A HYBRID METHODOLOGY IN PROCESS MODELING: "FROM-TO CHART BASED PROCESS DISCOVERY"

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

EREN ESGİN

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

FEBRUARY 2009

Approval of the Graduate School of Informatics

_____

Prof. Dr. Nazife BAYKAL
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Yasemin YARDIMCI
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Prof. Dr. Nazife BAYKAL	Asst. Prof. Pınar ŞENKUL
Co-Supervisor	Supervisor

Examining Committee Members

Assoc. Prof. Canan SEPİL	(METU, IE)	_____
Asst. Prof. Pınar ŞENKUL	(METU, CENG)	_____
Asst. Prof. Aysu BETİN CAN	(METU, II)	_____
Dr. Sevgi ÖZKAN	(METU, II)	_____
Asst. Prof. Tuğba TAŞKAYA TEMİZEL	(METU, II)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Eren ESGİN

Signature : _____

ABSTRACT

A HYBRID METHODOLOGY IN PROCESS MODELING:
"FROM-TO CHART PROCESS DISCOVERY"

Esgin, Eren
MS, Department of Information Systems
Supervisor: Asst. Prof. Pınar Şenkul
Co-Supervisor: Prof. Dr. Nazife Baykal

February 2009, 175 pages

The managing of complex business processes, which are changed due to globalization, calls for the development of powerful information systems that offer generic process modeling and process execution capabilities.

Even though contemporary information systems are more and more utilized in enterprises, their actual impact in automatizing complex business process is still limited by the difficulties encountered in design phase. Actually this design phase is time consuming, often subjective and incomplete.

In the scope of this study, a reverse approach is followed. Instead of starting with process design, the method of discovering interesting patterns from the navigation traces is taken as basis and a new data analysis methodology named "From-to Chart Based Process Discovery" is proposed.

In this hybrid methodology “from-to chart”, which is fundamentally dedicated to material handling issues on production floor, is used as the front-end to monitor the transitions among activities of a realistic event log and convert these raw relations into optimum activity sequence. Then a revised version of process mining, which is the back-end of this methodology, upgrades optimum activity sequence into process model.

Keywords: From-to Chart Based Process Discovery, Process Modeling, Process Mining, From-to Chart, Event Logs

ÖZ

SÜREÇ MODELLEMEDE HİBRİT BİR METODOLOJİ: "NEREDEN-NEREYE ÇİZELGESİNE DAYALI SÜREÇ BULUŞU"

Esgin, Eren
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Danışmanı: Yard. Doç. Pınar Şenkul
Tez Yardımcı Danışmanı: Prof. Dr. Nazife Baykal

Şubat 2009, 175 sayfa

Küreselleşme sonucunda dönüşüme uğrayan karmaşık iş süreçlerinin yönetimi için jenerik süreç modelleme ve süreç uygulama kabiliyetlerine sahip güçlü bilgi sistemlerinin geliştirilmesine ihtiyaç duyulmaktadır.

Güncel bilgi sistemlerinin kurumlarda yoğun olarak kullanılmasına rağmen, iş süreçlerinin otomasyonundaki gerçek etkileri tasarım safhasında karşılaşılan problemlerle kısıtlanmaktadır. Aslında bu tasarım safhası zaman alıcı, çoğunlukla öznel ve eksiktir.

Bu çalışmanın kapsamında, ters bir yaklaşım izlenmiştir. Süreç tasarımıyla başlamak yerine, olay kayıtlarından ilginç eğilimler keşfetme methodu esas alınıp "Nereden-nereye Çizelgesine Dayalı Süreç Buluşu" adında yeni bir veri analiz metodolojisi önerilmektedir.

Bu hibrit metodolojide, temel olarak üretim yerinde malzeme taşıma problemlerinin çözümünde yararlanılan “nereden-nereye çizelgesi” olay tarihçelerinde yer alan işlemler arasındaki geçişleri izlemek ve bu ham ilişkileri optimum işlem dizisine dönüştürmek amacıyla ön-uç olarak kullanılmaktadır. Daha sonra metodolojinin arka-ucu olan revize edilmiş süreç madenciliği ile, elde edilen optimum işlem dizisi süreç modeline dönüştürülür.

Anahtar kelimeler: Nereden-nereye Çizelgesine Dayalı Süreç Buluşu, Süreç Modelleme, Süreç Madenciliği, Nereden-nereye Çizelgesi, Olay Kayıtları

to my family…

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Asst. Prof. Pınar Şenkul for her guidance, encouragement and endless support.

I particularly would like to thank to Prof. Dr. Nazife Baykal for all her valuable helps, comments and contributions throughout the study.

I wish to thank to the members of my examining committee; Assoc. Prof. Canan Sepil, Asst. Prof. Aysu Betin Can, Asst. Prof. Tuğba Taşkaya Temizel and Dr. Sevgi Özkan for their their insightful comments in this study.

I would also like to express my sincere thanks to my bosses and colleagues at HAVELSAN for their patience and endless support. I am grateful to my all friends for their encouragement and patience throughout thesis.

Finally, I would like to express my appreciation for my family for the endless love and understanding they gave me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

AAT	:	Average Arc Traffic
BI	:	Business Intelligence
BPI	:	Business Process Improvement
BPR	:	Business Process Reengineering
CM	:	Casuality Metric
CRM	:	Customer Resource Planning
DS	:	Dependency Score
EPC	:	Event-based Process Chain
ERP	:	Enterprise Resource Planning
ES	:	Enterprise System
FSM	:	Finite State Machine
FTCBPD	:	From-to Chart Based Process Discovery
FUPD	:	Frequent Unconnected Patterns Discovery
GM	:	Global Metric
IT	:	Information Technologies
LM	:	Local Metric
LOC	:	Line of Code
MCT	:	Minimum Confidence Threshold
ML	:	Machine Learning
MST	:	Minimum Support Threshold
PAIS	:	Process-aware Information Systems
PDM	:	Product Data Management
RDMS	:	Relational Database Management System
SCM	:	Supply Chain Management
SNA	:	Social Network Analysis
SOA	:	Service-oriented Architecture
SOP	:	Standard Operation Procedure
TTL	:	Transition Top List
TTLL	:	Transition Top List Limit
WFMS	:	Workflow Management Systems

CHAPTER 1

INTRODUCTION

The rapid development in information technologies (IT) and the tendency towards more open economies has lead to the evolution of the globalization concept, with the suggestion that the world is one broad market, which can be accessed by all regions and societies. Globalization has been evaluated as a new era, which opens world market to rural regions and thereby new opportunities for growth at local level. The changes that are seen in the market field are so rapid, volatile and costly, that only the most flexible forms of organization and cooperation between business firms will be able to interpret these trends of change to become more competitive.

Ultimate step in this effective enterprise transformation should be to achieve full and consistent reengineering of process breakdown, organizational structure and business performance metrics, which are three major top-down operational management structures. With clarity of global business architecture with respect to “who does what, why, how and when”, mapping of business processes across all global value chain constituent is theoretically possible by the leverage affect of contemporary information systems (e.g. Enterprise Resource Planning-ERP, Workflow Management Systems-WFMS, Customer Relationship Management- CRM, Supply Chain Management- SCM and Product Data Management-PDM), which offer basic process modeling and process execution capabilities.

Even though contemporary information systems are intensively utilized in enterprises, their actual impact in automating complex business processes is still constrained by the difficulties encountered in the design phase [4]. Actually crucial problems are resulting from the discrepancy between process design (i.e. the construction of predefined reference process model) and process enactment (i.e. the actual execution of the process) [15]. Unfortunately process design is typically performed by a small group of consultants, managers and specialists [1]. Designed process model is influenced by the personal understandings, e.g.

models are often normative in the sense that they represent what “should” be done rather than describing the actual process, because of the lack of deep knowledge about the business process (domain) on hand (i.e. lengthy discussions with operational-level works and management are needed) [12]. As a result, process models tend to be rather incomplete, subjective and at a too high level [14].

As stated above, traditional process modeling in contemporary information systems concentrates on the design and configuration phases, while less attention is dedicated to the enactment phase and few organizations systematically hold runtime data which can be baseline for re-design [12]. In the scope of the thesis, it is aimed to develop a data analysis methodology, named “**From-to Chart Based Process Discovery**” (**FTCBPD**) to “reverse” this design-oriented approach. Instead of starting with a process design, FTCBPD methodology attempts to discover interesting patterns from the navigation traces, which can be handled as a main data source for end-user behavior analysis, and translate this discovered knowledge into process model [18]. In this perspective, FTCBPD methodology consists of two components: *from-to chart* and *process mining*.

From-to chart is an analytical tool, which is basically used in monitoring material handling routes between operations, machines, departments or work centers on the production floor [25]. It is a symmetric square matrix with the number of rows and columns equal to the number of operations, machines, departments or work centers in the problem. FTCBPD methodology inherits this tool from industrial engineering domain and implements it in a distinct field, process discovery, as the basic bookkeeping material in monitoring transitions among activities (transactions) occurred in process instances and figuring out if there exist any specific order of the occurrences for representing in process model.

As the second component, process mining is a type of association rule mining algorithm, which is used to distill behavioral process knowledge from a set of real time execution [14]. Although association rule mining algorithm represents intra-transaction relationships, process mining highlights the correlation between activities (transactions) according to timeline [22]. Therefore minimal information in event log has to essentially contain timestamp for each event, which can be used to extract additional causality information by sorting all log entries in the order they take place in a process instance.

Unlike contemporary information systems, process mining is not biased or restrictive by perceptions [12]. However if end-users alter the system doing things differently (e.g. finding out short cuts or changing standard operation procedures (SOPs) entirely), the event log can still deviate from the actual work being done [12]. Nevertheless it is useful in conformance

checking with reference models. Also note that process mining is not an instrument to (re)design business process directly [12]. The goal is to understand what is really happening in enactment phase and evaluate the conflictions with the reference model in diagnosis phase. This approach is vital for any re-design effort. The big picture view of FTCBPD methodology is visualized in Figure 1.1.

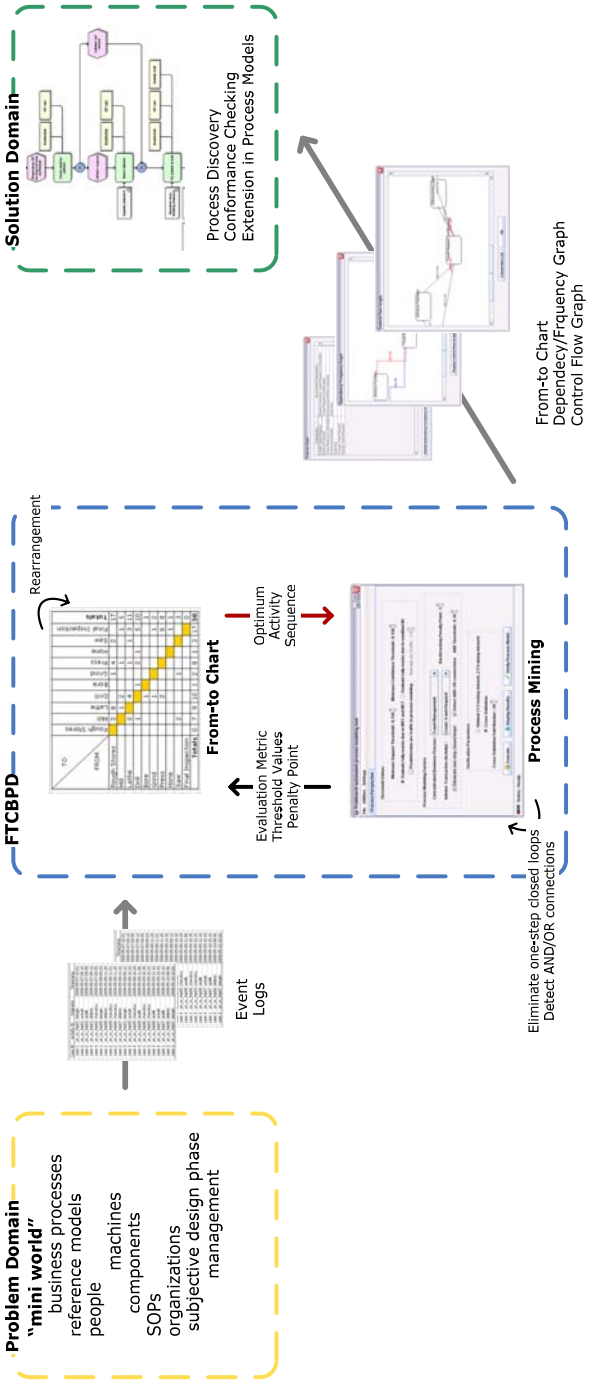


Figure 1.1 FTCBPD Methodology Big Picture View

As represented in Figure 1.1, FTCBPD is a hybrid methodology which integrates two components dedicated to distinct domains. Main cutting edge characteristic of FTCBPD methodology in process discovery field is the implementation of from-to chart, which is an analytical tool in material handling monitoring in plant layout issues, as the front-end to monitor the transitions among activities (transactions) in process instances, which are embedded in event logs, and convert the raw successive relations into an activity sequence, which is optimum with respect to moment (i.e. $flow \times distance$) minimization. Besides, process mining component is the back-end of FTCBPD methodology, which upgrades intermediate product, namely optimum activity sequence, into three process model representations with distinct information level: from-to chart, dependency/frequency graph and control flow graph.

Another novelty feature of FTCBPD methodology in process modeling field is the development of evaluation metrics (e.g. minimum support threshold (MST), minimum confidence threshold (MST) and modified lift), which are used as the major stick yard to control the level of robustness and complexity of the discovered process model from large amounts event logs. Instead of borrowing from statistics, these metrics are formulated by familiarizing with original formations in association rule mining.

The major contributions of FTCBPD methodology in process discovery fields are as follows:

i. Implementation of from-to chart as the basic bookkeeping entity in monitoring relations (transitions) among the activities occurred in event logs. Optimum activity sequence, which rearranged in a straight line formation, constitutes the backbone of discovered process model.
ii. Development of evaluation metrics, which are based on the original formations used in association rule mining. These metrics are fundamentally used in rule induction procedure.
iii. Verification of discovered process model with respect to completeness, soundness and average arc traffic metrics in accuracy and minimality perspectives.

This study is composed of seven chapters. Traditional approaches in workflow technology and process mining as a new era in process modeling are introduced in Chapter 2. Prior aspects and approaches in process mining are summarized in Chapter 3. Proposed methodology is explained in detail in Chapter 4. Evaluation of proposed methodology with respect to distinct key performance metrics are discussed in Chapter 5. Limitations and suggestions for future work are explained in Chapter 6. Finally, Chapter 7 concludes the work.

CHAPTER 2

PROBLEM ANALYSIS

2.1. Major Aspects in Enterprise Transformation

With the increasing interest and wide deployment of information systems, it is seen a growing demand for new emerging architectures and technologies that support effective enterprise transformation, which implies strategic business agility in terms of how efficiently an enterprise can respond to its competitors and how timely an enterprise can predict new opportunities that may arise in the future.

In the globalized economy, enterprises face sophisticated challenges that require rapid and possibly continual transformations. As a result, enterprises are intensively focused on the strategic management of fundamental changes with respect to markets, products and services. Additionally, this transformation typically has a direct impact on the business processes of the enterprise such that, there has been a tendency from data orientation to process orientation within the organizations [13]. Degree of this enterprise transformation may range from traditional Business Process Improvement (BPI), which is a common key word for techniques under the Business Intelligence (BI) technology [17], to paradigm shift in the processes supported by the enterprise. Fundamental of enabling this widespread enterprise transformation is the development of novel instruments and techniques for transforming the business processes of the enterprise.

2.2. Traditional Approach in Workflow Technology

Managing of complex business processes of today's organizations calls for the development of *process-aware information systems* (PAIS), as a way to bridge the perceived gap between organization and software through process technology by controlling and monitoring the flow of work [7]. ERP (Enterprise Resource Planning), WFMS (Workflow Management

Systems), CRM (Customer Relationship Management), SCM (Supply Chain Management) and PDM (Product Data Management) can be classified as PAIS [20]. While WFMS are generally evaluated as “a generic software instrument which allows for definition, execution, registration and control of workflows”, modern ERP, CRM and PDM systems also have a process engine, which embeds workflow technology [7].

Despite workflow technology’s promise on widely covering all tasks of an enterprise, it is burdened with a set of fundamental problems that cause a fair deal of inefficiencies in practical use. Major drawback of such an approach is that the prescribed process model (i.e. reference model) leads to an inherit lack of flexibility, which means a lack of a capability to yield to transform without loss of identity [20], and functionality surplus [18].

The requirement to develop a reference model as a prerequisite to use a new technology is an overwhelming vision to the managers of large, on-going projects [3]. The dilemma is that the more a project is problematic, the more it can benefit from the process technologies, but also willingness of project managers lessens towards devoting resources in new methods and tools [3].

Moreover the problem that is most responsible for the inflexibility of traditional workflow paradigm is its strongly push-oriented nature of routing, enforcing what to do instead of leaving the choice to the end-users [17]. As a result, a completely described workflow design phase is required to enact a given workflow process [1].

On the other hand, creating a workflow design is complicated, time-consuming process and typically, there are discrepancies between the actual workflow process and the process as perceived by the management [14]. The causality links in the mechanism (closed loop) that take place in traditional workflow technology can be modeled by a reinforcement cycle introduced by Senge in [32] as in Figure 2.1.

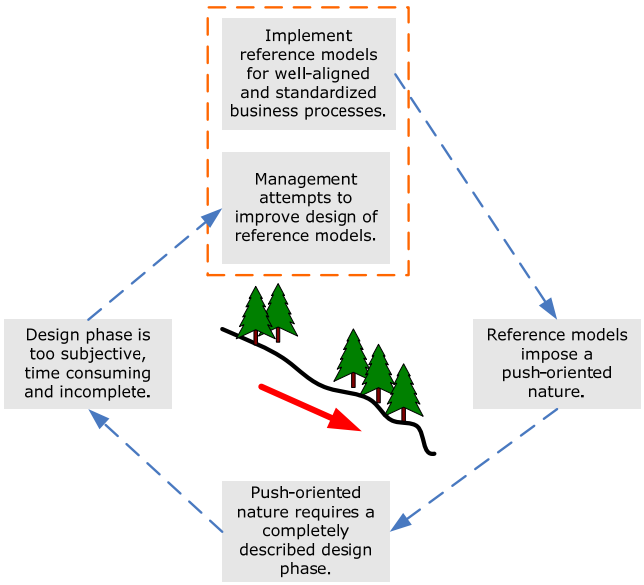


Figure 2.1 Reinforcement Cycle for Traditional Approach in Workflow Technology

In traditional workflow technology, the major concentration is on the design and configuration phases, which are driven by ideas of management on improving the business processes on hand, as represented in Figure 2.2. Less concentration is dedicated to the enactment phase and few enterprises systematically hold runtime data which is the baseline for re-design (i.e. the diagnosis phase is missed) [12]. Hence the initial design of traditional workflow technology is often subjective, incomplete and at a too high level [14].

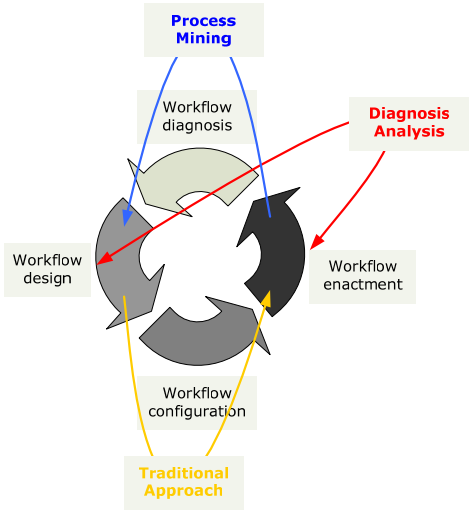


Figure 2.2 Traditional Workflow Technology Life Cycle [12]

Another crucial problem is the partitioning of work in order to make it easy to allocate, while this work is usually performed at a far more detailed level by the employees involved [17].

Furthermore, distribution of work is closely linked to authorization issue such that, employees are always presented with all activities they are allowed to perform.
Another shortcoming in traditional workflow technology is the concept of “context tunneling”, in which employees have no overview about the process as a whole [17]. As a result, this attitude enforces standardization in highly knowledge-intensive labor.

2.3. Process Mining

As stated above, modeling an existing process is influenced by personal understandings, e.g. models are often normative in the sense that they state what should be done rather than describing the actual process [12]. As a result models tend to be rather subjective. Instead of hand-designing the business process, it is proposed to “reverse” the process by a more objective and automated way of modeling to collect the information related to that business process and discover the underlying workflow process from the low-level information history, which is called *event log* [14].

The term *process mining* is used for this objective method of distilling structured process knowledge from a set of real executions (i.e. event logs) [1]. Hence, process mining comes up with an “a posteriori” process model, which explains end-user behaviors embedded in event logs, in contrast to the ideal picture drawn in “a priori” model [12].

Technically, process mining is a specialized form of association mining algorithm such that, patterns discovered by process mining indicate the correlations between activities while association rule represents intra-transaction (i.e. itemset) relations [22]. In association rule mining of transaction database, the mining results are about which items are brought together frequently, those items must come from the same transaction. While the outcomes of process mining are about which activities are brought together in a certain order by the same process instance, those activities come from different event logs [22].

Actually, process mining is not biased by perceptions or normative behavior unlike to traditional workflow technology. However, if end-users alter the system by performing activities differently (e.g. finding out short cuts or changing standard operation procedures (SOPs) entirely), the log can still diverge from the actual work being done [12]. Nevertheless, it is useful to confront man-made models with models discovered through process mining.

Additionally, process mining is not an instrument to directly re-design workflow processes [12]. The major aim is to understand what is really happening in the business process.

Despite the fact that process mining is not an instrument for design phase, it is clear that a good visualization of the existing workflow process is essential for any re-design effort.

Since the goal of process mining is to extract information about processes from event logs, it is assumed that minimal information in event records should represent these features [19]:

i. Each event refers to an activity (i.e. a clear-cut step in the process).
ii. Each event refers to a case (process instance). Processes are by definition case-based, i.e. every piece of work is performed for a specific case [15]. Examples of cases are mortgage, a repair claim, a tax declaration, a purchasing order or a request for travel.
iii. Each event can have a performer also referred to as originator (i.e. the end-user executing or initiating the activity).
iv. Events have a timestamp and are totally ordered by case identifier.

Any information system using transactional infrastructure such as ERP, CRM or WFMS offers this information in some form. Note that it does not mean the existence of a workflow management system. The only assumption made in FTCBPD methodology is that; it is possible to construct event logs from execution history. For instance, Table 2.1 shows an example of a log involving 21 events, seven activities and six originators.

Table 2.1 Event Log Example

Case ID	Activity	Originator	Timestamp
case 1	pr_sy_trip01	eesgin	2008-05-07:10.01
case 2	pr_sy_trip01	ucicek	2008-05-07:09.10
case 2	pr_sy_trip02	acelik	2008-05-07:09.55
case 4	pr_sy_trip01	dderici	2008-05-07:09.12
case 4	pr_sy_trip02	stavukcu	2008-05-07:09.55
case 1	pr_sy_trip05	eesgin	2008-05-08:09.03
case 1	pr_sy_trip08	dderici	2008-05-08:11.20
case 2	pr_sy_trip08	stavukcu	2008-05-08:11.01
case 2	pr_sy_trip06	stavukcu	2008-05-08:11.20
case 2	pr_sy_trip07	dderici	2008-05-08:14.32
case 3	pr_sy_trip01	eesgin	2008-05-08:09.12
case 3	pr_sy_trip05	ucicek	2008-05-08:10.32
case 4	pr_sy_trip05	stavukcu	2008-05-08:10.29
case 4	pr_sy_trip08	acelik	2008-05-08:11.56
case 1	pr_sy_trip06	stavukcu	2008-05-09:10.35
case 3	pr_sy_trip03	stavukcu	2008-05-09:10.22
case 3	pr_sy_trip08	acelik	2008-05-09:12.10
case 4	pr_sy_trip07	ucicek	2008-05-09:10.43
case 1	pr_sy_trip07	acelik	2008-05-10:15.10
case 3	pr_sy_trip06	dderici	2008-05-10:08.56
case 3	pr_sy_trip07	eesgin	2008-05-10:10.02

Event logs such as the one shown in Table 2.1 are used as the baseline for process mining. Process mining is distinguished into three perspectives:

a. The Process Perspective ("How?"):

The process perspective concentrates on the control-flow aspect, i.e. the ordering of the activities. The goal of process perspective is to find out a good behavioral characterization of all process instances on hand [19]. For the process perspective, the process instance and activity attributes of an event log is taken into consideration.

b. The Organizational Perspective ("Who?"):

The organizational perspective focuses on the originator attribute of an event log, i.e. which performers are concerned and how they are associated. The goal is to either structure the organization by classifying people in terms of roles (profiles) and organizational units or to show interactions between individual performers which are represented by social network analysis (SNA) [19].

c. The Case Perspective ("What?"):

The case perspective focuses on the features of cases. Cases can be featured by their patterns in the process or by the originators working on a case [19]. However, cases can also be characterized by the values of the corresponding data elements, e.g. if a case represents a travel request, it may be interesting to know type of the travel or destination. In other words, the case perspective handles the case as a whole and attempts to draw relations with respect to the various properties of a case [19].

In a simple scenario where workflows are handled from a "data perspective" only, classical mining techniques, such as the "market basket analysis", can be proper to discover interesting and potentially useful information about the business data on hand. However, these classical techniques do not fit to scenarios where one looks at workflows from a "control-flow perspective", in which the fundamental concentration is instead on the casual successive relations and on the constraints on the occurrence of the activities [4].

The main idea of process mining is to discover, monitor and improve real processes (not proposed or assumed processes) by extracting knowledge from event logs. Clearly process mining is relevant to operation procedure where much flexibility is allowed or required such that; the more ways in which end-users and organizations deviate, the more variability and the more interesting is to observe and analyze end-user behavior as they are executed [7]. In this aspect, it is considered three basic types of process mining:

a. Process Discovery:

The aim of process discovery is to extract information from event logs in the form of process models. These may be process models, e.g. an event- based process chain (EPC) or Petri

net, but also other models such as social networks (e.g. sociogram) or time-charts describing the performance (e.g. flow times) [8]. Process discovery does not require an a-priori model (i.e. reference model); however the discovered model may be used for delta analysis, i.e., comparing the discovered model representing the actual business process with the reference model representing the predefined business process [8].

This form of data analysis is termed process discovery, because inherent in every project is a business process (whether known or unknown, whether good or bad, whether stable or unpredictable) and for every business process there is some underlying model that is dedicated to describe it. The challenge in this technique is to use that information to describe the process in a form suitable for generic model-based process technologies and a representation that allows the domain expert to understand and evaluate the discovered process model [3].

b. Conformance Checking:

Unlike process discovery, conformance checking requires a-priori model to which it compares the observed behavior as recorded in the event log. Conformance checking may be used to perceive discrepancies but it is also possible to see which parts of the process are really used and where bottlenecks (e.g. closed loops) occur [21].

"Rediscovery problem" is a significant issue in conformance checking such that; the proposed mining algorithm is required to be able to discover a process model functionally equivalent to the source process model (i.e. reference model), on which the basis of complete event log is generated [11].

c. Extension:

Like conformance checking, there is a-priori model which is enriched with a new aspect or perspective discovered in the user behavior analysis. For instance, process mining applications may be implemented in ERP systems for revising system settings (i.e. Customizing[1]) or simplifying the personalization[2] of the system for the end-user [18].

As represented in Figure 2.3, contemporary information systems have improved to comprehensive information technology (IT) supported business solutions that extensively support and enhance organizations in their operations. However this situation is only true for

[1] Customizing in ERP terminology enables the end-user to select and parameterize, with respect to his duities and demands, the desired processes with the appropriate functionality from the set of various functionalities embedded to the information system.

[2] Personalization means to adjust the system to meet the work requirements of specific end-users or end-user groups. It aims to speed up and simplify the business transactions of system processes.

such systems that are well-aligned with organizational requirements [8]. Although this alignment phase sometimes implies unexpected configuration and customization efforts in the implementation process and it may result in significant “hidden” implementation costs [8], reference and enterprise models (i.e. transformed version of reference models) are generated as the documentation of this alignment phase. Actually, reference and enterprise models play a crucial role in representing re-engineered standard operation procedures (SOPs) implemented in business processes of the enterprise.

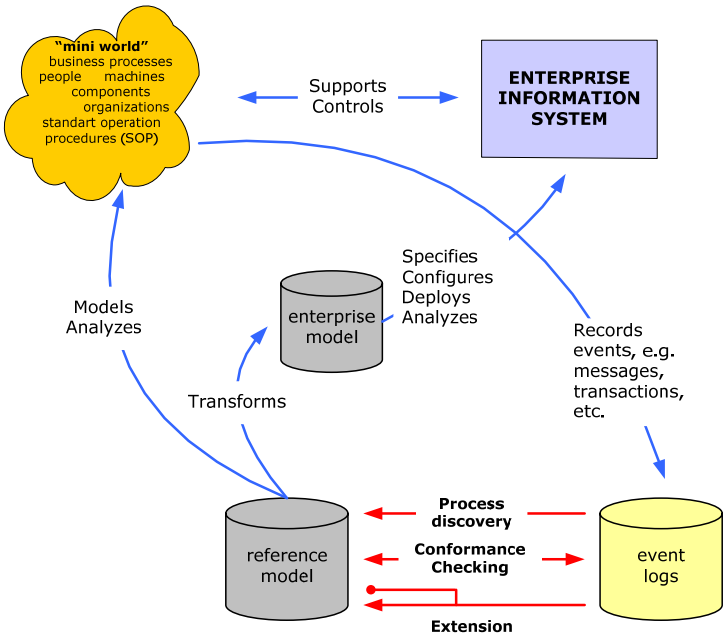


Figure 2.3 Types of Process Mining [7]

Unfortunately reference and enterprise models are only of limited use in supervising end-users. This is mainly due to a lack of flexibility in configuration of completely described process design. To overwhelm this problem, process mining aims to derive a “good” process model with as little information as possible in a posteriori manner (i.e. process mining is not a proactive methodology, since it requires a sufficient amount of runtime data to extract interesting information or patterns).

Beside the capabilities and advantages served by process mining, there are two crucial issues encountered in realistic event logs, which are basically inherited from data mining notion:

a. Completeness:

For larger and complicated business processes, mining is more difficult, since process model exhibits alternative and parallel patterns and then event log will typically not reflect all possible routes [12]. For instance, consider 10 activities which can be executed in parallel. The total number of permutation is 10! = 3628800. It is not realistic that each potential interleaving exists in the event log. Moreover, certain patterns through the process model may have a low probability (e.g. surprise-type relations) and therefore remain undetected. As a result the event log is not complete in the sense that it does not capture all possible behavior.

b. Noise:

Parts of the event log may be incorrect, incomplete, detoriated or refer to exceptions. Events can be logged incorrectly because of human or technical errors. Events can be missed in the event log if some of the activities are performed manually or handled by an exterior system or organizational unit [12]. Additionally events can imply to rare or undesired events [12]. For instance, in a hospital environment a patient started a treatment into hospital X and continues his treatment in the hospital Y; in the event log of hospital Y the treatment activities that occurred in the hospital X cannot be traced [14]. Clearly, exceptions which are traced in the event log for only once should not automatically become part of the discovered process model [12].

To tackle these issues, FTCBPD methodology is proposed in this study. Proposed methodology is relatively robust and has options to focus on the main process instead of attempting to model the full details of the behavior exhibited in the “locally complete” event logs. Basically this characteristic is accomplished by the end-user or domain expert through the adjustable probability threshold parameters in “Threshold Values” view at the selection screen of generic tool named ProMiner, in which FTCBPD methodology is realized.

CHAPTER 3

LITERATURE REVIEW

Process-oriented systems have been increasingly attracting data mining society, as the application of inductive process mining techniques become widespread in both the analysis of complex business processes and the design of new process models. Prior studies about process mining concentrate fundamentally on conceptual and algorithmic aspects of this area of data mining.

3.1. Conceptual Aspects in Process Mining

Prior studies concentrated on conceptual aspects in process mining generally emphasizes the significance of logging functionality in information systems and integration of clear-cut process mining perspectives (i.e. process, organizational and case perspectives in process mining) in order to constitute a hybrid perspective in process mining.

In [21], which is a report of the four workshops organized within the context of the Dagstuhl seminar on “The Role of Business Processes in Service Oriented Architectures”, Aalst firstly handles the issue of obtaining event logs in process-aware and traditional information systems. It is emphasized that the topic of holding event logs is closely related to the correlation of service-oriented architecture (SOA) messages. Just like a process engine needs to know which messages are relevant for an exact process instance, process mining is only possible if events are related with the right corresponding process instances by an attribute, e.g. caseID. After establishing the way in which process logs can be obtained, Aalst focuses on the classification of the various process mining techniques.

As stated in “Problem Analysis” chapter, traditional workflow management systems confront a crucial problem in business process alignment, since these information systems are typically considered too inflexible. In general, business process flexibility is the capability to

react to the external changes by adjusting only those components of a business process, which requires to be changed and keeping other parts stable.

In this aspect, [20] highlights the process mining usage in analysis and improvement of process flexibility. Since traditional information systems are unaware of the performed deviations and thus unable to record information about these deviations, the missing traceability limits the benefit of process mining, whose capabilities obviously depend on the content and the quality of the available event log.

Furthermore, existing mining approaches mainly handle process mining perspectives separately, whereas ad-hoc changes often constitute a combination of control-flow and data-flow adaptations. In [20], Aalst et al. propose the enrichment of obtained event logs, which basically hold syntactical information about the business process, with semantic information, i.e. context information. Consequently process mining techniques can be able to capture the reactive and stimulus entities for the change in event logs.

Likewise [20], [17] aims to handle both case and control-flow perspectives in an integrated fashion, which concentrates on the association between process mining and case handling paradigm. The basic idea in case handling paradigm is to lean the progress in process modeling on the availability of defined data elements. While production workflow is strongly process-oriented, the case handling paradigm basically concentrates on the case itself, which is the fundamental object to be produced.

In [8] Recker et al. concentrate on the development and application of a generic engineering process for configurable reference modeling. As enterprise systems (ES) are developed in a generic manner in order to provide rapid reactions to the requirements of a wide variety of organizations, industry sectors and countries, their implementation emerges the problem of alignment business process and information technologies. Alignment, however, implies indispensable configuration and customization effort in the implementation phase and may result in significant implementation costs that exceed the price of software licenses by significant factors. Process for model-driven ES configuration consists of the steps specification, configuration, transformation and deployment as well as the feedback loops controlling and consolidation.

3.2. Algorithmic Aspects In Process Mining

Prior studies based on algorithmic aspects in process mining specialize on the methodologies, which aim to build up behavioral structure with respect to the underlying

event logs. Approaches dedicated to control-flow perspective can be classified into three prescriptive types:

a. Correlation-based approaches aim to extract interesting correlations, frequent patterns, associations or casual connections among activities in the transaction database.
b. Classification-based approaches aim to induce a rule set, which is constituted of metrics introduced in correlation-based approaches, from event logs and build automatically a classifier to prescribe the type of log-based relations as causal (c), exclusive (e), parallel (p) and inverse casual relation (i).
c. Clustering-based approaches differ from previous approaches in taking account global constraints and performance measures on log traces beside mere “structural” local constraints. This is achieved by representing each event as a point at a properly identified space of features and constructing process model by combining sub-patterns (i.e. clusters), which hold executions sharing the same structure and the same unexpected behavior.

3.2.1. Correlation-based Approaches

The idea of applying process mining in the context of workflow management was first introduced in [10]. In this study, two issues are handled. The first issue is to discover a workflow graph generating activities appearing in a given workflow log. The second issue is to find the definitions of relation conditions. As a shortcoming, given the nature of workflow graphs there is not any requirement to identify the nature (AND or OR) of joins and splits. Moreover workflow graphs are acyclic. The only way to deal with pattern iteration is to enumerate all occurrences of a given activity. Therefore there is a crucial effort to associate activity occurrences belonged to the same activity type. Additional redundancies may occur in activity identification.

Cook and Wolf have investigated similar issues in the context of software engineering process. In [3] they describe three methodologies for process discovery ranging from the purely algorithmic to purely statistical: one using neural networks named RNET, one using a purely algorithmic approach named KTAIL and one Markovian approach named MARKOV. The approach underlying these three methods is to view the process discovery problem as one of grammar inference, in which sample sentences given in a language and some sentences specifically not in the language suppose a grammar that represents the language. In other words, the data describing the behavior of a process are viewed as sentences in a language; the grammar of this language is then limited to the formal process model.

Major shortcoming of grammar inference is that the grammar inference methods generally do not support seeding the underlying algorithm with pre-known information about the process model (or grammar). Often end-user will have domain knowledge about the process under study, and may formulate a backbone of the process model. Additionally grammar inference methods assume a single state machine. In the typical process model, activities generally occur concurrently, which produce an event stream that may have non-deterministic orderings of events.

The authors consider KTAIL and MARKOV methods as the most promising approaches, while RNET method is evaluated as to be insufficient mature to be used in practical applications. KTAIL method builds a finite state machine where states are complex if their futures (in terms of possible behaviors in the next k steps) are identical. Finite state machine (FSM) is the preferred representation in this study rather than other powerful representations such as Petri nets not to make software process prescribing more sophisticated. Actually FSMs are quite convenient and sufficiently powerful for describing historical patterns of actual behavior. Another significant point is results presented in [3] are limited to sequential behavior.

On the other hand, Weijters and Aalst propose a rediscovery technique in [1]. This technique can deal with noise and can also be used to validate workflow processes by uncovering and measuring the discrepancies between the prescriptive models (e.g. reference models) and actual process executions. Compared to Cook and Wolf's existing work, Weijters and Aalst focus on workflow processes with concurrent behavior, i.e. detecting concurrency is one of the fundamental concerns. Therefore AND/OR connectors are aimed to be explicitly distinguished in the process model. To accomplish this goal, WorkFlow nets, which are a subset of Petri nets, is combined with techniques from machine learning (ML). Moreover Weijters and Aalst propose metrics (i.e. local and global metrics), which are quiet different from the proposed metrics in [3] (i.e. entropy, event type counts, periodicity and causality), to find explicit representations for a broad range of process models.

Proposed technique in [1] is composed of three steps: Step (i) construction of dependency/frequency table, Step (ii) generation of a dependency/frequency graph out of a dependency/frequency table and Step (iii) reconstruction of the Workflow net out of dependency/frequency graph and dependency/frequency table.

Dependency/frequency table figures out the following information from event logs:

a. The overall frequency of task A (notation of $\#A$)
b. The frequency of task A directly preceded by task B (notation of $\#B < A$)

c. The frequency of task A directly followed by task B (notation of $\#A > B$)
d. A local metric that indicates the strength of the dependency relation between task A and task B (notation of $\$A \rightarrow^{L} B$)
e. A more global metric that indicates the strength of the dependency relation (notation of $\$A \rightarrow B$)

Sample dependency/frequency table for event type T6 is represented in Table 3.1:

Table 3.1 Dependency/Frequency Table for event type T6 (A=T6) [1]

B	#B	#B<A	#A>B	$A→^{L}B	$A→B
T10	1035	0	581	0.998	0.803
T5	3949	80	168	0.353	0.267
T11	1994	0	0	0	0.193
T13	1000	0	0	0	0.162
T9	1955	50	46	-0.041	0.161
T8	1994	68	31	-0.370	0.119
T3	3949	146	209	0.177	0.019
T6	1035	0	0	0	0.000
T7	959	0	0	0	-0.011
T12	994	0	0	0	-0.093
T1	1000	0	0	0	-0.246
T2	1994	0	0	0	-0.487
T4	1994	691	0	-0.999	-0.825

Local metric (d) expresses the tendency of succession relation by comparing the magnitude of $\#A > B$ and $\#B > A$ with total inter-traffic, $\#A > B + \#B > A + 1$, at a local level. The definition of local metric is as follows:

$$\$A \rightarrow^{L} B = (\#A > B - \#B > A)/(\#A > B + \#B > A + 1) \quad (eq.\ 3.1)$$

The last metric, global metric (e), is more global than local metric because not only direct following events are involved. The underlying intuition is as follows: If it is a frequent case that, when task A occurs, shortly later task B also occurs, then it is reasonable that task A causes the occurrence of task B. In a transaction stream, task A occurs before task B and n is the number of intermediary events between them, the $\$A \rightarrow B$ deÿÿndenÿÿ cÿÿnter is incremÿÿted with a factor σ^{n}. σ is a dependency fall factor (i.e. σ is in [0.0…1.0]).

After dependency/frequency table is constructed, dependency scores (DS) between tasks pairs say task X and Y (i.e. notation $DS(X,Y)$) is calculated. This dependency scores are used in the formulation of a heuristic rule, which is the baseline of dependency/frequency graph. Dependency score is calculated as follows:

$$DS(X,Y) = \left(\left(\$X \rightarrow^{L} Y \right)^2 + \left(\$X \rightarrow Y \right)^2 \right) \div 2 \quad (eq.3.2)$$

For each task pairs, dependency score is calculated. Then the heuristic rule is executed such that:

Given a task A, suppose X is the task for which $DS(A,X) = M$ is maximal.

Then $A \rightarrow Y$ if and only if $DS(A,Y) < 0.95 \times M$

Then $Y \rightarrow A$ if and only if $DS(Y,A) < 0.95 \times M$

As stated above $0.95 \times M$ is the upper bound for dependency scores. Threshold value of 0.95 is the only parameter used to ensure robustness for noise and concurrent processes. With respect to $0.95 \times M$ threshold value, relation among tasks are determined and represented in dependency/frequency graph. Dependency/frequency graph is a FSM-like graph-based representation, on which tasks are represented as blocks and relations among tasks are represented in arcs.

The last step is the upgrade of dependency/frequency graphs to Workflow nets. Major value-adding operation in this step is detecting the types of splits and joins. Dependency scores in dependency/frequency graph and information in the dependency/frequency table contain useful information to determine the types of splits and joins intuitively.

According to this intuitive operation, dependency score, which is approximately equal to total number of incoming or outgoing transitions of the underlying activity, implies an AND-connection, while dependency scores complementing each other to total number of incoming or outgoing transitions of the underlying activity implies an OR-connection. Unfortunately proposed mining technique in [1] have still problems with handling complex interconnected structures in combination with short loops. Proposed mining technique in this study is realized as a tool named “Little Thumb”.

Formal approaches stated above are based on an assumption of a weak notion of completeness (i.e. if one activity can be followed by another activity, this should occur at least once in the event log) and noise-free event log (i.e. everything that is registered in the

log is correct and erroneous-free). Actually impractical situations logs are rarely complete and noise free. Hence HeuristicMiner approach stated in [13] anticipates three threshold parameters to handle this issue: (i) the dependency score, (ii) the positive observation score, (iii) the relative to best threshold.

Approaches, which suffer from the drawback that the nature of splits and joins is not discovered, suppose that the threshold value usage is unnecessary for dependency relations according to "all activities connected" heuristic. As the major novelty of [13], Weijters et al. propose a measurement to express the type of splits and joins instead of the intuitive heuristic approaches like in [1]. According to this measurement:

$$A \Rightarrow B \wedge C \ = \left(\frac{|B > C| + |C > B|}{|A > B| + |A > C|} \right) \quad (eq.\ 3.3)$$

The $|A > B| + |A > C|$ notation indicates the number of positive observations and $|B > C| + |C > B|$ notation indicates the number of times activity B and C appear directly after each other.

In the following study [15], Weijters and Aalst introduce two additional parameters: noise factor N and a threshold value σ instead of dependency scores, which are calculated in [1] to determine the significance of the relation. The value σ is automatically calculated using the following equation: $\sigma = 1 + Round\left(\frac{N \times \#L}{\#T} \right)$. N is the noise factor with default value of 0.05, $\#L$ is the number of trace lines in the workflow log and $\#T$ is the number of activity types in the related business process. Frequencies stated in dependency/frequency table (parameters b and c) are compared with noise factor N and calculated σ to determine whether underlying relation is worth to be represented in dependency/frequency graph.

Weijters and Aalst enhance a significant novelty to the present approach in [1], which resides in the fact that they use a global learning approach, named "logistic regression model" and find a threshold value that can be used to detect direct successions in [14]. As the basic material, dependency/frequency table is used as in [1]. Addition to existing parameters in dependency/frequency table, the frequency of task B directly succeeded by another task A, but before the next appearance of B (i.e. notation of $B >>> A$) and the frequency of task A directly succeeded by another task B, but before the next appearance of A (notation of $A >>> B$) parameters are added.

Additionally local and global metrics introduced in [1], which indicates the strength of the relation locally and globally, are revised in [14] such that:

a. **The local metric (LM):** Considering tasks A and B, the local metric LM expresses the tendency of succession relation by comparing the magnitude of ($A > B$) versus ($B > A$).

$$LM = P - 1.96 \times \sqrt{\frac{P \times (1-P)}{N+1}} \; where \; P = \frac{|A > B|}{N+1} \quad N = |A > B| + |B > A| \quad (eq.3.4)$$

In general, it is concluded that LM can have a value (i) close to 1 when there is a perfect tendency of succession between X and Y, (ii) in the neighborhood of 0.5 when there is both a succession between X and Y and between Y and X, but a clear tendency cannot be identified and (iii) zero when there is no succession relation between X and Y.

b. **The global metric GM:** The previous measure LM was expressing the tendency of succession by comparing the magnitude of $(A > B)$ versus $(B > A)$ at a local level. Therefore, GM measure is built.

$$GM = \left(|A > B| - |B > A|\right) \times \frac{\#L}{\#A \times \#B} \quad (eq.3.5)$$

In conclusion, for determining the likelihood of succession between two events A and B, the GM metric is indeed a global metric because it takes into account the overall frequency of events A and B with respect to $\#L$, while the LM metric is a local metric because it compares the magnitude of $(B > A)$ with $(A > B)$.

c. **The causality metric CM:** The casualty metric is calculated as following: if task B occurs after task A and n is the number of events between A and B, then CM is incremented with a factor σ^n, where σ is a causality factor, σ is in [0.0,1.0]. The causality metric CM was firstly introduced in [1]. But it is labeled as “global” in this prior study. In [14] there is a clear distinction between causality and global identities of the metrics.

The idea of the logistic regression model is to combine these three metrics described above and to find a probability Π over which two tasks A and B can be considered to be in the direct succession relation. The form of logistics regression model is such that;

$$\log\left(\frac{\Pi}{(1-\Pi)}\right) = B_0 + B_1 \times LM + B_2 \times GM + B_3 \times CM \quad (eq.3.6)$$

where the ratio ($\frac{\Pi}{(1-\Pi)}$) represents the *odds*. The significance of individual logistic regression coefficients, B_is, is given by the Wald statistics which indicates significance in the model; that means all independent variables have a significant effect on direct succession predictability.

As stated above, relations represented in dependency/frequency graph are classified as direct succession and succession in [14] and the probability of these relations are determined by logistic regression model. Definitions of these relation types are given in "Proposed Method" chapter.
Global learning method proposed in [14] uses information contained in workflow logs to discover the direct successor relations between events. This method is able to find almost all direct connections in the presence of parallelism, noise and an incomplete log. In [16] Maruster, Weijters and Aalst implement another variant of this method on simulated hospital event logs, containing information about which medical actions took place over time. Technique in [16] does not work well for all kind of Workflow nets, as one experiment involving none-free-choice showed.

In [11], the goal of proposed method, named "alpha algorithm", is twofold: first of all, a mining algorithm is sought to rediscover sound Workflow nets, i.e. based on a complete workflow log the corresponding workflow process model can be derived without any extra behaviors. Second, given such an algorithm, it is aimed to indicate the class of workflow nets which can be rediscovered. Clearly, this class set should be as large as possible. Note that in the prior studies [1, 3, 14, 15, 16] there is not any mining algorithm which is able to rediscover all sound Workflow nets. As a way of representation, Maruster, Weijters and Aalst attempt to generate concrete Petri net for a broad range of process models rather than a set of dependency relation between events like in [1].

Actually the preliminary results presented in [1, 14, 15, 16] only provide heuristics and basically concentrate on issues such as noise, basic parallelism, basic closed loops. The approach described in [11] differs from these approaches in the sense that for the alpha

algorithm it is proven that for certain subclasses (e.g. non-free choice, basic and arbitrary loops, hidden tasks, noise, basic and complex parallelism) it is possible to find the right workflow model. Also alpha algorithm can mine timed workflow logs and calculate several kinds of timing information (e.g. waiting/synchronization times, flow times, utilization) to performance metrics. On the other hand, the major limitation of alpha algorithm is that certain kind of multiple tasks having the same title cannot be detected.

