 0.08$		Wait	0.002
Board Flight	$8*5 = 40$		Board Flight	1
Enquiry	$1*20 = 20$		Enquiry	0.5

Figure 4.4: Normalized unary utility significance for Airport dataset

a relation, utility of a relation and attenuation factor (so that longer distance relations affect the measurement less than direct following relationships).

For e.g. :- **For Luggage Check (1) ->Board Flight (5)**

$$= 0.5*19 + 19 * 0.25 + 19 * 0.25 + 19 * 0.25 + 19 * 0.25 + 19 * 0.25$$

$$= 33.25$$

For e.g. :- **Show Identification Proof (0) ->Collect Boarding Pass (2)**

$$= 1*5 + 5*0.75 + 5*0.75 + 5*0.75 + 5*0.75 + 5*0.75 + 5*0.75 + 5*0.75$$

$$= 31.25$$

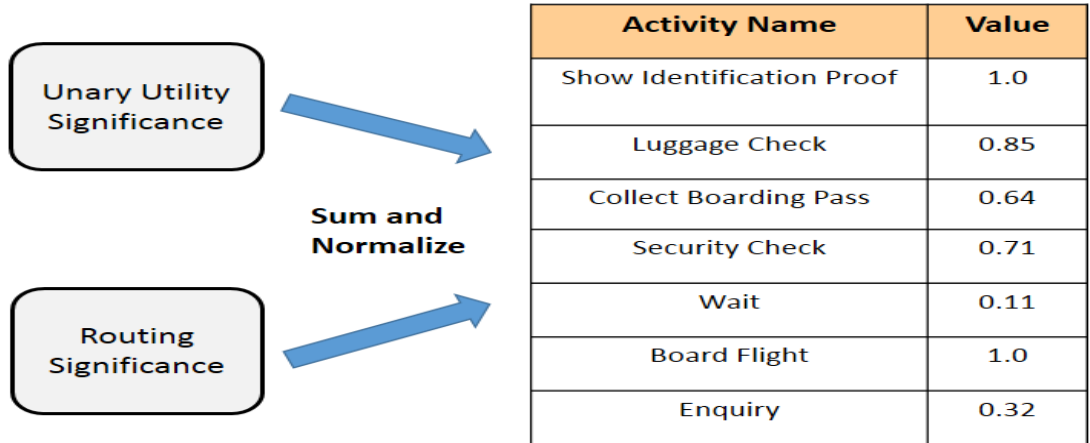
Table 4.6: Binary Utility Significance Metric for Airport Dataset

	0	1	2	3	4	5	6
0	0	35.0	31.25	3.75	11.25	2.5	1.25
1	0	0	35.0	22.5	17.5	33.25	10.0
2	0	0	0	35.0	30.0	80.0	3.75
3	0	0	0	0	33.75	25.0	5.0
4	0	0	0	0	0	40.0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	5.0	3.75	0

Table 4.7: Binary Correlation Metric for Airport Dataset

	0	1	2	3	4	5	6
0	0	0.11	0.353	0.926	0.749	0.235	0.176
1	0	0	0.701	0.631	0.11	0.113	3.5e-6
2	0	0	0	0.912	0.140	0.35	0.070
3	0	0	0	0	0.210	0.105	0.947
4	0	0	0	0	0	1.0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0.842	0.754	0

Similarly, we calculate binaryUtilitySignificance for other relations. The resultant binaryUtilitySignificance metric as shown in Table 4.6. Third, we calculate binaryCorrelation as the sum of Proximity, Endpoint, Data Value, Data Type and Originator correlation. Table 4.7 shows normalized binaryCorrelation metric. Fourth, we compute normalized unarySignificance and binarySignificance as shown in Figure 4.5 and Figure 4.6 respectively.