In [12], distinct tools, which are driven by different problem areas in process mining, are compared. Handled tools are EMiT, Little Thumb, InWoLvE and Process Miner:

- EMiT (Enhanced Mining Tool) is a graphical process model including all kinds of performance metrics. Because of its graphical-based structure, it is able to handle rediscovery problem effectively.
- Little Thumb, which is firstly introduced in [1], concentrates on incomplete logs and noise. However in a noisy and incomplete situation, one erroneous event can completely confuse the derivation of a right conclusion. For this reason Little Thumb attempts to develop a heuristic mining technique which is less sensitive for noise and the incompleteness of the log.
- Although approaches previously presented assume that a task name should be a unique identifier within the process, in the graphical models it is not possible to have multiple building blocks referring to the same task. InWoLvE (Inductive Workflow Learning via Examples) attempts to deal with duplicate tasks with lattice of task mappings in the workflow logs, which is inherited from machine learning and grammatical inference. Between the mappings there is a partial ordering (more general than/more specific than). The lattice is limited by a top or most general mapping (i.e. every task instance with name X is mapped to one single task node with name X) and a bottom or most specific element (the mapping is bijection between task instances in the event log and task nodes of the workflow model).
- The last tool, Process Miner, exploits the properties of block-structured workflows through rewriting rules. Block-structured models are composed of blocks which are nested. These building blocks of block-structured models can be differentiated into operators and constants. Operators construct the process main flow, while constants are the tasks or sub-workflows that are embedded inside the process flow.

In [19], Aalst et al. aimed to demonstrate the applicability of process mining in general and developed algorithms and tools in particular. The industrial application in this study involves one of the twelve offices of the Dutch National Public Works Department, which is primarily responsible for the construction and maintenance of the road and water infrastructure in its providence. The focus of this study is not limited to the control-flow perspective. In this case

study, the organizational and case perspectives are also handled. As a supporting tool, ProM framework, which integrates EMiT, Little Thumb and MiSoN tools, is used in this application.

As a hybrid methodology in process mining, Gomez et al. introduce “Application Usage Mining” concept in [18]. Application usage mining is explained as the examination of the user’s behavior in the business application systems (e.g. ERP) by applying the basic approaches, notions and methods inherited from Web Usage Mining. Basically this application indicates some significant differences, which are caused by the fact that there are some logical as well as technical gap between a web application of e.g. electronic shopping and a business application system. In a web application the connection between a guest and provider is not obligatory. The visitor has the freedom to navigate through the providers’ web pages. Along with the usage of a business application system the employee should optimally perform the assigned tasks and business processes of the enterprise.

3.2.2. Classification-based Approaches

Tackling the problem of process discovery at a more robust level is subsequently introduced by Weijters and Aalst in [14] using an empirical data-driven approach namely logistic regression model, which is able to detect the casual relations (i.e. direct successors) from event logs. However that logistic regression approach requires a global threshold value for deciding when there is a direct regression between two tasks. The usage of global threshold has the shortcoming of being too rigid, thus real relations may not be found out and false relations may be caught.

In this aspect, [2] aims to use machine learning techniques to stimulate classification rules for (i) casual relations and (ii) parallel/exclusive relations assuming the existence of noisy information in event log and imbalance in execution priorities.

The construction of a so-called dependency/frequency table from the event log information is the starting point of the method as in [1]. Afterwards three relational metrics, i.e. causality metric (CM), local metric (LM) and global metric (GM), are calculated for each task pair occurred in process instances. Relational metrics and dependency/frequency table materials are inherited from [1] and [14].

Actually the CM, LM and GM metrics have been developed specifically to be used as predictor attributes for determining decision rule sets. They are less practical for deciding between exclusive and parallel relations, for which it is required to develop other adequate predictors. Because the rule set to be induced using these three metrics as predictors must

be in a general condition, the normalized form of $|X > Y|$ and $|Y > X|$ are also taken into account as predictors. Thus YX and XY metrics are introduced.

Last operation in [2] is to detect the existing log-based relations between tasks by applying the predictive features of the introduced metrics to the learning material generated in dependency/frequency table. In this operation “Ripper” is chosen as the appropriate learning algorithm, which induces minimal description-length rule sets. It has been shown that Ripper is competitive with the intensively-used alternative algorithm, C4.5rules, in terms of error rates, but more capable than C4.5rules on noisy data.

Because of supervised nature of classification, a training dataset has to be provided, each of which has been labeled with a class. Each instance in training dataset is labeled corresponding to the log-based relations that can exist between two tasks: (c) for causal, (e) for exclusive, (p) for parallel and (i) for an inverse casual relation. Following rule set in Figure 3.1 is constructed for detecting log-based relations.

Rule1: IF XY=0 AND GM>=0 THEN class e [4734 pos, 32 neg]
Rule2: IF XY<=0.01 AND CM<=-0.35 AND YX<=0.04 THEN class e [486 pos, 0 neg]
Rule3: IF YX<=0.01 AND LM<=0.31 AND CM>=-0.02 AND CM<=0.04 THEN class e [3006 pos, 2 neg]
Rule4: IF YX<=0.01 AND CM<=-0.26 THEN class e [588 pos, 8 neg]
Rule5: IF YX<=0.01 AND XY<=0 AND CM>=-0.06 AND CM<=0.01 THEN class e [2704 pos, 7 neg]
Rule6: IF XY<=0.01 AND CM>=0.29 THEN class e [253 pos, 0 neg]
Rule7: IF XY>=0.01 AND YX>=0.02 THEN class p [5146 pos, 0 neg]
Rule8: IF XY>=0.02 AND CM>=-0.24 AND LM>=0.33 THEN class p [3153 pos, 0 neg]
Rule9: IF YX>=0.01 AND CM>=-0.26 AND CM<=-0.07 THEN class p [1833 pos, 1 neg]
Rule10: IF XY>=0.01 AND CM>=-0.24 AND CM<=-0.04 THEN class p [2227 pos, 3 neg]

Figure 3.1 Rule Set Constructed by Ripper [2]

As a result, the contribution of [2] can be seen as successfully complementing the work reported in [11]: it resolves shortcomings of the alpha algorithm, in dealing with issues about causality and parallel/exclusive relations exhibition in noise and incomplete process logs.

3.2.3. Clustering-based Approaches

Currently correlation and classification based techniques focus on structural aspects of the process and disregard all non-structural data that are still kept by many real systems, such as information about activity executors, timestamps, parameter values, as well as different performance measures.

In [6] Chiaravalloti et al. present an enhanced process mining approach, where different process variants (use cases) can be discovered by clustering log traces, based on both structural aspects and performance measures. For this aim, an information-theoretic framework is used, where the structural information as well as performance measures is represented by proper auxiliary domains, which is correlated to the central domain, namely D_T, of logged process instances. Hence beside the list of activity identifiers, each cluster is equipped with a number of metrics, which are meant to characterize some performance measures for the enactment phase at hand.

Major problem in this multi-dimensional extension is that there may well exist two auxiliary dimensions X, $X^I \in \{A, Z_1,\ldots Z_N\}$ with $X \neq X^I$, such that the best co-clustering solution for the pair D_T and D_X does not conform with the best co-clustering for D_T and $D_X{}^I$. The solution to jointly optimize all pair-wise loss functions is to linearly combine them in a global function, by using N+1 weights $ß_A$, $ß_1$,… $ß_N$ with the characteristic of $\beta_A + \sum_{i=1}^{N} \beta_i = 1$, which are meant to quantify the relevance that the corresponding auxiliary domain should have in concentrating on the co-clustering of the whole log data.

In parallel to [6], [5] continues on the way of the investigation of data mining techniques for process mining through hierarchical clustering of the event logs, in which each trace is seen as a point of a properly identified space of features. As a major distinction in this study, previous approach is extended to process mining by proposing an algorithm which is able to discover not only the behavioral structural of a given business process, but also enrich the discovered schema with some interesting global constraints, in order to provide the designer with a refined view of the process. In this aspect, local constraints basically reflect behavioral aspect, while global constraints are richer in nature and their representation strongly depends on the particular application domain of the modeled business process. Thus they are often expressed using other complex formalisms, mainly associated with clear semantics.

Lastly, [4] aims to precisely investigate the unconnected patterns, which are sets of the patterns that frequently occur together in some event log data, by designing and implementing efficient solutions for the frequent unconnected patterns discovery (FUPD) problem, in which a set of frequent patterns is given as input and all subsets of this set that are frequent as well have to be discovered.

Proposed technique in [4] can be used for singling out sets of arbitrary sub-processes that are very often executed together and may be abstractly seen as a sub-process in the workflow schema. Hence these unconnected patterns can be used by the system administrators to identify interesting and useful correlations among sub-processes which are apparently not related with each other.

CHAPTER 4

PROPOSED METHOD

4.1. From-to Chart as a Basic Analytical Technique in Plant Layout

The basic from-to chart is a square matrix for summarizing material handling between related operations, machines, departments or work centers on the production floor [25] with high volume production rate [35]. The sequence of operations is written down the left-hand side of the form and across the top. While the vertical sequence of activities is the “from” side of the matrix, the horizontal sequence of activities is the “to” matrix [26]. This analytical technique is extremely useful for [25]:

a. Designing relative locations of operations.
b. Demonstrating material flow patterns.
c. Showing degree of self-sufficiency of each operation.
d. Interpreting possible production control problems.
e. Planning interrelationships between several products, parts, materials, etc.
f. Representing quantitative relationships between operations and the related handling between them.
g. Evaluating alternative flow patterns.
h. Improving distances traveled during a process.

The number of rows and columns in the matrix is equal to the number of operations under consideration. Additionally operation titles are listed in identical order across the top of the columns and down the row on the left hand side of the matrix. Initial row or column sequence may represent geographical arrangement in the plant, logical arrangement of process flow or proposed sequence as represented in Figure 4.1.

TO / FROM	Rough Stores	Mill	Lathe	Drill	Bore	Grind	Press	Hone	Saw	Final Inspection	**Totals**
Rough Stores		2	8			1	4		2		17
Mill			1	2			1			1	5
Lathe		2		4			1		1	3	11
Drill		1			1		2	1		5	10
Bore				1							1
Grind				1						1	2
Press				2						6	8
Hone										1	1
Saw		2				1					3
Final Inspection											0
Totals	0	7	9	10	1	2	8	1	3	17	**58**

Figure 4.1 From-to Chart as a Basic Analytical Technique in Plant Layout [25]

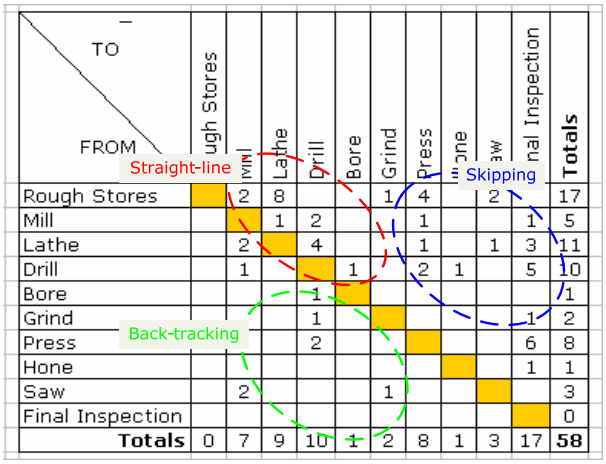
Basic data for entry into from-to chart are prepared by tabulating the flow paths of each concentrated part, product or material such that; for each move of related entity from operation i to operation j, current score at the (i, j)th element of matrix is incremented by one. Thus accumulated scores in each element represent the total number of moves from and to the underlying operation. Entry of data into the matrix can be done in several ways, depending on objective or desired result of the analysis [25]. Scores may also represent:

a. Number of moves between operations.
b. Quantity of material moved per time period.
c. Weight of material moved per time period.
d. Combination of quantity × weight per time period.
e. Ratio of total through each operation to each subsequent operation.
f. Move time.
g. Move cost.

Constructed from-to chart has to be analyzed for better arrangements of operations to reduce handling, costs, distances, production control problems, etc. [25]. Major use-cases occurred at from-to chart are as follows:

a. All entries below the diagonal indicate *back-tracking*, i.e., backwards from the order indicated by the numbers representing the operations.
b. All entries in the upper right or far right indicate *skipping* past several adjacent operations to get to their next operation.
c. Items moving from one operation to an adjacent operation result in the marks falling in the elements along and just above the diagonal. This represents *straight-line* (direct) flow.

Major use cases are visualized at prior from-to chart in Figure 4.2.



FROM \ TO	ugh Stores	Mill	Lathe	Drill	Bore	Grind	Press	Hone	Saw	nal Inspection	Totals
Rough Stores		2	8			1	4		2		17
Mill			1	2			1			1	5
Lathe		2		4			1		1	3	11
Drill		1			1		2	1		5	10
Bore				1							1
Grind				1						1	2
Press				2						6	8
Hone										1	1
Saw		2				1					3
Final Inspection											0
Totals	0	7	9	10	1	2	8	1	3	17	**58**

Figure 4.2 Major Use Cases Occurred at From-to Chart

Intuitively it is seen that the best layout can be devised by rearranging the columns and rows to put the elements with larger scores just above the diagonal and fewer ones below the line [25]. Actually this may be possible for one material, but it is not possible for all materials in production portfolio systematically.

According to [35], from-to chart is a descriptive material to reduce a large volume data into a workable formation such that, the construction of a from-to chart does not result directly in the solution of a layout problem. On the other hand, a more quantitative approach to minimize material handling is obtained by taking “moments” of the accumulated score at each element around the diagonal (i.e. “support point”) and aiming for the lowest moment total at from-to chart as stated in [25]. The number of elements away from the diagonal is used as the distance from the diagonal, i.e. moment arm[1]. Objective function to minimize the total moment of from-to chart is formulated as follows:

$$Min\ Z = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij} \times \left| j - i \right| \times p \quad (eq.\ 4.1)$$

Parameters stated in the objective function are:

f_{ij} indicates total move from operation i to operation j.

[1] To make moment computation simple, suppose all operations (machines) are of the same size and the distance between the working points of each pair of adjacent operation (machine) is one unit.

p is the back-tracking penalty point assigned to each entry below the diagonal. Back-tracking penalty point is doubled to enforce the model towards a straight line arrangement [26].
Range of back-tracking penalty is such that;

$$p = \begin{cases} 1 \ when \ \ j \geq i \\ 2 \ when \ i > j \end{cases}$$

4.2. From-to Chart Based Process Discovery Methodology (FTCBPD)

As stated in the "Problem Analysis" chapter, managing complex business processes calls for the development of powerful information systems, which are able to control and support the underlying processes. To support a structured business process, such process-aware information systems have to offer generic process modeling and process execution capabilities [2].

Although information systems that are not process-aware (e.g. ERP system), are built around a large set of database tables instead of designed around explicit process models, it seems a crucial tendency towards building information systems on a process layer, making logging at the right level is a standard functionality [21]. We would even like to claim that "logging should be first class citizen" for any information system that is used to support processes.

In this aspect, this study aims to develop a methodology named "**From-to Chart Based Process Discovery**" (**FTCBPD**), which is information system-independent. FTCBPD methodology does not assume the presence of a process engine as in process-aware information systems. The major requirement of this methodology is that it is possible to construct event logs, which can be pulled up and used as a main data source for process discovery.

FTCBPD methodology aims to discover interesting patterns from event logs in the form of straight-line, skipping and back-tracking type relations among activities (transactions) and convert this discovered knowledge into the form of business models as output (i.e. from-to chart, dependency/frequency graph and control flow graph) without any domain knowledge about the underlying business process.

In general, this is a very difficult challenge to meet. To scope the problem, we have concentrated our efforts on models of the process perspective, rather than, on models of the

relationships between artifacts produced by the business process or on the models of the roles and responsibilities of the actors in the process.

The novelty of FTCBPD methodology in process discovery field resides in the fact that; we use a global instrument, which is essentially designed for material flow analysis in industrial engineering domain, named “From-to Chart”, as baseline to monitor transitions among activities (transactions) occur in business processes. FTCBPD methodology inherits the square, symmetric matrix characteristic of from-to chart and applies some modifications and add-ins to this tool.

Major modifications and add-ins applied to classical from-to chart are listed below:

a. Instead of operation, machine or department titles, activity (transaction) titles retrieved from event logs are embedded as attribute titles to from-to chart matrix in FTCBPD methodology.
b. Although traditional from-to chart implementation uses original total tally marks directly in rearrangement phase of the matrix to find out the layout with minimum material handling, FTCBPD methodology compares total tally marks by evaluation metrics (i.e. minimum support threshold- MST, minimum confidence threshold- MCT and modified lift) prior to rearrangement of from-to chart.

   This comparison operation may reset tally mark values, which are “weak” according to local or global evaluation metrics (MST and MCT), or negativate tally marks, which implies that occurrence of successor activity is negatively correlated to the occurrence of predecessor activity.
c. In the traditional from-to chart implementation, the matrix is rearranged to minimize material handling by neighboring the operations with higher material inter-traffic. While rearranging the matrix, unchanging and doubled back-tracking penalty point is assigned to back-tracking use cases. On the other hand, this back-tracking penalty point is determined by end-user in FTCBPD methodology and can be adjusted for what-if analysis.
d. In the traditional from-to chart implementation while rearranging the matrix, there is not any insight about the initial operation in the operation sequence. Because parts, products or materials with distinct flow paths (production routes) are handled on the same production floor according to job shop philosophy.

   On the other hand, constructed from-to chart in FTCBPD methodology is populated by process instances belonged to a single business process. In other words, the constructed from-to chart is dedicated to a single business process. Hence the end-user can anticipate the initiator activity (transaction) in discovered process model.

Additionally this domain knowledge may reduce the complexity of rearrangement operation from $O(n!)$ to $O((n-1)!)$.

e. The traditional from-to chart implementation attempts to arrange operations in a straight line, thus the results in traditional from-to chart implementation are limited to sequential behavior. But business processes exhibit more sophisticated behaviors (i.e. basic parallelism) additional to sequential behavior.

   Although mining of non-observable AND/OR connections is difficult since they are not explicitly present in event log [13], FTCBPD methodology is able to detect and handle basic pair-wise parallelism with respect to AND threshold set in the "Process Modeling Factors" view. Additionally FTCBPD methodology imposes restrictions on the structure of basic one-step closed loops to guarantee the correction of discovered process model by "Eliminate one-step closed loops in discovered process model" application. Thus one-dimensional activity sequence, which is an "intermediate product" of rearrangement operation, is upgraded to two-dimensional process model.

FTCBPD methodology is realized in a tool named ProMiner. Selection screen of ProMiner is presented in Figure 4.3.

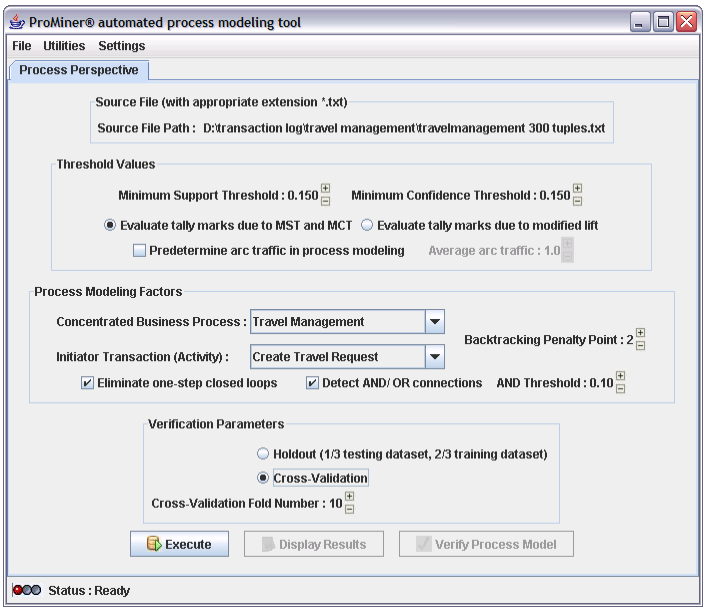


Figure 4.3 Selection Screen of ProMiner

Functionalities of the views (i.e. source file, threshold values, process modeling factors and verification factors) at the selection screen are given in APPENDIX C.

In ProMiner, process model can be visualized in four forms corresponding to four levels/steps of process discovery:

a. Initial State of From-to Chart

Initial state of the from-to chart represents the original tally mark values in FROMTOCHART table prior to evaluation of these tally marks. This information may be useful to understand the mechanism of evaluation and rearrangement operations in FTCBPD methodology. Screenshot of “Initial State of From-to Chart” frame is represented in Figure 4.4.

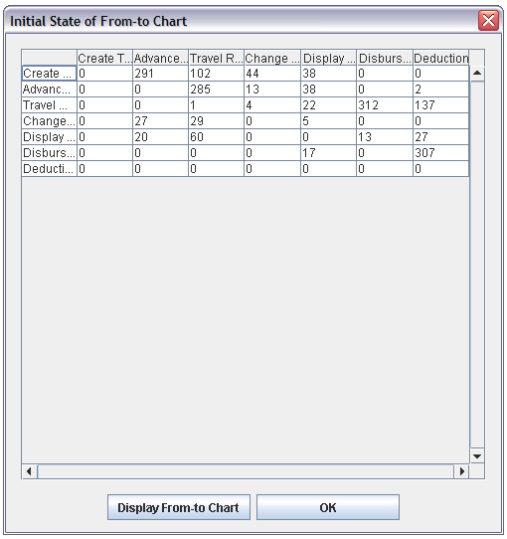


Figure 4.4 Initial State of From-to Chart Frame

b. From-to Chart

Constructed from-to chart textually summarizes the successive activities (transactions) of each activity (transaction) take place in the discovered process model. Fields of the output are “transaction” and “successive transactions”. Screenshot of “From-to Chart” frame is represented in Figure 4.5.

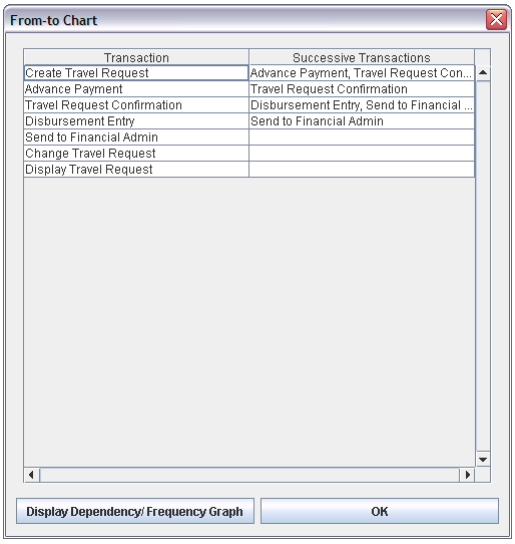


Figure 4.5 Discovered Process Model in From-to Chart Form

c. Dependency/Frequency Graph (D/F Graph)

Dependency/frequency graph is a kind of finite state machine (FSM) which visually represents activities as blocks, dependency scores of each transition among activities (i.e. number of repetition for the underlying transition and total outgoing transition number of the source activity) and type of these transitions.

Although there are more powerful representations than finite state machines (e.g. push-down automata, Petri nets etc.) which are arguably better suited for representing business process [3], finite state machines are quite convenient and sufficiently powerful for describing behavioral patterns of actual behavior. The underlying reason of this argument is that; the more powerful the representation, the more complex the discovery problem [3].

The color legend describing type of the transition is stated below:

- Red color indicates straight-line (direct) flow among activities (transactions).
- Blue color indicates transitions that skip past several adjacent activities (transactions) to get to their next activity (transaction).
- Green color indicates back-tracking.

Screenshot of “Dependency/Frequency Graph” frame is represented in Figure 4.6.

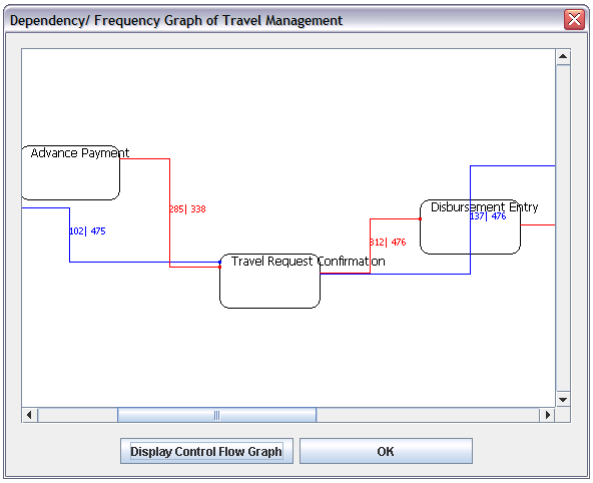


Figure 4.6 Discovered Process Model in Dependency/Frequency Graph Form

d. Control Flow Graph

If “Detect AND/OR connections” check-box in “Process Modeling Factors” view is marked, control flow graph can be visualized. In this output, pair-wise connection types, which are determined with respect to AND threshold, are represented in the discovered process model by linear connectors in addition to dependency scores (i.e. repetition number and confidence value of the underlying transition). The color legend describing type of the connection is stated below:

- Red color linear connector indicates AND-type connection.
- Blue color linear connector indicates OR-type connection.

The transitions among activities are in straight-line appearance instead of intermittent form used in dependency/frequency graph. Screenshot of “Control Flow Graph” frame is represented in Figure 4.7.

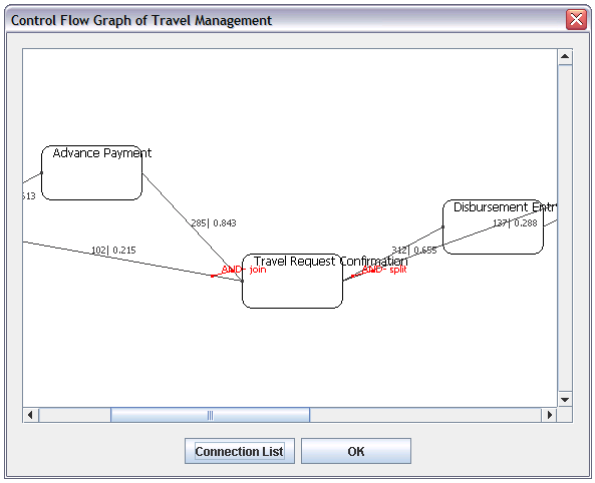


Figure 4.7 Discovered Process Model in Control Flow Graph Form

Moreover end-user can list connections and their characteristics (e.g. source transaction, connection type, connected transactions, connection score) by “Connection List” button on control flow graph frame.

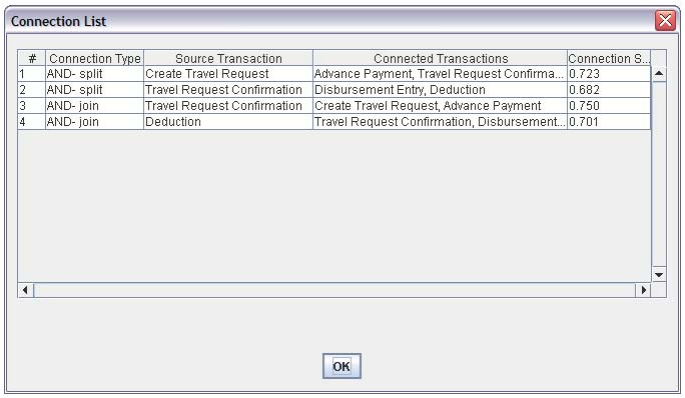


Figure 4.8 Connection List Frame

FCTBPD methodology is apparently a sequential approach (i.e. AND, OR and XOR connectors are merely used for distinguishing check-box marked or unmarked use cases), which is composed of nine operations:

1. “Create FROMTOCHART Table” operation aims to create FROMTOCHART database table and instantiate this table with null valued tuples.
2. “Populate FROMTOCHART Table” operation aims to mark the transitions, which are captured in transaction streams, into the appropriate element of FROMTOCHART. At the end of this operation, row and column total of each activity type is computed and then inserted into TALLYMARK table.

3. “Evaluate Tally Marks in FROMTOCHART Table” operation aims to prune down the rules by eliminating “weak” accumulated tally marks in FROMTOCHART table with respect to selected evaluation metric(s).
4. “Rearrange FROMTOCHART Table” operation aims to find out “the most frequent” activity sequence in training dataset by taking moment for each non-zero tally mark in FROMTOCHART table. Thus activities with higher inter-traffic are neighbored in the activity sequence with respect to the minimum moment objective function and this activity sequence acts as the backbone for discovered process model.
5. “Construct Process Model” operation aims to drive direct successive, successive and backtracking type transitions among activities (transactions) with respect to evaluated tally marks in FROMTOCHART. This form of process model is represented as dependency/frequency graph.
6. “Eliminate One-Step Closed Loops at Discovered Process Model” operation aims to detect and eliminate one-step closed loops, which are the bottlenecks (deadlocks) of the underlying business process.
7. “Construct AND/OR Connections at Discovered Process Model” operation aims to detect connection types (i.e. AND-split, AND-join, OR-split and OR-join) by interpreting discovered transitions (patterns) of each activity type in discovered process model with respect to AND threshold and represent these connections in control flow graph by linear connectors.
8. “Verify Discovered Process Model” operation aims to find out the best process model by challenging process models, which are discovered at each iteration, with respect to completeness verification metric. Additionally soundness and average arc traffic characteristics of optimum process model are reported at this operation.
9. “Report Process Instances” operation aims to construct transaction streams of each process instance in training and testing datasets and write down constructed transaction streams into a text file.

Algorithm of FTCBPD methodology is represented in Figure 4.8:

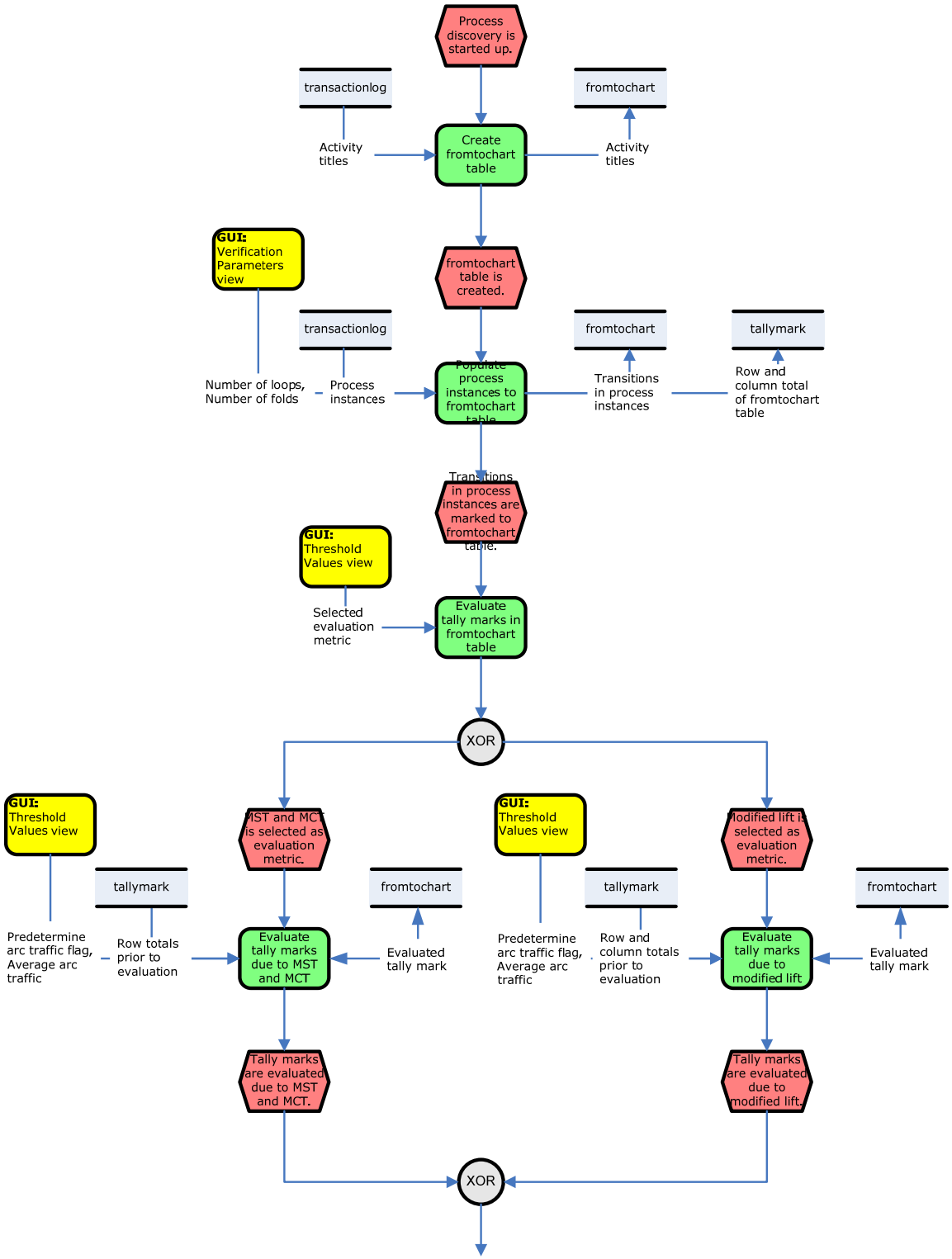


Figure 4.8 Event-Based Process Chain (EPC) Diagram for FTCBPD Methodology

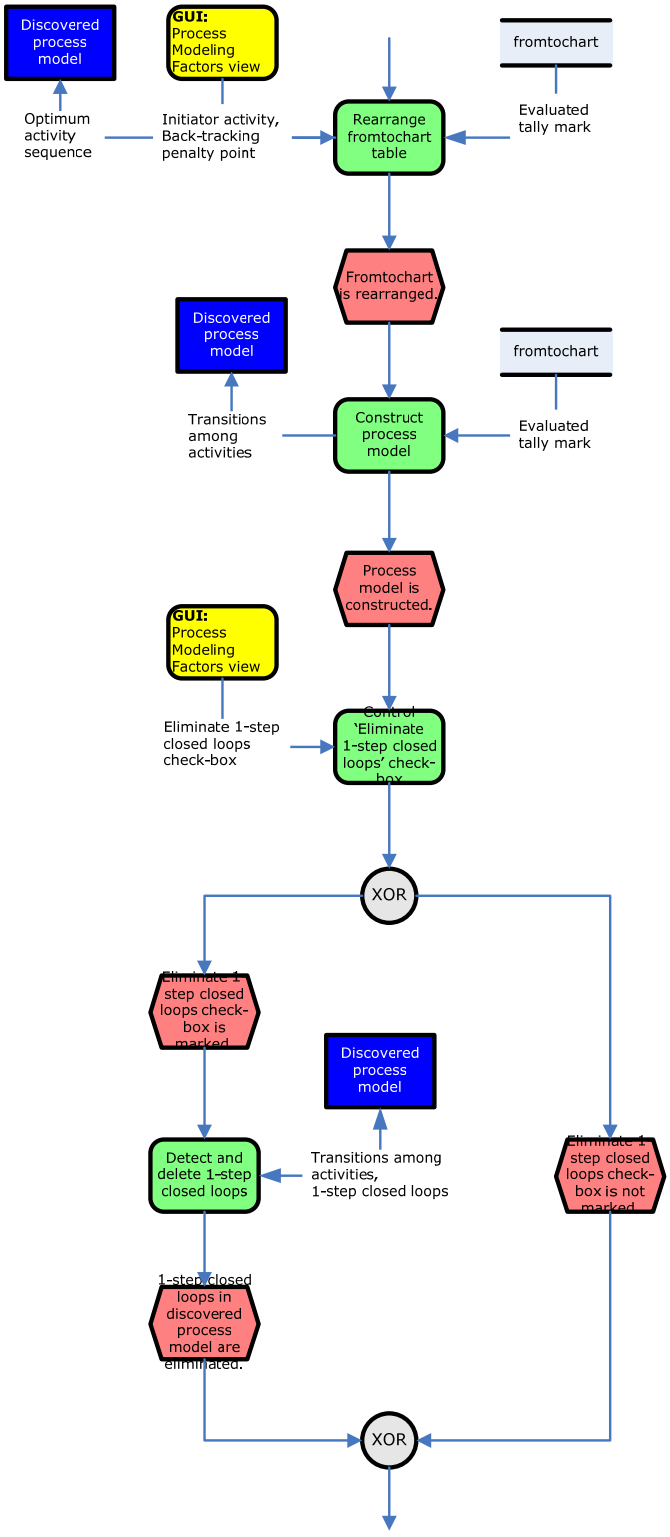


Figure 4.8 (continued)

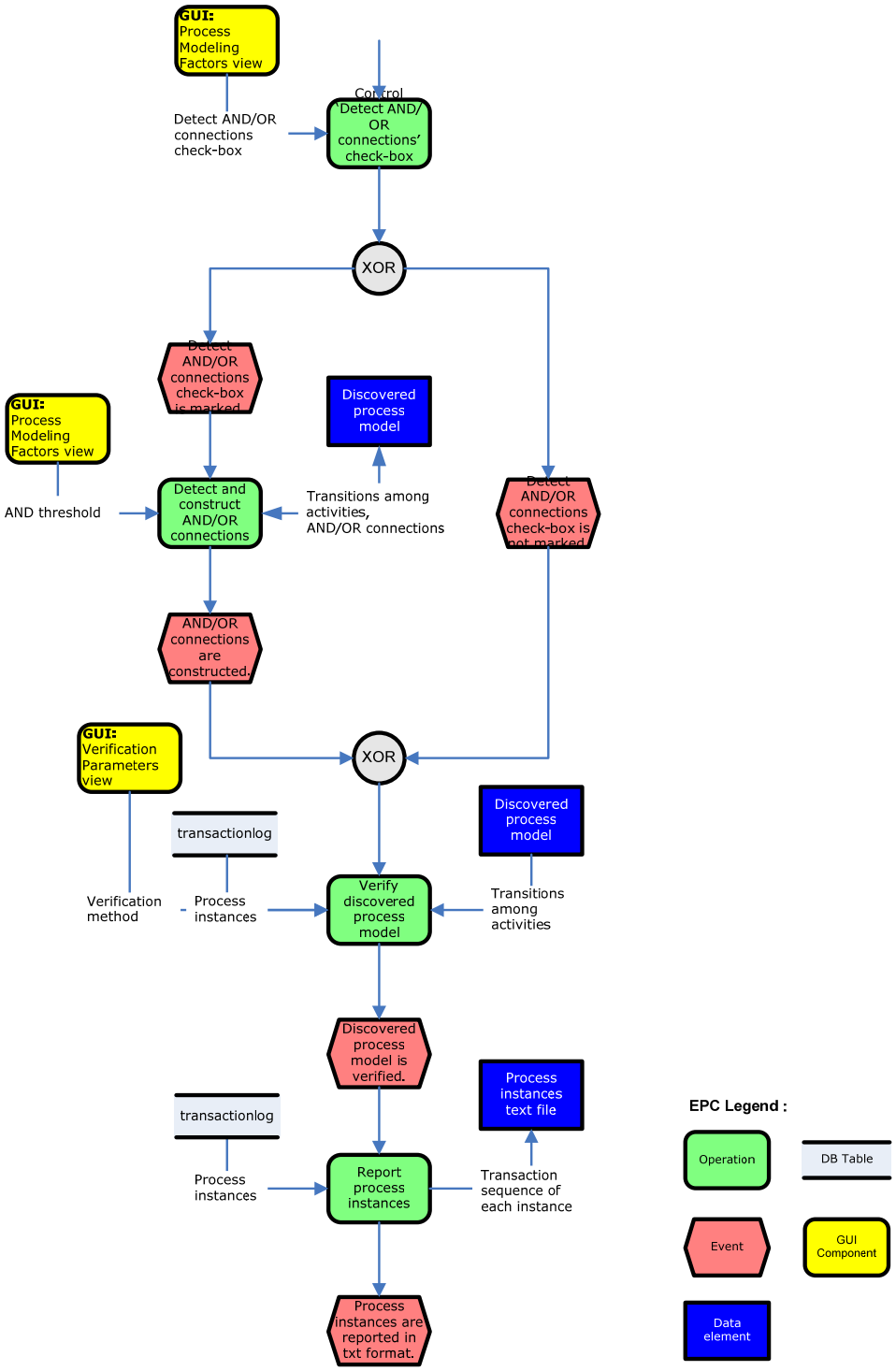


Figure 4.8 (continued)

4.2.1. Operational Perspective of FTCBPD Methodology

4.2.1.1. Create FROMTOCHART Table Operation

The starting point of FTCBPD methodology is the creation of a so-called FROMTOCHART table from the event log information. FROMTOCHART is the basic bookkeeping material, on which major transitions among activities (transactions) in process instances are marked (weaved) and direct succession, succession and back-tracking type relations between adjacent activities are interpreted. Structure of FROMTOCHART is stated in data dictionary, APPENDIX A.

As the first step, if an older version exists in database schema, FROMTOCHART table is dropped. Then activity titles are retrieved distinctly from TRANSACTIONLOG table, which stores event logs. Hence only activities (transactions), which contribute to the process instances, are represented as blocks at dependency/frequency graph and control flow graph. Retrieved activity titles are added dynamically to CREATE TABLE statement as attributes[2]. In addition, retrieved activity titles are added to INSERT statement dynamically to insert null tuples (i.e. tuples with not-null tcode attribute and null values at dynamically created activity type attributes) into FROMTOCHART table. Hence symmetric and square matrix characteristics of traditional from-to chart are reflected to FTCBPD methodology.

Pseudo-code for “Create FROMTOCHART Table” operation is stated below:

```
/* Pseudo-code of 'Create FROMTOCHART table' Operation
DROP FROMTOCHART table in thesis database
READ activity type values in TRANSACTIONLOG table
ADD activity type values into creation SQL statement
ADD activity type values into insertion SQL statement
CREATE FROMTOCHART table in thesis database
INSERT null tuples into FROMTOCHART table
```

[2] Header of CREATE TABLE statement is "CREATE TABLE FROMTOCHART (tcode VARCHAR (20) NOT NULL, ". This portion of the statement is constant.

Activity types, which take place in process instances, are added by " INTEGER (4) DEFAULT “0”, " tag dynamically to the CREATE TABLE statement.

Footer of CREATE TABLE statement is "PRIMARY KEY (tcode));" which defines the primary key of FROMTOCHART table.

Activity diagram of “Create FROMTOCHART Table” operation is stated at Figure 4.9:

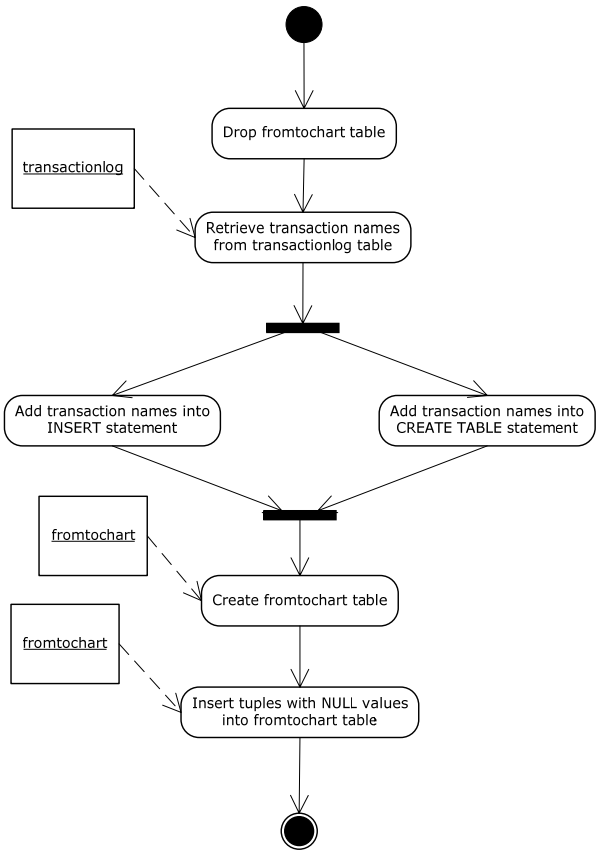


Figure 4.9 Activity Diagram of “Create FROMTOCHART Table” Operation

4.2.1.2. Populate FROMTOCHART Table Operation

Firstly complete log (transaction stream) belonged to a process instance has to be retrieved in the following way: Per process instance, all event log entries have to appear in the order in which they took place in timeline. In other words, event logs have to be arranged by process instances and then ordered by timestamp in ascending order. Thus transaction streams are constructed successfully. Sample transaction streams in Figure 4.10 indicate process instance identifier (i.e. caseID) and activities executed for the underlying process instance in the order of timestamp.

```
1100000006 : pr_sy_trip01, pr_sy_trip03, pr_sy_trip08, pr_sy_trip06, pr_sy_trip07,
1100000007 : pr_sy_trip01, pr_sy_trip05, pr_sy_trip08, pr_sy_trip06, pr_sy_trip07,
1100000008 : pr_sy_trip01, pr_sy_trip02, pr_sy_trip03, pr_sy_trip05, pr_sy_trip08, pr_sy_trip06, pr_sy_trip07,
1100000009 : pr_sy_trip01, pr_sy_trip02, pr_sy_trip05, pr_sy_trip08, pr_sy_trip07,
1100000010 : pr_sy_trip01, pr_sy_trip02, pr_sy_trip08, pr_sy_trip06, pr_sy_trip07,
```

Figure 4.10 Sample Transaction Streams

Retrieved process instance index (i.e. `retrievedIndex`) variable is used to determine the dataset (training or test), to which underlying process instance will be dedicated. After the dataset is determined, predecessor and successor activities are parsed for each transition in transaction streams, which are assigned to training dataset. Then current tally mark of (predecessor, successor)th element in FROMTOCHART table is incremented by one. As a result, all transitions among activities (transactions) in process instances, which are assigned to training dataset, are marked (weaved) to FROMTOCHART table.

As the last step, current row and column totals of each activity in FROMTOCHART table are calculated and inserted into TALLYMARK table, since these values are influenced by "Evaluate Tally Marks in FROMTOCHART Table" operation[3]. Also this information in TALLYMARK table is used in evaluation operation.

Pseudo-code of "Populate FROMTOCHART table" operation is as follows:

```
/* Pseudo-code of 'Populate FROMTOCHART table' Operation
INIT concentrated caseID to 0
INIT concentrated activity type TO null
INIT process instance index to 0
INIT predecessor TO null
INIT successor TO null
INIT transaction stream TO null
INIT row total to 0
INIT column total to 0
SET activity type set TO (READ activity type values in TRANSACTIONLOG table)
SET caseID set TO (READ caseID values in TRANSACTIONLOG table)
WHILE caseID set has more elements
       GET concentrated caseID FROM caseID set
       SET transaction stream TO (READ activity type of concentrated caseID
        ORDERED BY transaction log date AND transaction log date in TRANSACTIONLOG table)
       INCREMENT process instance index BY 1
       IF process instance index % fold number != 0 THEN
              FOR i=1 to (transaction stream length-1)
                     SET predecessor TO activity type at ith line of transaction stream
                     SET successor TO activity type at (i+1)th line of transaction stream
                     INCREMENT score of element (predecessor, successor) BY 1
                     INCREMENT i BY 1
              ENDFOR
       ENDIF
ENDWHILE
WHILE activity type set has more elements
       GET concentrated activity type FROM activity type set
       SET row total TO (READ row total of concentrated activity in FROMTOCHART table)
       SET column total TO (READ column total of concentrated activity in FROMTOCHART table)
       INSERT row total AND column total of concentrated activity type into TALLYMARK table
ENDWHILE
```

[3] Tally marks are reset to zero or multiplied by minus one in "Evaluate tally marks in FROMTOCHART table" operation. Thus current row and columns total of FROMTOCHART table diminish after this operation.

Activity diagram of “Populate FROMTOCHART Table” operation is stated at Figure 4.11:

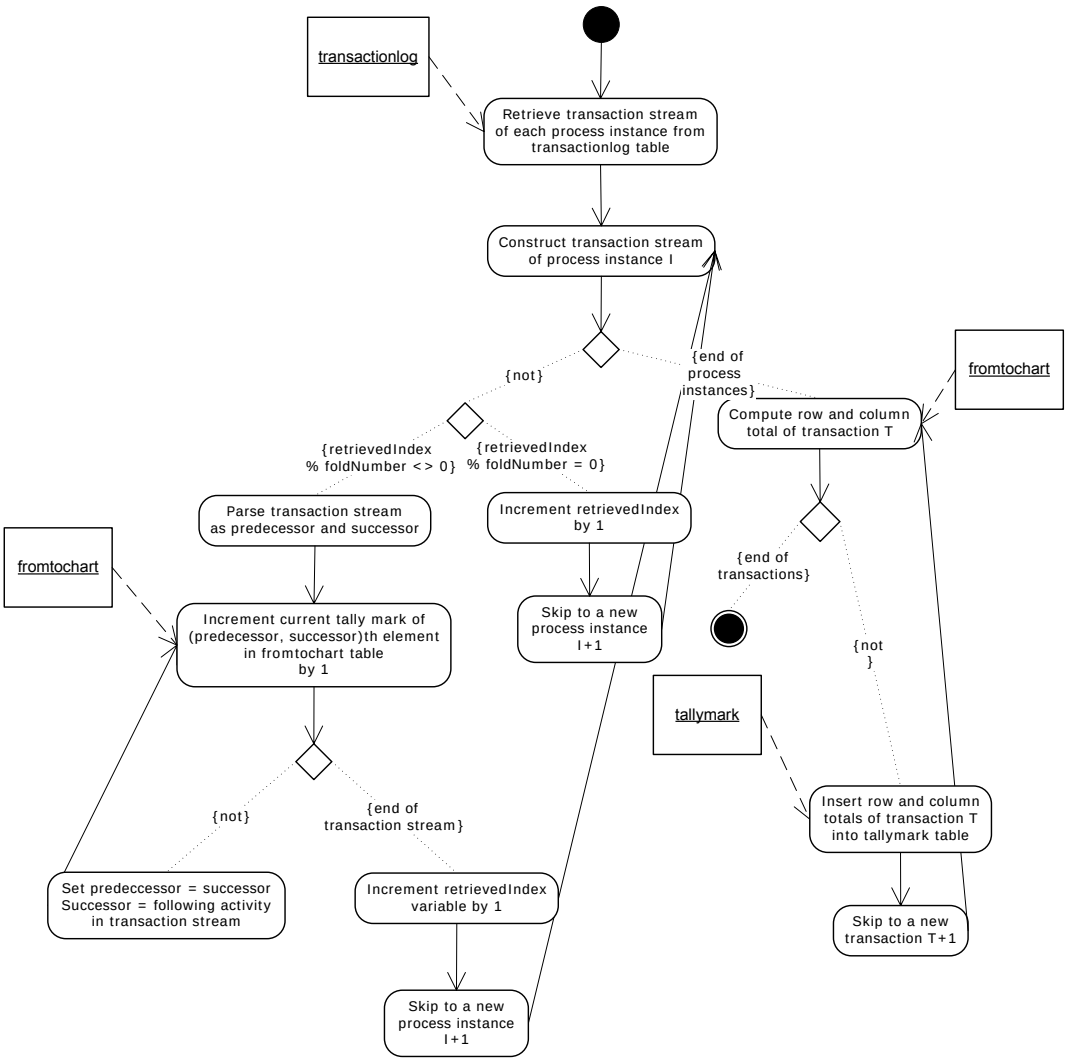


Figure 4.11 Activity Diagram of “Populate FROMTOCHART Table” Operation

4.2.1.3. Evaluate Tally Marks in FROMTOCHART Table Operation

While frequent pattern mining concentrates on the discovery of associations and correlations among entities in large transactional or relational datasets, which are overstated by massive amounts of data continuously being collected and stored, an indispensable requirement of rule induction procedure is emerged [24]. By “Evaluate Tally Marks in FROMTOCHART Table” operation, it is aimed to prune down the rules on the basis of some evaluation metrics.

In traditional from-to chart implementation, total score of each element is directly taken into consideration in rearrangement of the matrix. But by pruning down the "weak" scores prior to rearrangement, it is attempted to eliminate their effect on the optimum activity sequence in FTCBPD methodology. These evaluation metrics (except modified lift) are tuned by end-users. There are three evaluation metrics used in FTCBPD methodology:

a. Minimum Confidence Threshold

Minimum confidence threshold (MCT) is the ratio of transitions, which are from predecessor A to successor B, to total transitions which are initiated by activity A (i.e. row total of activity A in FROMTOCHART table). Basically, formulation of MCT is similar to classical confidence definition in association rule mining except predecessor and successor notations [23].

$$MCT = \frac{|A > B|}{|A > *|} \quad (eq.\ 4.2)$$

$$Confidence = \frac{|A \cup B|}{|A|} \quad (eq.\ 4.3)$$

Parameters stated in the metric are:

$|A > B|$ is total number of transitions from activity A to activity B.

$|A > *|$ is total number of transitions initiated by activity A (row total of activity A in FROMTOCHART table).

MCT takes into account the magnitude of $|A > B|$ and $|A > *|$. In this aspect, MCT is analogous to local metric (LM) stated in [14] and briefly explained in "Literature Review" chapter. LM expresses the tendency of succession relation by comparing the magnitude of $|A > B|$ versus $|B > A|$. The idea of LM measure concentrates on pair-wise moves. By this way, it is aimed to challenge activities dually, come up with the "strongest" pair-wise direct successive relations and then construct process model based on these discovered direct successive relations. The formulation of local metric (LM) is given in "Literature Review" chapter.

b. Minimum Support Threshold

The previous metric, MCT is expressing the tendency of succession by comparing the magnitude of $|A > B|$ and $|A > *|$ at local level. But for determining the likelihood of

succession between two activities A and B, minimum support threshold (MST) is indeed a global metric because it takes into account the overall process instances.

MST is the ratio of transitions, which are from predecessor A to successor B, to total number of process instances dedicated to training dataset (i.e. $\#L$). In initial version of MST, it is decided to place overall total of scores, which are marked to FROMTOCHART table, as the denominator of MST. However it would make the metric highly sensitive to number of activity types, which is a domain-dependent parameter (i.e. number of activity type is related to the underlying business process). Correlation between number of activity types and size stated in "Experimental Results" chapter highlights this implication. Since MST and MCT metrics are adjusted by end-users, end-users have to take into consideration the number of activity types while setting MST value in that case. Actually this case is seemingly unrealistic. As a result, MST is designated to be independent to the number of activity types.

As in MCT, MST is similar to classical support definition in association rule mining except predecessor and successor notations [23].

$$MST = \frac{|A > B|}{\#L} \quad (eq.\ 4.4)$$

$$Support = \frac{|A \cup B|}{\#T} \quad (eq.\ 4.5)$$

Parameters stated in the metric are:

$|A > B|$ is total number of transitions from activity A to activity B.

$\#L$ is total number of process instances in training dataset.

MST is similar to global metric (GM), which is stated in [14] and briefly explained in "Literature Review" chapter. GM expresses the likelihood of succession between two activities A and B by overall frequency of activities A and B. The difference of transitions between activity A and B in GM measure ($|A > B| - |B > A|$) aims to eliminate the effect of noise in event logs and monitor direct succession relation among pair-wise activities. The formulation of global metric (GM) is given in "Literature Review" chapter.

The level of robustness in discovered process model can be easily controlled by the end-user or domain expert through the probability threshold parameters in "Threshold Values"

view. These parameters can also be used to manage the complexity of the discovered model from large amounts of data; eliminating low-probability sequences, even if they are not due to the noise factor [3]. While extracting certain sequential patterns whose support and confidence value exceed a predefined MST and MCT, some sequential patterns with a high confidence value that do not satisfy MST are still interesting. This type of patterns is called *surprise*, whose occurrence rate differs significantly from the expected occurrence by treading every activities equal [22].

c. Modified Lift

Support and confidence measures are practical barriers, which are "supervised" by end-user. Thus lowered MST and MCT values may encourage FTCBPD methodology in representing all patterns exhibited in training dataset and they may turn into insufficient evaluation metrics at filtering out uninteresting rules. To tackle this weakness, a correlation measure named modified lift can be used to augment the support-confidence framework for process mining rules.

Modified lift is a simple correlation measure that is given as follows. The occurrence of activity B is independent of the occurrence of activity A if $P(A > B) = P(A > *) \times P(* > B)$; otherwise, activities A and B are dependent and correlated.

$$Modified\ lift(A,B) = \frac{|A > B| \times \#G}{|A > *| \times |* > B|} \quad (eq.\ 4.6)$$

$$lift(A,B) = \frac{P(A \cup B)}{P(A) \times P(B)} \quad (eq.\ 4.7)$$

Parameters stated in the metric are:

$|A > B|$ is total number of transitions from activity A to activity B.

$|A > *|$ is total number of transitions initiated by activity A (row total of activity A at FROMTOCHART table).

$|* > B|$ is total number of transitions attained to activity B (column total of activity B at FROMTOCHART table).

$\#G$ is gross total tally mark in FROMTOCHART table.

One of the modifications performed on original lift measure stated in [23] is predecessor and successor notations. But major modification is performed on the evaluation procedure of tally marks with respect to calculated lift value such that;

- If the resulting value of modified lift is greater than 1, then activities A and B are positively correlated, meaning that the occurrence of activity A triggers the occurrence of activity B. Thus tally mark of element (A, B) does not change.
- If the resulting value of modified lift is equal to 1, then activities A and B are independent and there is not any correlation between these activities. Thus tally mark of element (A, B) is reset to zero.
- If the resulting value of modified lift is less than 1, then activities A and B are negatively correlated, meaning that the occurrence of activity A discourages the occurrence of activity B. Thus tally mark of element (A, B) is multiplied by minus 1.

This “minus one” factor is inherited from “the Big M method” in linear programming. If a linear programming has any ≥ or = constraints, a starting basic feasible solution may not be readily apparent. The Big M method is a version of the Simplex Algorithm that first finds a basic feasible solution by adding “artificial” variables to the problem. The objective function of the original linear programming must, of course, be adjusted to ensure that the artificial variables are all equal to 0 at the conclusion of the simplex algorithm [27].

When the Big M method is applied, the major argument is to determine “how large M should be”. Because extremely large coefficients may dominate objective function and deviate the optimum solution. In this aspect, multiplying tally mark by minus 1 can be evaluated as a “realistic” value to instruct the rearrangement operation that activity A and B are negatively correlated without affecting optimum activity sequence. Consequently rearrangement operation tends to separate these activity pairs due to decrement in moment calculation.

d. Predetermine Arc Traffic in Process Model Feature

Theoreticians consider a process model to be “good” if it is accurate and minimal. By accurate they emphasize that the process model represent all and only the behaviors of the processes in the training dataset. Minimality implies that the process model contains the fewest number of activities (transactions) and transitions, which do not needlessly complicate the patterns identified with extra activities and transitions [3].

In this aspect, “Predetermine arc traffic in process model” feature is added to “Threshold Values” view to accomplish a predefined minimality level in discovered process model. This feature aims to construct a user-driven transition list (namely transition top list), whose limit

(TTLL) is determined by number of activity types (|AT|) and average arc traffic (AAT) as follows:

$$TTLL = \lfloor |AT| \times AAT \rfloor \quad (eq.4.8)$$

Parameters stated in the formula are:

$TTLL$ is the transaction top list limit.

$|AT|$ is the number of activity types in discovered process model.

AAT is average arc traffic parameter set in “Threshold Values” view.

Consequently TTL acts as an upper bound for transitions represented in discovered process model by only holding the transitions with higher evaluation metric value(s).

As the first step in “Rearrange Tally Marks in FROMTOCHART Table” operation, each tally mark in FROMTOCHART table is retrieved. Afterwards retrieved tally marks are challenged with selected evaluation metric(s). If tally mark is unsatisfactory with respect to evaluation metric(s) in this challenging step, then it is reset to zero (or multiplied by minus one). Otherwise, it is checked whether “Predetermine arc traffic in process model” feature is activated. If this feature is activated, then current size of transaction top list (TTL) is controlled. In the case of “full” TTL, the last element of TTL is compared with current tally mark. If current tally mark provides a better evaluation metric value, then the last element is replaced with the current tally mark and tally mark of popped element in FROMTOCHART table is reset to zero (or multiplied by minus one). Otherwise current tally mark is reset to zero (or multiplied by minus one). In the case of “empty” TTL, a new element tracing current tally mark is directly pushed into TTL.

Pseudo-code of “Evaluate Tally Marks in FROMTOCHART Table” operation is as follows:

```
/* Pseudo-code of 'Evaluate tally marks in FROMTOCHART table' Operation
INIT tally mark TO 0
INIT row total TO 0
INIT column total TO 0
INIT tally mark gross total TO 0
FOR i=1 to activity type number
        FOR j=1 to activity type number
                SET tally mark TO (READ score at element (i,j) in FROMTOCHART table)
                IF evaluation metric EQUAL MST and MCT THEN
                        SET row total TO row total of activity type i in TALLYMARK table
                        IF (tally mark < row total×MCT) OR
                         (tally mark < process instance number×MST) THEN
                                SET score at element (i,j) in FROMTOCHART table TO 0
                        ELSE
                                IF determine arc traffic EQUAL true THEN
                                        IF transition top list current length < limit THEN
                                                PUSH element (i,j) to transition top list
                                                 with (calculated MCT×MST)
                                                INCREMENT transition top list current length BY 1
                                        ELSE
                                                POP element with minimum (calculated MCT×MST)
                                                SET score at popped element in FROMTOCHART table TO 0
                                                PUSH element (i,j) to transition top list
                                                 with (calculated MCT×MST)
                                        ENDIF
                                ENDIF
                        ENDIF
                ELSE
                        SET row total TO row total of activity type i
                         in TALLYMARK table
                        SET column total TO column total of activity type j
                         in TALLYMARK table
                        CALCULATE tally mark gross total in FROMTOCHART table
                        IF ((tally mark×tally mark gross total) < (row total×column total)) THEN
                                SET score at element (i,j)
                                 in FROMTOCHART table TO tally mark×(-1)
                        ELSEIF ((tally mark×tally mark gross total) = (row total×column total)) THEN
                                SET score at element (i,j)
                                 in FROMTOCHART table TO 0
                        ELSE
                                IF determine arc traffic EQUAL true THEN
                                        IF transition top list current length < limit THEN
                                                PUSH element (i,j) to transition top list
                                                 with calculated modified lift
                                                INCREMENT transition top list current length BY 1
                                        ELSE
                                                POP element with minimum (calculated modified lift)
                                                SET score at popped element in FROMTOCHART table TO 0
                                                PUSH element (i,j) to transition top list
                                                 with calculated modified lift
                                        ENDIF
                                ENDIF
                        ENDIF
                ENDIF
                INCREMENT j BY 1
        ENDFOR
        INCREMENT i BY 1
ENDFOR
```

Activity diagram of “Evaluate Tally Marks in FROMTOCHART Table” operation is stated at Figure 4.12:

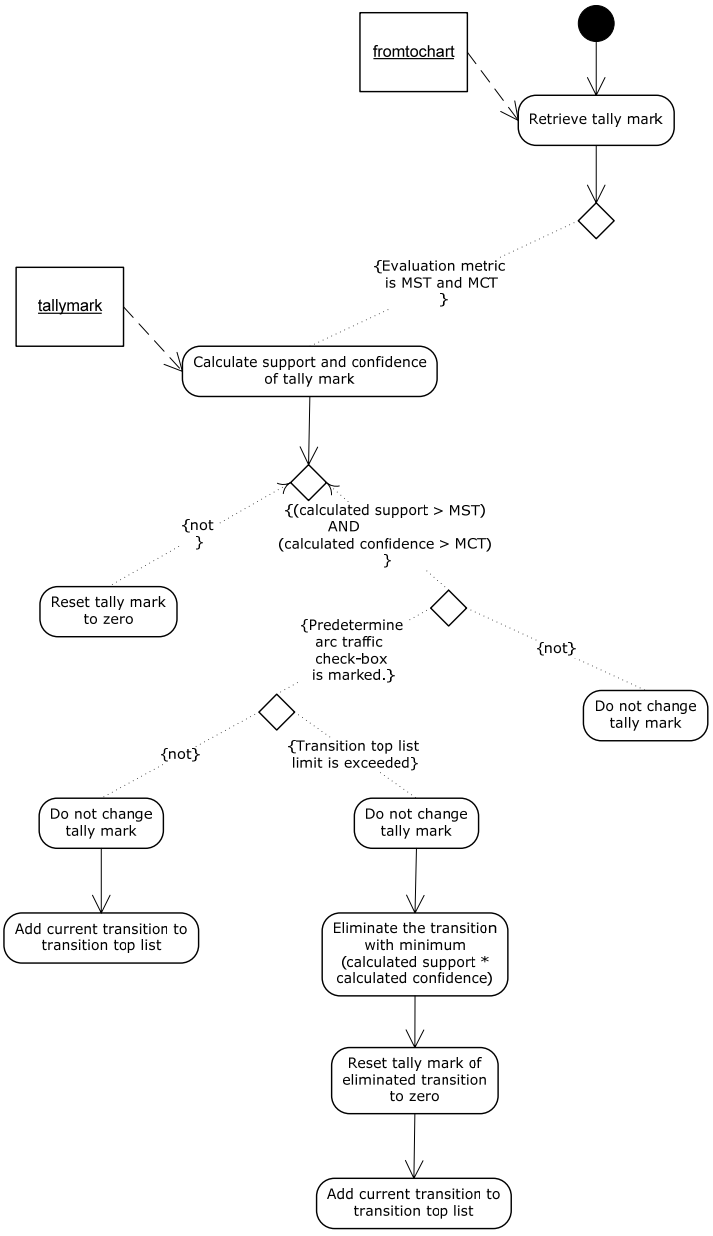


Figure 4.12 Activity Diagram of “Evaluate Tally Marks in FROMTOCHART Table” Operation

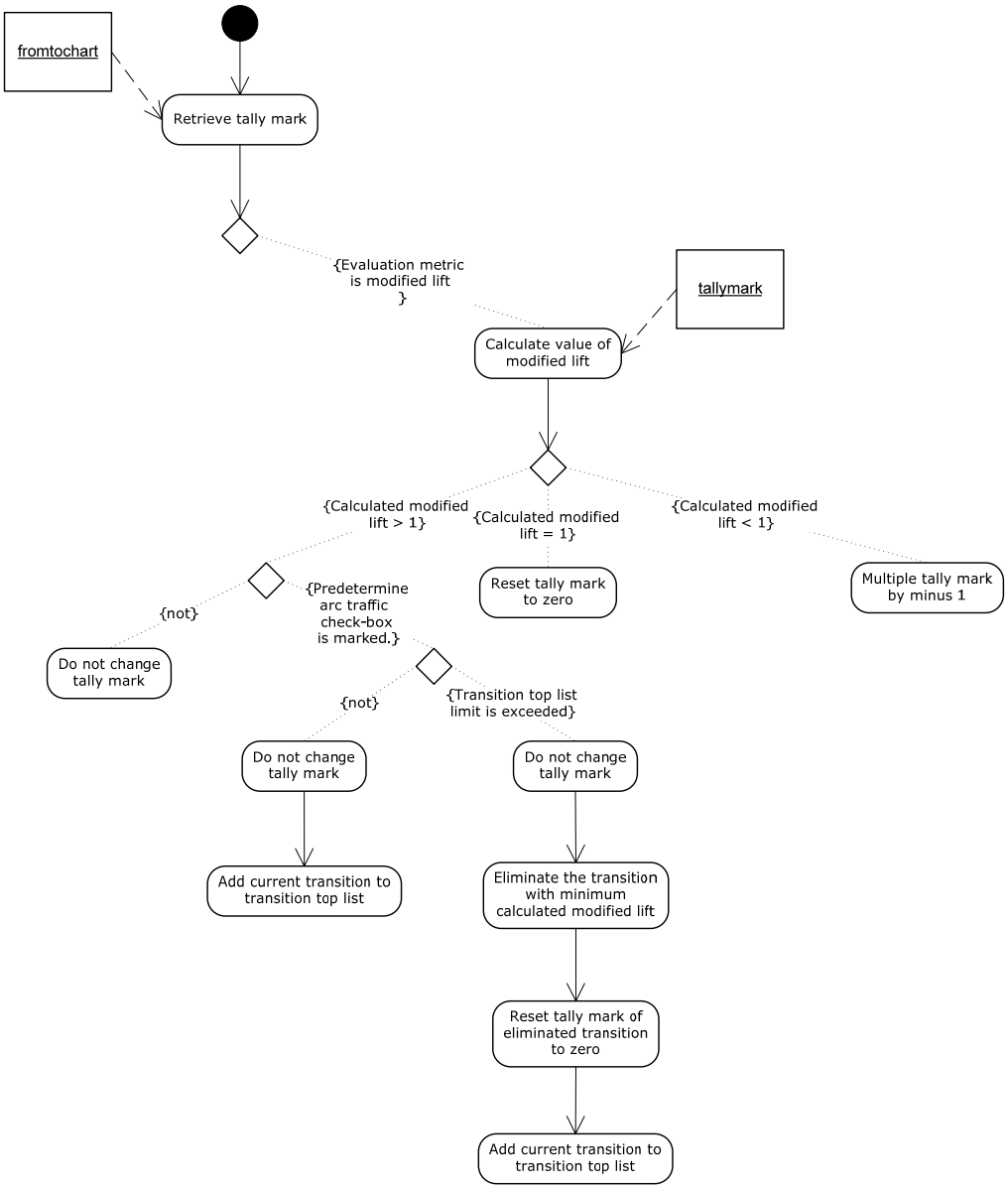


Figure 4.12 (continued)

4.2.1.4. Rearrange FROMTOCHART Table Operation

"Rearrange FROMTOCHART Table" operation is the "engine component" of FTCBPD methodology, which aims to find out the optimum activity (transaction) sequence that provides the minimum overall moment in FROMTOCHART table. Intuitively, the best sequence is accomplished by positioning the elements with higher tally marks along and above the diagonal. Hence straight-line transitions between the activities with higher inter-traffic are concentrated. Unfortunately this qualitative solution is not directly applicable in systematic manner, since neighboring two activities with high inter-traffic may lead to indispensable increase in total moment value due to other activities, which are also in a

strong interaction with these two activities. Therefore a permutative approach is implemented in rearranging the columns of FROMTOCHART table to find out “the most frequent” activity sequence.

Initial step in rearrangement of FROMTOCHART table is to generate activity sequences. This is performed by *Permutation object*, which creates activity sequences from the seed set, composed of activity types occurred in event logs. The length of generated activity sequence is determined by the number of existing activity types.

Each generated activity sequence is checked whether the first element matches the initiator activity (transaction) set by end-user in “Process Modeling Factors” view. If it does not match, this sequence is discarded. Otherwise activities in generated sequence are added to SELECT statement iteratively. After SELECT statement is prepared, tuples in FROMTOCHART are retrieved according to the order emphasized at the generated activity sequence. This order is crucial to calculate the “moment arm” accurately.

In the following step, each tuple’s moment value is calculated locally. The most significant notion in this step is the diagonal (i.e. support point), which is the nominal position of column, whose name matches to `tcode` attribute value of underlying row. Tally marks in the underlying row are multiplied by relative distance of related column to the diagonal ($|column - diagonal|$). In the case of back-tracking ($concentrated\ column < diagonal$), calculated moment value is multiplied by back-tracking penalty point, which is set by the end-user in “Process Modeling Factors” view. By this way, back-tracking relations are discouraged in the optimum activity sequence. Sample moment calculations are given in Figure 4.13.

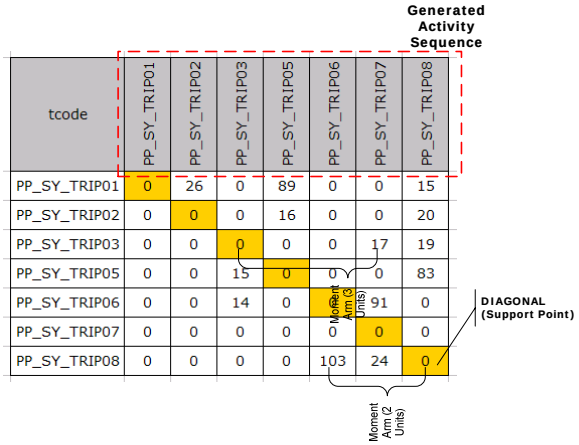


Figure 4.13 Sample Moment Calculations

$moment_{pr\_sy\_trip01} = 26 \times 1 + 89 \times 3 + 15 \times 6 = 383$

$moment_{pr\_sy\_trip03} = 17 \times 3 + 19 \times 4 = 127$

$moment_{pr\_sy\_trip05} = 15 \times 1 \times 2 + 83 \times 3 = 279$

$moment_{pr\_sy\_trip08} = 103 \times 2 \times 2 + 24 \times 1 \times 2 = 460$

$moment_{pr\_sy\_trip02} = 16 \times 2 + 20 \times 5 = 132$

$moment_{pr\_sy\_trip06} = 14 \times 2 \times 2 + 91 \times 1 = 147$

$moment_{pr\_sy\_trip07} = 0$

$total\ local\ moment = 1528$

Figure 4.13 (continued)

After all tuples are traversed, calculated total moment value is compared to local optimum moment value (i.e. minimum moment value during rearrangement operation). If it is less than local optimum moment value, then local optimum moment and local optimum activity sequence variables are revised such that; local optimum moment is equalized to calculated total moment value and local optimum activity sequence variable is set to current activity sequence. At the end of activity sequence generation, local optimum moment and local optimum activity sequence variables are assigned to global optimum variables. Process model represented in dependency/frequency graph and control flow graph is constructed according to this global optimum activity sequence variable.

Pseudo-code of “Rearrange FROMTOCHART Table” operation is as follows:

```
/* Pseudo-code of 'Rearrange FROMTOCHART table' Operation
INIT optimum minimum moment TO 0
INIT optimum activity sequence TO 0
CALL Permutation WITH available activity sequence
 RETURNING generated activity sequence set
WHILE generated activity sequence set has more elements
        GET concentrated activity sequence FROM generated activity sequence set
        ADD concentrated activity sequence to query
        IF first element of concentrated activity sequence EQUAL initiator activity THEN
                INIT diagonal TO 0
                INIT calculated moment TO 0
                INIT tally mark TO 0
                EXECUTE query
                FOR i=1 to row number of query result set
                        DETERMINE diagonal of row i
                        FOR j=1 to column number of query result set
                                SET tally mark TO
                                 (READ score READ score at element (i,j) in FROMTOCHART table)
                                IF j < diagonal of row i THEN
                                        calculated moment =
                                         calculated moment + tally mark × (diagonal-j) × bt.p.p.
                                ELSE
                                        calculated moment =
                                         calculated moment + tally mark × (j-diagonal)
                                ENDIF
                                INCREMENT j BY 1
                        ENDFOR
                        INCREMENT i BY 1
                ENDFOR
                IF first moment calculation EQUAL true THEN
                        SET minimum moment TO calculated moment
                        SET optimum activity sequence TO concentrated activity sequence
                ELSE
                        IF calculated moment < minimum moment
                                SET optimum minimum moment TO calculated moment
                                SET optimum activity sequence TO concentrated activity sequence
                        ENDIF
                ENDIF
        ENDIF
ENDWHILE
```

Activity diagram of “Rearrange FROMTOCHART Table” operation is stated at Figure 4.14:

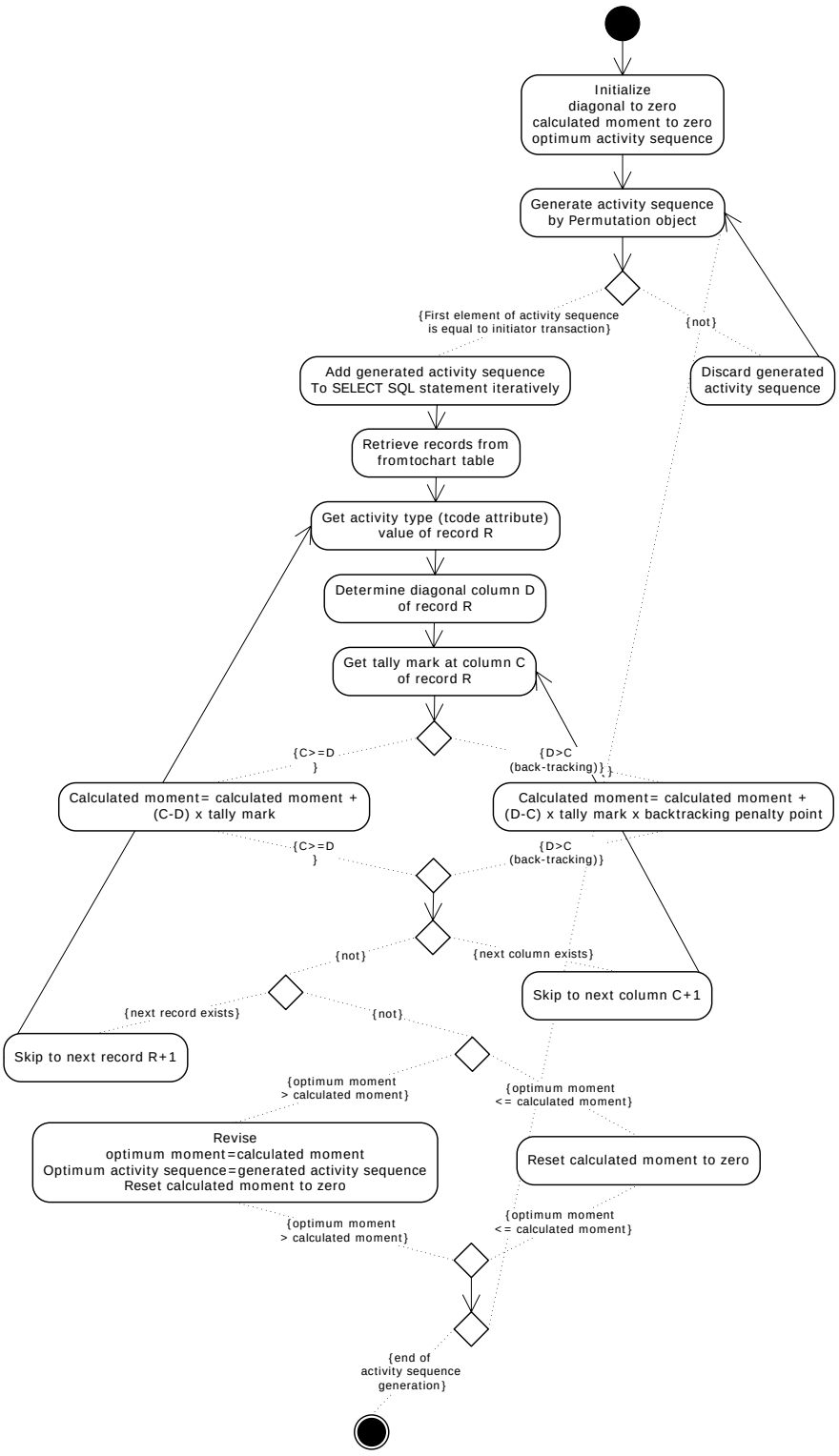


Figure 4.14 Activity Diagram of “Rearrange FROMTOCHART Table” Operation

4.2.1.5. Construct Process Model Operation

Operations up to “Construct Process Model” aim to find out the activity (transaction) sequence, which minimizes the distance that is taken by the transitions among activities, by locating the activities with higher inter-traffic in neighboring positions. Although this activity sequence is the backbone of discovered process model, it is based on the philosophy of arranging the operations in a straight line. Hence activity sequence is limited to sequential behavior. Consequently this outcome has to be upgraded to process model by major value-adding modifications such that;

a. Discovering one-to-one, one-to-many relations among activities (transactions) and constructing dependency/frequency graph with respect to these relations.
b. Monitoring the discovered successors and predecessors of the activities and eliminating one-step closed loops.
c. Discovering basic parallelisms in discovered process model and representing these relations by pair-wise AND/OR connectors in control flow graph.

In this aspect, “Construct Process Model” operation aims to discover one-to-one, one-to-many relations among activities (transactions) by concentrating on the evaluated tally marks in FROMTOCHART table. Evaluated tally marks are the essential stick yard to determine whether the underlying move is worth to be visualized in dependency/frequency graph and control flow graph such that;

- Tally mark at (i, j)th element, which is greater than zero (non-reset tally mark), highlights that the underlying transition is significant with respect to evaluation metric(s). Therefore successor of the transition (activity j) is added to the successor activity list of predecessor (activity i) and predecessor of the transition (activity i) is added to the predecessor activity list of successor (activity j).
- Negative or zero-value tally mark indicates that the transition is challenged by the evaluation metric(s). Therefore “Construct Process Model” operation skips these elements in FROMTOCHART table.

There are three types of relations presented in discovered process model:

a. Direct succession relation:

Let S be an optimum activity sequence over T, a set of activities, i.e. $S \in T^*$ (i.e. a permutation of T) and A, B $\in$ T. Then B *directly succeeds* A if and only if position of A (p_A) and position of B (p_B) at S are two successive integers such that; $p_B = p_A + 1$, where $p_A, p_B \in [1, |S|]$.

This implication is similar to direct succession relation definition in [14]. That definition states that;

> **Definition 3**: *(Direct succession relation)* Let W be a workflow log over T, i.e. $W \subseteq T^*$ and A, $B \in T$. Then B *directly succeeds* A (notation $A \rightarrow_W B$) if either:
>
> 1. $(A>B) > 0$ and $(B>A) = 0$
>
> or
>
> 2. $(A>B) > 0$ and $(B>A) > 0$ and $((A>B) - (B>A) \geq \sigma)$, $\sigma > 0$.

In fact, the philosophy of "a significant difference in transition numbers indicates direct succession among these activities" (i.e. σ) is implicitly emphasized in rearrangement operation such that; if the effect of relatively "larger" tally mark gained by positioning it closer to the diagonal compensates the effect of "fewer" tally mark that is exaggerated by back-tracking penalty point, rearrangement operation tends to place underlying activity pair at adjacent positions in the activity sequence. Thus a straight-line type relation among these activities is constructed.

b. Succession relation:

Let S be an optimum activity sequence over T, a set of activities, i.e. $S \in T^*$ (i.e. a permutation of T) and A, B $\in$ T. Then B *succeeds* A if and only if position of A (p_A) and position of B (p_B) at S are two integers such that; $p_B > p_A + 1$, where $p_A, p_B \in [1, |S|]$.

This implication is also similar to succession relation definition in [14]. That definition states that;

> **Definition 2**: *(Succession relation)* Let W be a workflow log over T, i.e. $W \subseteq T^*$. Let A, $B \in T^*$. Then:
>
> - *B succeeds A* (notation $A >_W B$) if and only if there is a trace $\theta = t_1 t_2 ... t_{n-1}$ and $i \in \{1,\ldots, n-2\}$ such that $\theta \in W$ and $t_i = a$ and $t_{i+1} = b$.
>
> In the log from Table 1, $A >_W F$, $F >_W G$, $B >_W C$, $H >_W G$, etc.
>
> - we denote $(A>B) = m$, $m \geq 0$, where m is the number of cases for which the relation $A >_W B$ holds. For example, if for the log W, the relation $A >_W B$ holds 100 times and the relation $B >_W A$ holds only 10 times, then $(A>B) = 100$ and $(B>A) = 10$.

Actually every unevaluated tally mark (i.e. non-reset or unnegativated tally mark) in FROMTOCHART table, in which predecessor and successor activities are not neighboring in optimum activity sequence, accomplishes the rule stated at succession relation.

c. Back-tracking relation:

Let S be an optimum activity sequence over T, a set of activities, i.e. $S \in T^*$ (i.e. a permutation of T) and A, B $\in$ T. Then B *backtracks* A if and only if position of A (p_A) and position of B (p_B) at S are two integers such that; $p_B < p_A$, where $p_A, p_B \in [1, |S|]$.

Three types of relations are visualized in Figure 4.15.

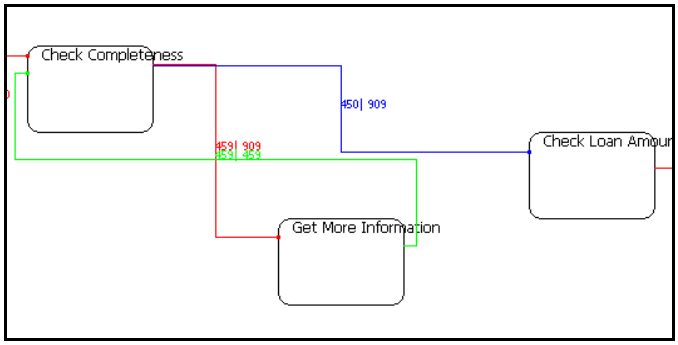


Figure 4.15 Relation Type Visualization in Dependency/Frequency Graph

Pseudo-code of "Construct Process Model" operation is as follows:

```
/* Pseudo-code of 'Construct process model' Operation
INIT tally mark TO 0
FOR i=1 to optimum activity sequence length
        FOR j=1 to optimum activity sequence length
                SET tally mark to (READ score at element (i,j) in FROMTOCHART table)
                IF tally mark > 0 THEN
                        INSERT activity type value of element j
                         at optimum activity sequence TO
                         successor activity list of element i at optimum activity sequence
                        INSERT activity type value of element i
                         at optimum activity sequence TO
                         predecessor activity list of element j at optimum activity sequence
                ENDIF
                INCREMENT j BY 1
        ENDFOR
        INCREMENT i BY 1
ENDFOR
```

Activity diagram of “Construct Process Model” operation is stated at Figure 4.16:

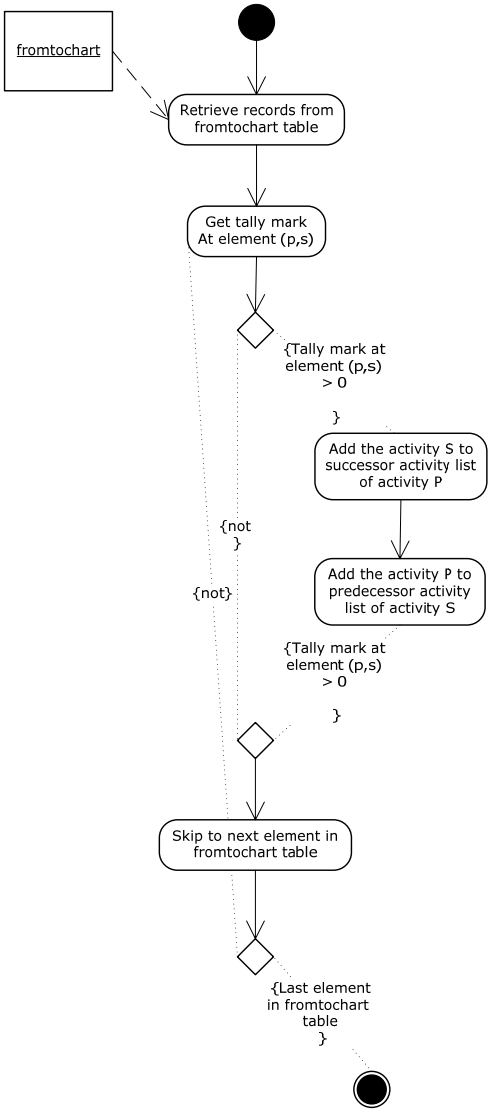


Figure 4.16 Activity Diagram of “Construct Process Model” Operation

4.2.1.6. Eliminate One-Step Closed Loops at Discovered Process Model Operation

Although the most important outcome from a process mining perspective is the discovery of the main flow in the underlying business process, it can be clearly be seen that detecting the existence of closed loops in discovered business process is also crucial in monitoring the bottlenecks, which cannot be deduced from the predefined process model.

If “Eliminate one-step closed loops at discovered process model” check-box in “Process Modeling Factors” view is marked, this operation is activated in FTCBPD methodology. Basic one-step closed loops[4] are identified and eliminated according the condition stated below:

> “If a successor of an activity has presented the underlying activity in its own successor activity list, then the transition is removed from the successor activity list of the activity at latter position in global optimum activity sequence.”

Pseudo-code of “Eliminate One-Step Closed Loops at Discovered Process Model” operation is as follows:

```
/* Pseudo-code of 'Eliminate one-step closed loops at discovered process model' Operation
INIT concentrated successor TO null
INIT j TO 0
FOR i=1 to optimum activity sequence length
       GET element i FROM optimum activity sequence
       WHILE successor activity list of element i has more elements
              GET concentrated successor
               FROM successor activity list of element i
              SET j TO (GET index of concentrated successor at optimum activity sequence)
              IF activity type value at element i EXISTS
               at successor activity list of element j THEN
                     REMOVE activity type value at element MIN(i,j)
                      FROM successor activity list of
                      element MAX(i,j) at optimum activity sequence
                     REMOVE activity type value at element MAX(i,j)
                      FROM predecessor activity list of
                      element MAX(i,j) at optimum activity sequence
              ENDIF
       ENDWHILE
       INCREMENT i BY 1
ENDFOR
```

[4] 0-step closed loops indicates the recursion, while 1-step closed loops is used for loops length of one activity (i.e. ABA).

Activity diagram of “Eliminate One-Step Closed Loops at Discovered Process Model” operation is stated at Figure 4.17:

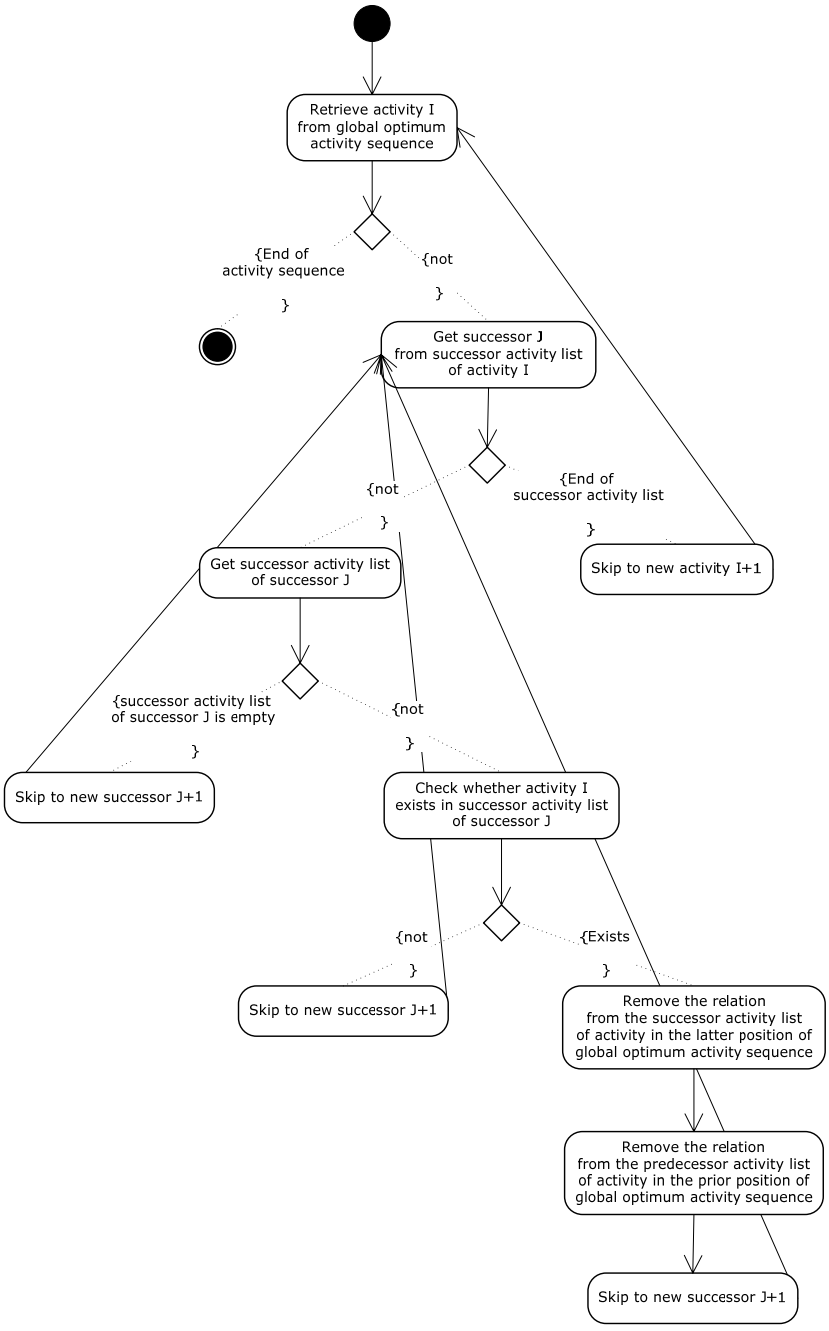


Figure 4.17 Activity Diagram of “Eliminate One-Step Closed Loops at Discovered Process Model” Operation

4.2.1.7. Construct AND/OR Connections at Discovered Process Model Operation

What we seek to infer from event log are recurring patterns of behavior, specifically those involving direct succession, succession and back-tracking. For our purposes, dependency/frequency graph provides a good staring point for expressing those relations. Although dependency/frequency graph is quiet convenient and sufficiently powerful for describing the patterns of actual behavior, additional representation is required to provide a useful process model, which prescribes business process in a grammar inference. In this aspect AND/OR connections are useful entities, which converts graph-oriented meta-model (i.e. dependency/frequency graph) into block-oriented meta-model (i.e. control flow graph), which is always well-formed and sound [12].

Block-structured models are made up from blocks which are nested. These building blocks of the meta-model can be differentiated into activities (transactions) and connectors. Activities (transactions) build the process flow, while connectors are the places, denotes the parallelism among predecessors or successors of underlying activities (transactions) [12]. Unfortunately, mining of these non-observable connections is difficult, since they are not present in the event log explicitly. However (i) information in the dependency/frequency graph and (ii) the dependency score of the transitions in the dependency/frequency graph contains useful information to indicate the types of these connections (AND or OR) in the discovered process model.

"Construct AND/OR Connections at Discovered Process Model" operation handles the transitions of underlying activity (transaction) stated in successor and predecessor activity lists in pair-wise fashion. There are two formulas of calculating connection scores with respect to the type of connections (join or split) as stated in [13]:

$$A \Rightarrow B \wedge C_{SPLIT} = \left(\frac{|B > C| + |C > B|}{|A > B| + |A > C|} \right) \quad (eq.\ 4.9)$$

$$A \Rightarrow B \wedge C_{JOIN} = \left(\frac{|B > C| + |C > B|}{|B > A| + |C > A|} \right) \quad (eq.\ 4.10)$$

While $A \Rightarrow B \wedge C_{SPLIT}$ term indicates the score of potential split-type connection between activities B and C, which are successors of activity A in discovered process model, $A \Rightarrow B \wedge C_{JOIN}$ term indicates the score of potential join-type connection between activities B and C, which are predecessors of activity A in discovered process model.

The denominators of formulas, $|A > B| + |A > C|$ and $|B > A| + |C > A|$, indicates the total number of outgoing and incoming observations and the numerator, $|B > C| + |C > B|$, indicates the number of times activity B and C appear directly after each other.

In quantitative aspect, connection scores, which are greater than the AND threshold set by the end-user in “Process Modeling Factors” view, implies an AND-connection between the related activities. Otherwise it is an OR-connection.

On the other hand detecting the type of a connection can be implemented intuitively such that; it is sufficient to interpret tally marks of predecessors or successors of underlying activity in FROMTOCHART table. In the case of an AND-connection, it is expected a “significant” positive value between the predecessors or successors. If it is an OR-connection, the patterns between the predecessors or successors do not appear in the matrix.

Additionally dependency score of a transition is a good indicator for determining connection type intuitively such as; dependency score, which is approximately equal to total number of incoming or outgoing transitions of the underlying activity, implies an AND-connection, while dependency scores complementing each other to total number of incoming or outgoing transitions of the underlying activity implies an OR-connection [1].

For instance, connection between “Display Travel Request” and “Advance Payment” activities (transactions) are OR-join, since total incoming transition number of “Travel Request Confirmation” activity (180 observations) is approximately equal to the aggregation of transition number outgoing from “Advance Payment” (117 observations) and transition number outgoing from “Display Travel Request” (28 observations). In other words;

$$|Travel\ \ Request Confirmation| \geq |Display\ Travel\ Request| + |Advance\ Payment|$$

This use case is represented in Figure 4.18.