**Figure 4.5:** Unary significance metric for Airport dataset

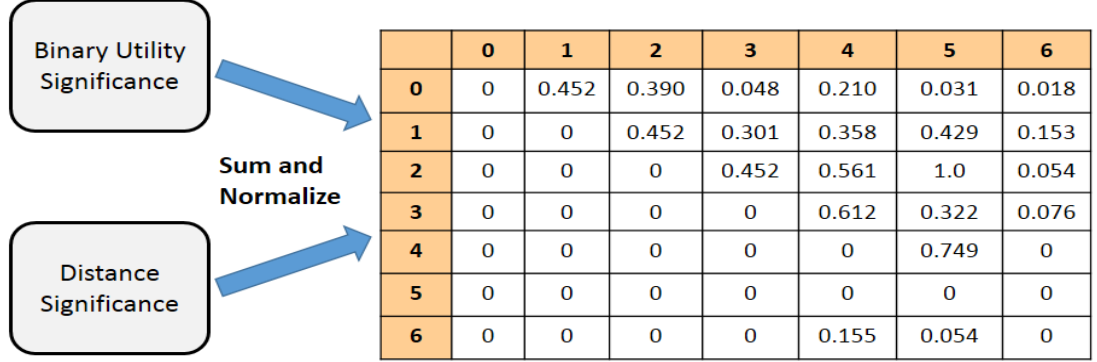


Figure 4.6: Binary significance metric for Airport dataset

We create process models at distinct values of edge and node filter after applying UBFM to the Airport dataset. The value inside the nodes represents the significance and the value on edges represents the *util* value. Process models are described as follows:

- (a) Figure 4.7 depicts the process model at node filter = 0.0 and edge filter = 0.0. The process model includes all activities and relations of the event log.