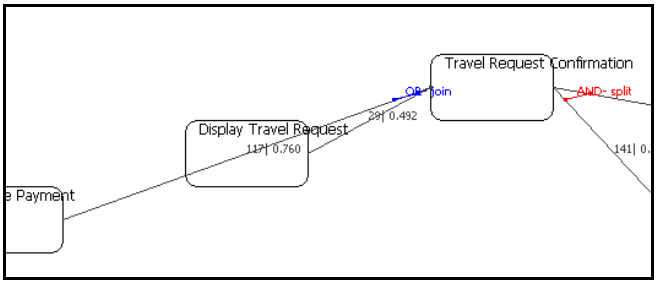


Figure 4.18 Intuitive Way in Determination of Connection Type

Pseudo-code of “Construct AND/OR Connections at Discovered Process Model” operation is as follows:

```
/* Pseudo-code of 'Construct AND/OR connections at discovered process model' Operation
INIT successor activity combination set TO null
INIT predecessor activity combination TO null
INIT concentrated combination TO null
FOR i=1 to optimum activity sequence length
       COMBINE successor activity list of element i BY
        2 RETURNING successor activity combination set
       WHILE successor activity combination set has more elements
              GET concentrated combination FROM successor activity combination set
              CALCULATE connection score of concentrated combination
              IF connection score > AND threshold THEN
                     ADD AND-split type connection TO connection list of element i
              ELSE
                     ADD OR-split type connection TO connection list of element i
              ENDIF
       ENDWHILE
       COMBINE predecessor activity list of element i BY
        2 RETURNING predecessor activity combination set
       WHILE predecessor activity combination set has more elements
              GET concentrated combination FROM predecessor activity combination set
              CALCULATE connection score of concentrated combination
              IF connection score > AND threshold THEN
                     ADD AND-join type connection TO connection list of element i
              ELSE
                     ADD OR-join type connection TO connection list of element i
              ENDIF
       ENDWHILE
       INCREMENT i BY 1
ENDFOR
```

Activity diagram of “Construct AND/OR Connections at Discovered Process Model” operation is stated at Figure 4.19:

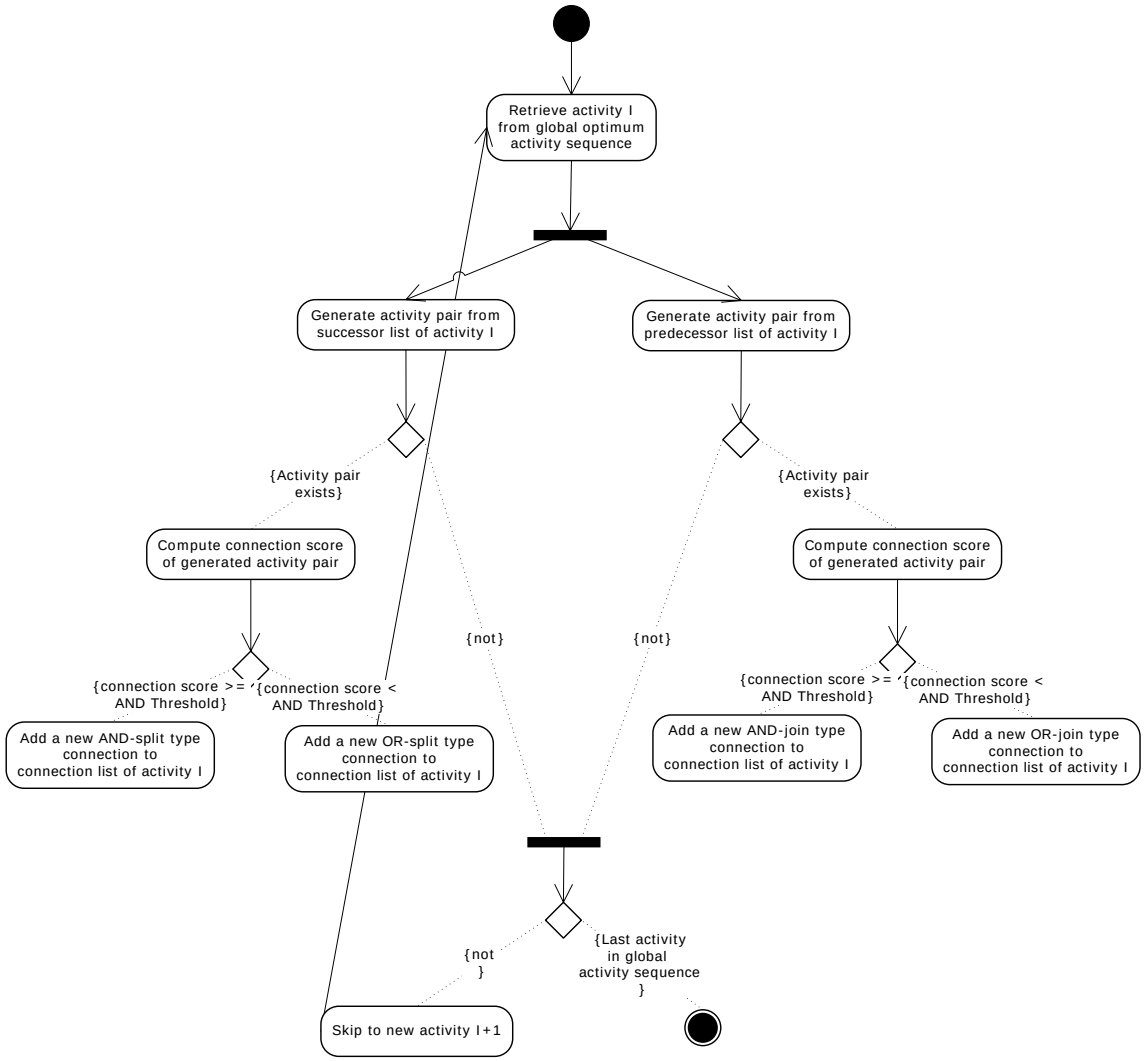


Figure 4.19 Activity Diagram of “Construct AND/OR Connections at Discovered Process Model” Operation

4.2.1.8. Verify Discovered Process Model Operation

After enhancing the notion of basic parallelism and elimination of one-step closed loops to global optimum activity sequence, “Verify Discovered Process Model” operation aims to check to ensure that discovered process model conforms its specifications and represents the behaviors exhibited at event logs. FTCBPD methodology is able to implement two verification methods:

a. Holdout

Holdout method reserves a certain amount of event log for testing and uses the remainder for training. In practical terms, it is common to hold one-third of the data out for testing and use the remaining two-third for training [24].

Unfortunately the sample used for training (or testing) may not be uniformly representative (i.e. a class of transition would necessarily be over-represented in the test dataset, since none of its instances is encountered in the training dataset). Also this non-uniform representation problem is exaggerated by surprise-type sequential patterns, which do not satisfy support threshold while accomplishing a high confidence. Therefore there should be one simple check that might be worthwhile: each of the behavior sub-patterns in the full dataset should be represented in about the right proportion in the training and testing datasets. To handle this limitation, a variation of holdout method named *threefold cross-validation*, in which the holdout method is repeated three times, is implemented in FTCBPD methodology.

b. Cross-Validation

In k-fold cross-validation, the original data are randomly partitioned into k mutually exclusive subsets or folds, D_1, D_2, D_3… D_{kn} each of approximately equal size [23]. Training and testing is performed k times. In iteration i, partition D_i is reserved as the test set and remaining partitions are collectively used to train the model. The variance of the resulting accuracy estimate is diminished as fold number is increased [14].

Unlike the holdout and random sub-sampling methods, each sample is used for the same number of times for training and once for testing. Therefore, the representation problem in holdout method is demoted by this characteristic of cross-validation [23].

In general, stratified 10-fold cross-validation is recommended as verification method, since ten is about the appropriate number of folds to get the best estimate [24]. In FTCBPD methodology fold number is determined by the end-user in “Verification Parameters” view.

For a given set of event logs, there are an infinite number of possible process models that can be constructed. Goal of FTCBPD methodology is to construct a “good” process model. But how do we measure this goodness? To answer this question, the meaning of “good” term must be first understood:

Theoreticians consider a process model to be good if it is both *accurate* and *minimal*. By accurate it is meant that process model represents all legal sentences in the language and rejects all illegal sentences [3]. For our purposes, accuracy implies that we are modeling all

and only the behaviors of the business process. By minimal, theoreticians emphasize that the process model exhibits the fewest number of activities necessary [3].

Actually accuracy and minimality are in a sense, in conflict. The simplest way to achieve accuracy is to discover an independent path through the process model for each sample dataset, which typically refers to a high degree of redundancy. On the other hand, the simplest way to achieve minimality is to create a single activity (transaction) with self-transitions for each possible input [3].

Like [3], notions such as *fitness* and *appropriateness* have been quantified in [7]. An event log and process model "fit" if the process model can generate each trace in the event log. In other words, the process model should be able to discover every transition observed in the event log [7]. Unfortunately a high degree of fitness only does not imply that the model is indeed suitable, since spaghetti-like process model do not provide meaningful information. Therefore a second dimension is introduced: appropriateness. Appropriateness attempts to answer the question "Does the model reflect the observed process in a suitable way?" and evaluate the process model from a structural and behavior perspective. In [7] it is stated that a "good" process model should somehow be minimal in structure to clearly exhibit the described behavior, referred to as structural appropriateness, and minimal in behavior in order to represent as closely as possible what actually occurs, which is called behavioral appropriateness.

In the domain of FTCBPD methodology, this definition of "goodness" is particularly applicable. Thus a "goodness" definition with respect to FTCBPD methodology is one that:

a. Fully handles the sample behaviors it is given, subject to constraints on noise in the dataset.
b. Successfully describes patterns made up of direct successive, successive and back-tracking type relations.
c. Does not needlessly confuse the patterns identified with extra activities (transactions) and transitions.

FTCBPD methodology to compensate the argument about measurement of the process model goodness provides a conformance checker that supports three tuning verification metrics stated below:

a. Completeness

When it comes to process mining, the notion of completeness is very important. Like in any data mining or machine learning context one cannot assume to have monitored all possibilities in the "training material" (i.e. event log at hand) [7].

A mining algorithm could be very precise in the sense that it assumes that only the sequences in the log are precise. This implies that the algorithm actually does not provide more insights that what is already in the event log. It seems better to use Occam's Razor stated in [7], i.e., "one should not increase, beyond what is necessary, the number to entities required to explain anything", to seek the simplest model that can describe what is in the event log. Therefore FTCBPD methodology assumes that the event log is "locally complete".

Different algorithms assume different notions of completeness. These notions illustrate the different attempts to strike a balance between overfitting and underfitting. A model is overfitting if it does not generalize and only allows for the precise behavior recorded in the event log. This overfitting term implies that the corresponding mining technique assumes a very strong notion of completeness: "If it is not in the event log, it is not possible." An underfitting model over-generalizes the patterns seen in the event log, i.e., it allows for more behavior even when there are no indications in the event log that highlights this additional behavior [7].

In FTCBPD methodology, completeness metric refers to the percentage of transitions in discovered process model that are corresponding with some transition in the event log as stated in [5]:

$$completeness(PM, L_P) = \frac{\left|\left\{s \middle| s \in L_p \wedge s \middle| = PM\right\}\right|}{\left|\left\{s \middle| s \in L_P\right\}\right|} \quad (eq.\ 4.11)$$

Parameters stated in the metric are:

L_P is a test dataset of event log for business process P.

s is a transition in L_P.

PM is a disjunctive process model discovered by FTCBPD methodology.

b. Soundness

In the case of non-uniform distribution of transitions among training and testing datasets, an underfitting process model may be constructed, which allows for more behavior, having no

compliant representation in the training dataset. Thus a verification metric, named soundness is dedicated to interpret this type of tendencies.

Soundness is the percentage of transitions having no corresponding complement in the event log as stated in [5]:

$$soundnesss(PM, L_P) = \frac{\left|\left\{s \middle| s = PM \wedge \exists \neg s \in LP \; s.t. s = PM\right\}\right|}{\left|\left\{s \middle| s \in PM\right\}\right|} \quad (eq.\ 4.12)$$

Given two real numbers μ and σ between 0 and 1 (typically σ is small where μ is close to 1) it is said that PM is;

- σ-sound with respect to L_P, if $soundnesss(PM, L_P) \leq \sigma$ i.e. the smaller the sounder
- μ-complete with respect to L_P, if $completenesss(PM, L_P) \geq \mu$ i.e. the larger the more complete.

It is aimed to discover an alternative process model PM for a given business process P which is σ-sound and μ-complete, for some given σ and μ. Actually it is easy to see that a trivial process model satisfying the above conditions always exists, consisting in the union of exactly one process model modeling each of the instances in LP. However such model would not be a syntactic view of business process P, because of its size being $|PM| = |LP|$, where $|LP| = \{s \mid s \in L\}$. We therefore introduce a third metric, named average arc traffic, which acts as an upper bound on the size of PM.

c. Average Arc Traffic

Process model with a large soundness metric value implies that discovered process model allows for more behavior even when there are not any indications in training dataset. Although transitions and activities (transactions) arise from the commonality of activities and relations highlighted in the training dataset, there may exist no compliance in transaction stream assigned to testing dataset because of the non-uniform representation problem. Hence discovered process model attains the status of underfitting “spaghetti-like” representation.

In minimality perspective, average arc traffic interprets discovered process model in term of the number of transitions per activity (transaction) in discovered process model:

$$average\ arc\ traffic(PM) = \frac{\left|\{s | s \in PM\}\right|}{\left|\{a | a \in PM\}\right|} \quad (eq.\ 4.13)$$

Parameters stated in the metric are:

s is a transition in PM.

a is an activity (transaction) in PM.

PM is a disjunctive process model discovered by FTCBPD methodology.

Verification of discovered process model is performed fundamentally with respect to completeness metric instead of a multi-objective fashion that takes into account completeness, soundness and average arc traffic metrics simultaneously. Discovered process models constructed at each iteration of process modeling are challenged with respect to the completeness value. At the end of process modeling session, process model with the maximum completeness value is subjected as the optimum process model. Soundness, average arc traffic and average arc length characteristics of discovered process model are also reported in verification operation.

Furthermore “Predetermine arc traffic in process model” feature in “Threshold Values” view enables the end-user to designate an upper bound for the minimality perspective prior to executing process modeling. FTCBPD methodology takes this structural requirement (constraint) into consideration in evaluation of tally marks with respect to underlying evaluation metric(s). Tally marks, which are not worth to be tracked in transition top list (i.e. transition top list is an array, whose size is determined by average arc traffic and activity type number in FROMTOCHART table), are discarded. Hence transition traffic in discovered process model is held beneath a predetermined level.

Pseudo-code of “Verify Discovered Process Model” operation is as follows:

```
/* Pseudo-code of 'Verify discovered process model' Operation
INIT concentrated caseID to 0
INIT process instance index to 0
INIT predecessor TO null
INIT successor TO null
INIT transaction stream TO null
INIT concentrated activity type TO null
INIT row total to 0
INIT column total to 0
INIT handled transition number TO 0
INIT missed transition number TO 0
INIT completeness TO 0.000
INIT soundness TO 0.000
INIT average arc traffic TO 0.00
SET activity type set TO (READ activity type values in TRANSACTIONLOG table)
SET caseID set TO (READ caseID values in TRANSACTIONLOG table)
WHILE caseID set has more elements
        GET concentrated caseID FROM caseID set
        SET transaction stream TO (READ activity type of concentrated caseID
         ORDERED BY transaction log date AND transaction log date in TRANSACTIONLOG table)
        INCREMENT process instance index BY 1
        IF process instance index % fold number == 0 THEN
                FOR i=1 to (transaction stream length-1)
                        SET predecessor TO activity type at ith line of transaction stream
                        SET successor TO activity type at (i+1)th line of transaction stream
                        FOR j=1 to optimum activity sequence length
                                IF activity type value at element j at optimum activity sequence
                                 EQUAL predecessor THEN
                                        IF successor EXISTS
                                         at successor activity list of element j THEN
                                                INCREMENT handled transition number BY 1
                                                SET transition
                                                 at successor activity list of element j TO observed
                                        ELSE
                                                INCREMENT missed transition number BY 1
                                        ENDIF
                                ENDIF
                                INCREMENT j BY 1
                        ENDFOR
                        INCREMENT i BY 1
                ENDFOR
        ENDIF
ENDWHILE
CALCULATE completeness of discovered process model
CALCULATE soundness of discovered process model
CALCULATE average arc traffic of discovered process model
```

Activity diagram of “Verify Discovered Process Model” operation is stated at Figure 4.20:

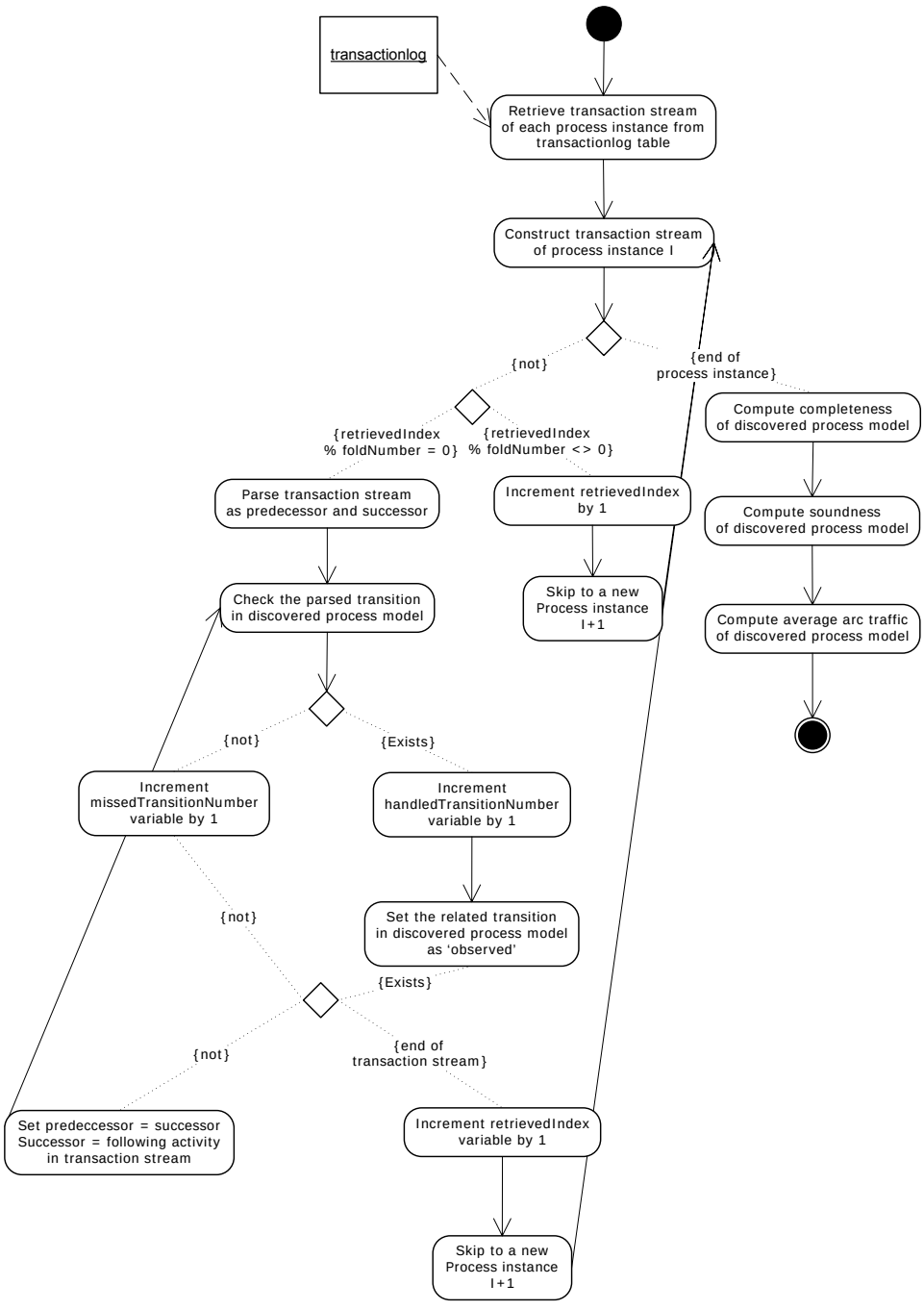


Figure 4.20 Activity Diagram of “Verify Discovered Process Model” Operation

4.2.1.9. Report Process Instances Operation

At the end of process modeling, ProMiner constructs transaction streams (i.e. a form that is independent of originator and timestamp parameters) of all process instances in training and

testing datasets and writes constructed transaction streams on a text file under the program folder.

Pseudo-code of “Report Process Instances” operation is as follows:

```
/* Pseudo-code of 'Report process instances' Operation
INIT concentrated caseID to 0
CREATE output text file
SET caseID set TO (READ caseID values in TRANSACTIONLOG table)
WHILE caseID set has more elements
       GET concentrated caseID FROM caseID set
       SET transaction stream TO (READ activity type of concentrated caseID
        ORDERED BY transaction log date AND transaction log date in TRANSACTIONLOG
       table)
       PRINT concentrated caseID TO output text file
       FOR i=1 to transaction stream length
              PRINT activity type at ith position of transaction stream
              INCREMENT i BY 1
       ENDFOR
ENDWHILE
```

Activity diagram of “Report Process Instances” operation is stated at Figure 4.21:

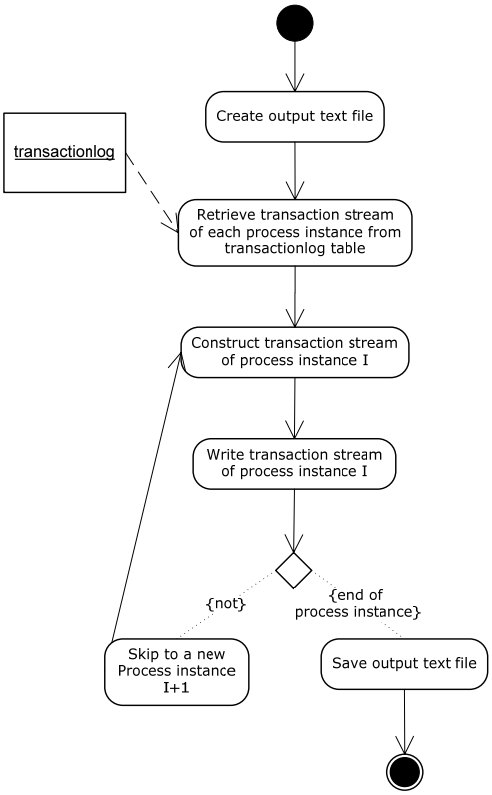


Figure 4.21 Activity Diagram of “Report Process Instances” Operation

4.2.2. Technical Perspective of ProMiner Implementation

FTCBPD methodology is basically a database application, which refers to a particular database (i.e. thesis database) and the associated program that implements the database queries and updates.

Application program, which is developed in Java, consists of 42 classes. These classes serve different functionalities as follows:

- Interface involves the functionalities provided to an interfacing entity (e.g. displaying discovered process model in the form of dependency/frequency graph).
- Algorithmic/Data manipulation process involves the functionalities provided to transform data item to create another one by means of mathematical and/or logical operations (e.g. from-to chart implementation).
- Permanent data access/storage involves the functionalities provided to an interfacing entity to access (read/write) permanent group or collection of related and self-contained data in the world.

According to functionality types stated above, size distribution of the classes with respect to functionality types are summarized in terms of line of code (LOC) in Table 4.1.

Table 4.1 Size Distribution of Classes in Application Program

Class Name	Interface Size	Algorithmic/Data Manipulation Process Size	Permanent Data Access/Storage Size	Total Size (LOC)
BlockCoordinates			38	38
BlockDiagram	282			282
BlockDiagramwithConnections	288			288
Combinatoric		96		96
CombinatoricException		8		8
ConnectionFrame	28			28
ConnectionLinkedList			81	81
ConnectionNode			84	84
ConnectionTable			121	121
DefineBusinessProcessFrame	22			22
DefineTcodeFrame	86			86
DependencyFrequencyGraph	32			32
FromtoChartImplementation		694	347	1041
FromtoChartImplementationThread		68		68
FromtoChartRearrangementFactors	299			299
ImportTableOperation			814	814
InitialFromtoChartFrame	26			26
InitialFromtoChartTable			301	301
LinkedList			6	6
MainGUI	106			106
MenuBar	416			416
MyFileFilter	235			235
OperationThread		106		106
Permutations		173		173
PredecessiveTcodeLinkedList			116	116
PredecessiveTcodeNode			64	64
ProcessModel	33			33
ProcessModelFrame	26			26
ProcessModelTable			99	99

Class Name	Interface Size	Algorithmic/Data Manipulation Process Size	Permanent Data Access/Storage Size	Total Size (LOC)
ProcessPerspective	231			231
Prototype		17		17
SourceFile	31			31
StatusBar	55			55
SuccessiveTcodeLinkedList			116	116
SuccessiveTcodeNode			84	84
TcodeObject			39	39
ThresholdValues	112			112
ThresholdValuesforProcessPerspective	162			162
TransitionLinkedList			96	96
TransitionNode			64	64
VerificationFactors	123			123
WrapLogOperation		91	45	136

Thesis database component of FTCBPD methodology is managed by MySQL 4.1.8 relational database management system (RDBMS). Data dictionary of thesis database is given in APPENDIX A and entity relationship diagram is given in APPENDIX B.

CHAPTER 5

EXPERIMENTAL RESULTS

In this section, FTCBPD methodology is briefly evaluated with respect to three key performance metrics. Our evaluation is based on experiments that are conducted on distinct industrial applications varying on each of the below enumerated parameters.

5.1. Basic Concepts in Experimental Procedure

Applications in this study involve three distinct business processes. First one is the travel management business process, which aims to manage all travel activities including booking trips and handling of expenses associated with travel. This business process is built as a sub-component of the finance module (i.e. SAP FI/CO) on a widespread ERP-system[1] the software product SAP R/3[2] and it is on live at 64 provincial offices of a public institution till 2004. Reference model, which is designed for this business process, consists of seven activity types, which are “create travel request”, “change travel request”, “display travel request”, “advance payment”, “travel request confirmation”, “disbursement entry” and “deduction”. As stated in [21], ERP-systems like SAP and PeopleSoft are based on a large set of database tables, instead of designing around explicit processes. Therefore information about a process instance (case) is scattered around various tables and functions recording different aspects at different levels (i.e. while TRANSACTIONLOG table in ERP-system holds the execution timestamp and originator parameters of an underlying transaction without any case identifier, TABLELOG table holds the operational information (insert, delete and update) of a case with respect to timestamp and originator). Typically an application, named “Log wrapper”, is developed to link these information sources to concrete

[1] The abbrevation “ERP” means Enterprise Resource Planning.

[2] SAP and R/3 are trademarks of SAP AG, Systems. Applications and Products in Data Processing, Neurottstr. 16, 69190 Walldorf, Germany.

event logs prior to process modeling[3]. As a result it is often difficult, time-consuming and labor-intensive to collect the right event logs in travel management business process.

Second business process presents the activity pattern of repair, whose event logs are obtained from Process Mining Research Group at Eindhoven University of Technology [33]. This business process consists of nine activity types, which are “first contact”, “make ticket”, “arrange survey”, “inform client survey”, “survey”, “repair ready”, “ready inform client”, “send ticket to financial administration” and “ticket ready”.

The last business process is about credit card application, which is terminated by credit card delivery or rejection notification. Likewise in repair example, event logs that belong to credit card application business process are obtained from Process Mining Research Group at Eindhoven University of Technology [33]. This business process consists of eight activity types, which are “receive application”, “check for completeness”, “get more information”, “check loan amount”, “perform checks”, “make decision”, “notify result” and “deliver credit card”.

Parameters varying in the experiments are classified into two groups: *case-related* and *execution-related parameters*. Case related parameters are the characteristics of the concentrated event logs such as:

a. **The number of activity types in the business process.** Three different business processes with 7, 8 and 9 activity types are used in the experiments.

b. **The amount of information in the event logs.** Event logs with 100, 200, 300, 400, 500 and 600 process instances for each business process are obtained.

As well, execution-related parameters are the inputs set by the end-user at the selection screen of ProMiner. It is observed that at least three parameters may strongly influence the key performance metrics:

a. **Evaluation metric in “Threshold Values” view.** Ten evaluation metrics with MST=0.100 and MCT=0.100, MST=0.125 and MCT=0.100, MST=0.150 and MCT=0.100, MST=0.175 and MCT=0.100, MST=0.200 and MCT=0.100, MST=0.100 and MCT=0.275, MST=0.100 and MCT=0.450, MST=0.100 and MCT=0.625,

[3] “Log wrapper” application fills the “missing” case identifier of each TRANSACTIONLOG table record by retrieving the caseID attribute value of the record in TABLELOG table with a maximum timestampthat is smaller than timestamp value of underlying TRANSACTIONLOG table record.

MST=0.100 and MCT=0.800 and modified lift values are conducted in the experiments.

b. **Verification method in "Verification Parameters" view.** Six verification methods with cross-validation with 10 folds, cross-validation with 20 folds, cross-validation with 30 folds, cross-validation with 40 folds, cross-validation with 50 folds and holdout values are conducted in the experiments.

c. **Back-tracking penalty point in "Process Modeling Factors" view.** Five use cases with back-tracking penalty point=1, back-tracking penalty point=2, back-tracking penalty point=3, back-tracking penalty point=4 and back-tracking penalty point=5 are conducted in the experiments.

While monitoring the effects of the parameters on the key performance metrics, ceteris paribus[4] policy is qualified to individually construct the logical connection between key performance metrics and concentrated parameter by ruling out the effects of remaining parameters which may override this logical connection. Hence the effect of a single independent variable on the dependent variable can be isolated. According to "with other things the same" nature of ceteris paribus policy, default values are determined for the execution-related parameters as stated in Table 5.1.

Table 5.1 Default Values for Execution-related Parameters

Execution-related Parameter	Default Value
Evaluation Metric in "Threshold Values" view	MST=0.100 and MCT=0.100
Verification Method in "Verification Parameters" view	Cross-Validation with 10 folds
Back-tracking Penalty Point in "Process Modeling Factors" view	Back-tracking penalty point=2

Another significant notion in the experimental procedure is the *key performance metrics*, which are used to compare and evaluate the effects of case-related and execution-related parameters in FTCBPD methodology. There are three perspectives in determining key performance metrics such as:

[4] Ceteris paribus is a Latin phrase, literally translated as "with other things the same". It is commonly rendered in English as "all other things being equal."

a. Accuracy Perspective

The ultimate aim of FTCBPD methodology is to construct a process model that successfully identifies the patterns made up with sequencing, selection and iteration in the event-logs. In this aspect, accuracy performance metric aims to measure the percentage of transitions extracted from training dataset that are compliant with some transition in testing dataset. Hence accuracy is represented by completeness verification metric introduced in “Proposed Method” chapter.

b. Minimality Perspective

Key performance metrics in minimality perspective aim to monitor structural complexity of the discovered process model. Actually these metrics are meaningful in the case of “less effective” evaluation metrics (i.e. MST and MCT with lower values) such that, “less effective” evaluation metrics lower the practical barriers and this setting encourages the methodology to exhibit all behaviors represented in the training dataset. Hence FTCBPD methodology comes up with process model having a high level accuracy. On the other hand, the “spaghetti-like” process model with a high level in arc traffic is useless to visualize the backbone of the underlying business process. Key performance metrics in minimality perspective are represented by soundness and average arc traffic verification metrics introduced in “Proposed Method” chapter.

c. Non-functional Perspective

In fact, non-functional requirements (constraints) as processing time and size are not primarily taken into consideration in “goodness” definition of the discovered process model. Additionally classification of risky business process re-engineering (BPR), which is represented in Figure 5.1, as a long-term organization change strengthens this implication.

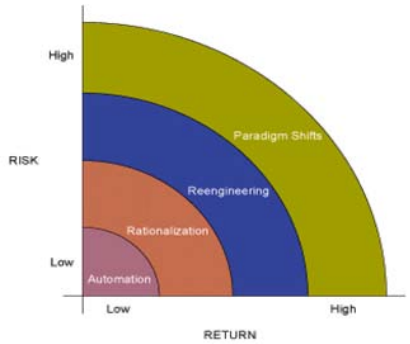


Figure 5.1 Forms of Organizational Change

On the other hand, process time and size are useful metrics to monitor the operational bottlenecks of FTCBPD methodology and future work of this study can be figured by these key performance metrics.

As a result, based on the 462 executions (i.e. 3 business processes × 6 amount of material × 22 execution-related parameters + 3 activity types × 22 execution-related parameters) performed on a machine with Intel(R) Pentium(R) M processor 1.60 GHz and 2.0 GHz RAM, it is possible to highlight the effects of case-related and execution-related parameters on the key performance metrics in part 5.2 and 5.3 In part 5.4, it is aimed to monitor the effectiveness of modified lift as an evaluation metric in FTCBPD methodology. In the last part, an inter-evaluation is performed to compare FTCBPD methodology with the approaches stated in prior studies.

5.2. Evaluation Based on Amount of Information in the Event Logs

In this experiment, key performance metrics (i.e. completeness, soundness, average arc traffic, processing time and size) varying with respect to the amount of information in the event logs and execution-related parameters (i.e. evaluation metric, verification methods and back-tracking penalty point) are measured for each business process. Tabulated form of obtained key performance metrics' measurements are represented in APPENDIX J.

5.2.1. The Effect of Number of Process Instances on the Learning Scheme

As the rule of thumb, successful data mining involves far more than selecting learning algorithm and running it over the available data. For one thing, many learning schemes have various parameters, which are handled as execution-based parameters in this study, and appropriate values must be chosen for these. In most cases, results can be improved by a suitable choice of parameters and this appropriate choice depends on the size of available data at hand.

In this aspect, it is desired as much of the data as possible for training to analyze user behaviors on a wider spectrum, monitor major interesting patterns in this data and get a good process model. On the other hand, there is a dilemma in this experiment such that; according to the runtime data of the concentrated business processes, completeness and

number of process instances are negatively-correlated according to correlation coefficient[5] stated below:

$$correlation\ coefficient = \frac{S_{PA}}{S_P \times S_A} \quad (eq.\ 5.1)$$

$$where\ S_{PA} = \frac{\sum_i \left(p_i - \bar{p}\right) \times \left(a_i - \bar{a}\right)}{n-1} \quad S_P = \frac{\sum_i \left(p_i - \bar{p}\right)^2}{n-1} \quad S_A = \frac{\sum_i \left(a_i - \bar{a}\right)^2}{n-1}$$

Calculated correlation coefficient of completeness and number of process instances with respect to execution-related parameters and business processes are tabulated in Tables 5.2 and 5.3.

Table 5.2 Correlation between Number of Process Instances and Completeness
With respect to Minimum Support Threshold (MST) value and Business Process

Business Process	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2
Travel Management	-0.904	-0.599	-0.582	-0.146	0.391
Credit Card Application	-0.937	-0.937	-0.937	-0.937	-0.937
Repair	-0.768	-0.768	-0.768	-0.768	-0.768

Table 5.3 Correlation between Number of Process Instances and Completeness
With respect to Minimum Confidence Threshold (MCT) value and Business Process

Business Process	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
Travel Management	-0.100	-0.764	-0.952	-0.278	-0.234
Credit Card Application	-0.937	-0.937	-0.946	-0.946	-0.937
Repair	-0.768	0.341	-0.655	0.000	-0.768

Correlation between number of process instances and completeness is visualized in Figure 5.2, 5.3 and 5.4.

[5] Correlation coefficient measures the statistical correlation between two entities. Calculated correlation coefficient ranges from 1 for perfectly correlated results, through 0 when there is no correlation at all, to -1 when the results are perfectly correlated negatively.

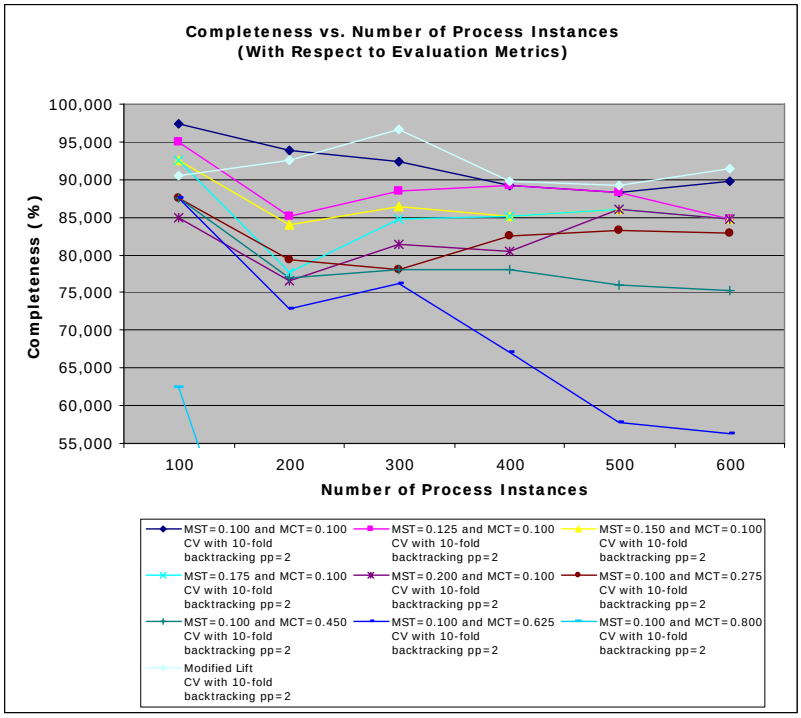


Figure 5.2 Correlation between Number of Process Instances and Completeness With respect to Evaluation Metrics for Travel Management Business Process (Details of this figure are given in Table J3.1 of Appendix J3)

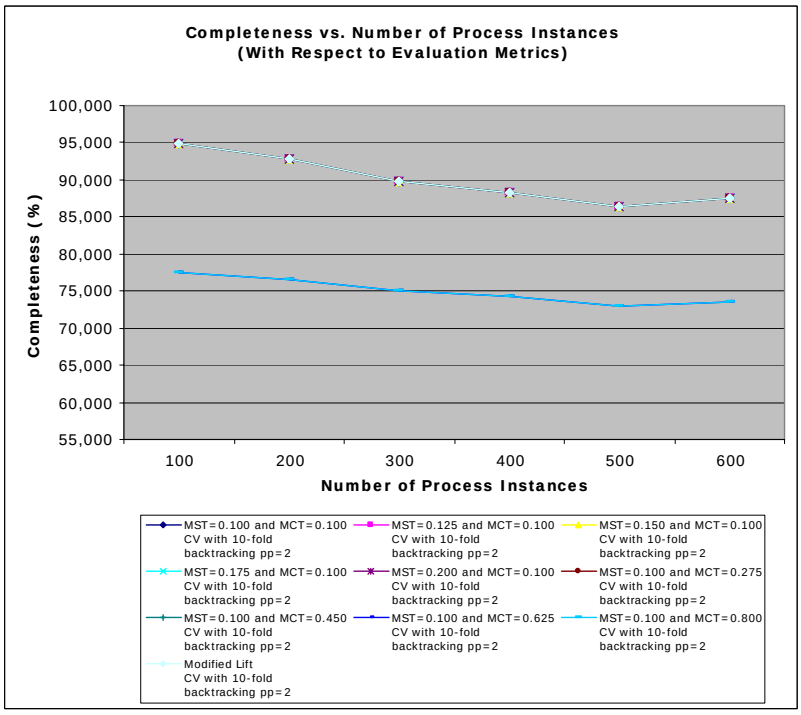


Figure 5.3 Correlation between Number of Process Instances and Completeness With respect to Evaluation Metrics for Credit Card Application Business Process (Details of this figure are given in Table J1.1 of Appendix J1)

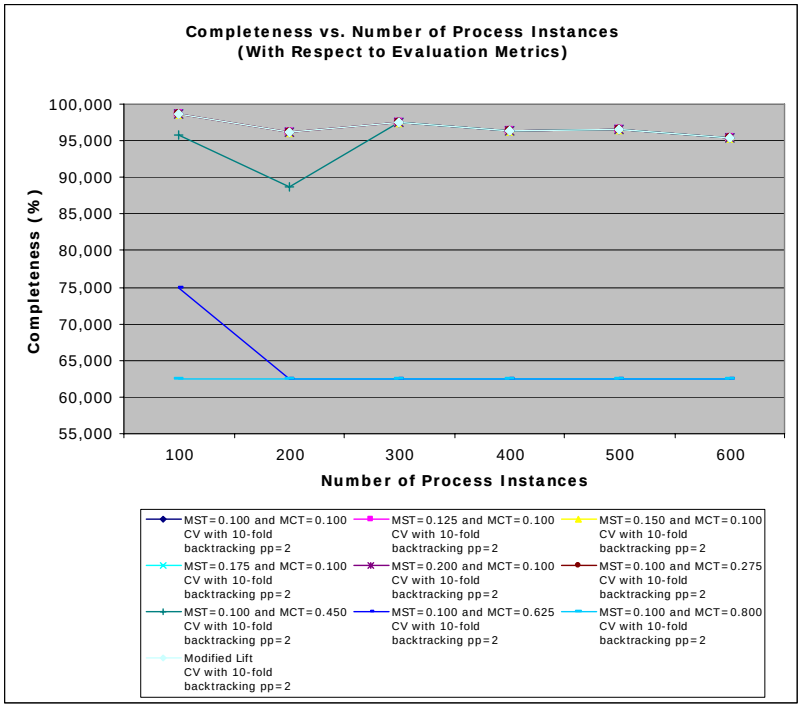


Figure 5.4 Correlation between Number of Process Instances and Completeness With respect to Evaluation Metrics for Repair Business Process (Details of this figure are given in Table J2.1 of Appendix J2)

The underlying catalyst of the negative correlation between number of process instances and completeness is the addition of “fresh” data in each trial, which exhibits new patterns and behaviors.

Increase in the number of process instances directly implies a higher minimum support threshold (MST), since this case-related parameter is the denominator of minimum support threshold (MST) formula. Consequently “periodic” patterns (i.e. transitions appeared nearly in every process instance) are not affected in the training dataset by this higher MST value. On the other hand, “niche” patterns (i.e. transitions not appeared nearly in every process instance) are mostly pruned in evaluation operation.

While the variety of the transitions in the training dataset increases with the effect of the “fresh” data added at each trial, relatively “niche” and “weak” transitions (i.e. non-periodic transitions with smaller support and confidence values) are challenged by the higher MST value. Hence discovered process models are constructed upon the “standard” and “strong” transitions (i.e. transitions with larger support and confidence values), which are uniformly presented in the training and testing datasets.

Pruning of relatively “niche” transitions in the training dataset means decrease in total number of transitions in the discovered process model and decrease in the average arc traffic key performance metric. This result can also be monitored by the negative correlation between number of process instances and average arc traffic especially in travel management business process, which constitutes of transitions with smaller support and confidence values in Tables 5.4 and 5.5.

Table 5.4 Correlation between Number of Process Instances and Average Arc Traffic
With respect to Minimum Support Threshold (MST) value and Business Process

Business Process	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2
Travel Management	-0.660	-0.967	-0.917	-0.655	0.925
Credit Card Application	0.000	0.000	0.000	0.000	0.000
Repair	0.000	0.000	0.000	0.000	0.000

Table 5.5 Correlation between Number of Process Instances and Average Arc Traffic
With respect to Minimum Confidence Threshold (MCT) value and Business Process

Business Process	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
Travel Management	-0.263	-0.710	-0.875	-0.146	0.690
Credit Card Application	0.000	0.000	0.000	0.000	0.000
Repair	0.000	0.828	-0.655	0.000	0.000

Correlation between number of process instances and average arc traffic in travel management business process is visualized in Figure 5.5.

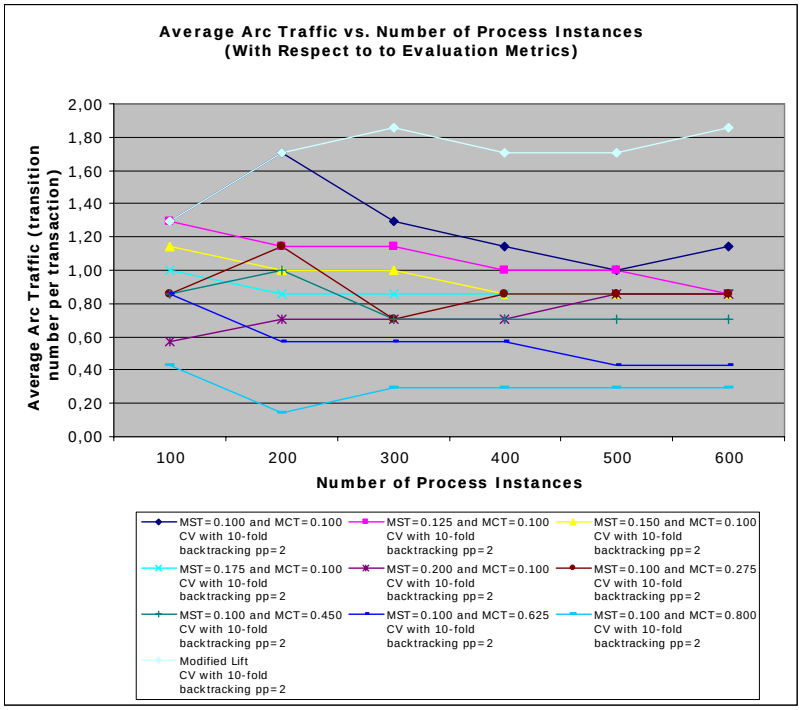


Figure 5.5 Correlation between Number of Process Instances and Average Arc Traffic
With respect to Evaluation Metrics for Travel Management Business Process
(Details of this figure are given in Table J3.3 of Appendix J3)

Decrease in the average arc traffic (total transition number of the discovered process model) may result in a less sound process model, since discovered process model with more transitions is potential to exhibit more behaviors even when there are no indications in the testing dataset. In this aspect, there is a negative correlation between number of process instances and soundness especially in travel management business process in Tables 5.6 and 5.7.

Table 5.6 Correlation between Number of Process Instances and Soundness
With respect to Minimum Support Threshold (MST) value and Business Process

Business Process	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2
Travel Management	0.131	-0.131	-0.655	-0.809	0.000
Credit Card Application	0.000	0.000	0.000	0.000	0.000
Repair	0.000	0.000	0.000	0.000	0.000

Table 5.7 Correlation between Number of Process Instances and Soundness
With respect to Minimum Confidence Threshold (MCT) value and Business Process

Business Process	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
Travel Management	-0.655	-0.655	-0.655	0.000	-0.123
Credit Card Application	0.000	0.000	0.000	0.000	0.000
Repair	0.000	0.000	0.000	0.000	0.000

Correlation between number of process instances and soundness in travel management business process is visualized in Figure 5.6.

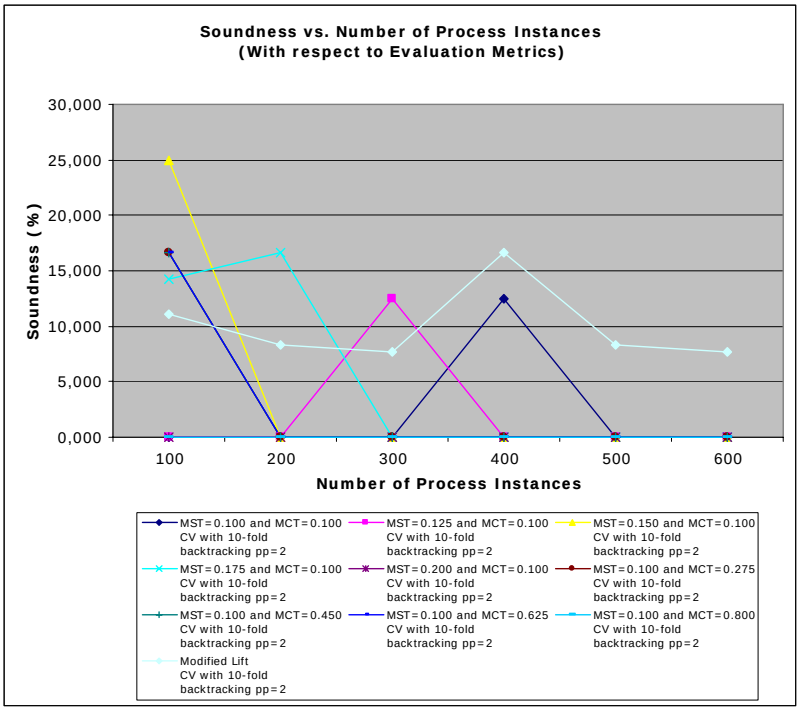


Figure 5.6 Correlation between Number of Process Instances and Soundness
With respect to Evaluation Metrics for Travel Management Business Process
(Details of this figure are given in Table J3.2 of Appendix J3)

While discovered process model turns into a simpler formation with respect to structural reduction in the training dataset, leverage effect of the “fresh” data results in encountering more “niche” patterns (transitions) at the process instances dedicated to the testing dataset with respect to closed system perspective. Hence testing dataset turns into a more overfitting formation. Actually the representation gap between training and “overfitted” testing datasets would be exacerbated by the fact that; the behavior would be necessarily be over-represented in the testing dataset since none of its instances made it into the training

dataset. As a result, increase in the amount of the information leads to decrease in completeness key performance metric.

To conclude, the amount of information available in the event logs is a case-related characteristic, which is independent of FTCBPD methodology. Thus it is not seemingly reasonable to draw conclusions specific to the capabilities of FTCBPD methodology. Although more process instances enable the underlying data mining algorithm to transform this data into more accurate knowledge, negative correlation between number of process instances and completeness notices a dilemma. The underlying reason of this dilemma is the representation discrepancies between "fresh" data added at each trial. This outcome is also motivated by improper random sampling of each pattern (transition) in both training and testing datasets.

The mechanism triggered by increase in number of process instances is modeled by a reinforcement cycle as in Figure 5.7:

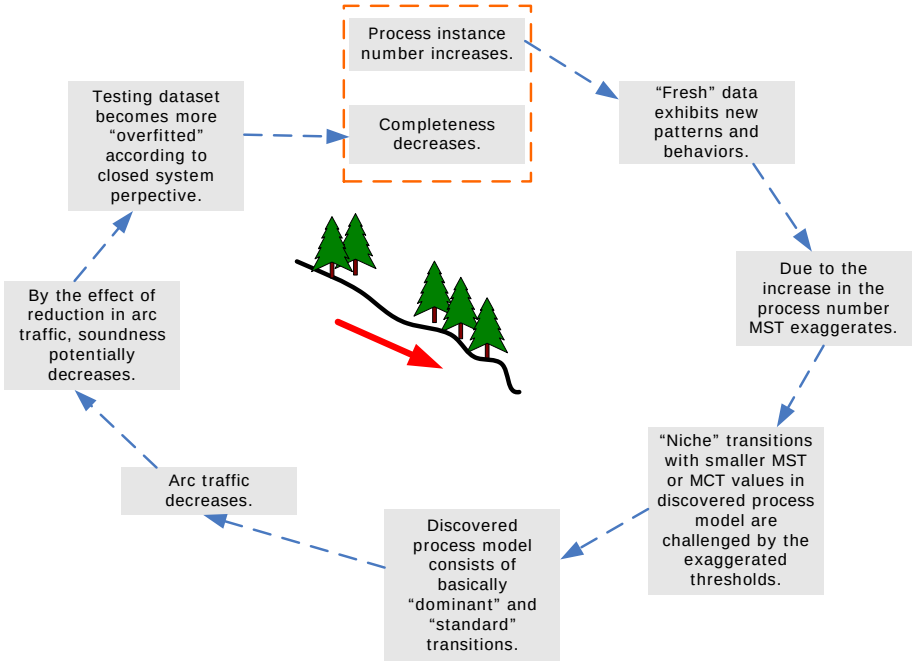


Figure 5.7 Reinforcement Cycle for the Effect of Number of Process Instances On the Learning Scheme

5.2.2. The Effect of Threshold Values on the Learning Scheme

Minimum support and minimum confidence thresholds (MST and MCT) are the probability threshold parameters to control the level of the robustness of discovered process model.