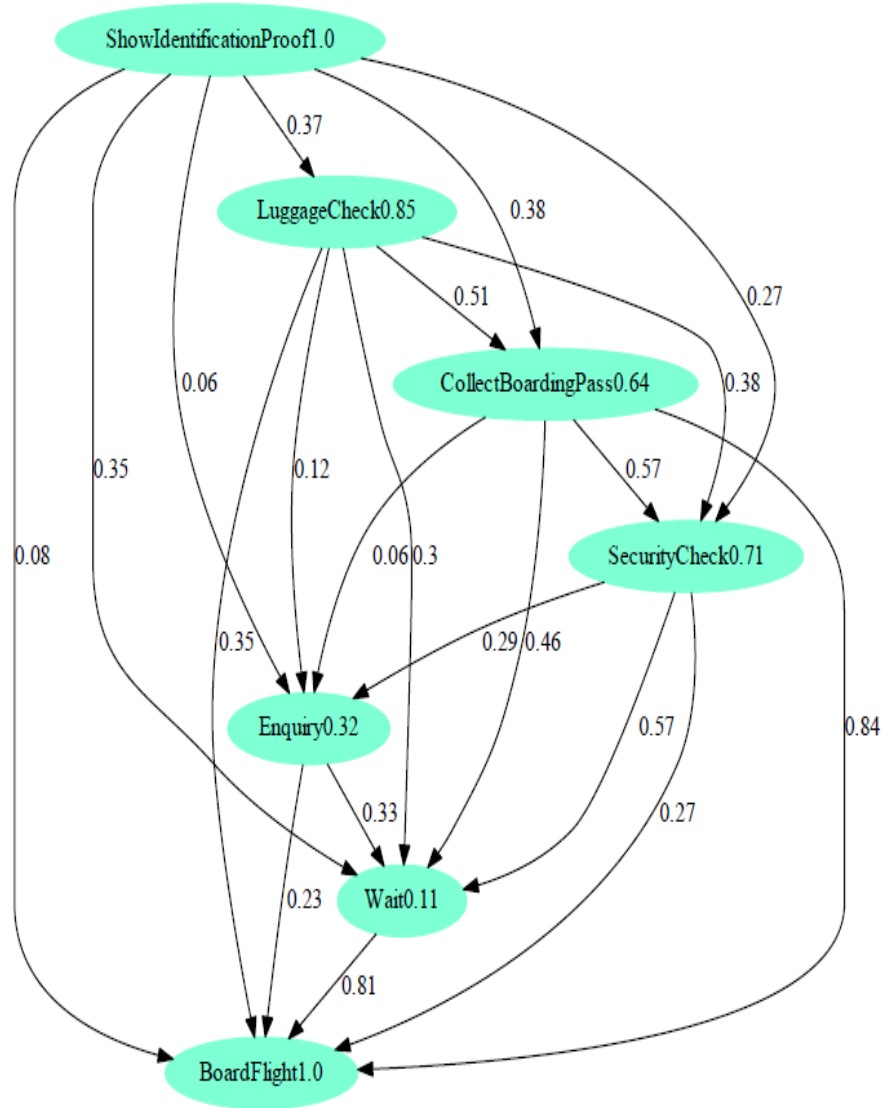


Figure 4.7: Process model created using UBFM at edge filter = 0.0 and node filter = 0.0

(b) Figure 4.8 shows the process model at node filter = 0.25 and edge filter = 0.25.

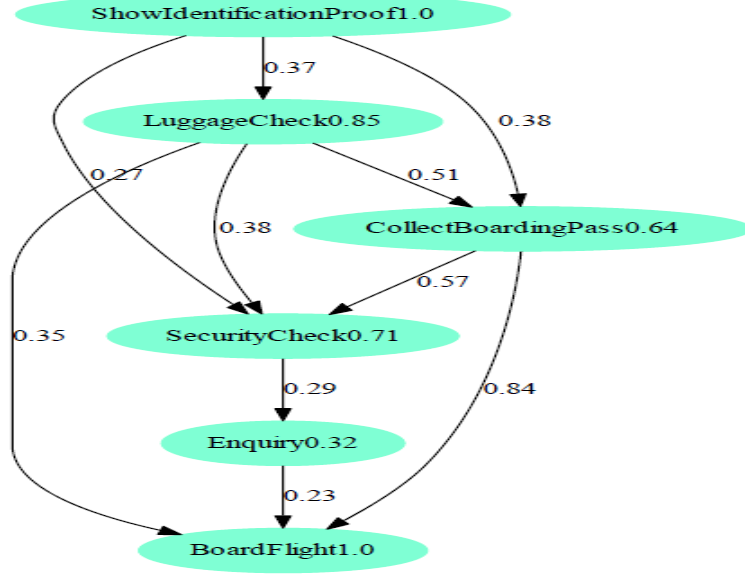


Figure 4.8: Process model created using UBFM at edge filter = 0.25 and node filter = 0.25

(c) Figure 4.9 depicts the process model at node filter = 0.5 and edge filter = 0.5.

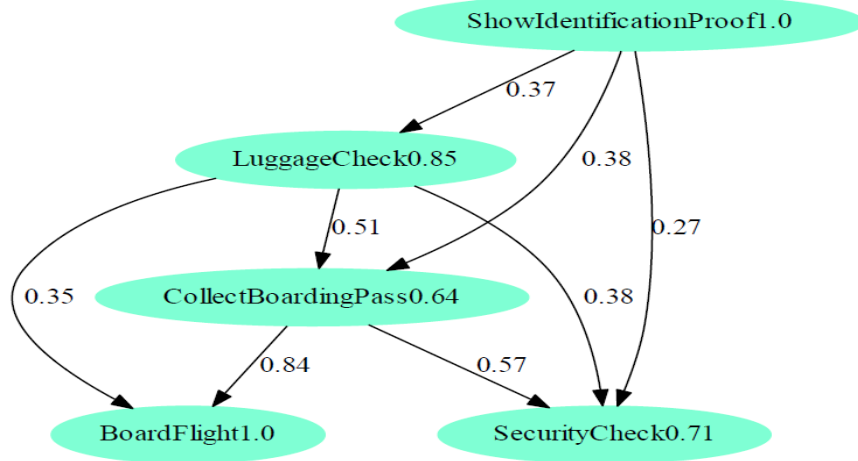


Figure 4.9: Process model created using UBFM at edge filter = 0.5 and node filter = 0.5

(d) Figure 4.10 shows the process model at node filter = 0.75 and edge filter = 0.75.