Additionally these parameters can be used to control the complexity of discovered process model from large amounts of data, ignoring low-probability transitions.

In “Evaluate Tally Marks in FROMTOCHART Table” operation, support and confidence values of the tally marks at each element in FROMTOCHART table are compared with set MST and MCT values and “significant” tally marks with respect to MST and MCT (i.e. tally marks that are not reset in the evaluation operation) are worth to be represented as a transition in discovered process model somehow[6].
According to the functionality of MST and MCT in FTCBPD methodology, increase in threshold values results in pruning the “weak” transitions with smaller support or confidence values in the business processes (e.g. travel management). On the other hand, business processes with “strong” transitions (e.g. credit card application and repair) are not affected by low increase in threshold values.

This causality is strengthened by the negative correlation between threshold values and average arc traffic key performance metric in Tables 5.8 and 5.9.

Table 5.8 Correlation between Minimum Support Threshold (MST) and Average Arc Traffic With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	-0.917	0.000	0.000
200	-0.937	0.000	0.000
300	-1.000	0.000	0.000
400	-0.972	0.000	0.000
500	-0.866	0.000	0.000
600	-0.707	0.000	0.000

Table 5.9 Correlation between Minimum Confidence Threshold (MCT) and Average Arc Traffic With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	-0.894	-0.866	-0.926
200	-0.988	-0.866	-0.945
300	-0.928	-0.866	-0.866
400	-0.991	-0.866	-0.866
500	-0.992	-0.866	-0.866
600	-0.994	-0.866	-0.866

[6] This “somehow” term denotes that, the existence of the transitions is clarified in evaluation operation, but the length of these transitions and relative positions of the activities (transactions) are not determined yet. This issue is handled at rearrangement operation and finalized by optimum activity sequence, which is the intermediate product of FTCBPD methodology.

According to experimental results, average arc traffic is more sensitive to the alterations in MCT, since range of MCT is wider than the MST's (i.e. MCT varies in the range of [0.100, 0.800], while MST range is [0.100, 0.200]). Correlation between threshold values and average arc traffic is visualized in Figures 5.8 - 5.13.

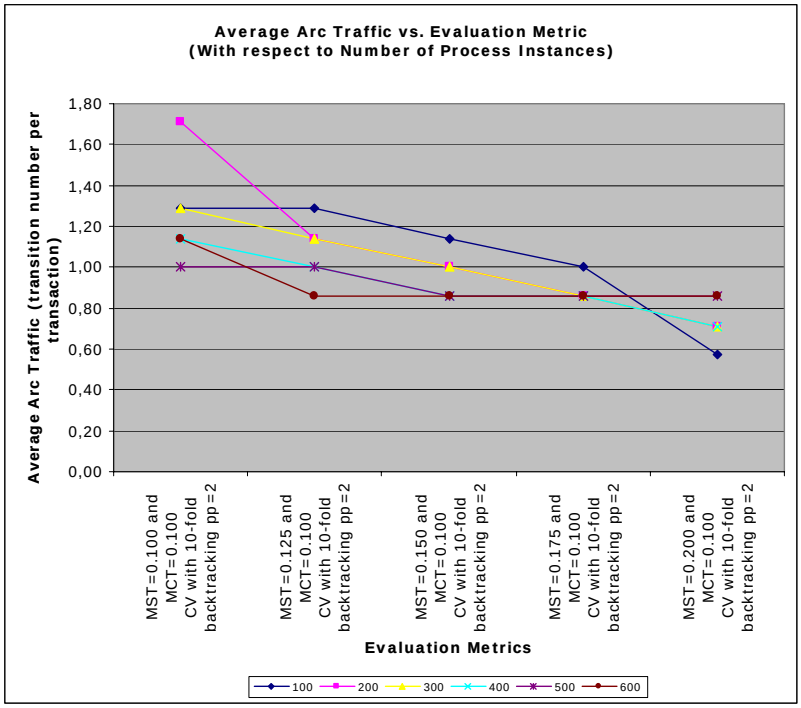


Figure 5.8 Correlation between Evaluation Metrics and Average Arc Traffic
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.3 of Appendix J3)

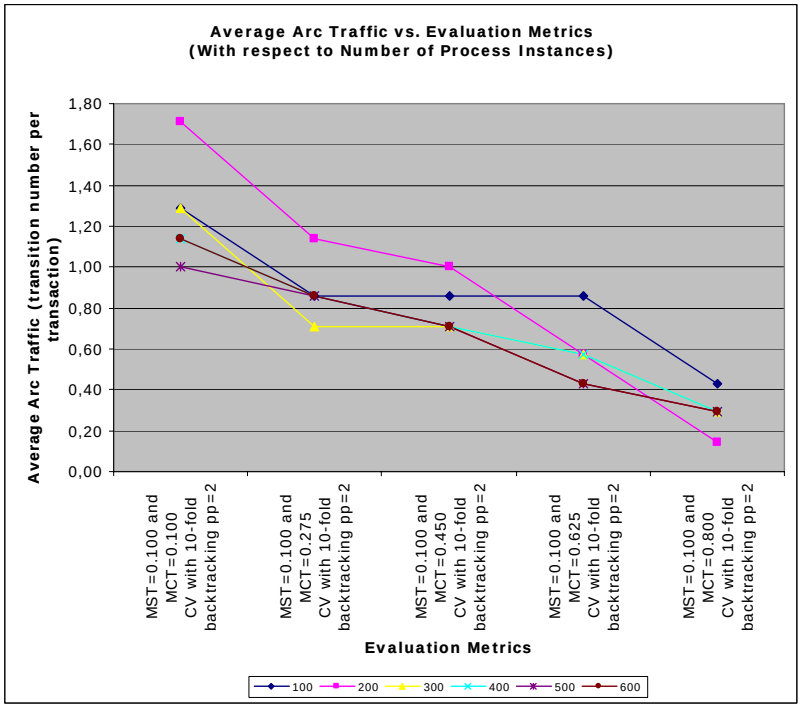


Figure 5.9 Correlation between Evaluation Metrics and Average Arc Traffic
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.3 of Appendix J3)

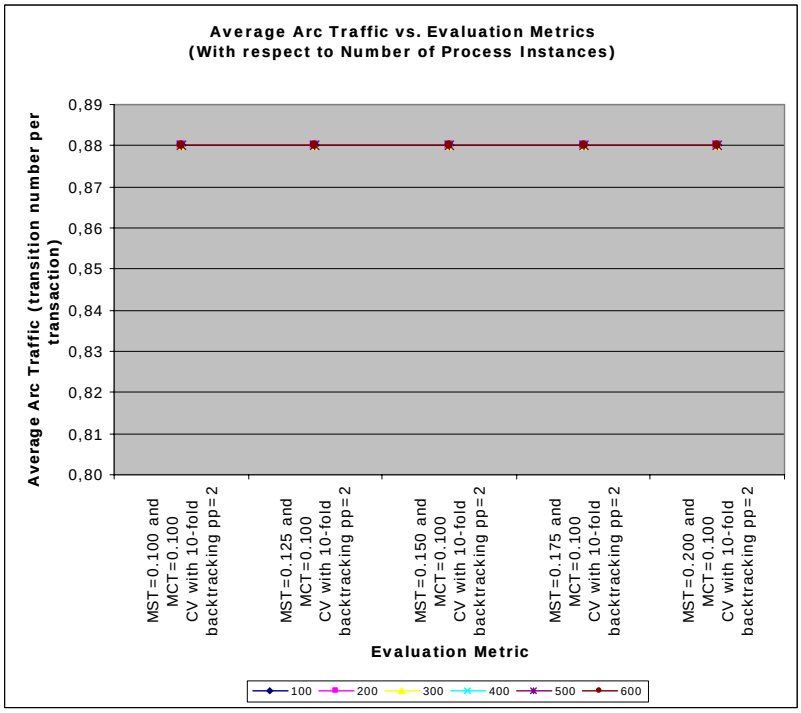


Figure 5.10 Correlation between Evaluation Metrics and Arc Traffic
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.3 of Appendix J1)

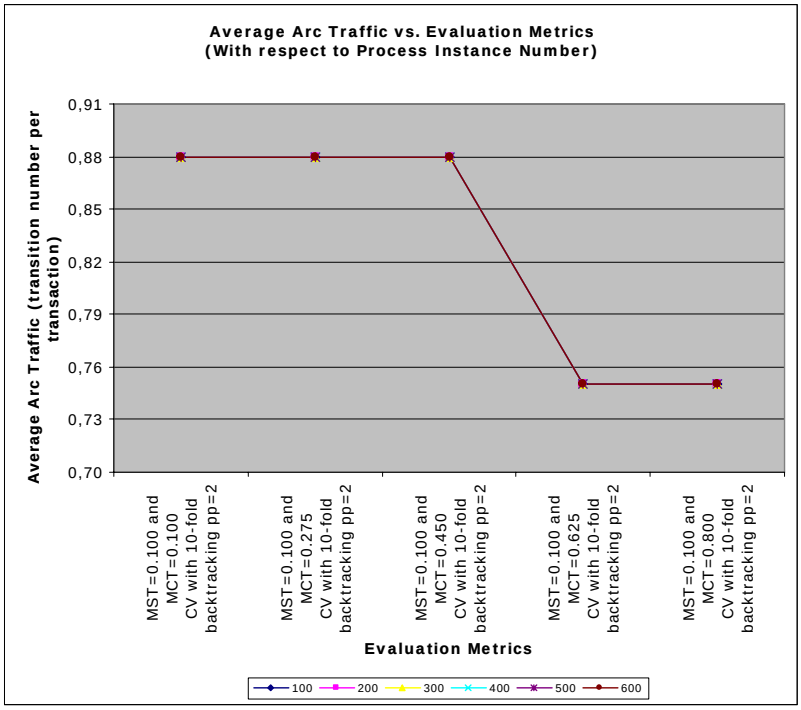


Figure 5.11 Correlation between Evaluation Metrics and Arc Traffic
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.3 of Appendix J1)

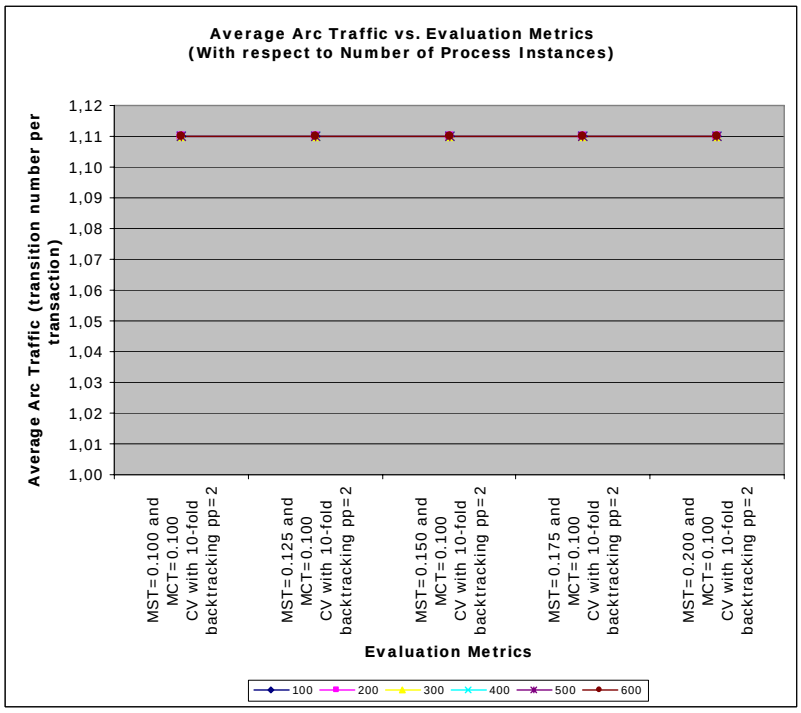


Figure 5.12 Correlation between Evaluation Metrics and Arc Traffic
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.3 of Appendix J2)

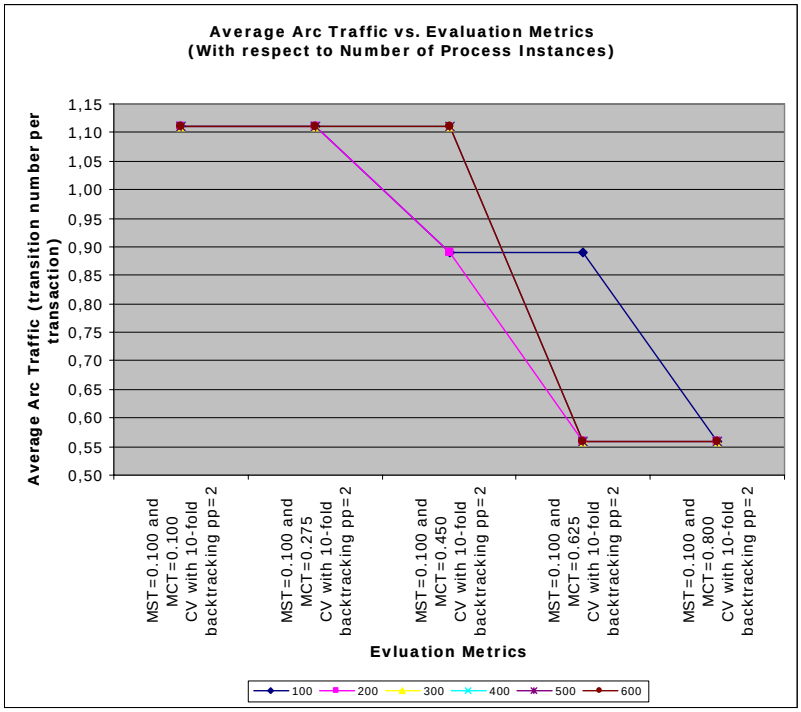


Figure 5.13 Correlation between Evaluation Metrics and Arc Traffic
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.3 of Appendix J2)

Pruning of relatively “weak” transitions in the discovered process model means the loss of extra-ordinary behaviors, which are not over-represented in the testing dataset. Thus this tendency in minimality perspective potentially results in numerically decrease in soundness. As a result, the major reflection of increase in threshold values for the same amount of information in event logs is the decrease in completeness, since crucial increase in threshold values may prune the “standard” and “strong” transitions, which are uniformly represented in the testing dataset. Consequently discovered process model cannot correspond to the most of the behaviors in testing dataset. Negative correlation between thresholds values and completeness arc traffic strengthens this implication in Tables 5.10 and 5.11.

Table 5.10 Correlation between Minimum Support Threshold (MST) and Completeness
With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	-0.928	0.000	0.000
200	-0.960	0.000	0.000
300	-0.989	0.000	0.000
400	-0.940	0.000	0.000
500	-0.866	0.000	0.000
600	-0.707	0.000	0.000

Table 5.11 Correlation between Minimum Confidence Threshold (MCT) and Completeness
With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	-0.848	-0.866	-0.919
200	-0.861	-0.866	-0.921
300	-0.849	-0.866	-0.866
400	-0.926	-0.866	-0.866
500	-0.963	-0.866	-0.866
600	-0.971	-0.866	-0.866

Likewise in average arc traffic, completeness is more sensitive to the alterations in MCT, since structural reduction with respect to pruning of transitions by the exaggerating MCT value results in significant representation gap between discovered process model and testing population. Correlation between threshold values and completeness is visualized in Figures 5.14 - 5.19.

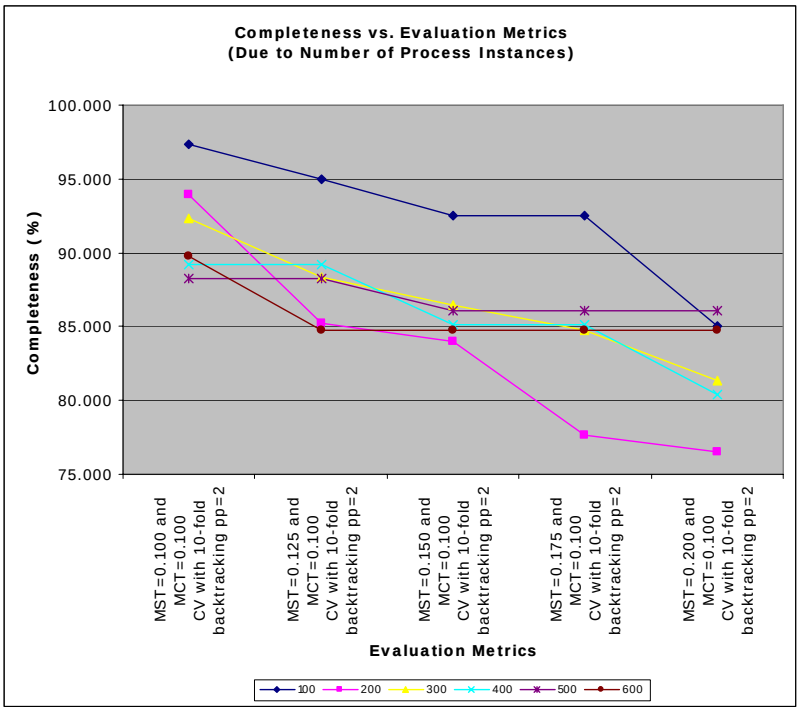


Figure 5.14 Correlation between Evaluation Metrics and Completeness
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.1 of Appendix J3)

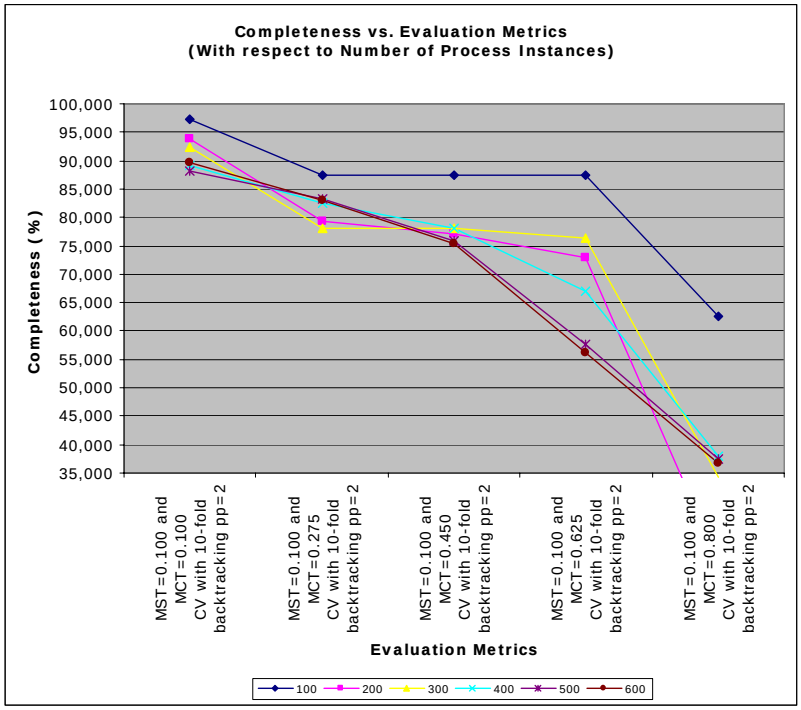


Figure 5.15 Correlation between Evaluation Metrics and Completeness
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.1 of Appendix J3)

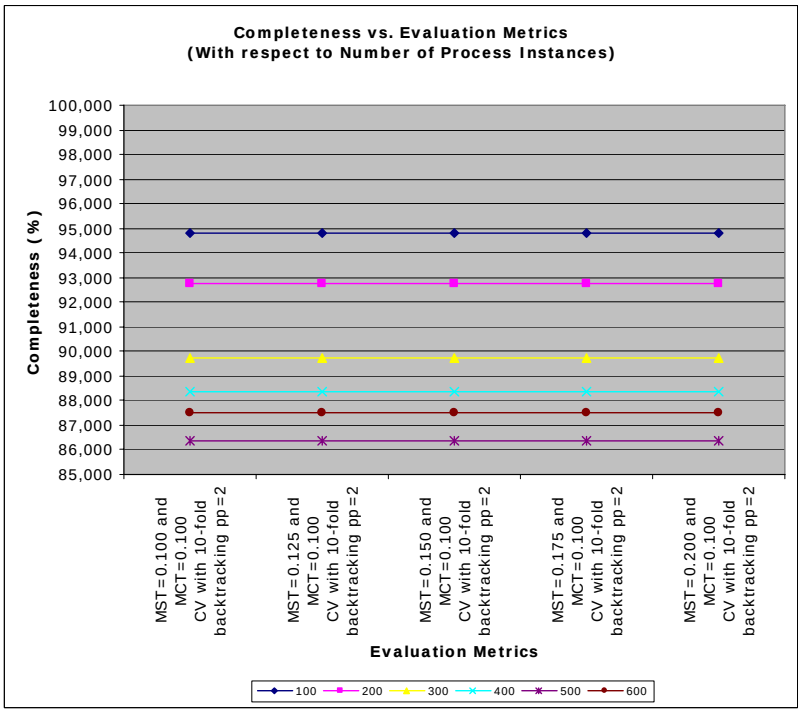


Figure 5.16 Correlation between Evaluation Metrics and Completeness
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.1 of Appendix J1)

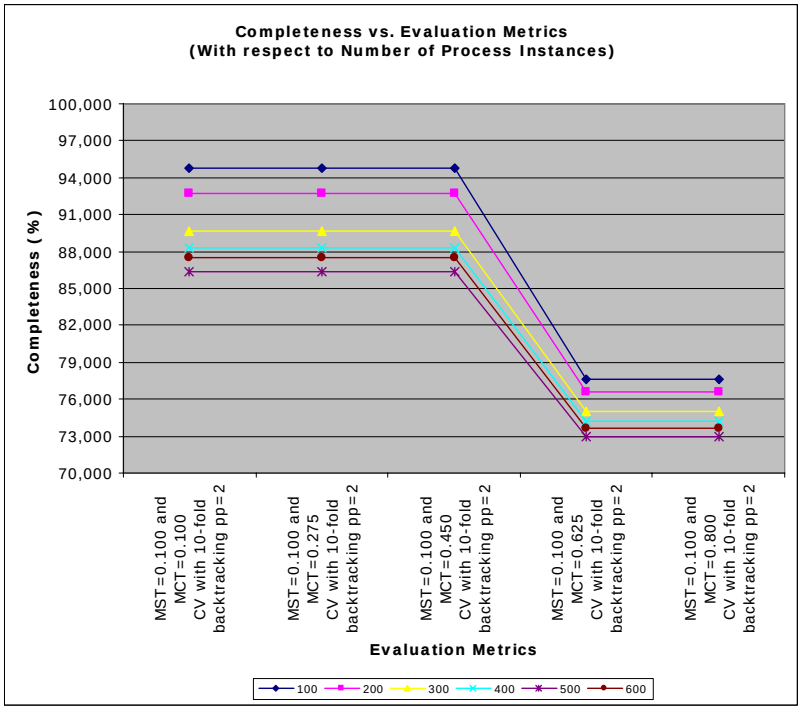


Figure 5.17 Correlation between Evaluation Metrics and Completeness
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.1 of Appendix J1)

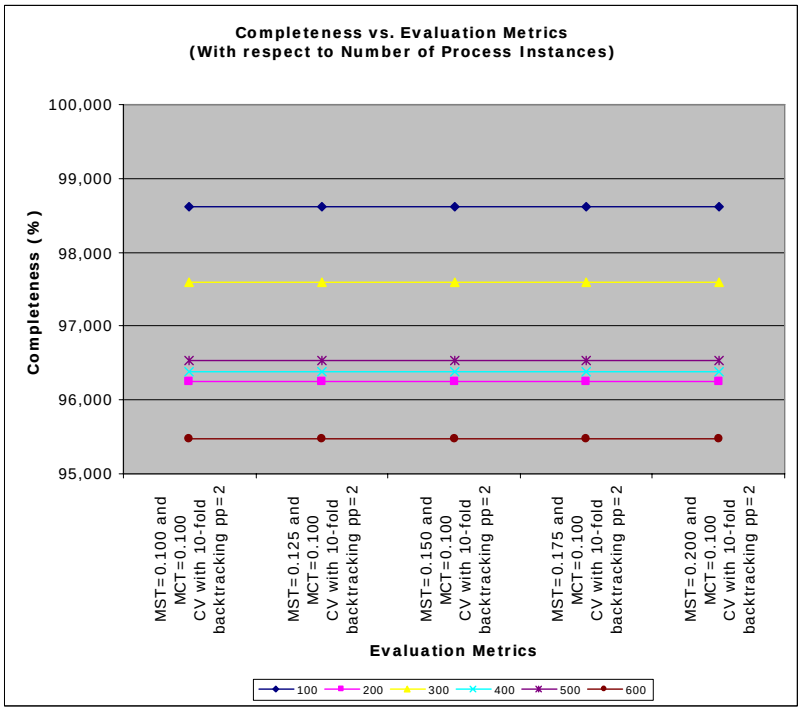


Figure 5.18 Correlation between Evaluation Metrics and Completeness
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.1 of Appendix J2)

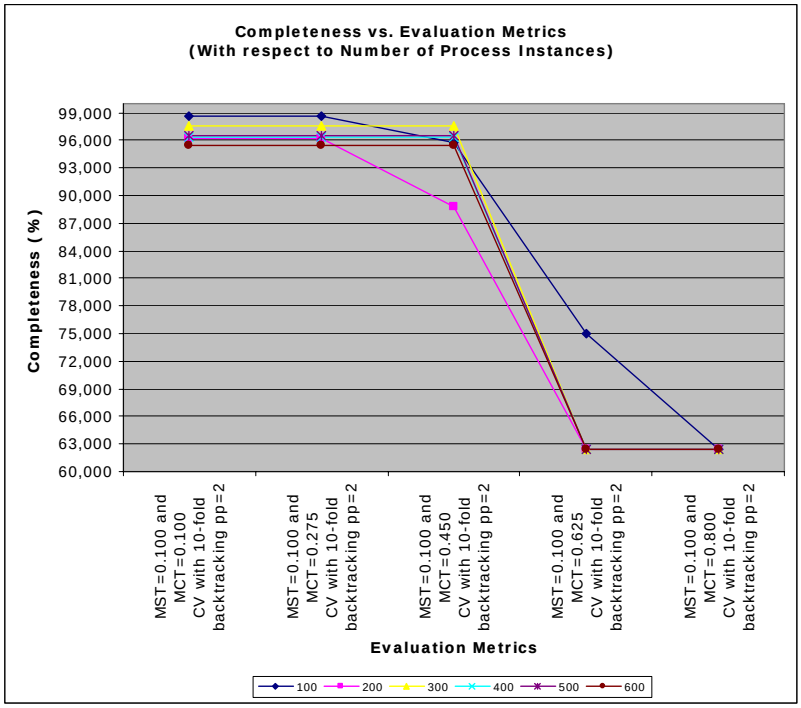


Figure 5.19 Correlation between Evaluation Metrics and Completeness With respect to Number of Process Instances for Repair Business Process (Details of this figure are given in Table J2.1 of Appendix J2)

To conclude, minimum support and minimum confidence thresholds (MST and MCT) are the sole stick yard in determining the representation of the underlying transition in discovered process model. Thus the level of robustness and complexity of discovered process model are controlled by these execution-related parameters of FTCBPD methodology. Moreover MST and MCT satisfy robustness for the amount of noise.

In general, slightly increasing MST and MCT values initially eliminate low-probability transitions. This situation is seemingly affirmative in minimality perspective. But after these threshold values reach to critical levels, “standard” and “strong” behaviors (transitions) are started to be challenged. As a result, all key performance metrics (completeness, soundness and average arc traffic) are negatively affected by the increase in threshold values after the critical level.

The mechanism triggered by increase in threshold values can be modeled by a reinforcement cycle as in Figure 5.20.

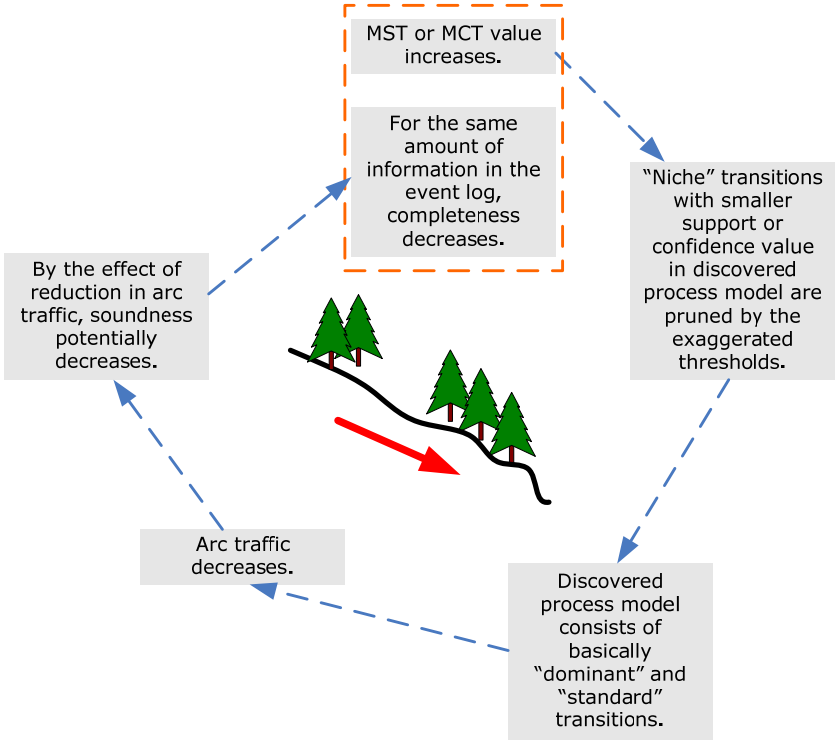


Figure 5.20 Reinforcement Cycle for the Effect of the Threshold Values On the Learning Scheme

5.2.3. The Effect of Fold Number on the Learning Scheme

According to the fold number tuned by the end-user in "Verification Parameters" view, the initial event log is randomly partitioned into mutually exclusive subsets or folds each of approximately equal size. Then each sample is used the same number of times for training (i.e. fold number – 1 times) and once for testing. In this aspect, cross-validation is an iterative method with an iteration number, which is equal to fold number set in the selection screen.

On the other hand, holdout verification method implemented in FTCBPD methodology is a version of threefold cross-validation, in which the data is divided randomly into three parts, two-thirds for training and one-third for testing, and repeat the procedure three times so that in the end, every instance has been used exactly once for testing.

For the same amount of information in the event log, the fold number increase implies decrease in the size of each fold (i.e. decrease in the number of process instances per fold).

Thus lesser process instances are dedicated for testing, while more process instances are dedicated for training. Actually the major effect of fold number on the learning scheme emphasizes this shrinkage in the size of testing dataset, in which some process instances with “niche” and “weak” transitions are shifted from testing to training dataset. Moreover FTCBPD methodology tends to fill this testing dataset with “standard” and ordinary behaving process instances to maximize the completeness by accomplishing uniform representation in training and testing datasets according to the iterative fashion of verification method.

Unfortunately for the same threshold values (MST = 0.100 and MCT = 0.100), transferred (shifted) process instances do not provide any significant information gain (i.e. new patterns and behaviors that are unknown in the prior trials with lesser fold number), since support or confidence values of the transitions in these transferred process instances are challenged by the default threshold values. Consequently transition traffic (i.e. average arc traffic) in discovered process model is not influenced by this data transfer according to shrinkage in testing dataset. This mechanism is strengthened by zero-valued correlation coefficient between fold number and average arc traffic in Table 5.12.

Table 5.12 Correlation between Fold Number and Average Arc Traffic
With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	0.000	0.000	0.000
200	0.000	0.000	0.000
300	0.000	0.000	0.000
400	0.000	0.000	0.000
500	0.000	0.000	0.000
600	0.000	0.000	0.000

Unchanging average arc traffic with respect to alteration in fold number is visualized in Figures 5.21, 5.22 and 5.23.

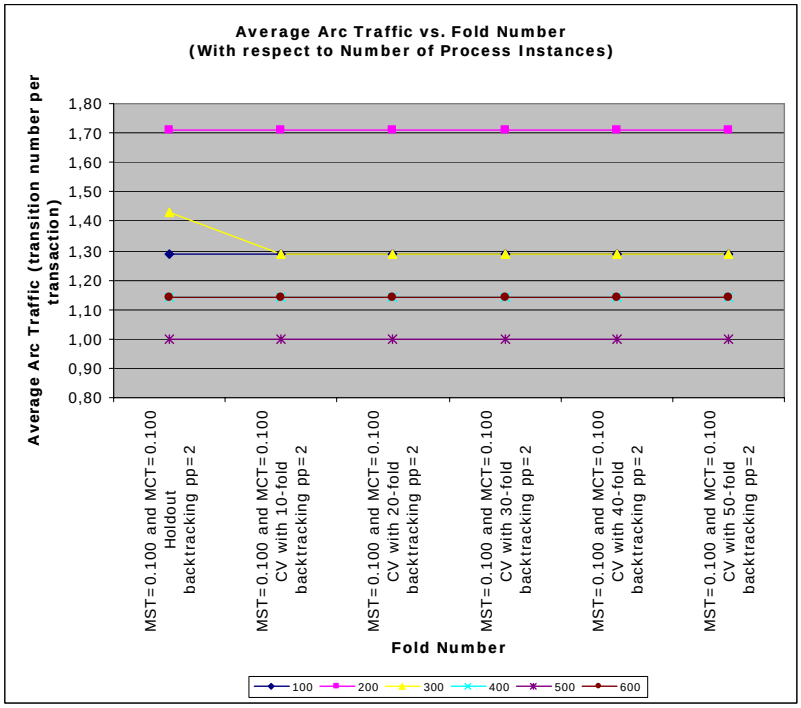


Figure 5.21 Correlation between Fold Number and Average Arc Traffic
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.3 of Appendix J3)

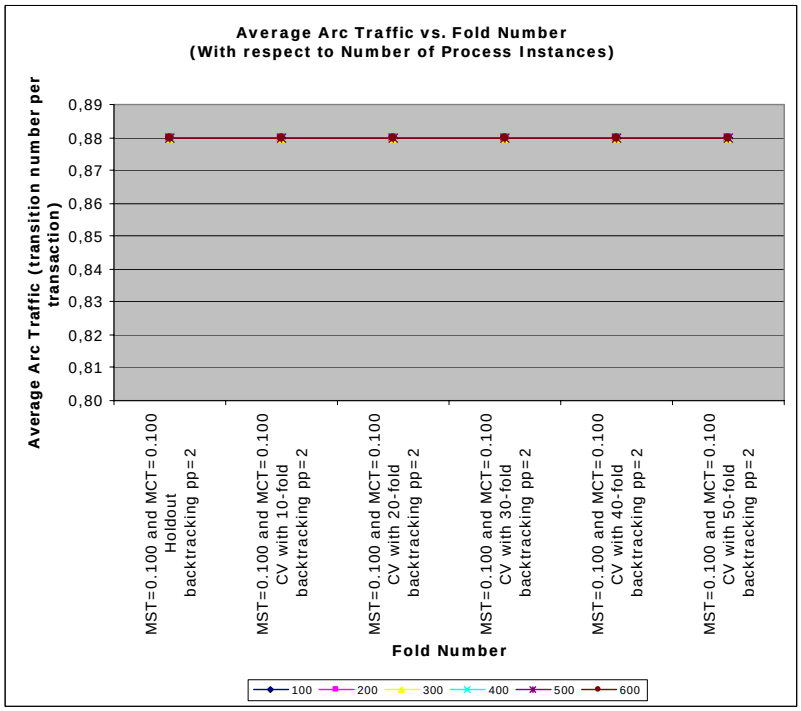


Figure 5.22 Correlation between Fold Number and Average Arc Traffic
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.3 of Appendix J1)

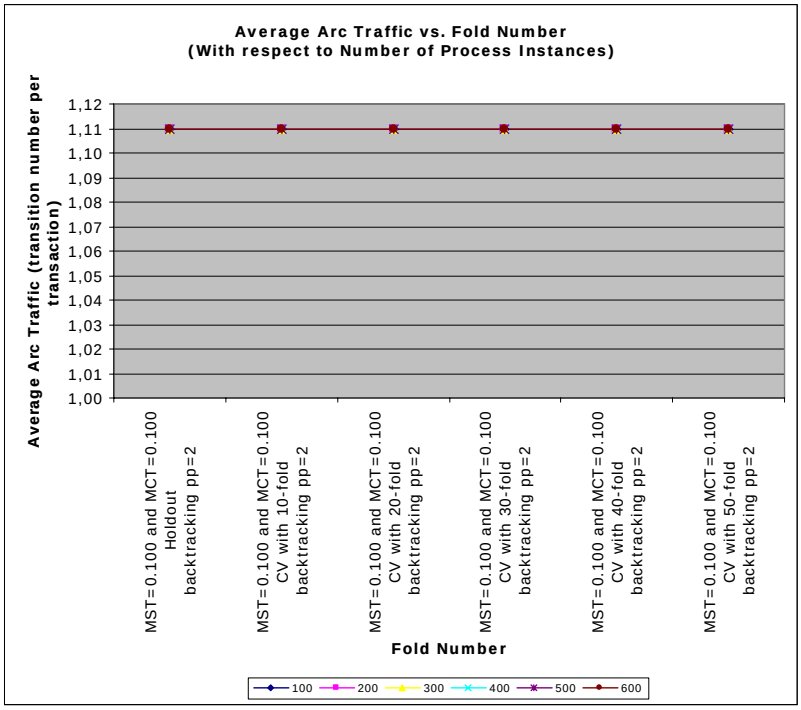


Figure 5.23 Correlation between Fold Number and Arc Traffic
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.3 of Appendix J2)

While transferring process instances from testing dataset to training does not enhance any new patterns and behaviors in the discovered process model, major effect of fold number alteration on the learning scheme can be monitored in soundness and completeness key performance metrics.

“Niche” transitions in the testing dataset makes testing dataset overfitting at the prior trials. Thus discovered process model cannot exhibit the behaviors, which occur in “overfitted” testing dataset. Fortunately by transferring these “niche” transitions to training dataset according to the shrinkage in the size of testing dataset, the representation gap between training and testing datasets is reversed. Consequently while process instances in testing dataset represent “standard” behaviors, training dataset becomes overrepresented by the leverage effect of the transferred process instances with “niche” transitions.

As a result, discovered process model becomes sounder and more complete with respect to the outcomes at the prior trials with lesser fold number. Positive correlation between fold number and soundness highlights this implication in Table 5.13.

Table 5.13 Correlation between Fold Number and Soundness
With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	0.847	0.886	0.821
200	0.765	0.886	0.454
300	0.934	0.669	0.336
400	0.711	0.000	0.000
500	0.626	0.000	0.000
600	0.929	0.000	0.000

Correlation between fold number and soundness is visualized in Figures 5.24, 5.25 and 5.26.

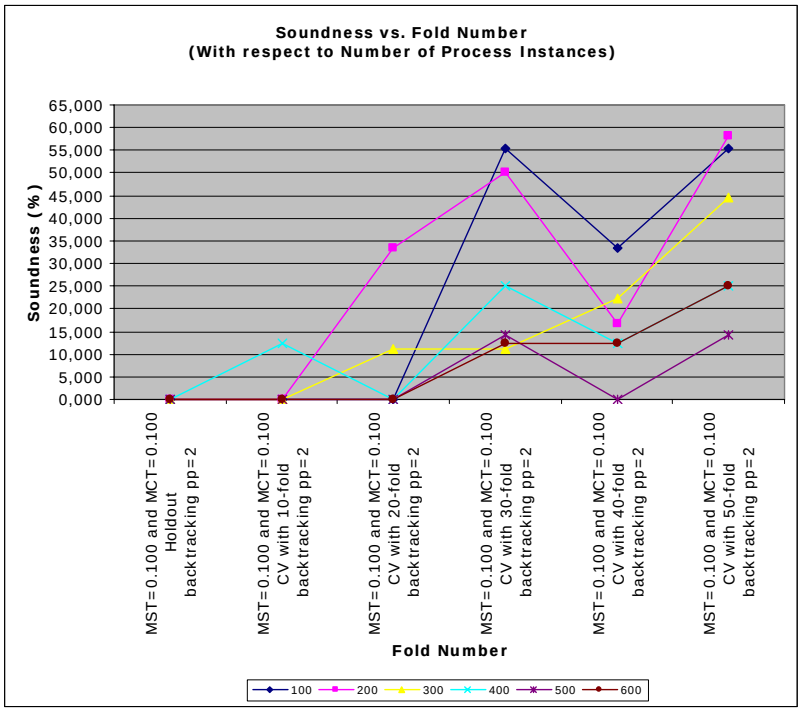


Figure 5.24 Correlation between Fold Number and Soundness
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.2 of Appendix J3)

Event logs with larger than 300 process instances in credit card application and repair business processes indicate an uncorrelated relationship between fold number and soundness. This situation may occur due to the proper random sampling in both training and testing datasets and standardization of behaviors (transitions) in “fresh” data.

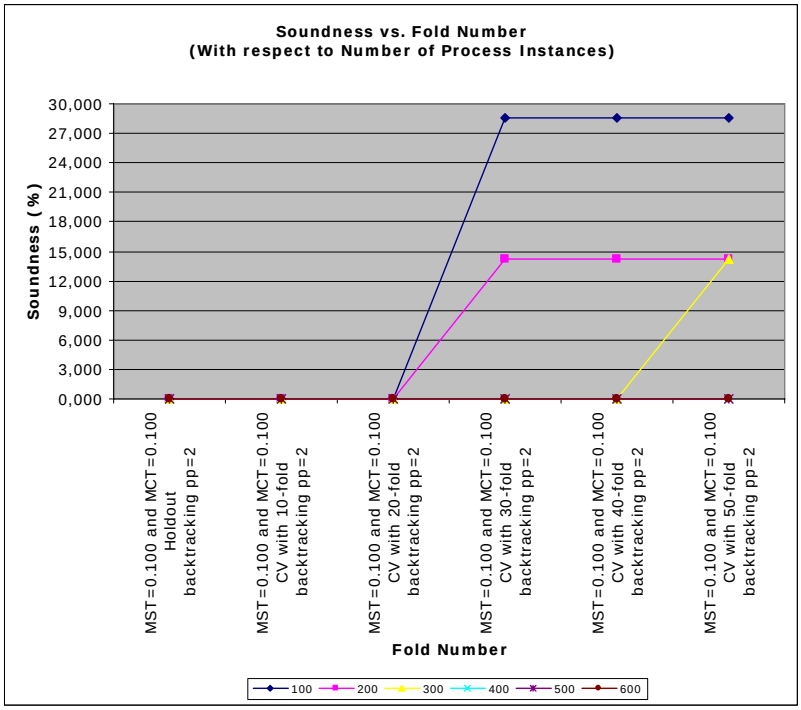


Figure 5.25 Correlation between Fold Number and Soundness
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.2 of Appendix J1)

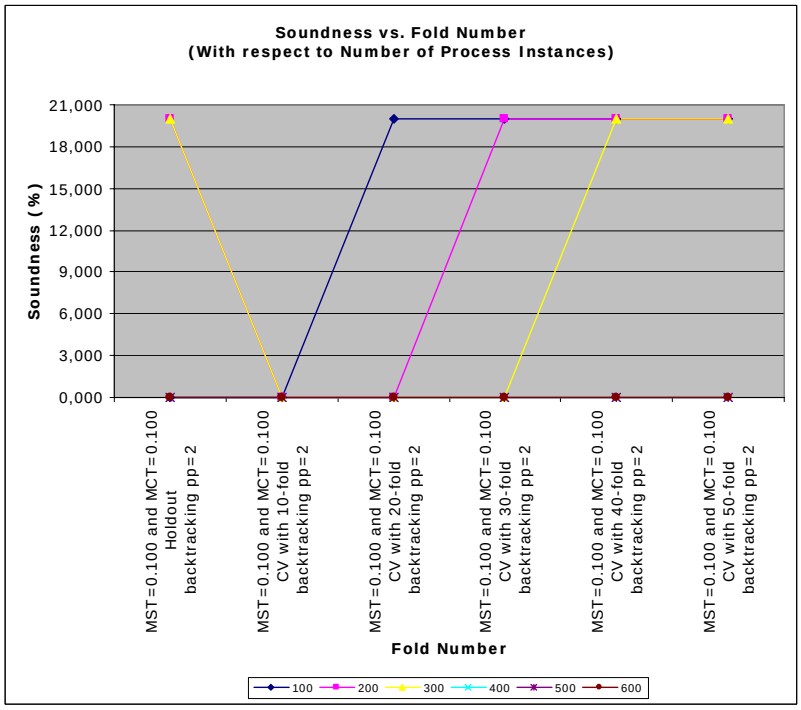


Figure 5.26 Correlation between Fold Number and Soundness
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.2 of Appendix J2)

Positive correlation between fold number and completeness is tabulated in Table 5.14.

Table 5.14 Correlation between Fold Number and Completeness
With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	0.783	0.870	0.810
200	0.915	0.740	0.478
300	0.920	0.972	0.891
400	0.928	0.870	0.782
500	0.963	0.969	0.889
600	0.929	0.837	0.867

Correlation between fold number and completeness is visualized in Figures 5.27, 5.28 and 5.29.

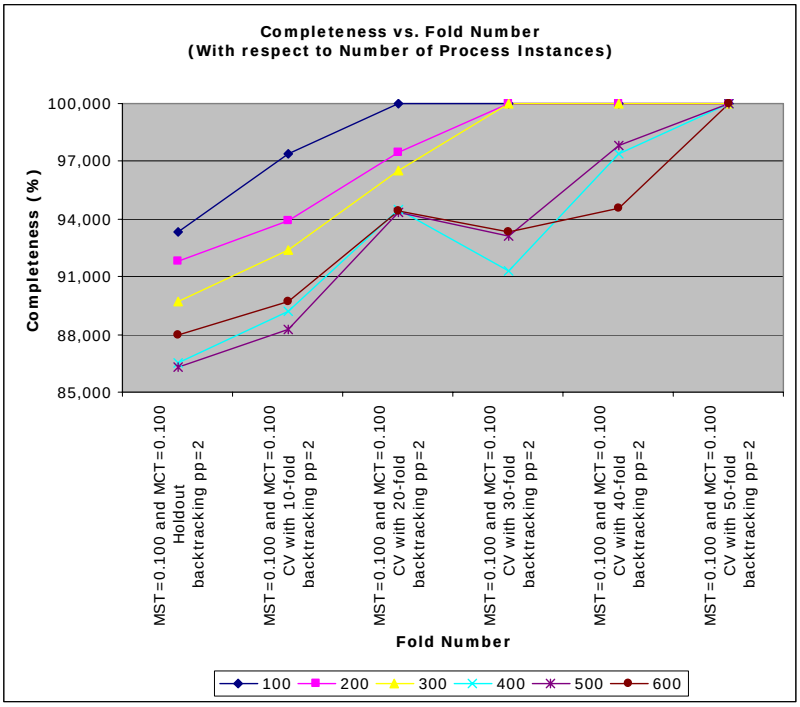


Figure 5.27 Correlation between Fold Number and Completeness
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.1 of Appendix J3)

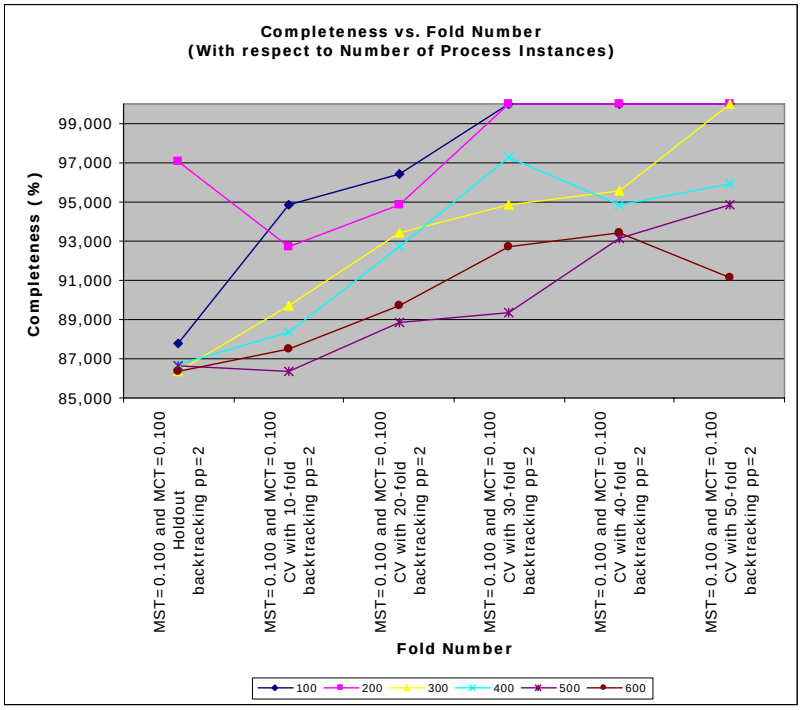


Figure 5.28 Correlation between Fold Number and Completeness
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.1 of Appendix J1)

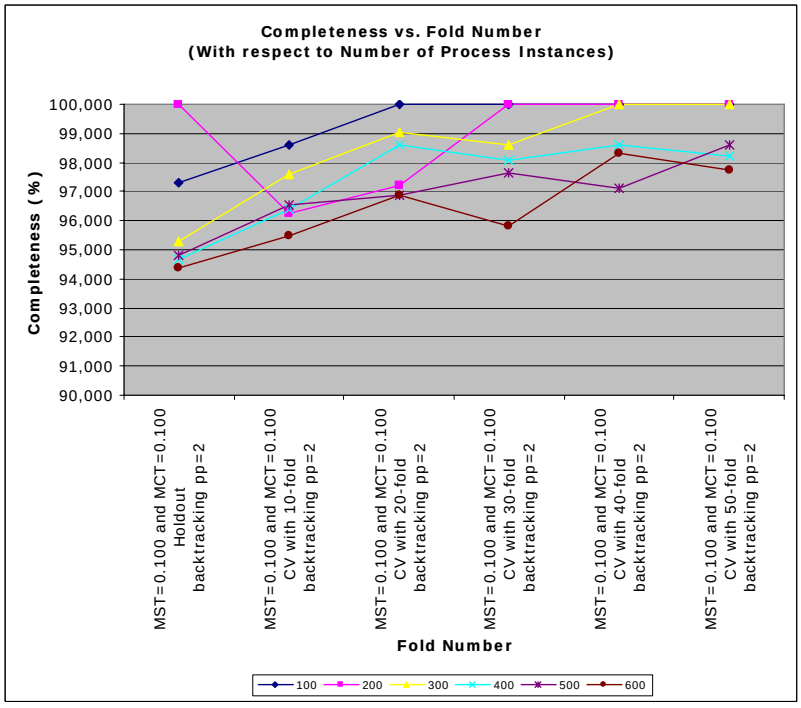


Figure 5.29 Correlation between Fold Number and Completeness
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.1 of Appendix J2)

To conclude, cross-validation and holdout (threefold cross-validation), which are embedded to FTCBPD methodology, are two standard methods of data mining in predicting the performance of discovered process model. With respect to the increase in fold number (and shrinkage in the size of fold), single fold dedicated for testing is turned into a more flexible formation (bag), in which FTCBPD methodology easily fills with process instances exhibiting "standard" behaviors. Additionally iterative fashion in both cross-validation and holdout methods (i.e. fold number parameter of verification method indicates for how many times process modeling is executed) encourages FTCBPD methodology in this learning tendency. Actually this result indicates a parallelism with the model testing part of [14], which states that; the variance of the resulting estimate is reduced as fold number, k is increased. As a result, discovered process model turns into a sounder and more complete status without any increase in average arc traffic (i.e. any behavior gain).

The mechanism triggered by increase in fold number is modeled by a reinforcement cycle as in Figure 5.30.

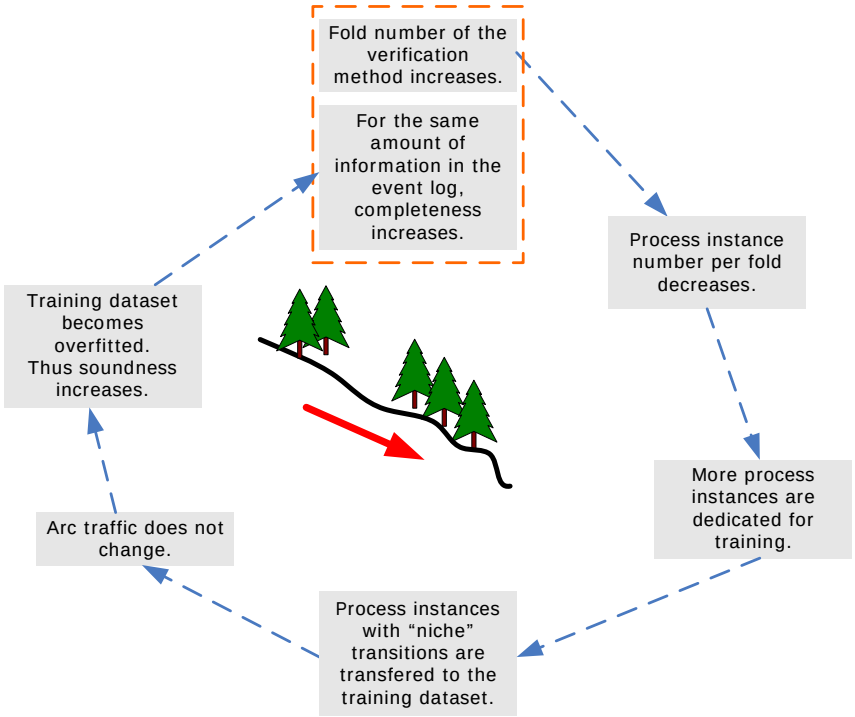


Figure 5.30 Reinforcement Cycle for the Effect of the Fold Number On the Learning Scheme

Beside the effect on the learning scheme, fold number of the underlying verification method directly affects the total processing time, since fold number indicates the iteration number for

how many times process modeling is executed. As a result, total processing time can be formulated such that;

$$total\ \ processing\ \ time = unit\ \ processing\ \ time \times\ fold\ \ number \quad (eq.\ 5.2)$$

Positive correlation between fold number and total processing time strengthens this implication in Table 5.15.

Table 5.15 Correlation between Fold Number and Total Processing Time
With respect to Number of Process Instances and Business Process

Number of Process Instances	Travel Management	Credit Card Application	Repair
100	1.000	0.997	0.998
200	1.000	0.998	0.998
300	1.000	0.998	0.997
400	1.000	0.996	0.998
500	1.000	0.997	0.997
600	1.000	0.998	0.995

Linearity of the relation between total processing time and fold number highlights the perfect correlation tabulated above. Correlation between fold number and total processing time can be visualized in Figures 5.31, 5.32 and 5.23.

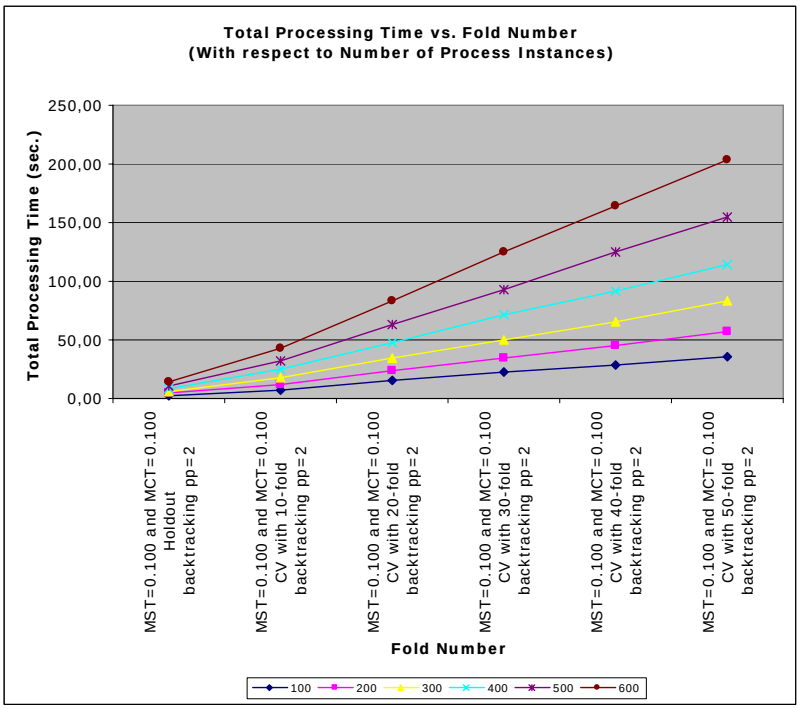


Figure 5.31 Correlation between Fold Number and Total Processing Time
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.4 of Appendix J3)

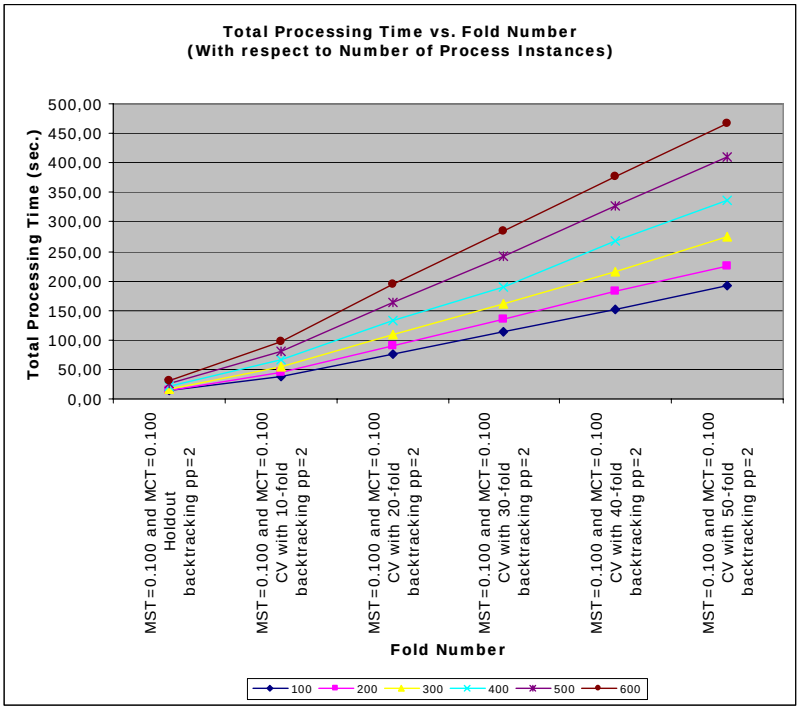


Figure 5.32 Correlation between Fold Number and Total Processing Time
With respect to Number of Process Instances for Credit Card Application Business Process
(Details of this figure are given in Table J1.4 of Appendix J1)

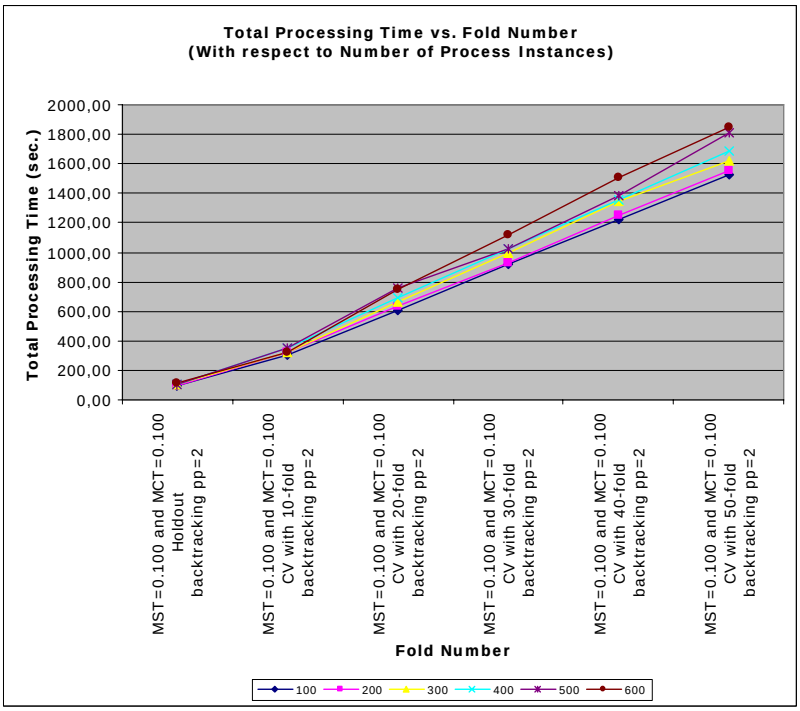


Figure 5.33 Correlation between Fold Number and Total Processing Time
With respect to Number of Process Instances for Repair Business Process
(Details of this figure are given in Table J2.4 of Appendix J2)

5.2.4. The Effect of Number of Process Instances on Non-Functional Key Performance Metrics

In operational perspective, the amount of information in the event logs influences the running time of “Populate FROMTOCHART Table”, “Verify Discovered Process Model” and “Report Process Instances” operations. This can be drawn from the operational-level complexity stated in part 5.3.1.

At “Populate FROMTOCHART Table” operation transaction streams of the process instances, which are dedicated to training dataset, are constructed by grouping the tuples in TRANSACTIONLOG table by case identifier (e.g. caseID) in the ascending order of timestamp(i.e. transaction log date and time). Afterwards transitions in each stream are inserted (weaved) into the appropriate element of FROMTOCHART table. In this aspect if there are more process instances in TRANSACTIONLOG table, it implies more grouping by case identifier. Thus it requires more running time to complete “Populate FROMTOCHART Table” operation.

Similarly at “Verify Discovered Process Model” operation, transaction streams of the process instances, which are dedicated to testing dataset, are constructed by grouping the tuples in TRANSACTIONLOG table by case identifier in the ascending order of time stamp. Thus increase in number of process instances implies increase in running time for “Verify Discovered Process Model” operation.

At “Report Process Instances” operation, transaction stream of each process instance is constructed and every constructed transaction stream is written on a text file. As at “Populate FROMTOCHART Table” operation, running time of “Report Process Instances” operation is dependent to the number of process instances in TRANSACTIONLOG table.

As a result, unit processing time is increased by increase in the number of process instances. Nearly perfect correlation between number of process instances and total process time highlights this implication in Tables 5.16 and 5.17.

Table 5.16 Correlation between Number of Process Instances and Total Processing Time
With respect to Minimum Support Threshold (MST) value and Business Process

Business Process	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2
Travel Management	0.989	0.986	0.984	0.984	0.984
Credit Card Application	0.991	0.994	0.993	0.990	0.988
Repair	0.629	0.981	0.985	0.995	0.990

Table 5.17 Correlation between Number of Process Instances and Total Processing Time
With respect to Minimum Confidence Threshold (MCT) value and Business Process

Business Process	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
Travel Management	0.988	0.987	0.987	0.986	0.985
Credit Card Application	0.993	0.988	0.989	0.990	0.992
Repair	0.892	0.987	0.985	0.963	0.938

Correlation between number of process instances and total processing time is visualized in Figures 5.34, 5.35 and 5.36.

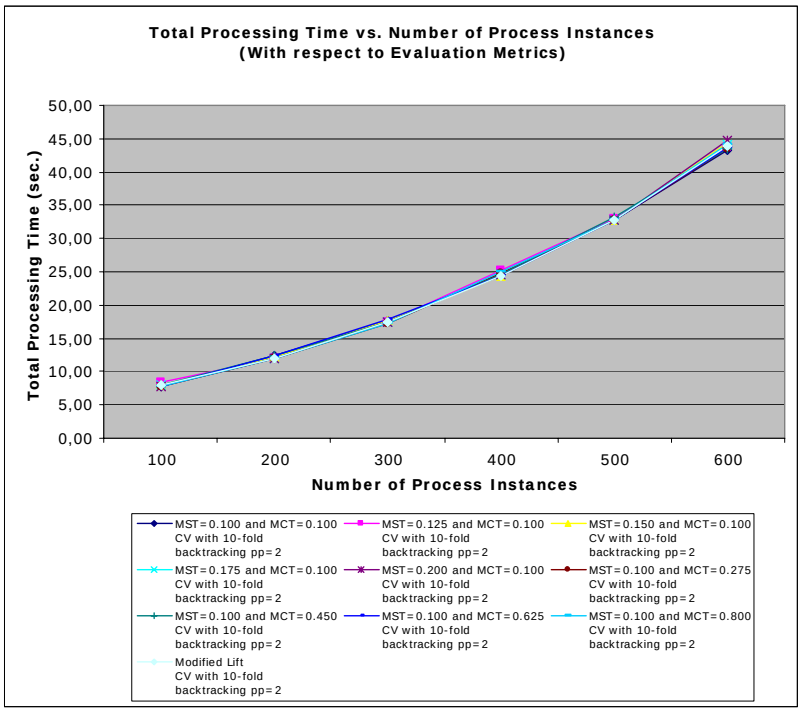


Figure 5.34 Correlation between Number of Process Instances and Total Processing Time
With respect to Number of Process Instances for Travel Management Business Process
(Details of this figure are given in Table J3.4 of Appendix J3)

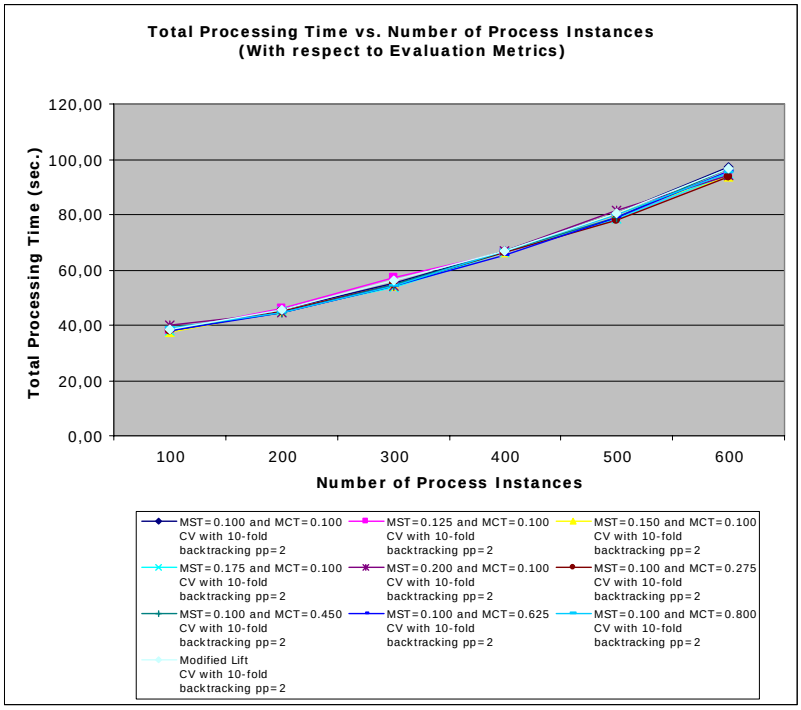


Figure 5.35 Correlation between Number of Process Instances and Total Processing Time With respect to Number of Process Instances for Credit Card Application Business Process (Details of this figure are given in Table J1.4 of Appendix J1)

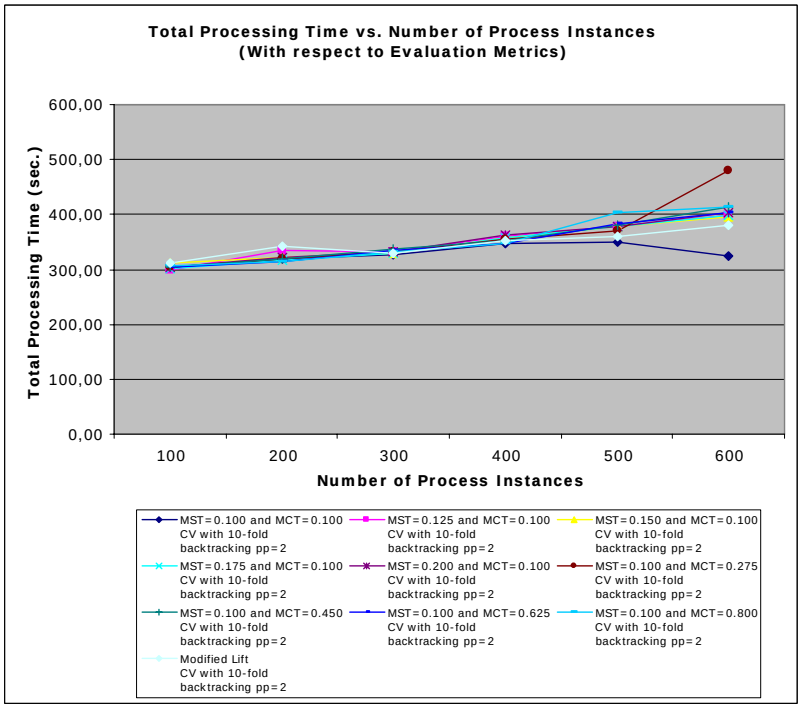


Figure 5.36 Correlation between Number of Process Instances and Total Processing Time With respect to Number of Process Instances for Repair Business Process (Details of this figure are given in Table J2.4 of Appendix J2)

Another effect of number of process instances is on the total size of the database, which consists of TRANSACTIONLOG, FROMTOCHART, INITIALFROMTOCHART, TALLYMARK, TCODES and BUSINESSPROCESSES tables. Number of process instances case-related parameter basically dominates the size of TRANSACTIONLOG table, which holds imported event logs, since increase in number of process instances directly implies increase in the line of event logs. Other tables compose the autonomous portion of total size. Correlation between number of process instances and total size highlights a perfect positive relationship between size and number of process instances in Table 5.18.

Table 5.18 Correlation between Number of Process Instances and Total Size
With respect to Business Process

Travel Management	Credit Card Application	Repair
1.000	1.000	1.000

The size of TRANSACTIONLOG table is expanded by the increase in number of process instances. The unchanging residue between total size and the size of TRANSACTIONLOG table indicates autonomous portion. Correlation between number of process instances and total size is visualized in Figures 5.37, 5.38 and 5.39.

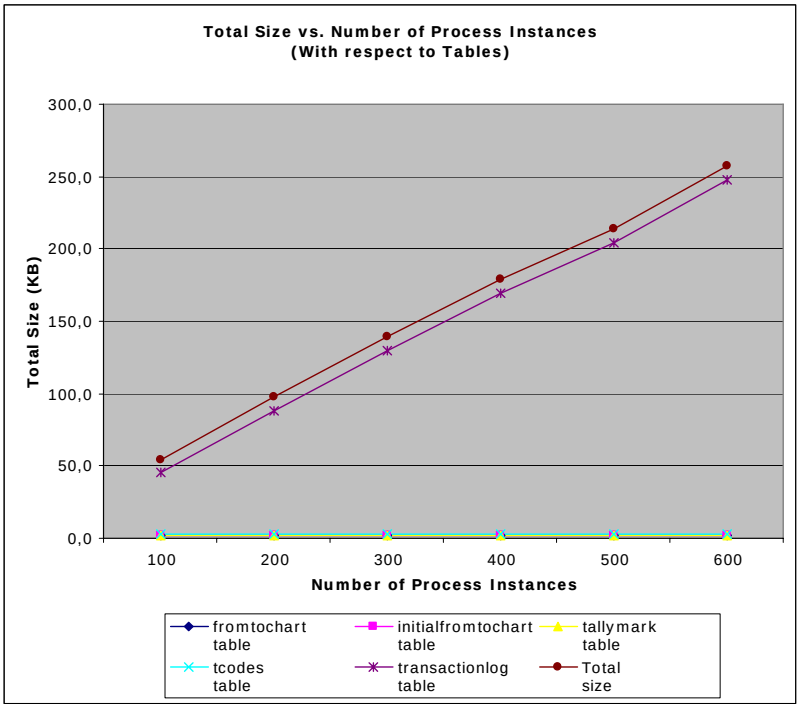


Figure 5.37 Correlation between Number of Process Instances and Total Size
With respect to Tables for Travel Management Business Process
(Details of this figure are given in Table J3.5 of Appendix J3)

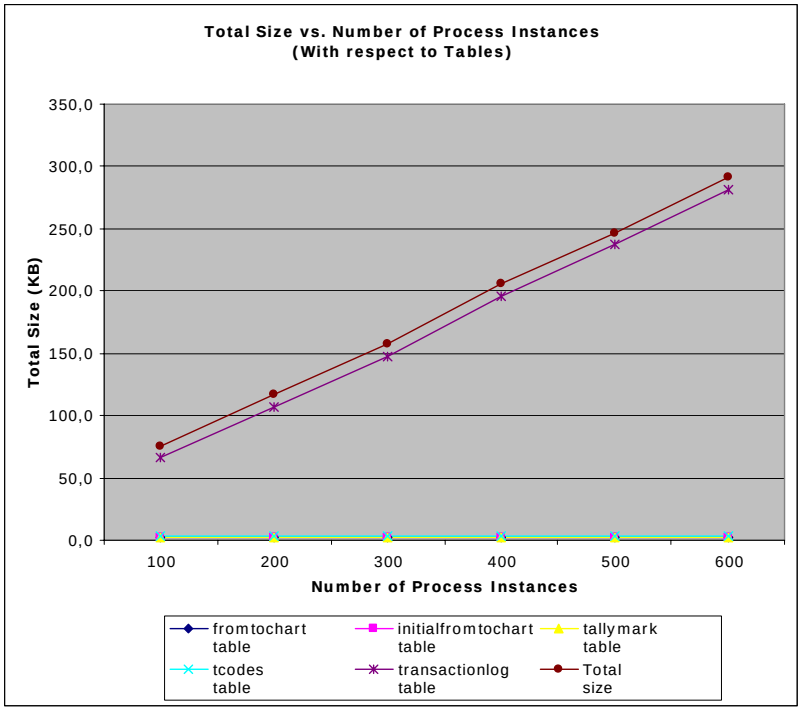


Figure 5.38 Correlation between Number of Process Instances and Total Size With respect to Tables for Credit Card Application Business Process (Details of this figure are given in Table J1.5 of Appendix J1)

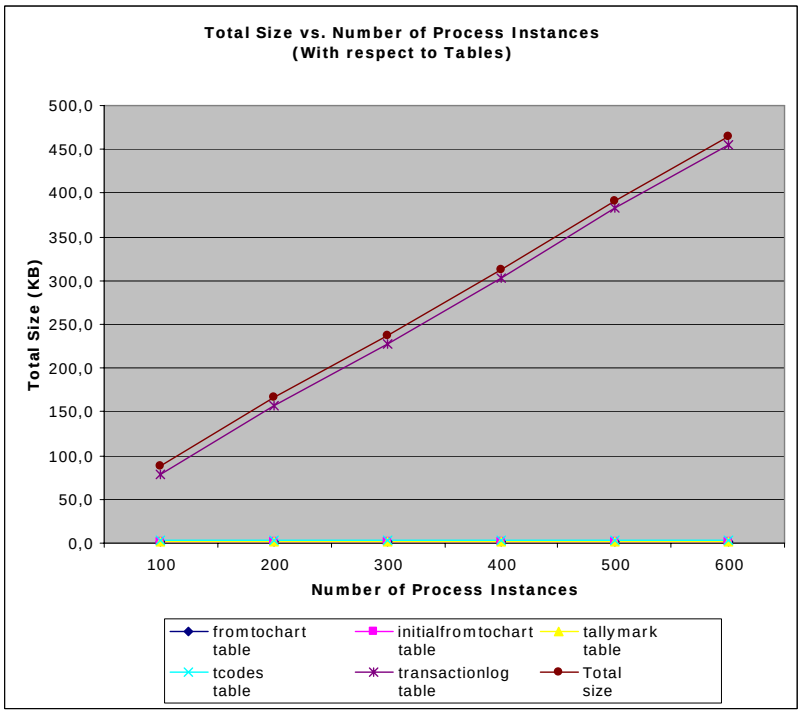


Figure 5.39 Correlation between Number of Process Instances and Total Size With respect to Tables for Repair Business Process (Details of this figure are given in Table J2.5 of Appendix J2)

5.3. Evaluation Based on Number of Activity Types

In this experiment, key performance metrics (i.e. completeness, soundness, average arc traffic, process time and size) varying according to the number of activity types and execution-related parameters (i.e. evaluation metrics, verification methods and back-tracking penalty point) are measured for each industrial application. Number of process instances is set to 500 as the default value for the amount of information in the event logs.

Number of activity types is a domain-dependent parameter, which indicates the complexity of concentrated business process. For business processes with larger number of activity types, discovering process model is potentially much more difficult and time-consuming, since process model may exhibit alternative and parallel routings due to this structural complexity. Additionally, event logs typically do not contain all possible combinations. Hence a completeness problem may emerge in mining process model.

According to correlation coefficient, there is not any clear evidence that the number of activity types have a significant influence on the key performance metrics in accuracy and minimality perspectives (i.e. completeness, soundness and average arc traffic). This result is in parallel with the outcomes stated in [12]. Calculated correlation coefficient of number of activity types and execution-related parameters are in Tables 5.19, 5.20, 5.21 and 5.22.

Table 5.19 Correlation between Number of Activity Types and Completeness
With respect to Fold Number

MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
0.882	0.767	0.305	0.549	-0.141	-0.259

Table 5.20 Correlation between Number of Activity Types and Average Arc Traffic
With respect to Minimum Support Threshold (MST) Value

MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2
0.478	0.478	0.352	0.256	0.350

Table 5.21 Correlation between Number of Activity Types and Average Arc Traffic
With respect to Minimum Confidence Threshold (MCT) Value

MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
0.900	0.612	0.404	0.584	-0.700

Table 5.22 Correlation between Number of Activity Types and Soundness
With respect to Back-tracking Penalty Point

MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
0.000	0.000	0.000	0.000	0.000

On the other hand, major effect of number of activity types is on the non-functional key performance metrics (i.e. processing time and size). Tabulated form of obtained key performance metrics' measurements are represented in APPENDIX K.

5.3.1. The Effect of Number of Activity Types on Non-Functional Key Performance Metrics

In processing time aspect, describing the effort with "Big O notation", which indicates the worst-case run time for an algorithm, is an appropriate way to determine the key (case-related or execution-related) parameters and define approximate total processing time in terms of these parameters. According to Big O notation, the complexity of each operation (and sub-operation) in FTCBPD methodology is tabulated in Table 5.23.

Table 5.23 Operational-level Complexity of FTCBPD Methodology

	Operation	Complexity
1	Create FROMTOCHART Table	$O(\|AT\|)$
2	Populate FROMTOCHART Table	$O(\|AT\|\times\|PI\|+\|AT\|^2)$
2.1	Populate FROMTOCHART Table	$O(\|AT\|\times\|PI\|)$
2.2	Populate TALLYMARK Table	$O(\|AT\|^2)$
3	Evaluate Tally Marks in FROMTOCHART Table	$O(\|AT\|^2)$
3.1	Evaluate Tally Marks due to MST and MCT	$O(\|AT\|^2)$
3.2	Evaluate Tally Marks due to modified lift	$O(\|AT\|^2)$
4	Rearrange FROMTOCHART Table	$O(\|AT\|!)$
5	Construct Process Model	$O(\|AT\|^2)$
6	Eliminate One-Step Closed Loops at Discovered Process Model	$O(2\times\|AT\|^2)$
6.1	Detect One-Step Closed Loops	$O(\|AT\|^2)$
6.2	Delete One-Step Closed Loops	$O(\|AT\|^2)$
7	Construct AND/OR Connections at Discovered Process Model	$O(2\times\|AT\|^2)$
7.1	Construct Split-type Connections	$O(\|AT\|^2)$
7.2	Construct Join-type Connections	$O(\|AT\|^2)$
8	Verify Discovered Process Model	$O(\|AT\|^3\times\|PI\|)$
9	Report Process Instances	$O(\|AT\|\times\|PI\|)$

$|AT|$: number of activity types occurred in event logs
$|PI|$: number of process instances in event logs

“Create FROMTOCHART Table” and “Evaluate Tally Marks in FROMTOCHART Table” operations are on the order of |AT| and $|AT|^2$, since row and column count of “squared” FROMTOCHART table is determined by the number of activity types of the concentrated business process.

Additionally “Populate FROMTOCHART Table”, “Verify Discovered Process Model” and “Report Process Instances” operations are directly related to number of process instances and number of activity types, since these case-related parameters affect the amount information in the event logs. The effect of number of process instances is stated in part 5.2.4. In addition to number of process instances, the number of activity types increments total trace lines in event logs according to the assumption of “The more activity types, the longer transaction streams”[7].

“Rearrange FROMTOCHART Table” operation, which is the “engine component” of FTCBPD methodology, discovers the intermediate product, optimum activity sequence with respect to minimum moment value. But this crucial operation is the bottleneck of FTCBPD methodology in processing time aspect, because of factorial complexity. The major reason of order |AT|! is the Permutation object that generates activity sequences, which are used as an input to determine relative location of diagonal and calculate moment value for each activity arrangement. The activity sequence length is determined by the width of FROMTOCHART table (i.e. attribute number of FROMTOCHART table) and this attribute number is designated by the number of activity types occur in the event logs at “Create FROMTOCHART Table” operation.

At “Construct Process Model”, “Eliminate One-Step Closed Loops in Discovered Process Model” and “Construct AND/OR Connections in Discovered Process Model” operations, optimum process model is tracked by a one-dimensional array type data structure, which is structurally independent from FROMTOCHART table. Each element in the array holds predecessors and successors of the underlying activity (transaction) by distinct linked-lists[8] embedded into the array element and the maximal limit of these linked-lists is the number of activity types (i.e. the zero-step closed loops are also handled). Therefore these operations are the order of $|AT|^2$ (i.e. O(|AT|) order for linear search in one-dimensional array and other O(|AT|) order for linear search in successor (predecessor) activity linked-list).

[7] Number of activity types acts as the average (standard) length of a single transaction stream.

[8] These linked-lists upgrade one-dimension array type process model into two-dimension structure.

Positive correlation between number of activity types and processing time strengthens the complexity determinations in Tables 5.24 and 5.25.

Table 5.24 Correlation between Number of Activity Types and Process Time
With respect to Minimum Support Threshold (MST) value

MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2
0.927	0.920	0.923	0.921	0.924

Table 5.25 Correlation between Number of Activity Types and Process Time
With respect to Minimum Confidence Threshold (MCT) value

MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
0.921	0.921	0.921	0.918	0.925

Optimum activity sequence and each activity (object) embedded to this sequence is represented in Figure 5.40.

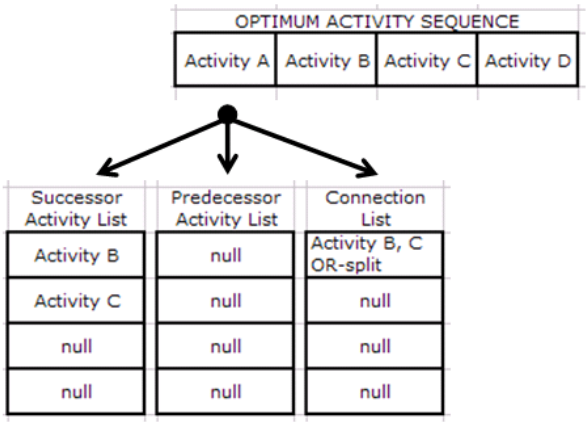


Figure 5.40 Data Structure of Optimum Activity Sequence and Activity Object

Correlation between number of activity types and processing time is visualized in Figure 5.41.

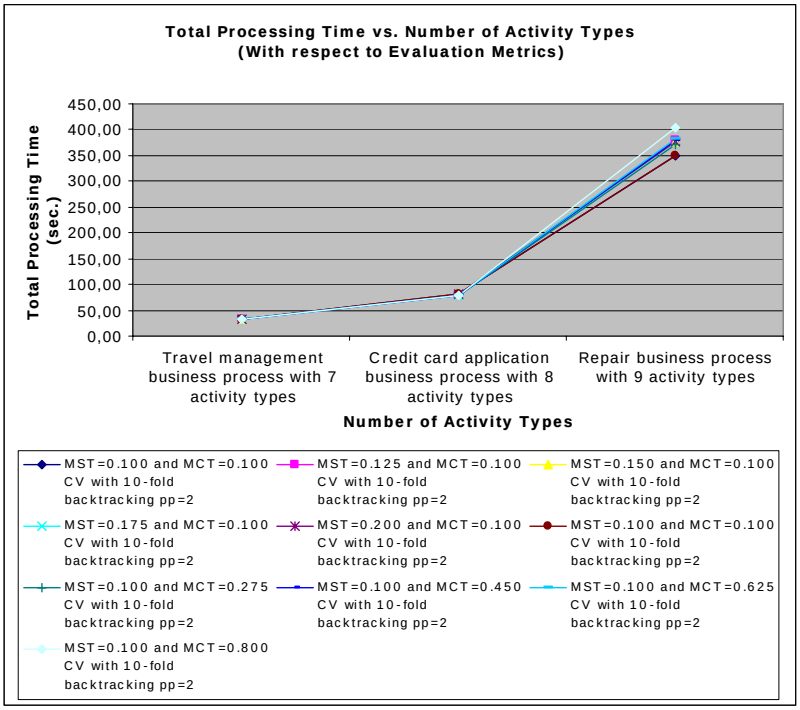


Figure 5.41 Correlation between Number of Activity Types and Total Processing Time With respect to Evaluation Metrics
(Details of this figure are given in Table K.4 of Appendix K)

As stated in prior experiments, total processing time is determined by number of process instances, number of activity types and verification method fold number parameters. In order to notice the effect of these parameters on processing time, a linear regression is performed in WEKA (Waikato Environment for Knowledge Analysis), which is a data mining workbench developed by Machine Learning (ML) Group of the University of Waikato [34]. According to this linear regression, total processing time of FTCBPD methodology can be predicted as follows:

$$unit\ processing\ time = 0.0001 \times |AT|! - 0.6062 \quad (eq.\ 5.3)$$

$$total\ processing\ time = \left(0.0001 \times |AT|! - 0.6062\right) \times FN \quad (eq.\ 5.4)$$

Parameters stated in the predictor are:

$|AT|$ is the number of activity types of the concentrated business process.

FN is the fold number of the verification method.

Unfortunately the effect of number of process instances (|PI|) emphasized in operational-level complexity determinations is eliminated by number of activity types (|AT|) in processing time predictor, because of the running time coefficient of rearrangement operation. As a result, FTCBPD methodology is limited by the complexity of concentrated business process with respect to processing time. Results of the related linear regression are stated in APPENDIX D.

Another effect of number of activity types is on the total size of the database. This case-related parameter basically dominates TRANSACTIONLOG table, since total trace lines in event logs are exaggerated by number of activity types for the same number of process instances. Positive correlation between number of activity types and size, which is worth of 0.940, strengthens this implication. Additionally, relationship between number of activity types and processing time is visualized in Figure 5.42.

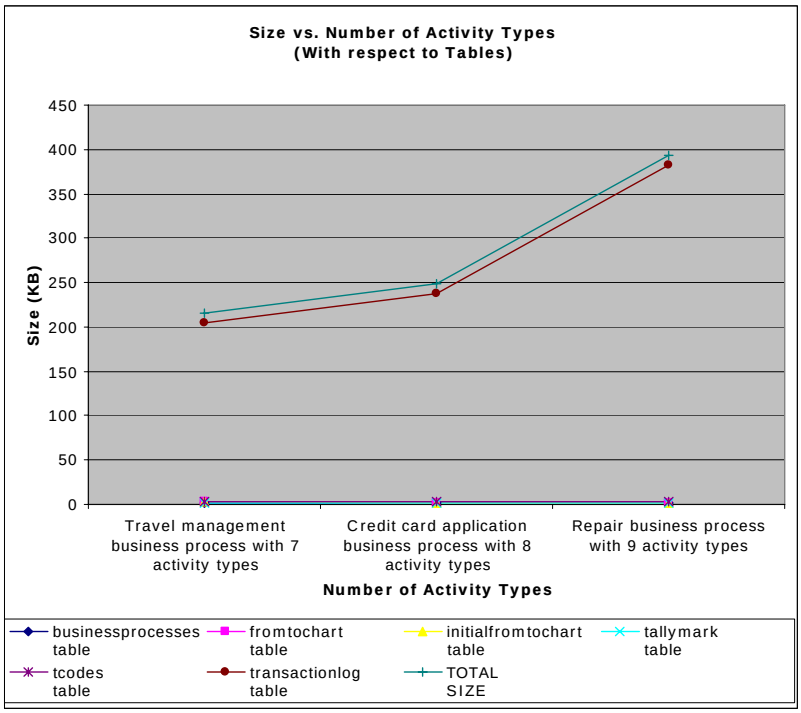


Figure 5.42 Correlation between Number of Activity Types and Total Size With respect to Tables
(Details of this figure are given in Table K.5 of Appendix K)

The size of TRANSACTIONLOG table is exaggerated by the increase in number of process instances. The unchanging residue between total size and the size of TRANSACTIONLOG table indicates autonomous portion, which is composed of the total size of FROMTOCHART, INITIALFROMTOCHART, TALLYMARK, TCODES and BUSINESSPROCESSES tables.

While the size of TRANSACTIONLOG table is determined by number of process instances and number of activity types, any significant change does not occur in FROMTOCHART table with respect to the increase in number of activity types. But FROMTOCHART table is structurally dependent to this parameter, since column number is designated by number of activity types occur in the event logs.

Similar to total processing time, a linear regression is performed in WEKA for modeling size requirement. According to this linear regression, total size of a process modeling application can be predicted as follows:

$$total\ \ size = 0.0717 \times |AT| \times |PI| + 7.2921 \quad (eq.5.5)$$

Parameters stated in the predictor are:

$|AT|$ is the average (standard) length of a transaction stream.

$|PI|$ is the number of process instances in the imported event logs.

According to the discovered predictor for total size, $|AT|$ is the total trace lines dedicated to a single process instance, while $|AT| \times |PI|$ indicates approximate total trace lines in the event logs. In this aspect, 0.0717 is the unit size (i.e. indexing and data) of a tuple in TRANSACTIONLOG table and 7.2921 autonomous portion is the total size of other tables. Results of the related linear regression are stated in APPENDIX E.

5.4. Evaluation Based on Modified Lift

The initial modification performed on original lift metric in association rule mining is the addition of predecessor and successor notations as the probability terms. But the major modification is the evaluation procedure of the underlying tally mark with respect to calculated lift value.

The first version named “modified lift two-branch use” has two branches similar to “supervised” minimum support and minimum confidence thresholds (MST and MCT). While tally marks with positive correlation are not altered, negatively correlated or non-correlated tally marks are reset to zero.

The second lift version named “modified lift three-branch use” attempts to reflect the negative correlation to rearrangement operation. For this reason, tally marks with negative correlation are multiplied by minus one to make these tally marks attractive in “Rearrange

FROMTOCHART Table" operation, since this operation tends to increase the "moment arm" of the underlying activity (transaction) pairs with negative tally mark with respect to moment minimizing objective function. Hence activities (transactions) with negative correlation are assigned to non-neighboring (relatively distant) positions at optimum activity sequence. On the other hand, reduction in local moment value according to separating activity (transaction) pairs with negative tally mark may be compensated by extra increase in total moment value due to rearrangement of other activities (transactions) that are positively correlated with these activities.

Pseudo-codes of "modified lift two-branch use" and "modified lift three-branch use" evaluation metrics are stated below:

```
SET tally mark TO READ score at element (i,j) in FROMTOCHART table
SET row total TO row total of activity i in TALLYMARK table
SET column total TO column total of activity j in TALLYMARK table
CALCULATE tally mark gross total in FROMTOCHART table
IF ((tally mark × tally mark gross total) <= (row total × column total)) THEN
        SET score at element (i,j) in FROMTOCHART table TO 0
ELSE
        DO NOTHING
ENDIF
```

```
SET tally mark TO READ score at element (i,j) in FROMTOCHART table
SET row total TO row total of activity i in TALLYMARK table
SET column total TO column total of activity j in TALLYMARK table
CALCULATE tally mark gross total in FROMTOCHART table
IF ((tally mark × tally mark gross total) < (row total × column total)) THEN
        SET score at element (i,j) in FROMTOCHART table TO tally mark × (-1)
ELSEIF ((tally mark × tally mark gross total) = (row total × column total)) THEN
        SET score at element (i,j) in FROMTOCHART table TO 0
ELSE
        DO NOTHING
ENDIF
```

5.4.1. Comparison of Two Versions of Modified Lift

In behavioral perspective of discovered process model (i.e. transition number, transition variety in discovered process model and pattern of these transitions), methodologically there is not a significant distinction between two versions of modified lift evaluation metric. Because the sole decision rule in representing the behaviors (transitions) in discovered process model is the tally mark of the underlying transition to be greater than zero and modified lift two-branch use and three-branch use propose the same strategy in handling tally marks with positive correlation. Therefore multiplying the tally mark by minus one or resetting it to zero does not make any sense in completeness, soundness and average arc traffic key performance metrics.

While additional evaluation procedure (i.e. multiplying negatively correlated tally mark by minus one) does not result in any enhancement in behavioral structure of the discovered process model, average length of the transitions may be incremented by the tendency

towards separating activity pairs in rearrangement of activity sequence. Thus a new measurement, named *average transition length*, is introduced to monitor the effect of modified lift on discovered process model.

$$average\ \ transition\ \ length = \frac{TLT}{TTN} \quad (eq.5.6)$$

Parameters stated in this formula are:

TLT is the total length of transitions represented in discovered process model.

TTN is the total number of transitions in discovered process model.

To determine whether additional evaluation procedure results in significant effects on discovered process model with respect to average transition length, a dependent t-test[9] is performed in this experiment. According to the differences obtained for each 36 dependent observations at travel management business process, calculated t-value (2.726) is greater than tabled t-value (2.03). Hence it is concluded that; additional evaluation procedure (i.e. multiplying by minus one) significantly increases average (and total) length of the transitions in the discovered process model. Dependent means t-test can be summarized as follows:

Table 5.26 Dependent t-test Summary

t-Test: Paired Two Sample for Means

	Modified Lift 3-Branch Use	**Modified Lift 2-Branch Use**
Mean	1.668888889	1.594722222
Variance	0.029484444	0.006048492
Observations	36	36
Pearson Correlation	0.332668196	
Hypothesized Mean Difference	0	
df	35	
t Stat	2.726018917	
P(T<=t) one-tail	0.00497253	
t Critical one-tail	1.68957244	
P(T<=t) two-tail	0.00994506	
t Critical two-tail	2.030107915	

Related information about dependent means t-test is stated in APPENDIX F.

[9] "Dependent" term denotes that the second observation is related to the first since the same training and testing datasets are being handled.

5.4.2. Effect of Modified Lift on Learning Scheme

In this experiment, it is aimed to concentrate on the characteristics of modified lift and compare this metric with MST and MCT evaluation metrics with respect to key performance metrics (i.e. completeness).

The underlying assumptions in this experiment are as follows:

- Every activity type in the event logs appears in each process instance for only once. In other words, the same activity cannot be used in two different parts of a process cycle. Hence number of activity types ($|AT|$) acts as the standard length of a transaction stream dedicated to a single process instance and $|AT| \times |PI|$ indicates approximate total trace lines in the event logs.
- According to the first assumption, gross total tally mark ($\#G$) marked into FROMTOCHART table can be formalized such that;

$$\#G = |AT| \times \#L \quad (eq.5.7)$$

Parameters stated in the predictor are:

$\#G$ is gross total tally mark in FROMTOCHART table.

$|AT|$ is the number of activity types in event logs.

$\#L$ is the number of process instances in event logs.

Firstly some algebraic conversions are applied to modified lift formula as follows:

$$Modified\ lift = \frac{|A > B| \times \#G}{|A > *| \times |* > B|} \quad where\ \#G = |AT| \times \#L \quad (eq.5.8)$$

$$Modified\ lift = \frac{|A > B|}{|A > *|} \times \frac{|AT| \times \#L}{|* > B|} \quad (eq.5.9)$$

$$Modified\ lift = conf(A,B) \times \frac{|AT| \times \#L}{|* > B|} \quad (eq.5.10)$$

$$Modified\ lift = conf(A,B) \times \frac{|AT| \times \#L}{|A > B| \times \xi} \quad where\ \xi \in [1, |AT| \text{-} 1] \quad (eq.5.11)$$

$$Modified\ lift = conf(A,B) \times \frac{|AT|}{\frac{|A > B|}{\#L} \times \xi} \quad (eq.5.12)$$

$$Modified\ lift = \frac{conf(A,B)}{sup(A,B)} \times \frac{|AT|}{\xi} \quad (eq.5.13)$$

In equation 5.9, the first multiplier is turned into explicit form of confidence formulation (i.e. $conf(A,B)$), while the numerator of the second multiplier, gross total tally mark, is converted to $|AT| \times \#L$ notation according to the second assumption. Then explicit form of confidence formulation is replaced with implicit form in equation 5.10.

In equation 5.11, the denominator of second multiplier, $|* > B|$, is turned into $|A > B| \times \xi$, where ξ denotes the overall ratio of transition from activity (transaction) A to activity (transaction) B, denoted as $|A > B|$, to total incoming traffic towards activity (transaction) B.

In equation 5.12, explicit form of support formulation (i.e. $sup(A,B)$) is introduced at the second multiplier by shifting total number of process instances, denoted as $\#L$, to the denominator. Finally explicit form of support formulation is replaced with implicit form in equation 5.13.

As a result, it is noticed in equation 5.13 that; modified lift tends to represent the transitions with higher confidence and lower support value according to the converted modified lift

formula. Actually this finding is in parallel with [23], which refers original lift as the lift of the association (correlation) rule $A \Rightarrow B$.

Secondly it is aimed to interpret the relationship between modified lift and other developed evaluation metrics, MST and MCT. For this reason, two dependent t-tests are performed to determine whether there is a significant performance difference between modified lift and MST, MCT evaluations metrics with respect to completeness metric.