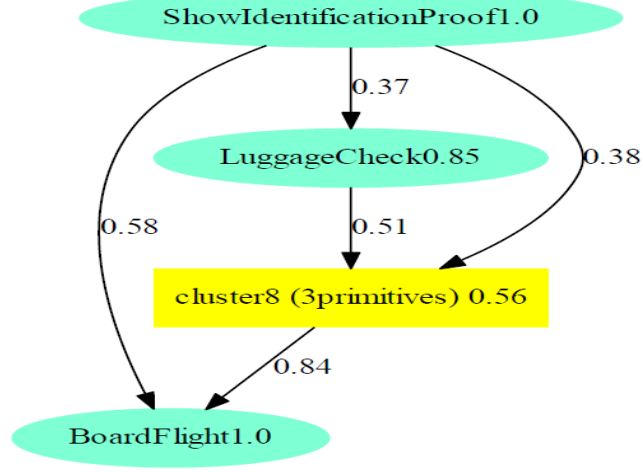


Figure 4.10: Process model created using UBFM at edge filter = 0.75 and node filter = 0.75

The process model created by UBFM and traditional FM differs. Some of the points are as follows:

- (a) Node 'Wait' gets pruned in UBFM but it is there in FM at node filter = 0.25 and edge filter = 0.25 as shown in Figure 4.8 and 4.11 respectively. The reason for removal of node is that 'Wait' is a LUHF node. Unary significance of 'Wait' in UBFM is 0.11 whereas, its unary significance in FM is 0.61.
- (b) Node 'Enquiry' gets pruned in FM at node filter = 0.25 and edge filter = 0.25 but it is present in UBFM as shown in Figure 4.8 and 4.11 respectively. The reason is that 'Enquiry' is a HULF node, it occurs only 1 time in full dataset. Unary significance of 'Enquiry' in UBFM is 0.32 whereas, its unary significance in FM is 0.14.
- (c) Edge 'Luggage Check' -> 'Board Flight' gets pruned in FM at node filter = 0.25 and edge filter = 0.25 as shown in Figure 4.8 and 4.11 respectively. The distance from 'Luggage Check' to 'Board Flight' is high as compared to other edges. Due to low attenuation its binary frequency significance is less. Therefore, it is pruned by FM but it is present in UBFM due to its high utility.

- (d) Edge ‘Show Identification Proof’ ->‘Security Check’ has utility value of 1. The edge is present at edge filter = 0.25 and node filter = 0.25 but gets pruned at edge filter = 0.75 and node filter = 0.75 in UBFM as shown in Figure 4.8 and 4.10 respectively. Despite of having low utility, edge is still present in the process model because of its very high utility value. It depicts that the edge is of high importance to the organization. For example, utility threshold is defined by vendor as Rs 40. A supermarket vendor sells 100 breads at a profit of Rs 1/bread. Utility value of bread or total profit to the vendor by selling bread is Rs 100. Despite of having low per unit utility, it is still categorized as a profitable item.

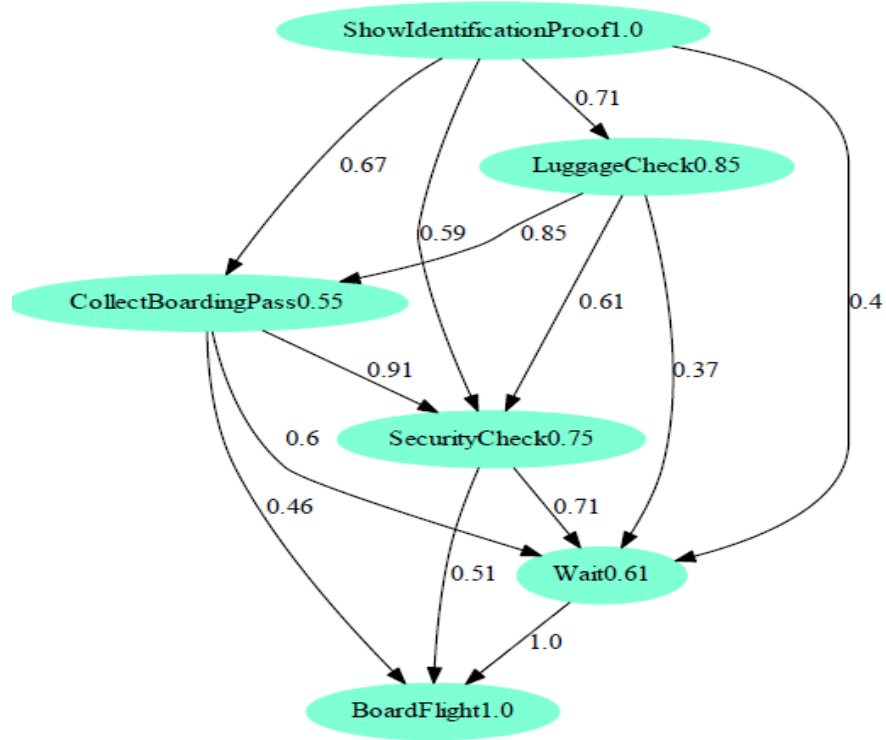


Figure 4.11: Process model created using FM at edge filter = 0.25 and node filter = 0.25

4.2.2 BPI 2014 Dataset Results

We perform experiment on large BPI 2014 dataset using UBFM and FM. We choose $\lambda_p = 0.6$, $\lambda_r = 0.7$, $ur = 0.75$, *attenuation* = linear and *distance* = 4 for

4.2 Experimental Results

Table 4.8: External Utility Provided by Business Process Analyst

Activities / Precedence Relation	Frequency	Per-unit External Utility
Quality Indicator Fixed	7,791	25
Status Change	50,914	0.5
Update	35,969	0.5
Assignment ->Closed	8,082	1.0

our experiment. Two of the inputs to UBFM are edge utility (eu) and node utility (nu). We provide default utility for all activities and relation between activities as 5, to give them equal importance initially. We assign external utility to some of activities and relations as shown in Table 4.8. We construct process models at distinct values of edge and node filter after applying UBFM in Algorithm 2.