In the first t-test, MST value is gradually increased from 0.100 up to 0.200 while MCT value is set to 0.100. According the differences obtained for each 30 observations at travel management business process, calculated t-value (4.72) is greater than tabled t-value (2.05). Hence it is concluded that; discovered process model evaluated by modified lift three-branch use has a better level of correspondence to the behaviors exhibited in event logs than gradually increasing MST. Dependent means t-test can be summarized as follows:

Table 5.27 Dependent t-test Summary

t-Test: Paired Two Sample for Means

	Modified Lift	**Gradually Increasing MST**
Mean	91.7085	86.84816667
Variance	6.225648534	22.54805214
Observations	30	30
Pearson Correlation	-0.130931841	
Hypothesized Mean Difference	0	
df	29	
t Stat	4.715121687	
P(T<=t) one-tail	2.80E-05	
t Critical one-tail	1.699126996	
P(T<=t) two-tail	5.60E-05	
t Critical two-tail	2.045229611	

In the second t-test, MCT value is gradually increased from 0.100 up to 0.200 while MST value is set to 0.100. According the differences obtained for each 30 observations, calculated t-value (2.81) is greater than tabled t-value (2.05). Hence it is concluded that; discovered process model evaluated by modified lift three-branch use has a better level of correspondence to the behaviors exhibited in event logs than gradually increasing MCT. Dependent means t-test can be summarized as follows:

Table 5.28 Dependent t-test Summary

t-Test: Paired Two Sample for Means

	Modified Lift	Gradually Increasing MCT
Mean	91.7085	89.3435
Variance	6.225648534	12.72714509
Observations	30	30
Pearson Correlation	-0.125668257	
Hypothesized Mean Difference	0	
df	29	
t Stat	2.814011752	
P(T<=t) one-tail	4.35E-03	
t Critical one-tail	1.699126996	
P(T<=t) two-tail	8.70E-03	
t Critical two-tail	2.045229611	

To conclude, although modified lift three-branch use has a better performance than MST and MCT evaluation metrics in accuracy perspective, the closeness of calculated t-value to tabled t-value in the second dependent t-test (i.e. 2.81 rather than 4.71) emphasizes the tendency of modified lift three-branch use towards representing the transitions with high confidence and low support value, which are called surprised-type behaviors. Additionally this closeness of process modeling executions parameterized with gradually increasing MCT to the executions parameterized with modified lift can be figured out in Figure 5.43 - 5.48.

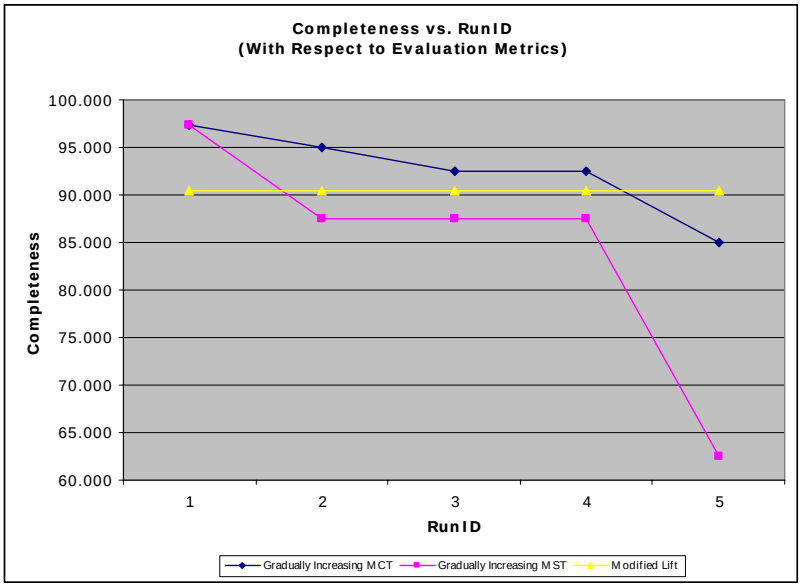


Figure 5.43 Residual Analysis between Modified Lift and Gradually Increasing MST/MCT
For 100 Process Instances
(Details of this figure are given in observations table of Appendix G and H)

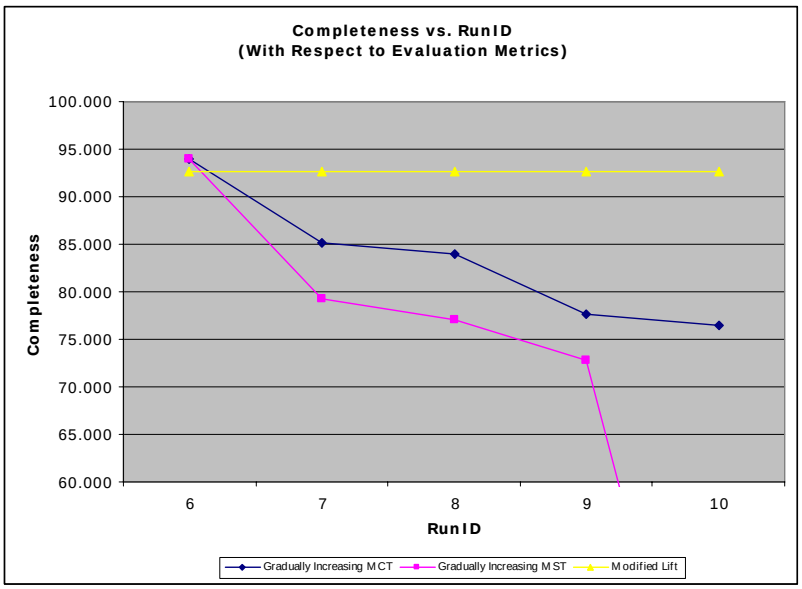


Figure 5.44 Residual Analysis between Modified Lift and Gradually Increasing MST/MCT For 200 Process Instances
(Details of this figure are given in observations table of Appendix G and H)

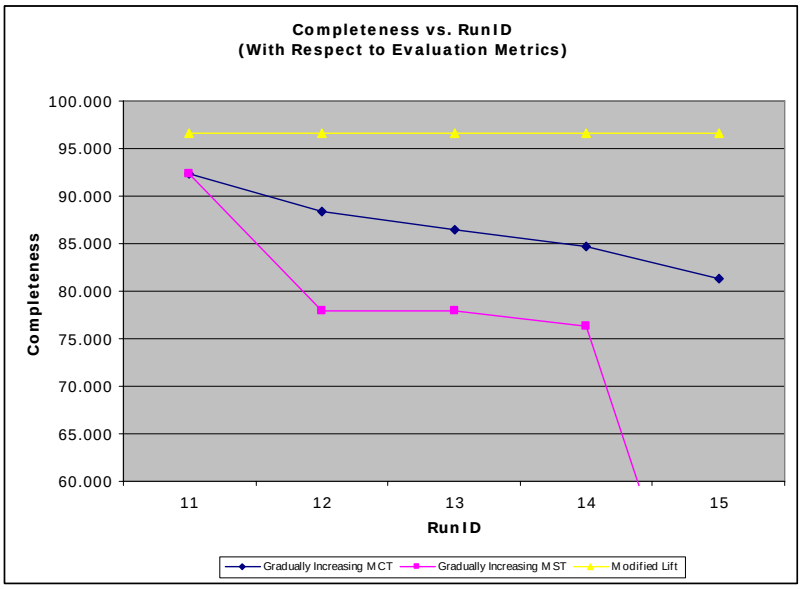


Figure 5.45 Residual Analysis between Modified Lift and Gradually Increasing MST/MCT For 300 Process Instances
(Details of this figure are given in observations table of Appendix G and H)

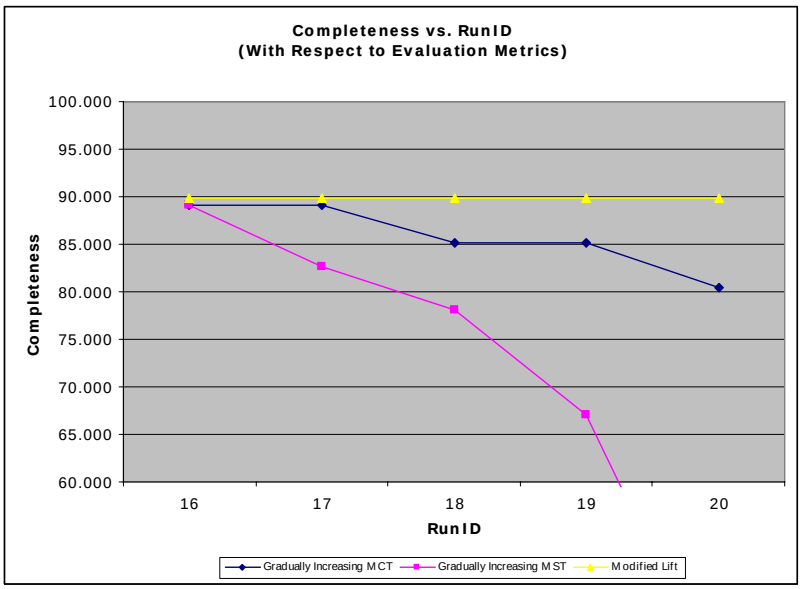


Figure 5.46 Residual Analysis between Modified Lift and Gradually Increasing MST/MCT For 400 Process Instances
(Details of this figure are given in observations table of Appendix G and H)

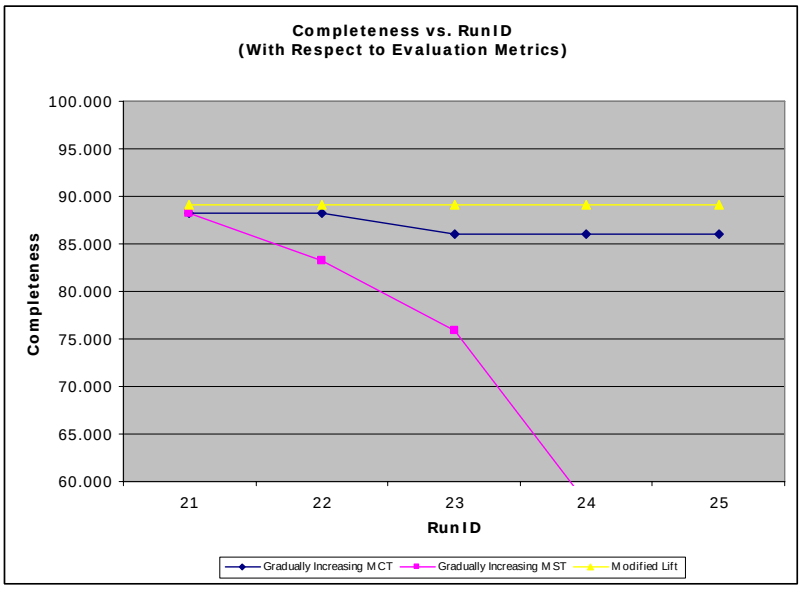


Figure 5.47 Residual Analysis between Modified Lift and Gradually Increasing MST/MCT For 500 Process Instances
(Details of this figure are given in observations table of Appendix G and H)

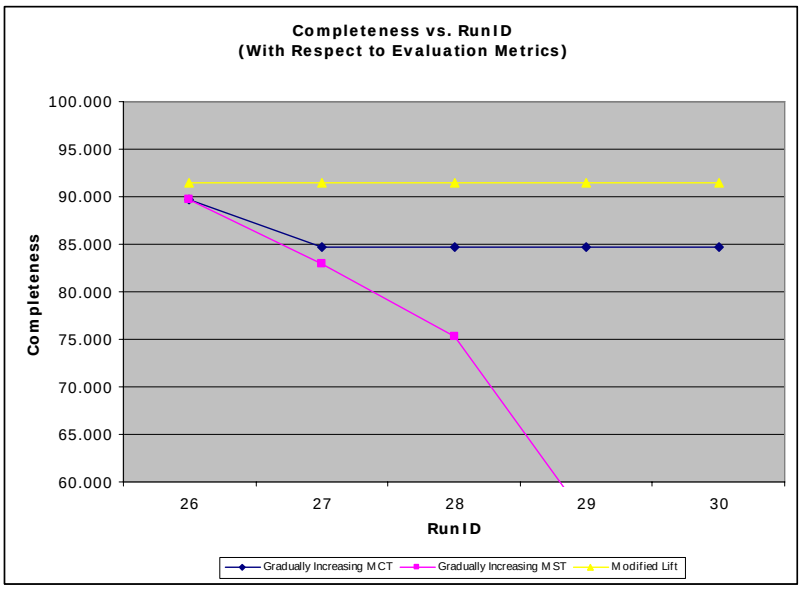


Figure 5.48 Residual Analysis between Modified Lift and Gradually Increasing MST/MCT
For 600 Process Instances
(Details of this figure are given in observations table of Appendix G and H)

Related information about dependent means t-tests are stated in APPENDIX G and H.

5.5. Comparison of FTCBPD Methodology with Prior Approaches

The fundamental requirement in process discovery field is the existence of a bookkeeping entity, which holds the moves (transitions) among activities (transactions) occurred in event logs. Thus "significant" patterns can be monitored and process model can be built up onto these patterns. The structure of this bookkeeping entity may range from unsophisticated two-dimensional array type data structures to database tables.

As stated in [1, 12, 14, 15] transitions among activities (transactions) captured in transaction streams are handled in *dependency/frequency table*, which is dedicated to a single activity (transaction). Thus number of dependency/frequency tables created in the underlying problem is determined by the number of activity types occurred in event logs.

In this table, interactions of the underlying activity (transaction) with other activities (transactions) are evaluated by different parameters (i.e. $\#A$, $\#B < A$, $\#A > B$, $\$A \rightarrow^{L} B$ and $\$A \rightarrow B$) as stated in "Literature Review" chapter. According to calculated dependency score in Little Thumb [1] or local, global and causality metric values in logistic regression model [14], which are based on dependency/frequency table, pair-wise transitions, which are

worth to be represented in discovered process model, are determined. Consequently process model is shaped by these driven transitions.

In this methodology, *from-to chart*, which is an essential analytical tool in material handling monitoring on production floor, is implemented as the bookkeeping material. Unlike to prior approaches, there is only one from-to chart and all captured transitions (moves) among activities (transactions) are populated (weaved) into this table. Therefore instead of traversing all relatively “small in size” tables in order to construct process model, this from-to chart serves a “big picture view” for more readily analyzing the underlying business process. Additionally, “Rearrange FROMTOCHART Table” operation aims to find out “the most frequent” activity sequence in event logs by taking into consideration overall inter-traffics among activities in from-to chart according to an optimization fashion. Although dimensions of from-to chart is theoretically dependent to number of activity types occurred in event logs, any significant correlation between number of activity types and the size of FROMTOCHART table is discovered as stated in part 5.3.1.

Another distinction between FTCBPD methodology and prior approaches is the evaluation metrics, which are used to prune down the “weak” patterns. Prior approaches have a tendency towards the development their own “unsupervised” metrics, which are borrowed from statistics. By combining metric values a threshold value is found out, and then significance or types of the relations are determined according this threshold value. In FTCBPD methodology, it is aimed to enhance metrics, which are inherited from association rule mining (i.e. support, confidence and lift), with predecessor and successor notations in order to shift concentration onto inter-transaction relations (patterns) (i.e. correlation between activities) rather than intra-transaction perspective in association rule mining.

As the last argument in this comparison, verification procedure of discovered process model can be handled. Theoreticians consider a process model to be good if it is both accurate and minimal. In this aspect, while discovered process model fully accounts for the sample behaviors given in training and testing population and successfully classifies types of relations (transitions), the structure of process model is not unnecessarily complicated by represented relations (transitions). In prior approaches, performance of process modeling is solely quantified by the correspondence of discovered process model to the behaviors exhibited in testing population. Minimality concept is not directly taken into consideration.

Likewise in prior studies, FTCBPD methodology determines the best process model with respect to completeness metric, which refers to the percentage of transitions that are compliant with some transitions in the event logs. On the other hand, “Predetermine arc

traffic in process model” feature enables the end user to declare an upper bound for transition number (i.e. arc traffic) in discovered process model. Thus FTCBPD methodology represents the most significant transitions, which are held in transition top list (TTL) in the end of process modeling session.

The results for default values of execution-related parameters given in Table 5.1 result in an average performance of 96.936. Optimum control flow graphs of the underlying business processes are given in APPENDIX I.

CHAPTER 6

LIMITATIONS AND FUTURE WORK

FTCBPD methodology attempts to distill a structured process description in the form of from-to chart, dependency/frequency graph and control flow graph from a set of real executions. In this study, apriori model (i.e. reference model) is not taken into consideration; however, the discovered process model may be used for delta analysis, i.e., comparing the mined model representing the actual business process with the reference model representing the predefined process, as a future research. In this delta analysis, actual and proposed business processes may be converted into from-to chart representations. The size of these from-to charts may be normalized to the size of the activity type union set. For the columns and rows added through the normalization, the corresponding elements in the normalized from-to chart are set to zero. By subtracting these matrixes, it can be detected discrepancies between actual and proposed process models. As an obvious way to capture the degree of discrepancies, sum of squares of elements in subtraction process matrix can be used in terms of distance metric. As a result, it is possible to monitor which sub-patterns are really performed and where bottlenecks are occurred.

Another future research is about representing the connections in the process model. As stated in the following chapters, “Construct AND/OR Connections in Discovered Process Model” operation handles the transitions of underlying activity (transaction) in successor and predecessor activity lists dually. Then connection score of each neighboring activity pairs are calculated and calculated connection score is compared with AND threshold to determine type of the connection. Afterwards, split or join type connections are visualized at control flow graph by linear connectors in a pair-wise fashion. In the case of multiple connections of an activity (transaction), it may be time-consuming and complex to combine these connections into a simpler formation.

For instance in Figure 6.1, there are three connectors combining three predecessors of “Travel Request Confirmation” activity (i.e. Connection number is calculated by dual

combination of successor or predecessor number, $C\binom{3}{2} = 3\ connections$) such that; an AND-join between “Advance Payment” and “Display Travel Request” activities, an OR-join between “Change Travel Request” and “Advance Payment” activities and an OR-join between “Change Travel Request” and “Display Travel Request” activities as stated in connection list, Figure 6.2. In a further version of FTCBPD methodology, connectors can be consolidated according to “dispersion property” in logic as follows:

$$\left[\left(Advance\ Payment \wedge Display\ Travel\ Request\right) \vee Change\ Travel\ Request\right]$$

Actually this reconstruction in representing connections may require a new block type like “places” in Petri-nets (i.e. “places” correspond to a condition that can be used as pre- or post-condition for tasks in Petri-nets) instead of linear connectors in control flow graph.

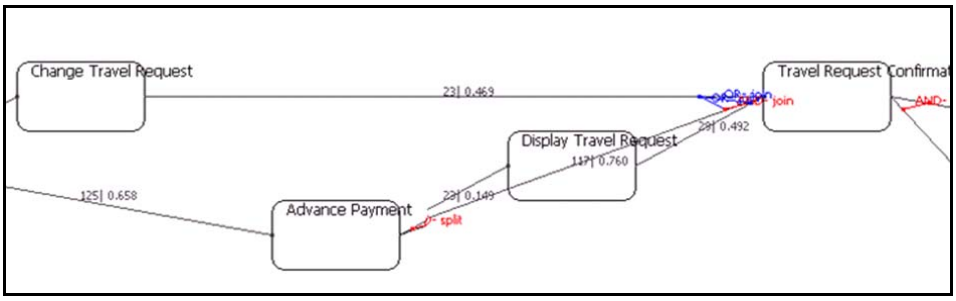
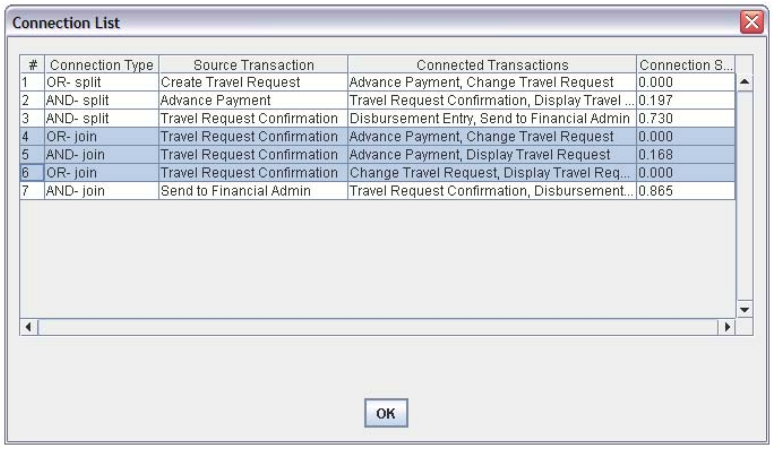


Figure 6.1 Multiple Connections among
Predecessors of “Travel Request Confirmation” Activity



#	Connection Type	Source Transaction	Connected Transactions	Connection S...
1	OR- split	Create Travel Request	Advance Payment, Change Travel Request	0.000
2	AND- split	Advance Payment	Travel Request Confirmation, Display Travel ...	0.197
3	AND- split	Travel Request Confirmation	Disbursement Entry, Send to Financial Admin	0.730
4	OR- join	Travel Request Confirmation	Advance Payment, Change Travel Request	0.000
5	AND- join	Travel Request Confirmation	Advance Payment, Display Travel Request	0.168
6	OR- join	Travel Request Confirmation	Change Travel Request, Display Travel Req...	0.000
7	AND- join	Send to Financial Admin	Travel Request Confirmation, Disbursement...	0.865

Figure 6.2 Connection List of Underlying Control Flow Graph

Although the challenge in process discovery is to derive a “good” process model that spans significant behaviors highlighted in the navigation traces in the form of log, protocol or trace data, the major limitation of process discovery performed by FTCBPD methodology is the total processing time, which is dominated by “Rearrange FROMTOCHART Table” operation. Calculating local moment for each generated activity sequence is just like traversing all local optimum points. However, finding out the activity sequence with the minimum moment value is not the whole story such that; optimum activity sequence is limited to sequential behavior and this one-dimensional activity sequence has to be upgraded to two-dimensional process model, which represents direct successive, successive and back-tracking type relations. Consequently future search can be concentrated on the heuristics handling permutative issues to improve FTCBPD methodology in processing time direction.

CHAPTER 7

CONCLUSION

In the increasingly globalized economy, enterprises face complex challenges that can require rapid and possibly continual transformation. As a result, more and more enterprises are focused on the strategical management of fundamental changes with respect to markets, products and services. Consequently the fundamental to enabling this transformation of an enterprise is the development of new information systems for comprehensive IT-supported business solutions that presumptively support and enhance enterprises in business process perspective.

Although contemporary information systems lead to significant efficiency gains in transformed business processes, they are developed in a generic way, which is driven by explicit process models (i.e. reference models). Hence a completely specified and well-conducted design phase is required to enact a transformed business process. Actually creating a design is a complicated, time-consuming phase and typically, there are discrepancies between the actual business process and the processes as perceived by the management. Therefore this study aims to propose a data analysis methodology, named "**From-to Chart Based Process Discovery**" (**FTCBPD**) to "reverse" this design-oriented approach.

Instead of starting with a process design, FTCBPD methodology attempts to distill interesting patterns from the navigation traces, which can be pulled up as a main data source for end-user behavior analysis, and translate these discovered patterns into process model with low-level domain knowledge. Structurally FTCBPD methodology integrates two components: from-to chart and process mining.

From-to chart, which is inherited from industrial engineering domain, is a fundamental analytical tool used in monitoring material handling routes between operations, machines,

departments or work centers on the production floor. In addition to its monitoring functionality, from-to chart can be turned into a decision support instrument in optimization of plant layout with respect to minimum material handling. The novelty of FTCBPD methodology in process discovery field resides in the fact that; this analytical tool is used as the baseline to monitor transitions among activities (transactions) occurs in business process and activity (transaction) sequence, which is rearranged in a straight line according to optimization capability, constitutes the backbone of discovered process model.

As the second component, process mining is a type of association rule mining algorithm, which is used to distill a structured process description from a set of real execution. In order to extract causality relations among activities (transactions), minimal information in event log has to contain time stamp for each event, which is used to sort all log entries in the order they take place in a process instance.

Unlike to design-oriented contemporary information systems, process mining is not biased or normative by perceptions. However if end-users alter the system doing things differently, the event log can still deviate from the actual work being done. Moreover process mining is not an instrument to (re-)design business process directly. The goal is to understand what is really going on in enactment phase and evaluate the conflictions with the reference model in diagnosis phase.

The major functionality of this process mining component in FTCBPD methodology is to upgrade one-dimensional optimum activity sequence, which assimilates “average” behaviors and patterns highlighted in event logs, into three process model representations:

i. From-to chart textually summarizes direct successive, successive and back-tracking relations among activities (transactions).
ii. Dependency/frequency graph is a form of finite state machine (FSM), which visualizes the transitions among activities (transactions) with dependency scores.
iii. Control flow graph enriches dependency/frequency graph by exhibiting type of the connections (i.e. AND-split, AND-join, OR-split and OR-join) between predecessors and successor pairs of each activities (transactions) in discovered process model.

As another cutting-edge feature of FTCBPD methodology, the evaluation metrics (i.e. minimum support threshold-MST, minimum confidence threshold-MCT and modified lift) used in process mining component can be emphasized. These metrics act as the major stick yard to control the level of robustness and complexity of the discovered model from large amounts of data; ignoring low-probability sequences, even if they are not the result of noise.

While formulating these metrics, it is attempted to familiarize them with original formations in association rule mining.

In order to test the performance of FTCBPD methodology, event logs of three distinct business processes with varying complexity level and amount of information (i.e. process instance number) are obtained. According to 396 test runs, the average percentage of transitions that were correctly handled in discovered process model is 96.936[1]. This level of correspondence implies that; FTCBPD methodology has a relatively good performance in discovering process model like logistic regression model with 94.3 completeness value stated in [14].

On the other hand, major limitation of FTCBPD methodology is the total processing time, which is highly dependent on the complexity level (i.e. number of activity types) of concentrated business process. The underlying reason of this shortcoming is the rearrangement operation with order |AT|!, whose running time is exaggerated by activity type number.

To conclude, this study aims to develop a data analysis methodology in the mature field of process discovery. Although several generic process mining tools (i.e. ProM, Little Thumb, EMIT and InWoLvE) are dedicated to this field, we attempt to handle process discovery issue by FTCBPD methodology, which integrates components from distinct disciplines.

[1] This value is the average of completeness obtained in travel management (97.368), credit card application (94.828) and repair (98.611) business processes.

REFERENCES

1. Weijters, T. A. J. M. M., & Aalst, W. M. P. v. d. (2003). Rediscovering Workflow Models from Event-Based Data Using Little Thumb. *Integrated Computer-Aided Engineering, 10*(2), 151-162.

2. Maruster, L., Weijters, T., & Bosch, A. v. d. (2006). A Rule-Based Approach for Process Discovery. *Data Mining and Knowledge Discovery, 13*(1), 67-87.

3. Cook, J. E., & Wolf, A. L. (1996). Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology (TOSEM), 7*(3), 215-249.

4. Greco, G., Guzzo, A., Luigi, P., & Sacca, D. (2004). *Mining Expressive Process Models by Clustering Workflow Traces.* Paper presented at the PAKDD 2004, Heidelberg.

5. Greco, G., Guzzo, A., Manco, G., & Sacca, D. (2007). Mining Unconnected Patterns in Workflows. *Information Systems, 32*(5), 685-712.

6. Chiravalloti, A. D., Greco, G., Guzzo, A., & Pontieri, L. (2006). An Information-Theoretic Framework for Process Structure and Data Mining. *Data Warehousing and Knowledge Discovery, 4081*, 248-259.

7. Aalst, W. M. P. v. d., Rubin, V., Dongen, B. F., E., K., & Gunther, C. W. (2007). Process Mining: A Two-Step Approach Using Transition Systems and Regions. *Business Process Management, 4714*, 375-383.

8. Recker, J., Mendling, J., Aalst, W. M. P. v. d., & Rosemann, M. (2006). Model-Driven Enterprise Systems Configuration. *Advance Information Systems Engineering, 4001*, 369-383.

9. Kindler, E., Rubin, V., & Schafer, W. (2006). *Process Mining and Petri Net Synthesis*. Paper presented at the Business Process Management Workshop.

10. R., A., D., G., & Leymann, F. (1998). *Mining Process Models from Workflow Logs*. Paper presented at the Sixth International Conference on Extending Database Technology.

11. Aalst, W. M. P. v. d., Weijters, T. A. J. M. M., & Maruster, L. (2004). Workflow Mining: Discovering Process Models from Event Logs. *Transaction on Knowledge and Data Engineering, 16*(9), 1128-1142.

12. Aalst, W. M. P. v. d., Dongen, B. F., Herbst, J., L., M., Schimm, G., & Weijters, T. A. J. M. M. (2003). Workflow Mining: A Survey of Issues and Approaches. *Data & Knowledge Engineering, 47*(2), 237-267.

13. Weijters, T. A. J. M. M., Aalst, W. M. P. v. d., & Medeiros, A. K. A. (2006). *Process Mining with the HeuristicMiner Algorithm.* Paper presented at the BETA Working Paper Series, WP 166,, Eindhoven University of Technology.

14. Maruster, L., Weijters, T. A. J. M. M., Aalst, W. M. P. v. d., & Bosch, A. v. d. (2002). Process Mining: Discovering Direct Successors in Process Logs. *Lecture Notes in Computer Science, 2534*, 364-373.

15. Weijters, T. A. J. M. M., & Aalst, W. M. P. v. d. (2003). Process Mining Discovering Workflow Models from Event-Based Data. *Integrated Computer-Aided Engineering, 10.*

16. Maruster, L., Aalst, W. M. P. v. d., Weijters, T. A. J. M. M., Bosch, A. v. d., & Daelemans, W. (2001). *Automated Discovery of Workflow Models for Hospital Data.* Paper presented at the Proceeding BNAIC-01.

17. Gunther, C. W., & Aalst, W. M. P. v. d. (2006). *Process Mining in Case Handling Systems*. Paper presented at the Multikonferenz Wirtschaftsinformatik 2006.

18. Gomez, J. M., Kassem, G., Rauntenstrauch, C., & Melato, M. (2003). Analysis of User's Behaviour in Very Large Business Application Systems with Methods of the Web Usage Mining- A Case Study on SAP® R/3®. *Advance in Web Intelligence, 2663*, 954-964.

19. Aalst, W. M. P. v. d., Reijers, H. A., Weijters, T. A. J. M. M., van, D. B. F., de, M. A. K. A., Song, M., et al. (2007). Business Process Mining: An Industrial Application. *Information Systems, 32*(5), 713-732.

20. Aalst, W. M. P. v. d., Gunther, C., Recker, J., & Reichert, M. (2006). *Using Process Mining to Analyze and Improve Process Flexibility*. Paper presented at the BPMDS'06.

21. Aalst, W. M. P. v. d. (2006). *Workshop Report: Process Mining, Monitoring Processes and Services.* Paper presented at the Dagstuhl Seminar Proceedings, Dagstuhl, Germany.

22. Zhao, Q., & Bhowmick, S. S. (2003). *Sequential Pattern Mining: a Survey*. Singapore: Nanyang Technological University.

23. Han, J., & Kamber, M. (2006). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

24. Witten, I. H., & Frank, E. (2000). *Data Mining Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco: Morgan Kaufmann.

25. Apple, J. M. (1972). *Material Handling Systems Design*. New York: The Ronald Press Company.

26. Meyers, F. E., & Stephens, M. P. (2005). *Manufacturing Facilities Design and Material Handling*. New Jersey: Pearson Prentice Hall.

27. Winston, W. L., & Goldberg, J. B. (2004). *Operations Research: Applications and Algorithms*: Thomson Delmar Learning.

28. Deitel, H. M., & Deitel, P. J. (2005). *Java How to Program*. New Jersey: Pearson Prentice Hall.

29. Sahni, S. (2000). *Data Structures, Algorithms and Applications in Java*. Singapore: Mc Graw Hill.

30. Dalbey, J. (2008). Pseucode Standard, from http://notes.ump.edu.my/fkee/Hazizulden/BEE1222/Handouts/Pseudocode%20Standard.pdf

31. Elmasri, R., & Navathe, S. B. (2003). *Fundamentals of Database Systems*. Boston: Pearson Addison Wesley.

32. Senge, P. M. (1990). *Beşinci Disiplin*. İstanbul: Yapı Kredi Yayınları.

33. Eindhoven University of Technology (2008). Process Mining, from http://prom.win.tue.nl/research/wiki/

34. The University of Waikato (2008). WEKA Software, from http://www.cs.waikato.ac.nz/ml/weka/

35. Francis, R. L., McGinnis L. F. & White, J. A. (1992). *Facility Layout and Location: An Analytical Approach*. New Jersey: Prentice Hall.

APPENDICES

APPENDIX A Data Dictionary

TRANSACTIONLOG table is used to bookkeep the event logs retrieved from concentrated information system. Event logs are the minimal information assumed to be present in FTCBPD methodology. Structure of TRANSACTIONLOG table is stated below:

Table A.1 Structure of TRANSACTIONLOG Table

Attribute	Data Type	Length	Primary Key	Description
transactionlogdate	DATE		Yes	Execution date of the underlying event.
transactionlogtime	TIME		Yes	Execution time of the underlying event.
uname	VARCHAR	12	Yes	Originator of the underlying event.
tcode	VARCHAR	20	Yes	Activity type of the underlying event.
caseID	BIGINT	10	Yes	Case identifier of the underlying event. Tuples are grouped by caseID to construct each process instance (cycle).

FROMTOCHART table is the basic material (environment) of FTCBPD methodology. Predecessor and successor relations between activities (transactions) captured in TRANSACTIONLOG table are marked into FROMTOCHART table. Then “significant” transitions are monitored by “Evaluate Tally Marks in FROMTOCHART Table” operation and direct successive, successive and back-tracking relations are determined due to these driven “significant” transitions. Except `tcode` attribute, attributes are added to FROMTOCHART dynamically not to deal with the activity types that do not take place in event logs. Structure of FROMTOCHART table is stated below:

Table A.2 Structure of FROMTOCHART Table

Attribute	Data Type	Length	Primary Key	Description
tcode	VARCHAR	20	Yes	Source activity of the related transition. This attribute resembles "from" side of the from-to chart.
successortcode	INT	4	No	Total number of transitions from source to destination activity. This attribute is dynamically added to FROMTOCHART table, if underlying 'successortcode' activity type exists in event logs.

INITIALFROMTOCHART table is used to hold the original tally mark values in FROMTOCHART table prior to “Evaluate Tally Marks in FROMTOCHART Table” operation. In other words, INITIALFROMTOCHART table is like an initial view of FROMTOCHART table. Tally marks held in INITIALFROMTOCHART table belong to the process model with the highest completeness in current process modeling run. Structurally INITIALFROMTOCHART table is similar to FROMTOCHART table as stated below:

Table A.3 Structure of INITIALFROMTOCHART Table

Attribute	Data Type	Length	Primary Key	Description
tcode	VARCHAR	20	Yes	Source activity of the related transition. This attribute resembles "from" side of the from-to chart.
successortcode	INT	4	No	Total number of transitions from source to destination activity. This attribute is dynamically added to INITIALFROMTOCHART table, if underlying 'successortcode' activity type exists in event logs.

TALLYMARK table is used to hold row and column totals of the activity types in FROMTOCHART table prior to “Evaluate Tally Marks in FROMTOCHART Table” operation. Row total of the underlying activity type is crucial in confidence calculation, while column total is used in modified lift calculation. TALLYMARK table is populated just after population of FROMTOCHART table not to lose the original tally mark values. Structure of TALLYMARK table is stated below:

Table A.4 Structure of TALLYMARK Table

Attribute	Data Type	Length	Primary Key	Description
tcode	VARCHAR	20	Yes	Activity type of underlying tuple.
rowtotaltallymark	INT	4	No	Row total of underlying activity type in FROMTOCHART before evaluation.
columntotaltallymark	INT	4	No	Column total of underlying activity type in FROMTOCHART before evaluation.

BUSINESSPROCESSES table is a domain table, which hold business process definitions that are defined in the system prior to process modeling. Structure of BUSINESSPROCESSES table is stated below:

Table A.5 Structure of BUSINESSPROCESSES Table

Attribute	Data Type	Length	Primary Key	Description
businessprocess	VARCHAR	25	Yes	Business process definition.

TCODES table is another domain table, that holds the conversion of each activity type code, which belongs to a predefined business process in BUSINESSPROCESSES table. Since activity types are tracked by “language-independent” transaction codes in TRANSACTIONLOG table, FTCBPD methodology is localized in a flexible manner by TCODES table. This conversion is used especially in dependency/frequency and control flow graphs. Structure of TCODES table is stated below:

Table A.6 Structure of TCODES Table

Attribute	Data Type	Length	Primary Key	Description
tcode	VARCHAR	20	Yes	Activity type of underlying tuple.
businessprocess	VARCHAR	25	Yes	Related business process. Foreign key of BUSINESSPROCESSES and TCODES relation.
ktext	VARCHAR	59	No	Definition of activity type.

APPENDIX B Entity/Relationship (ER) Diagram

Beside the classical representation of entity/relationship (ER) diagram, ER diagram of thesis database is in a relationless form. In fact, whenever an attribute of one entity type (i.e. table) refers to another entity type (i.e. table), some relationships exist. On the other hand in thesis database, a tuple in a table does not directly refer to another tuple in a distinct table except the relationship between TCODES and BUSINESSPROCESSES tables.

Operations in FTCBPD methodology (similar to aggregate functions added to SQL-99 for more statistical calculations) convert “relatively raw” data into a more value-added formation and then populate this “intermediate” information into a distinct table. For instance, “Populate FROMTOCHART Table” operation constructs transaction streams by grouping tuples in TRANSACTIONLOG table due to caseID attribute and ordering these grouped tuples by time stamp (i.e. transactionlogdate and transactionlogtime attributes). Then each transition that exists in transaction streams are inserted into FROMTOCHART table. Consequently this value-chain-based approach implies three information levels as represented in ER diagram.

On the other hand, a relationship exists between TCODES and BUSINESSPROCESSES tables to hold the assignment of transaction code to the business process.

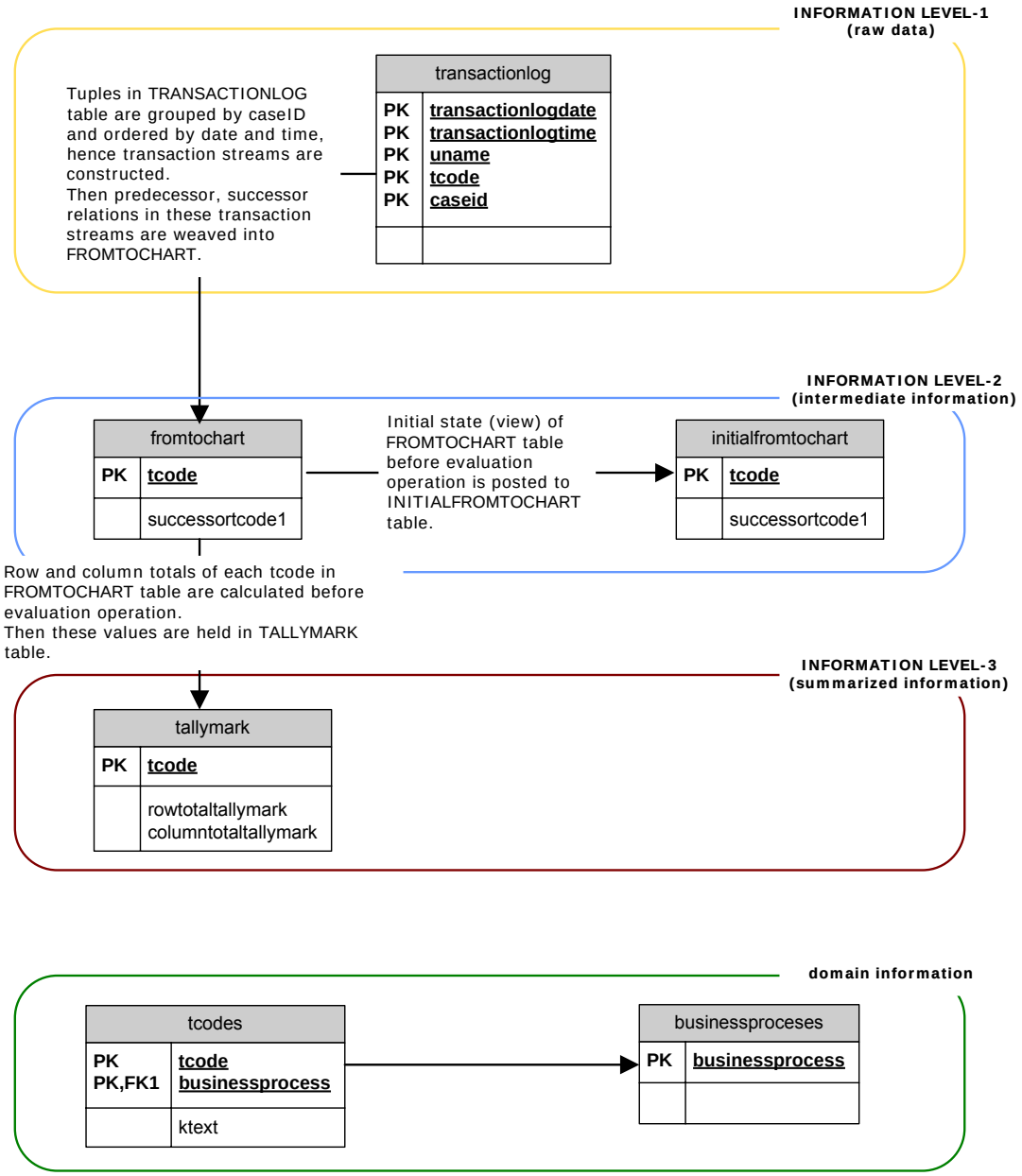


Figure B.1 ER Diagram of thesis Database

APPENDIX C An Example Run at ProMiner

Step 1: Importing Event Logs

First of all, source text file has to be constituted by order of transaction date, transaction time, originator, activity type and case identifier parameters. Additionally these parameters have to be separated by a semicolon (";") in the source file. A view of source file example is represented in Figure C.1.

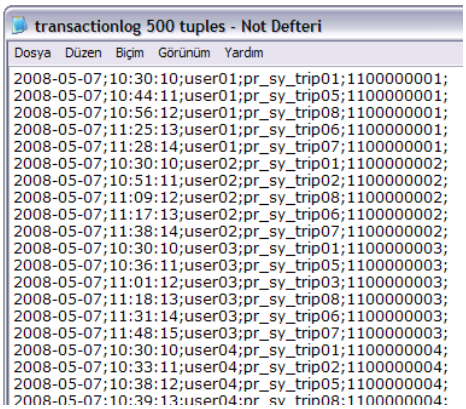


Figure C.1 A view of source file example

In order to import event logs, select "Utilities> Import Tables> TransactionLog Table" short cut from the menu bar of ProMiner as represented in Figure C.2.

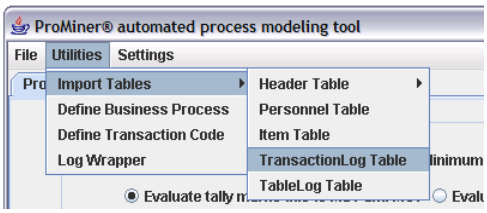


Figure C.2 Import TRANSACTIONLOG table Application

After clicking on "TransactionLog Table" menu item, a file chooser is displayed. End-user has to select the appropriate source file at this file chooser. By selecting the file, ProMiner replaces the existing records in TRANSACTIONLOG table with the prepared records in

source file. Thus ProMiner only takes the imported records into consideration in process model.

In this example run, 500 process instances worth of 2379 records are imported to TRANSACTIONLOG table. Additionally the file path of the underlying source file is reflected to “Source File” view at the selection screen.

Figure C.3 “Source File” View at Selection Screen

Step 2: Defining Business Process

If it is the first run of the discovered business process (domain), end-user has to introduce the business process to ProMiner. In order to define business process, select “Utilities> Define Business Process” short cut from the menu bar of ProMiner as represented in Figure C.4.

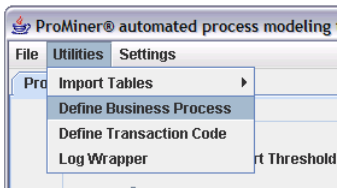


Figure C.4 Define Transaction Code Application

After clicking on “Define Business Process” menu item, a “Define Business Process” pop-up is displayed. At this pop-up, business process definition has to be entered. If business process definition exceeds the anticipated limit, an error message is displayed. Thus application is not completed successfully.

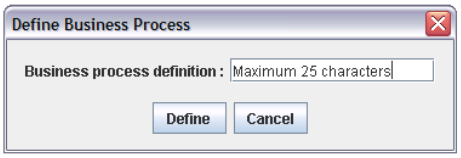


Figure C.5 Define Business Process Pop-up

Step 3: Defining Transaction Codes of Discovered Business Process

If it is the first run of the discovered business process (domain), end-user has to introduce the activity (transaction) codes domain information to ProMiner. In order to define transaction codes, select “Utilities> Define Transaction Code” short cut from the menu bar of ProMiner as represented in Figure C.6.

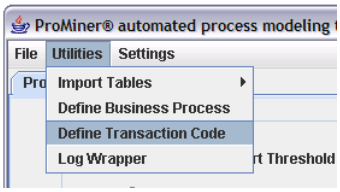


Figure C.6 Define Transaction Code Application

After clicking on “Define Transaction Code” menu item, a “Define Transaction Code” pop-up is displayed. At this pop-up, transaction code and transaction code definition data has to be entered and related business process has to be chosen from list of available business processes in the combo box. If one of the entered values exceeds the anticipated limit, an error message is displayed. Thus application is not completed successfully.

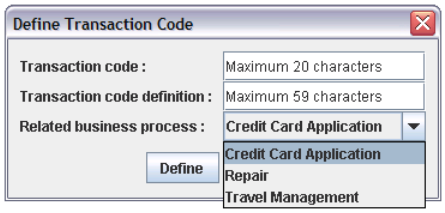


Figure C.7 Define Transaction Code Pop-up

Step 4: Setting the Parameters in Selection Screen

Prior to executing process modeling, end-user has to set execution-related parameters in selection screen. Firstly, evaluation metrics and their threshold values are set by the radio buttons on “Threshold Values” view. Actually, threshold setting part is only activated in “Evaluate tally marks due to MST and MCT” evaluation metric selection. “Predetermine arc traffic in process modeling” check-box is marked in order to declare an upper-bound for transition number in discovered process model.

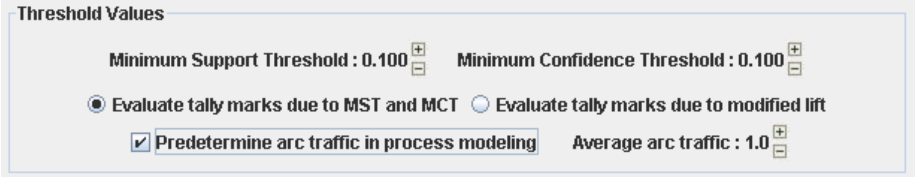


Figure C.8 “Threshold Values” View at Selection Screen

Afterwards, concentrated business process is selected from the list of predefined business processes in the system. Due to the business process selection, activity (transaction) list at initiator transaction (activity) combo box is revised. End-user proposes the first transaction at discovered process model.

In order to lessen back-tracking transitions in discovered process model, end-user can set higher penalty points to back-tracking type transitions by buttons at “Back-tracking penalty point” field. Additionally one-step closed loops can be detected and eliminated at discovered process model by marking “Eliminate one-step closed loops” check-box.

As the last parameter on “Process Modeling Factors” frame, AND/OR connection can be denoted on control flow graph due to AND threshold in the case of marking “Detect AND/OR connections” check-box. If this check-box is not marked, end-user cannot visualize control-flow graph.

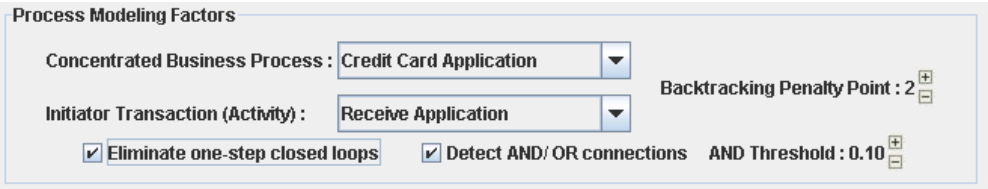


Figure C.9 “Process Modeling Factors” View at Selection Screen

At “Verification Parameters” view, end-user proposes verification method by radio buttons. In the case of cross-validation selection, fold number is set by buttons. For holdout method, the fold number is set to 3, because of threefold cross-validation.