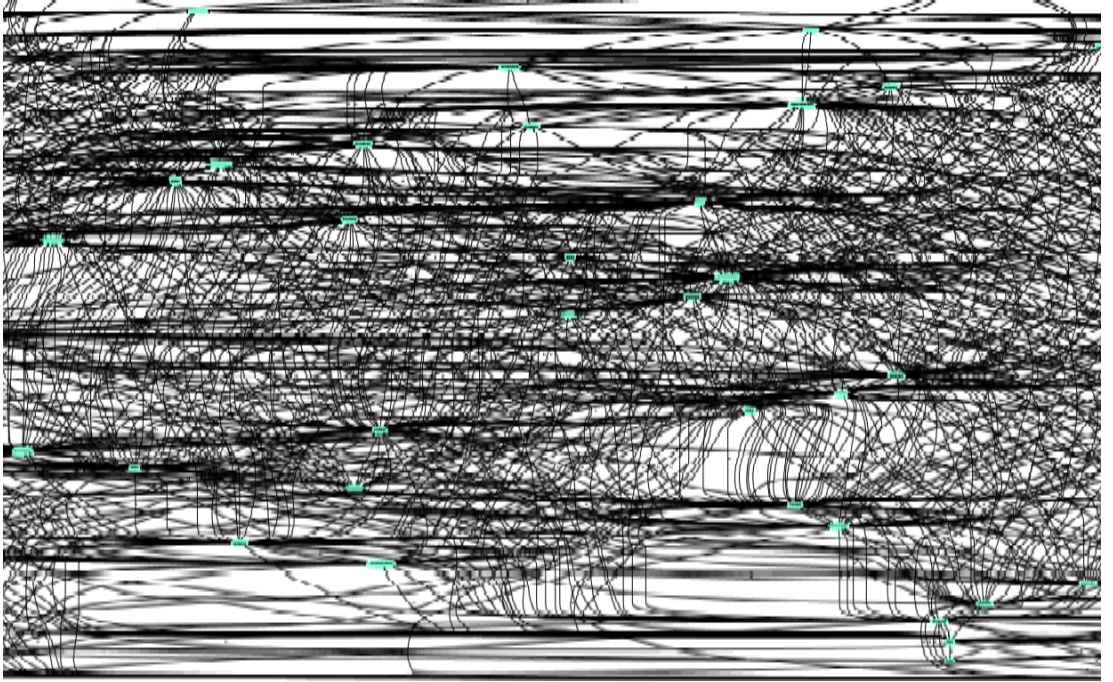


Figure 4.12: Process model derived from event-logs in BPI 2014 dataset discovered using UBFM at edge filter = 0.0 and node filter = 0.0

The value inside the nodes represents the significance and the value on edges represents the *sigCor* value as depicted in Figure 4.13. UBFM discovers a complex spaghetti like process model at node filter = 0.0 and edge filter = 0.0 as shown

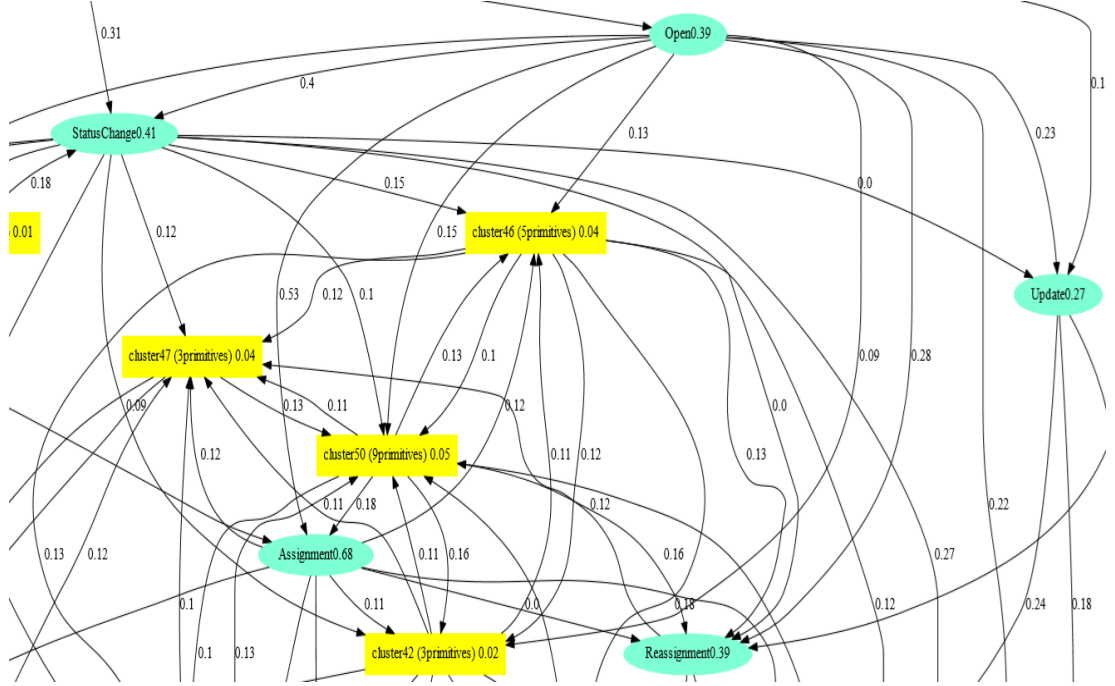


Figure 4.14: Snapshot of process model derived from event-logs in BPI 2014 dataset discovered using FM at edge filter = 0.5 and node filter = 0.25

4.13. Function *removeLowUtilityclusters()* removes low utility clusters from process model which further simplifies process model based on economic utility in Figure 4.13.

The line chart in Figure 4.15 depicts number of edges present after different steps in UBFM as values of edge and node filter increases. There are 952 distinct precedence relations present in the event log within a distance of 4. We make two important inferences from the graph in Figure 4.15. Firstly, we observe that for a particular function how the number of edges present in process model varies with increase in node and edge filter values. Secondly, we also notice that when we build a process model corresponding to a node and edge filter value, how the edges are pruned after each step. Removal of edges in *conflictResolution()* is independent of node and edge filter values but dependent on λ_p and λ_r . Therefore, a straight line is present for *conflictResolution()* in Figure 4.15.

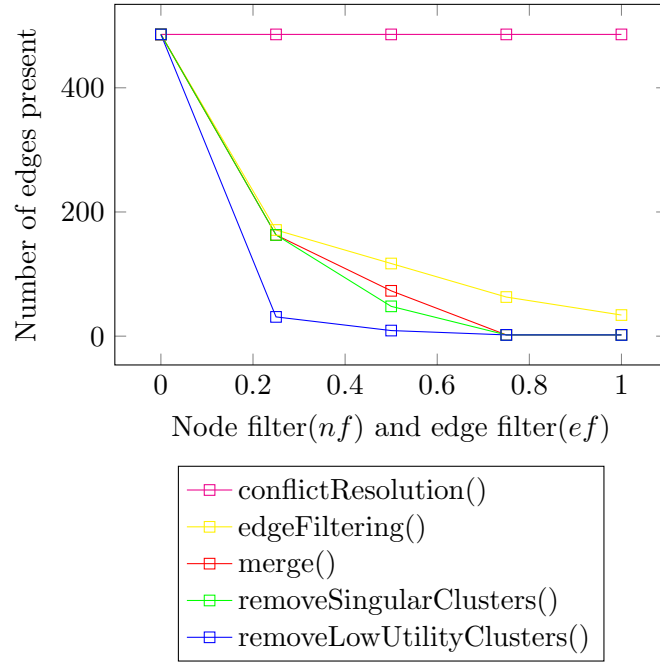


Figure 4.15: A line chart shows decrease in number of edges at different steps in UBFM at increasing value of node and edge filter

5

Limitations and Future Work

UBFM takes as input lots of initial parameters, so lot of time and effort goes in finding the values for that parameters. Future work includes finding the default settings.

We apply utility concept in control flow discovery. Future work includes application of utility concept in organizational and data flow perspective.

UBFM can be made more efficient in terms of time and space complexity.

The Fuzzy miner algorithm requires several parameters such as λ_p , λ_r , *utilityratio*, *attenuation* and *distance*. Currently we have taken default values $\lambda_p = 0.6$, $\lambda_r = 0.7$, *utilityratio* = 0.75, *attenuation* = linear and *distance* = 4. One of the limitations of the work is that we did not try for a large number of different parameter values. In future we plan to conduct experiments to gain insights on optimal parameter values for certain data characteristics.

Quantifying utility is non-trivial. In our current approach, we have not discussed challenges and solutions associated with eliciting utility for activities and events. In future we plan to propose practical solutions enabling a business or process analyst to specify utility in different domains.

In our work we have used real world BPI challenge 2014 dataset which is not a very large dataset, consisting of only 466737 records. Therefore, in future we plan to validate our approach on bigger and other publicly available datasets. ^{1 2}.

¹<http://www.win.tue.nl/bpi/2012/challenge>

²<http://www.win.tue.nl/bpi/2013/challenge>

6

Conclusion

The traditional FM simplifies spaghetti models based on frequency of activities and precedence relationships. The amount of detail to be shown in the output of the traditional FM algorithm can be adjusted using node and edge filters. Increasing the value of node and edge filters removes activities and relationships which have low significance and low correlation. In traditional FM algorithm, the extracted process model consists of nodes and edges of equal value. However, real life event logs often contain activities and precedence relations which are HULF and LUHF. Therefore, the concept of utility of activities and precedence relations should also be taken into account while simplifying process models. We incorporate the utility concept in traditional FM algorithm, so as to come up with a new UBFM algorithm. We perform experiment on Airport synthetic dataset to show detailed working of UBFM. Also, we conduct a series of experiments on real-world dataset to demonstrate that the proposed approach is effective. We observe that UBFM mines successfully the high utility activities and precedence relations within utility threshold.

References

- [1] WIL VAN DER AALST. **Process mining: Overview and opportunities.** *ACM Transactions on Management Information Systems (TMIS)*, **3**(2):7, 2012. ix, 3
- [2] WIL MP VAN DER AALST. **Process-aware information systems: Lessons to be learned from process mining.** In *Transactions on petri nets and other models of concurrency II*, pages 1–26. Springer, 2009. 1
- [3] CHRISTIAN W GÜNTHER AND WIL MP VAN DER AALST. **Fuzzy mining—adaptive process simplification based on multi-perspective metrics.** In *Business Process Management*, pages 328–343. Springer, 2007. 1, 4, 9, 12, 15
- [4] MARLON DUMAS, WIL M VAN DER AALST, AND ARTHUR H TER HOFSTEDE. *Process-aware information systems: bridging people and software through process technology.* John Wiley & Sons, 2005. 2
- [5] WMP VAN DER AALST. **Process mining: discovery, conformance and enhancement of business processes. 2011.** 2
- [6] WIL VAN DER AALST, TON WEIJTERS, AND LAURA MARUSTER. **Workflow mining: Discovering process models from event logs.** *Knowledge and Data Engineering, IEEE Transactions on*, **16**(9):1128–1142, 2004. 2, 11, 12
- [7] WIL MP VAN DER AALST AND ANA KARLA A DE MEDEIROS. **Process mining and security: Detecting anomalous process executions and**

- checking process conformance.** *Electronic Notes in Theoretical Computer Science*, **121**:3–21, 2005. 3
- [8] JONATHAN E COOK AND ALEXANDER L WOLF. **Automating process discovery through event-data analysis.** In *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pages 73–73. IEEE, 1995. 11
- [9] JONATHAN E COOK AND ALEXANDER L WOLF. *Process discovery and validation through event-data analysis.* PhD thesis, Citeseer, 1996. 11
- [10] JONATHAN E COOK AND ALEXANDER L WOLF. **Discovering models of software processes from event-based data.** *ACM Transactions on Software Engineering and Methodology (TOSEM)*, **7**(3):215–249, 1998. 11
- [11] RAKESH AGRAWAL, DIMITRIOS GUNOPOULOS, AND FRANK LEYMAN. *Mining process models from workflow logs.* Springer, 1998. 11
- [12] AJMM WEIJTERS, WIL MP VAN DER AALST, AND AK ALVES DE MEDEIROS. **Process mining with the heuristics miner-algorithm.** *Technische Universiteit Eindhoven, Tech. Rep. WP*, **166**:1–34, 2006. 12
- [13] RAKESH AGRAWAL, RAMAKRISHNAN SRIKANT, ET AL. **Fast algorithms for mining association rules.** In *Proc. 20th int. conf. very large data bases, VLDB*, **1215**, pages 487–499, 1994. 12
- [14] KE WANG, SENQIANG ZHOU, AND JIAWEI HAN. **Profit mining: From patterns to actions.** In *Advances in Database TechnologyEDBT 2002*, pages 70–87. Springer, 2002. 12
- [15] RAYMOND CHAN, QIANG YANG, AND YI-DONG SHEN. **Mining high utility itemsets.** In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 19–26. IEEE, 2003. 12
- [16] YING LIU, WEI-KENG LIAO, AND ALOK CHOUDHARY. **A fast high utility itemsets mining algorithm.** In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 90–99. ACM, 2005. 12

REFERENCES

- [17] SMITA R LONDHE, RUPALI A MAHAJAN, AND BHAGYASHREE J BHOYAR. **Overview on Methods for Mining High Utility Itemset from Transactional Database.** *International Journal of Scientific Engineering and Research (IJSER)*, **1**(4). 12, 13
- [18] S SHANKAR, T PURUSOTHAMAN, S JAYANTHI, AND NISHANTH BABU. **A fast algorithm for mining high utility itemsets.** In *Advance Computing Conference, 2009. IACC 2009. IEEE International*, pages 1459–1464. IEEE, 2009. 12