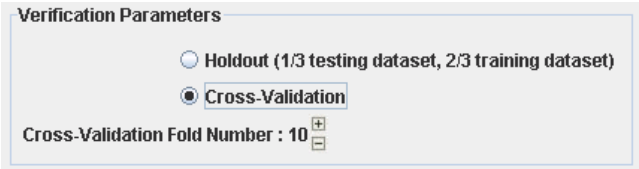


Figure C.10 Verification Parameters" View at Selection Screen

In the end of setting operation, end-user clicks on  at the bottom of the selection screen to start up the process modeling. In this example run, parameters are set as presented in Figure C.11

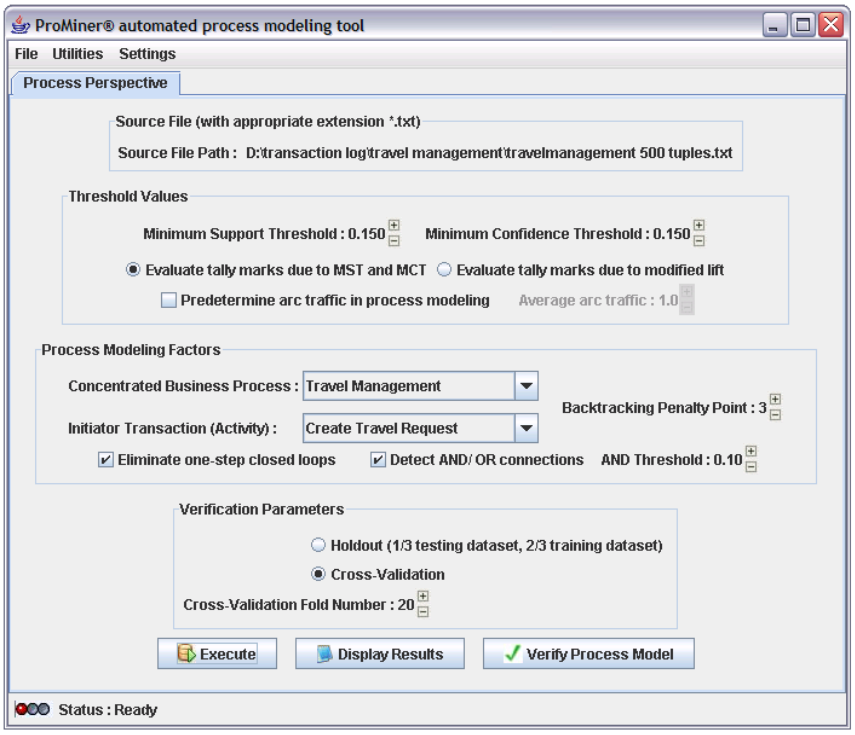


Figure C.11 Selection Screen View of ProMiner

Step 5: Executing Process Modeling by FTCBPD Methodology

Step 5.1: Creating and Populating FROMTOCHART Table

Firstly, FROMTOCHART table is created in thesis database by a dynamic manner such that; activity types existing in TRANSACTIONLOG table are distinctly selected. Then new attributes are added to CREATE TABLE SQL statement in the name of selected activity

types additional to tcode attribute. In this aspect, WYSIWYC (what you see is what you create) philosophy is implemented in FROMTOCHART table creation.

According to cross-validation with 20 folds verification method, 475 process instances in training dataset is grouped by case identifier and then lined up by transaction date and time in ascending order. Thus transaction streams are constructed. Afterwards, transitions in constructed transaction streams are inserted into FROMTOCHART table. In the end of populating FROMTOCHART table, row and column totals of activity types are calculated due to original (un-evaluated) tally marks and inserted into TALLYMARK table. Initial states of FROMTOCHART and TALLYMARK tables are presented in Table C.1 and Table C.2.

Table C.1 Initial State of FROMTOCHART Table

tcode	pr_sy_trip01	pr_sy_trip05	pr_sy_trip08	pr_sy_trip02	pr_sy_trip03	pr_sy_trip06	pr_sy_trip07
pr_sy_trip01	0	288	104	45	38	0	0
pr_sy_trip05	0	0	284	12	37	0	2
pr_sy_trip08	0	0	1	4	22	318	131
pr_sy_trip02	0	28	28	0	5	0	0
pr_sy_trip03	0	19	60	0	0	13	28
pr_sy_trip06	0	0	0	0	18	0	312
pr_sy_trip07	0	0	0	0	0	0	0

Table C.2 Initial State of TALLYMARK Table

tcode	rowtotaltallymak	columntotaltallymark
pr_sy_trip01	475	0
pr_sy_trip05	335	335
pr_sy_trip08	476	477
pr_sy_trip02	61	61
pr_sy_trip03	120	120
pr_sy_trip06	330	331
pr_sy_trip07	0	473

Step 4.2: Evaluating Tally Marks in FROMTOCHART Table due to MST and MCT

Basic criteria in this operation can be formulated as follows:

```
IF (accumulated tally mark < total process instance (475) × MST (0.150)) OR
   (accumulated tally mark < rowtotaltallymark of activity type j in TALLYMARK Table × MCT (0.150)) THEN
        Reset tally mark to zero
ELSE
        Do nothing
ENDIF
```

Firstly tally marks that are smaller than 71.25 (calculated support threshold) are reset to zero globally as follows:

Table C.3 Evaluating Tally Marks due to Calculated Support Threshold

tcode	pr_sy_trip01	pr_sy_trip05	pr_sy_trip08	pr_sy_trip02	pr_sy_trip03	pr_sy_trip06	pr_sy_trip07	Calculated Support Threshold
pr_sy_trip01	0	288	104	45	38	0	0	71.25
pr_sy_trip05	0	0	284	12	37	0	2	71.25
pr_sy_trip08	0	0	1	4	22	318	131	71.25
pr_sy_trip02	0	28	28	0	5	0	0	71.25
pr_sy_trip03	0	19	60	0	0	13	28	71.25
pr_sy_trip06	0	0	0	0	18	0	312	71.25
pr_sy_trip07	0	0	0	0	0	0	0	71.25

Afterwards, tally marks that are smaller than local calculated confidence threshold are reset to zero. Theoretically calculated support threshold, global metric of FTCBPD methodology, acts as a upper-bound for tally marks in FROMTOCHART table, since total process instance is the maximum row total for unrepeated activity types. Therefore resetting operations are mostly performed due to this metric. Final state of FROMTOCHART table is presented in Table C.4.

Table C.4 Evaluating Tally Marks due to Calculated Confidence Threshold

tcode	pr_sy_trip01	pr_sy_trip05	pr_sy_trip08	pr_sy_trip02	pr_sy_trip03	pr_sy_trip06	pr_sy_trip07	Calculated Confidence Threshold
pr_sy_trip01	0	288	104	0	0	0	0	71.25
pr_sy_trip05	0	0	284	0	0	0	2	50.25
pr_sy_trip08	0	0	0	0	0	318	131	71.4
pr_sy_trip02	0	0	0	0	0	0	0	9.15
pr_sy_trip03	0	0	0	0	0	0	0	18
pr_sy_trip06	0	0	0	0	0	0	312	49.5
pr_sy_trip07	0	0	0	0	0	0	0	0

Step 4.3: Rearranging FROMTOCHART Table

The core of this operation is the Permutation object, which generates activity sequences in the same number of activity types in FROMTOCHART table. If the initial element of generated activity sequence is different from the initiator transaction code set in Step 2, this outcome is discarded. Otherwise moment of this activity sequence is calculated according to

this objective function as stated in “Proposed Method” chapter:

$$Min\ Z = \sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij} \times \left| j - i \right| \times p$$

For instance, the first generated activity sequence is {pr_sy_trip01, pr_sy_trip05, pr_sy_trip08, pr_sy_trip02, pr_sy_trip03, pr_sy_trip06, pr_sy_trip07}.
Thus calculated moment for this activity sequence is as follows:

$$moment_{pr\_sy\_trip01} = 1 \times 288 + 2 \times 104$$
$$moment_{pr\_sy\_trip05} = 1 \times 284$$
$$moment_{pr\_sy\_trip08} = 3 \times 318 + 4 \times 131$$
$$moment_{pr\_sy\_trip06} = 1 \times 312$$
$$Calculated\ \ moment = 2570$$

If calculated moment value of one of the consequently generated activity sequences is smaller than the local optimum moment (i.e. minimum moment value during rearrangement operation), then local optimum activity sequence is revised. Since activities pr_sy_trip02, and pr_sy_trip03 increments the moment arms of activities pr_sy_trip06 and pr_sy_trip07, these activities have to be shifted to the end of activity sequence. Hence global optimum activity sequence is {pr_sy_trip01, pr_sy_trip05, pr_sy_trip08, pr_sy_trip02, pr_sy_trip03, pr_sy_trip06, pr_sy_trip07} with 1672 moment value.

Step 4.4: Constructing Process Model

After rearrangement of FROMTOCHART table, if tally mark of an element is greater than zero, this element indicates that the underlying transition represented by this element is worth to be taken into consideration in process model. Therefore from-to chart, which is a textual visualization form of discovered process model, is constructed as follows:

Table C.5 From-to Chart

Transaction Code	Successive Transaction Codes
pr_sy_trip01	pr_sy_trip05, pr_sy_trip08
pr_sy_trip05	pr_sy_trip08
pr_sy_trip08	pr_sy_trip06, pr_sy_trip07
pr_sy_trip06	pr_sy_trip07
pr_sy_trip07	
pr_sy_trip02	
pr_sy_trip03	

Step 4.5: Eliminating One-Step Closed Loops

Since there are not any one-step closed loops (i.e. one-step closed loops visually corresponds to elements with non-zero tally mark value that are in mirror-image position according to diagonal), this operation is skipped in this example.

Step 4.6: Constructing AND/OR Connections in Discovered Process Model

In this operation, connection score for each predecessor and successor pairs of every activity type in discovered process model is calculated. Connections, whose score is less than AND threshold set in Step 2, is labeled as OR-connection, otherwise connections are labeled as AND-threshold. Connection scores are calculated according this formula as stated in “Proposed Method” chapter:

$$A \Rightarrow B \wedge C_{SPLIT} = \left(\frac{|B > C| + |C > B|}{|A > B| + |A > C|} \right)$$

$$A \Rightarrow B \wedge C_{JOIN} = \left(\frac{|B > C| + |C > B|}{|B > A| + |C > A|} \right)$$

$$connection\ \ score_{pr\_sy\_trip01 \Rightarrow pr\_sy\_trip05 \wedge pr\_sy\_trip08\ \ SPLIT} = \frac{284+0}{288+104} = 0.723 > 0.1\ then\ \ AND-split$$

$$connection\ \ score_{pr\_sy\_trip08 \Rightarrow pr\_sy\_trip06 \wedge pr\_sy\_trip07\ \ SPLIT} = \frac{312+0}{318+131} = 0.693 > 0.1\ then\ \ AND-split$$

$$connection\ \ score_{pr\_sy\_trip08 \Rightarrow pr\_sy\_trip01 \wedge pr\_sy\_trip05\ \ JOIN} = \frac{288+0}{104+284} = 0.740 > 0.1\ then\ \ AND-join$$

$$connection\ \ score_{pr\_sy\_trip07 \Rightarrow pr\_sy\_trip06 \wedge pr\_sy\_trip08\ \ SPLIT} = \frac{318+0}{312+131} = 0.716 > 0.1\ then\ \ AND-split$$

As a result, there occur four AND-type connections in control flow graph of discovered process model.

In the end of process modeling, two extra buttons are enabled next to “Execute” button. By clicking on 

button, end-user can display the process model in distinct four forms that are ranged in complexity order as follows:

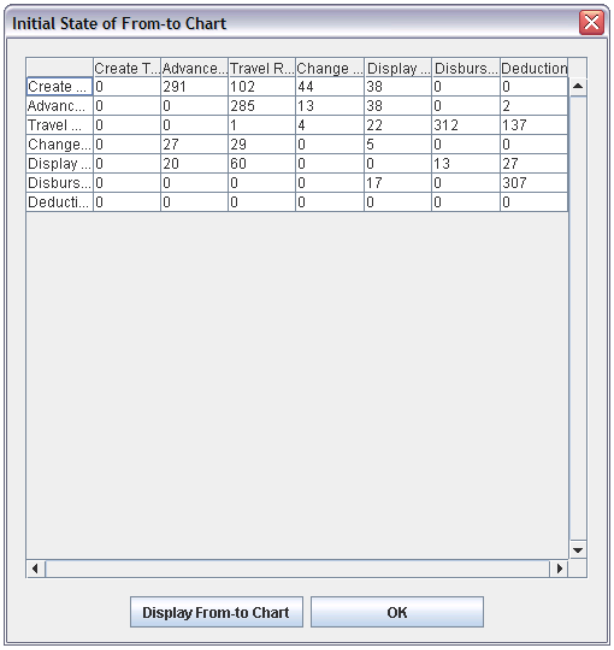


Figure C.12 Initial State of From-to Chart

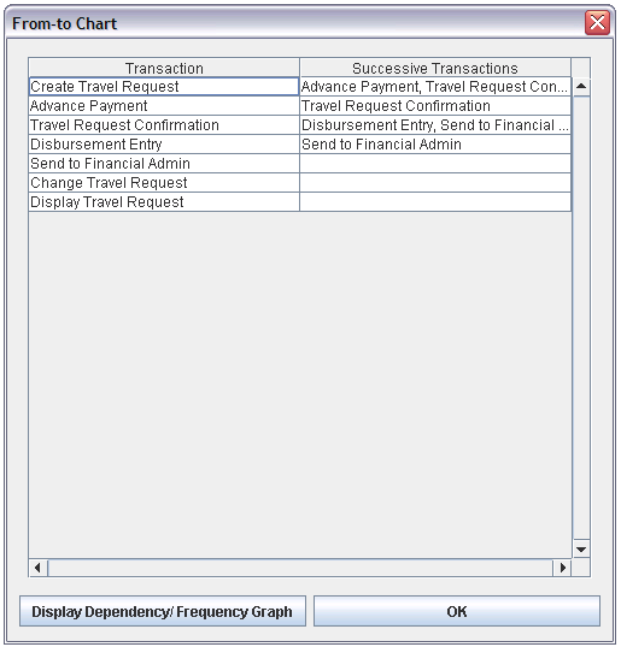


Figure C.13 Discovered Process Model in From-to Chart Form

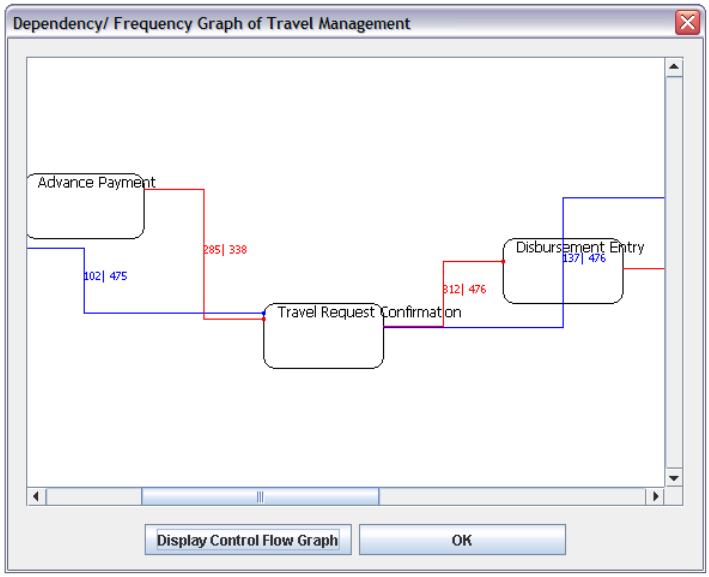


Figure C.14 Discovered Process Model in Dependency/Frequency Graph Form

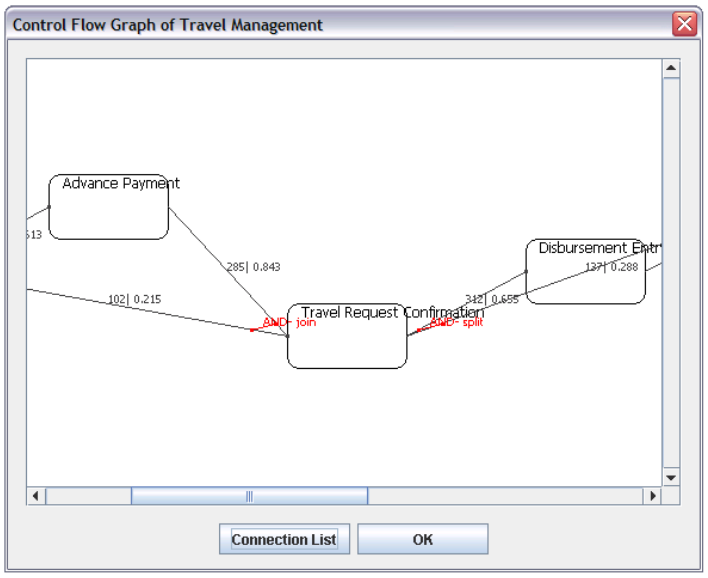


Figure C.15 Discovered Process Model in Control Flow Graph Form

By clicking on  on previous frame, connections that take place on control flow graph can be listed as follows:

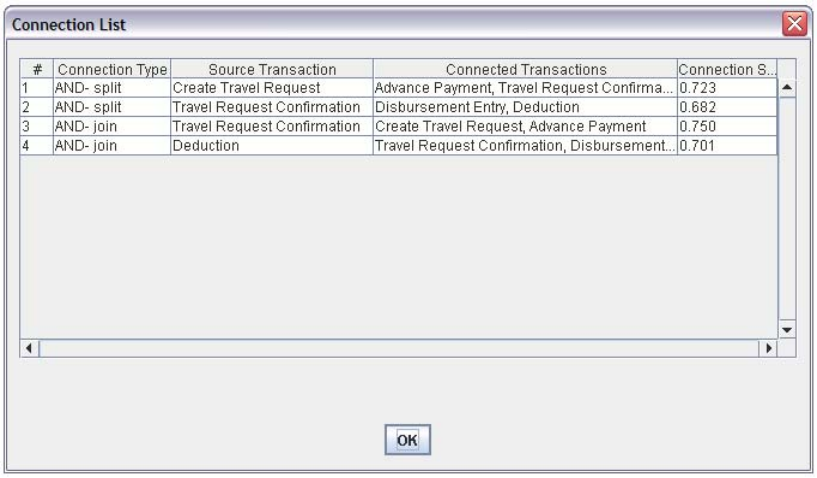


Figure C.16 Connection List Frame

For more details about the outputs of ProMiner, please read “From-to Chart Based Process Discovery (FTCBPD)” part in “Proposed Method” chapter.

Step 6: Verifying Discovered Process Model

According to 25 process instances (i.e. one-twentieth of 500 process instances due to cross-validation with 20 folds verification method) 83 transitions in test dataset is handled correctly in discovered process model, while 6 transitions are missed. Thus completeness percentage of discovered process model is 93.258. Additionally end-user can learn more about other performance values (i.e. soundness, arc traffic, average transition length) by clicking on  button on the selection screen.

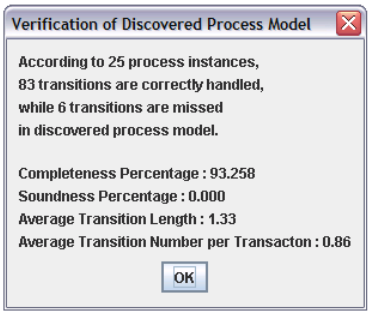


Figure C.17 Performance Values of Discovered Process Model

Step 7: Displaying Transaction Streams

In the end of process modeling, ProMiner builds up all transaction streams (i.e. a form that is independent of originator and time stamp parameters) of all process instances (process instances in training and testing datasets) and writes built transaction streams under the program folder as follows:

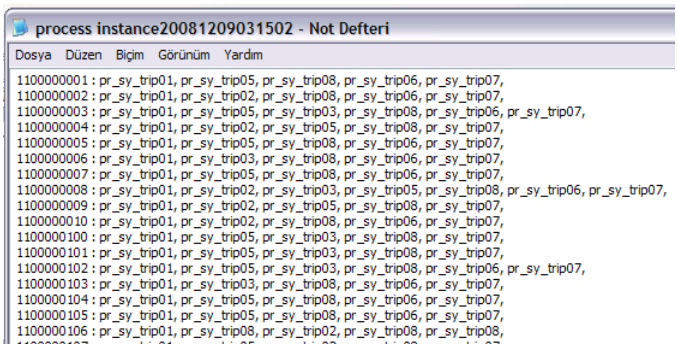


Figure C.18 Transaction Streams File

APPENDIX D Linear Regression for Processing Time

As starting point, a linear equation for unit processing time, yi, is parametrized according to operational-level complexity of FTCBPD methodology.

$$y_i = |AT|! \times c_1 + |AT|^3 \times |PI| \times c_2 + |AT|^2 \times c_3 + |AT| \times |PI| \times c_3 + |AT| \times c_4 + c_5$$

Afterwards a linear regression is performed in WEKA to determine the coefficients, cjs. Run information of the linear regression is stated below:

```
=== Run information ===

Scheme:         weka.classifiers.functions.LinearRegression -S 0 -R
1.0E-8
Relation:     unitProcessTime
Instances:    396
Attributes:   6
              |AT|!
              |AT|^3*|PI|
              |AT|^2
              |AT|*|PI|
              |AT|
              unitProcessTime
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Linear Regression Model

unitProcessTime =

      0.0001 * |AT|! +
      0      * |AT|^3*|PI| +
     -0.6062

Time taken to build model: 0.07 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9971
Mean absolute error                      0.5308
Root mean squared error                  1.1023
Relative absolute error                  3.9817 %
Root relative squared error              7.6486 %
Total Number of Instances                396
```

APPENDIX E Linear Regression for Size

```
=== Run information ===

Scheme:         weka.classifiers.functions.LinearRegression -S 0 -R
1.0E-8
Relation:     size
Instances:    18
Attributes:   2
              |AT|*|PI|
              size
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Linear Regression Model

size =

      0.0715 * |AT|*|PI|+
     -4.4412

Time taken to build model: 0.01 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9233
Mean absolute error                     34.3501
Root mean squared error                 41.3324
Relative absolute error                 40.0428 %
Root relative squared error             38.4517 %
Total Number of Instances               18
```

APPENDIX F Dependent Means T-Test for Two Versions of Modified Lift

In this dependent means t-test, it is aimed to interpret the effect of additional evaluation procedure (i.e. multiplying negatively correlated tally mark by minus one) on discovered process model. As the measure of this effect, average length of transition is used. As a test bed, 36 observations of modified and classical lift evaluation metrics are compared in this test.

Calculated t-value is used to decide between two statistical hypotheses:

- H_0: There is not a clear distinction between two versions of mofied lift evaluation metrics ($\mu_{\text{MODIFIED LIFT TWO-BRANCH}} = \mu_{\text{MODIFIED LIFT TWO-BRANCH}}$).
- H_A: The population evaluated due to modified lift three-branch use has a higher average transition length than modified lift two-branch use ($\mu_{\text{MODIFIED LIFT THREE-BRANCH}} > \mu_{\text{MODIFIED LIFT TWO-BRANCH}}$).

36 observations varying due to the amount of information in event log (i.e. process instance number) and verification method fold number are as follows:

Table F.1 Observations

RunID	Run Characteristics	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use	D
1	Holdout backtracking pp=2 100 process instances	2.09	1.55	0.54
2	CV with 10-fold backtracking pp=2 100 process instances	1.44	1.44	0.00
3	CV with 20-fold backtracking pp=2 100 process instances	1.60	1.60	0.00
4	CV with 30-fold backtracking pp=2 100 process instances	1.44	1.29	0.15
5	CV with 40-fold backtracking pp=2 100 process instances	1.44	1.44	0.00
6	CV with 50-fold backtracking pp=2 100 process instances	1.44	1.44	0.00
7	Holdout backtracking pp=2 200 process instances	1.92	1.54	0.38
8	CV with 10-fold backtracking pp=2 200 process instances	1.54	1.62	-0.08

RunID	Run Characteristics	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use	D
13	Holdout backtracking pp=2 300 process instances	2.00	1.54	0.46
14	CV with 10-fold backtracking pp=2 300 process instances	1.93	1.64	0.29
15	CV with 20-fold backtracking pp=2 300 process instances	1.93	1.64	0.29
16	CV with 30-fold backtracking pp=2 300 process instances	1.93	1.64	0.29
17	CV with 40-fold backtracking pp=2 300 process instances	1.93	1.64	0.29
18	CV with 50-fold backtracking pp=2 300 process instances	1.93	1.64	0.29
19	Holdout backtracking pp=2 400 process instances	1.62	1.62	0.00
20	CV with 10-fold backtracking pp=2 400 process instances	1.62	1.62	0.00

RunID	Run Characteristics	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use	D
25	Holdout backtracking pp=2 500 process instances	1.71	1.71	0.00
26	CV with 10-fold backtracking pp=2 500 process instances	1.62	1.62	0.00
27	CV with 20-fold backtracking pp=2 500 process instances	1.62	1.62	0.00
28	CV with 30-fold backtracking pp=2 500 process instances	1.62	1.62	0.00
29	CV with 40-fold backtracking pp=2 500 process instances	1.62	1.62	0.00
30	CV with 50-fold backtracking pp=2 500 process instances	1.62	1.62	0.00
31	Holdout backtracking pp=2 600 process instances	1.73	1.64	0.09
32	CV with 10-fold backtracking pp=2 600 process instances	1.62	1.62	0.00

RunID	Run Characteristics	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use	D
9	CV with 20-fold backtracking pp=2 200 process instances	1.54	1.62	-0.08
10	CV with 30-fold backtracking pp=2 200 process instances	1.54	1.62	-0.08
11	CV with 40-fold backtracking pp=2 200 process instances	1.54	1.62	-0.08
12	CV with 50-fold backtracking pp=2 200 process instances	1.54	1.62	-0.08

RunID	Run Characteristics	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use	D
21	CV with 20-fold backtracking pp=2 400 process instances	1.62	1.62	0.00
22	CV with 30-fold backtracking pp=2 400 process instances	1.62	1.62	0.00
23	CV with 40-fold backtracking pp=2 400 process instances	1.62	1.62	0.00
24	CV with 50-fold backtracking pp=2 400 process instances	1.62	1.62	0.00

RunID	Run Characteristics	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use	D
33	CV with 20-fold backtracking pp=2 600 process instances	1.62	1.62	0.00
34	CV with 30-fold backtracking pp=2 600 process instances	1.62	1.62	0.00
35	CV with 40-fold backtracking pp=2 600 process instances	1.62	1.62	0.00
36	CV with 50-fold backtracking pp=2 600 process instances	1.62	1.62	0.00

t-value Calculation:

$$t = \frac{\sum D}{\sqrt{\frac{n \times \sum D^2 - \left(\sum D\right)^2}{n-1}}}$$

The parameters used in the formula are:

D is the difference for a single observation.

n is the total observation number.

$$t = \frac{2.67}{\sqrt{\frac{36 \times 1.1307 - 7.1289}{35}}} = \frac{2.67}{\sqrt{\frac{33.5763}{35}}} = \frac{2.67}{0.9795} \cong 2.7260$$

Dependent means t-test is summarized as follows:

Table F.2 Dependent Means T-Test Summary

t-Test: Paired Two Sample for Means

	Modified Lift 3-Branch Use	Modified Lift 2-Branch Use
Mean	1.668888889	1.594722222
Variance	0.029484444	0.006048492
Observations	36	36
Pearson Correlation	0.332668196	
Hypothesized Mean Difference	0	
df	35	
t Stat	2.726018917	
P(T<=t) one-tail	0.00497253	
t Critical one-tail	1.68957244	
P(T<=t) two-tail	0.00994506	
t Critical two-tail	2.030107915	

Since $t_{0.05,\ 35}$ (2.0301) is smaller than calculated t-value (2.7260), null hypothesis, H_0, is

rejected. Thus it is concluded that, the population evaluated due to modified lift three-branch use has a higher average transition length than modified lift two-branch use ($\mu_{\text{MODIFIED LIFT THREE-BRANCH}} > \mu_{\text{MODIFIED LIFT TWO-BRANCH}}$). In other words, additional evaluation procedure (i.e. multiplying negatively correlated tally mark by minus one) significantly increases average (and total) length of the transitions in the discovered process model.

APPENDIX G Dependent Means T-Test for Gradually Increasing MST

In this dependent means t-test, it is aimed to interpret basic discrepancies in completeness key performance metric between minimum support threshold (MST) and modified lift evaluation metrics. Therefore 30 observations evaluated by gradually increasing MST and modified lift are compared in this test.

Calculated t-value is used to decide between two statistical hypotheses:

1. H_0: There is not a clear distinction between gradually increasing MST and modified lift with respect to completeness ($\mu_{\text{GRADUALLY INCREASING MST}} = \mu_{\text{MODIFIED LIFT}}$).
2. H_A: The population evaluated due to modified lift has a higher completeness ratio than slightly increasing MST ($\mu_{\text{MODIFIED LIFT}} > \mu_{\text{GRADUALLY INCREASING MST}}$).

30 observations varying due to the amount of information in event log (i.e. number of process instances) are as follows:

Table G.1 Observations

RunID	Run Characteristics	Gradually Increasing MST	Modified Lift	D
1	100 PIN CV with 10-fold backtracking pp=2	97.368	90.476	6.892
2	100 PIN CV with 10-fold backtracking pp=2	95.000	90.476	4.524
3	100 PIN CV with 10-fold backtracking pp=2	92.500	90.476	2.024
4	100 PIN CV with 10-fold backtracking pp=2	92.500	90.476	2.024
5	100 PIN CV with 10-fold backtracking pp=2	85.000	90.476	-5.476
6	200 PIN CV with 10-fold backtracking pp=2	93.902	92.593	1.309
7	200 PIN CV with 10-fold backtracking pp=2	85.185	92.593	-7.408
8	200 PIN CV with 10-fold backtracking pp=2	83.951	92.593	-8.642
9	200 PIN CV with 10-fold backtracking pp=2	77.632	92.593	-14.961
10	200 PIN CV with 10-fold backtracking pp=2	76.543	92.593	-16.050
11	300 PIN CV with 10-fold backtracking pp=2	92.373	96.610	-4.237
12	300 PIN CV with 10-fold backtracking pp=2	88.393	96.610	-8.217
13	300 PIN CV with 10-fold backtracking pp=2	86.441	96.610	-10.169
14	300 PIN CV with 10-fold backtracking pp=2	84.746	96.610	-11.864
15	300 PIN CV with 10-fold backtracking pp=2	81.356	96.610	-15.254
16	400 PIN CV with 10-fold backtracking pp=2	89.189	89.865	-0.676
17	400 PIN CV with 10-fold backtracking pp=2	89.189	89.865	-0.676
18	400 PIN CV with 10-fold backtracking pp=2	85.135	89.865	-4.730
19	400 PIN CV with 10-fold backtracking pp=2	85.135	89.865	-4.730
20	400 PIN CV with 10-fold backtracking pp=2	80.405	89.865	-9.460
21	500 PIN CV with 10-fold backtracking pp=2	88.235	89.189	-0.954
22	500 PIN CV with 10-fold backtracking pp=2	88.235	89.189	-0.954
23	500 PIN CV with 10-fold backtracking pp=2	86.096	89.189	-3.093
24	500 PIN CV with 10-fold backtracking pp=2	86.096	89.189	-3.093
25	500 PIN CV with 10-fold backtracking pp=2	86.096	89.189	-3.093
26	600 PIN CV with 10-fold backtracking pp=2	89.732	91.518	-1.786
27	600 PIN CV with 10-fold backtracking pp=2	84.753	91.518	-6.765
28	600 PIN CV with 10-fold backtracking pp=2	84.753	91.518	-6.765
29	600 PIN CV with 10-fold backtracking pp=2	84.753	91.518	-6.765
30	600 PIN CV with 10-fold backtracking pp=2	84.753	91.518	-6.765

t-value Calculation:

$$t = \frac{\sum D}{\sqrt{\frac{n \times \sum D^2 - \left(\sum D\right)^2}{n-1}}}$$

The parameters used in the formula are:

D is the difference for a single observation.

n is the total observation number.

$$t = \frac{145.810}{\sqrt{\frac{30 \times 1633.097 - 21260.556}{29}}} = \frac{145.810}{\sqrt{\frac{27732.359}{29}}} = \frac{145.810}{30.924} \cong 4.7151$$

Dependent means t-test is summarized as follows:

Table G.2 Dependent Means T-Test Summary

t-Test: Paired Two Sample for Means

	Modified Lift	**Gradually Increasing MST**
Mean	91.7085	86.84816667
Variance	6.225648534	22.54805214
Observations	30	30
Pearson Correlation	-0.130931841	
Hypothesized Mean Difference	0	
df	29	
t Stat	4.715121687	
P(T<=t) one-tail	2.80E-05	
t Critical one-tail	1.699126996	
P(T<=t) two-tail	5.60E-05	
t Critical two-tail	2.045229611	

Since $t_{0.05,\ 29}$ (2.0452) is smaller than calculated t-value (4.7151), null hypothesis, H_0, is rejected. Thus it is concluded that, completeness obtained by modified lift is better than the performance of gradually increasing MST in accuracy perspective ($\mu_{\text{MODIFIED LIFT}} > \mu_{\text{GRADUALLY INCREASING MST}}$).

APPENDIX H Dependent Means T-Test for Gradually Increasing MCT

In this dependent means t-test, it is aimed to interpret basic discrepancies in completeness key performance metric between minimum confidence threshold (MCT) and modified lift evaluation metrics. Therefore 30 observations evaluated by gradually increasing MCT and modified lift are compared in this test.

Calculated t-value is used to decide between two statistical hypotheses:

3. H_0: There is not a clear distinction between gradually increasing MCT and modified lift with respect to completeness ($\mu_{\text{GRADUALLY INCREASING MCT}} = \mu_{\text{MODIFIED LIFT}}$).
4. H_A: The population evaluated due to modified lift has a higher completeness ratio than slightly increasing MCT ($\mu_{\text{MODIFIED LIFT}} > \mu_{\text{GRADUALLY INCREASING MCT}}$).

30 observations varying due to the amount of information in event log (i.e. process instance number) are as follows:

Table H.1 Observations

RunID	Run Characteristics	Gradually Increasing MCT	Modified Lift	D
1	100 PIN CV with 10-fold backtracking pp=2	97.368	90.476	6.892
2	100 PIN CV with 10-fold backtracking pp=2	97.368	90.476	6.892
3	100 PIN CV with 10-fold backtracking pp=2	95.000	90.476	4.524
5	100 PIN CV with 10-fold backtracking pp=2	87.500	90.476	-2.976
6	200 PIN CV with 10-fold backtracking pp=2	93.902	92.593	1.309
7	200 PIN CV with 10-fold backtracking pp=2	90.244	92.593	-2.349
8	200 PIN CV with 10-fold backtracking pp=2	90.000	92.593	-2.593
9	200 PIN CV with 10-fold backtracking pp=2	85.526	92.593	-7.067
10	200 PIN CV with 10-fold backtracking pp=2	82.500	92.593	-10.093
11	300 PIN CV with 10-fold backtracking pp=2	92.373	96.610	-4.237
12	300 PIN CV with 10-fold backtracking pp=2	91.071	96.610	-5.539
13	300 PIN CV with 10-fold backtracking pp=2	88.393	96.610	-8.217
15	300 PIN CV with 10-fold backtracking pp=2	83.929	96.610	-12.681
16	400 PIN CV with 10-fold backtracking pp=2	89.189	89.865	-0.676
17	400 PIN CV with 10-fold backtracking pp=2	89.189	89.865	-0.676
18	400 PIN CV with 10-fold backtracking pp=2	89.189	89.865	-0.676
19	400 PIN CV with 10-fold backtracking pp=2	89.189	89.865	-0.676
20	400 PIN CV with 10-fold backtracking pp=2	84.459	89.865	-5.406
21	500 PIN CV with 10-fold backtracking pp=2	88.235	89.189	-0.954
22	500 PIN CV with 10-fold backtracking pp=2	88.235	89.189	-0.954
23	500 PIN CV with 10-fold backtracking pp=2	88.235	89.189	-0.954
25	500 PIN CV with 10-fold backtracking pp=2	88.235	89.189	-0.954
26	600 PIN CV with 10-fold backtracking pp=2	89.732	91.518	-1.786
27	600 PIN CV with 10-fold backtracking pp=2	87.892	91.518	-3.626
28	600 PIN CV with 10-fold backtracking pp=2	87.892	91.518	-3.626
29	600 PIN CV with 10-fold backtracking pp=2	87.892	91.518	-3.626
30	600 PIN CV with 10-fold backtracking pp=2	87.892	91.518	-3.626

t-value Calculation:

$$t = \frac{\sum D}{\sqrt{\frac{n \times \sum D^2 - \left(\sum D\right)^2}{n-1}}}$$

The parameters used in the formula are:

D is the difference for a single observation.

n is the total observation number.

$$t = \frac{70.950}{\sqrt{\frac{30 \times 782.308 - 5033.903}{29}}} = \frac{70.950}{\sqrt{\frac{18435.333}{29}}} = \frac{70.950}{25.213} \cong 2.8140$$

Dependent means t-test is summarized as follows:

Table H.2 Dependent Means T-Test Summary

t-Test: Paired Two Sample for Means

	Modified Lift	Gradually Increasing MCT
Mean	91.7085	89.3435
Variance	6.225648534	12.72714509
Observations	30	30
Pearson Correlation	-0.125668257	
Hypothesized Mean Difference	0	
df	29	
t Stat	2.814011752	
P(T<=t) one-tail	4.35E-03	
t Critical one-tail	1.699126996	
P(T<=t) two-tail	8.70E-03	
t Critical two-tail	2.045229611	

Since $t_{0.05,\ 29}$ (2.0452) is smaller than calculated t-value (2.8140), null hypothesis, H_0, is rejected. Thus it is concluded that, completeness obtained by modified lift is better than the performance of gradually increasing MCT in accuracy perspective ($\mu_{\text{MODIFIED LIFT}} > \mu_{\text{GRADUALLY INCREASING MCT}}$).

Although, previous and current dependent means t-tests emphasize that; discovered process model based on modified lift evaluation metric serves better correspondence to the behaviors exhibited in testing population, closeness of calculated t-value (2.81) to tabled t-

value (2.04) in this test (i.e. smallness of difference in each observation, namely D_i, to zero) highlights the tendency of modified lift towards representing transitions (patterns) with higher confidence and lower support values.

APPENDIX I Optimum Control Flow Graphs

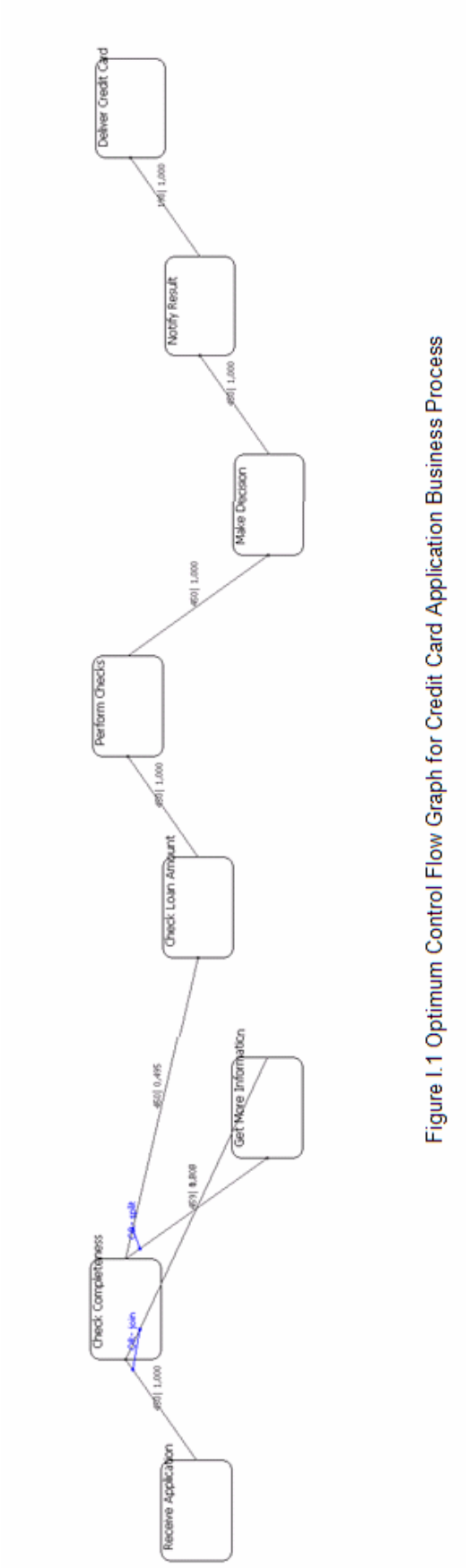


Figure I.1 Optimum Control Flow Graph for Credit Card Application Business Process

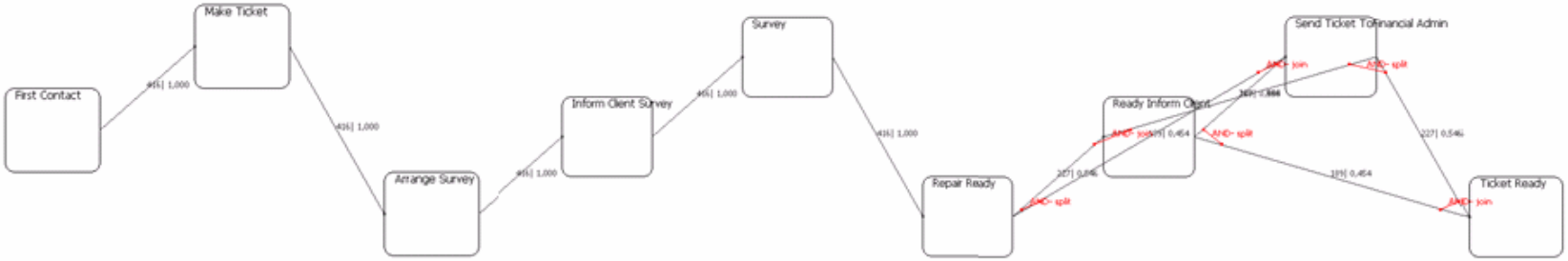


Figure I.2 Optimum Control Flow Graph for Repair Business Process

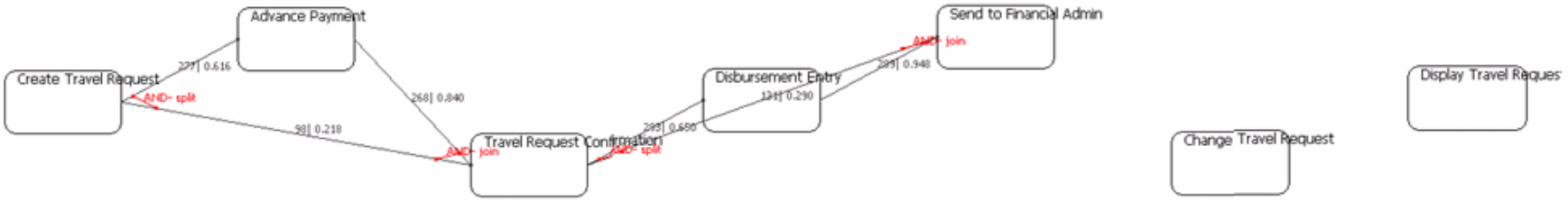


Figure I.3 Optimum Control Flow Graph for Travel Management Business Process

APPENDIX J1 Runtime Data (With respect to Number of Process Instances) Credit Card Application Business Process

TABLE J1.1 – COMPLETENESS (%)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	94.828	94.828	94.828	94.828	94.828	94.828	94.828	94.828	77.586	77.586	94.848
200	92.742	92.742	92.742	92.742	92.742	92.742	92.742	92.742	76.613	76.613	92.742
300	89.706	89.706	89.706	89.706	89.706	89.705	89.706	89.706	75.000	75.000	89.706
400	88.339	88.339	88.339	88.339	88.339	88.339	88.339	88.339	74.205	74.205	88.339
500	86.364	86.364	86.364	86.364	86.364	86.364	86.364	86.364	72.995	72.995	86.364
600	87.500	87.500	87.500	87.500	87.500	87.500	87.500	87.500	73.611	73.611	87.500

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	87.815	94.828	96.429	100.000	100.000	100.000
200	97.064	92.742	94.828	100.000	100.000	100.000
300	86.364	89.705	93.407	94.828	95.556	100.000
400	86.714	88.339	92.742	97.297	94.828	95.918
500	86.640	86.364	88.889	89.381	93.151	94.828
600	86.364	87.500	89.706	92.742	93.407	91.139

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	94.828	94.828	94.828	94.828	94.828
200	92.742	92.742	92.742	92.742	92.742
300	89.706	89.705	89.706	89.706	89.706
400	88.339	88.339	88.339	88.339	88.339
500	86.364	86.364	86.364	86.364	86.364
600	87.500	87.500	87.500	87.500	87.500

TABLE J1.2 – SOUNDNESS (%)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
300	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
400	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	0.000	0.000	0.000	28.571	28.571	28.571
200	0.000	0.000	0.000	14.286	14.286	14.286
300	0.000	0.000	0.000	0.000	0.000	14.286
400	0.000	0.000	0.000	0.000	0.000	0.000
500	0.000	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000	0.000

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000
300	0.000	0.000	0.000	0.000	0.000
400	0.000	0.000	0.000	0.000	0.000
500	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000

TABLE J1.3 – ARC TRAFFIC (transition number per transaction)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88
200	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88
300	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88
400	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88
500	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88
600	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	0.88	0.88	0.88	0.88	0.88	0.88
200	0.88	0.88	0.88	0.88	0.88	0.88
300	0.88	0.88	0.88	0.88	0.88	0.88
400	0.88	0.88	0.88	0.88	0.88	0.88
500	0.88	0.88	0.88	0.88	0.88	0.88
600	0.88	0.88	0.88	0.88	0.88	0.88

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	0.88	0.88	0.88	0.88	0.88
200	0.88	0.88	0.88	0.88	0.88
300	0.88	0.88	0.88	0.88	0.88
400	0.88	0.88	0.88	0.88	0.88
500	0.88	0.88	0.88	0.88	0.88
600	0.88	0.88	0.88	0.88	0.88

TABLE J1.4 – TOTAL PROCESSING TIME (sec.)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	37.86	37.86	37.56	38.15	40.16	37.86	37.92	38.44	38.06	39.06	38.63
200	45.48	45.85	45.12	45.30	44.77	45.48	44.95	44.50	44.42	44.70	45.73
300	55.25	57.45	54.01	53.70	54.20	55.25	54.70	54.70	54.42	54.32	56.05
400	66.59	66.19	66.23	66.11	66.65	66.59	66.29	66.62	65.51	66.87	66.90
500	80.34	78.33	80.49	78.54	81.53	80.34	78.17	79.20	79.18	79.89	80.44
600	97.20	94.58	93.68	94.63	94.18	97.20	93.89	96.84	95.68	95.16	96.51

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	13.42	37.86	75.67	113.590	151.56	192.11
200	14.35	45.48	90.80	135.68	182.46	225.83
300	17.09	55.25	110.14	162.22	216.16	275.02
400	22.41	66.59	133.42	189.34	266.60	336.81
500	25.89	80.34	163.65	242.53	327.95	409.24
600	31.75	97.20	193.94	284.19	377.85	467.92

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	39.22	37.86	39.42	38.86	39.88
200	46.79	45.48	46.15	46.17	46.09
300	54.86	55.25	55.59	55.74	55.27
400	66.65	66.59	66.58	66.52	66.78
500	80.74	80.34	83.54	80.64	80.56
600	96.87	97.20	96.78	98.32	97.64

TABLE J1.5 – TOTAL SIZE (KB)

|P|	businessprocesses table	fromtochart table	transactionlog tuple number	initialfromtochart table	tallymark table	tcodes table	transactionlog table	TOTAL SIZE
100	2.1	2.3	848	2.2	2.2	2.9	66.2	77.9
200	2.1	2.3	1696	2.3	2.2	2.9	107.4	119.2
300	2.1	2.3	2544	2.3	2.2	2.9	147.7	159.5
400	2.1	2.3	3392	2.3	2.2	2.9	195.9	207.7
500	2.1	2.3	4240	2.3	2.2	2.9	237.1	248.9
600	2.1	2.3	5088	2.3	2.2	2.9	281.3	293.1

APPENDIX J2 Runtime Data (With respect to Number of Process Instances) Repair Business Process

TABLE J2.1 – COMPLETENESS (%)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	98.611	98.611	98.611	98.611	98.611	98.611	98.611	95.833	75.000	62.500	98.611
200	96.250	96.250	96.250	96.250	96.250	96.250	96.250	88.750	62.500	62.500	96.250
300	97.596	97.596	97.596	97.596	97.596	97.596	97.596	97.596	62.500	62.500	97.596
400	96.382	96.382	96.382	96.382	96.382	96.382	96.382	96.382	62.500	62.500	96.382
500	96.543	96.543	96.543	96.543	96.543	96.543	96.543	96.543	62.500	62.500	96.543
600	95.474	95.474	95.474	95.474	95.474	95.474	95.474	95.474	62.500	62.500	95.474

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	97.321	98.611	100.000	100.000	100.000	100.000
200	100.000	96.250	97.222	100.000	100.000	100.000
300	95.296	97.596	99.038	98.611	100.000	100.000
400	94.657	96.382	98.611	98.077	98.611	98.214
500	94.792	96.543	96.875	97.656	97.115	98.611
600	94.361	95.474	96.875	95.833	98.333	97.727

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	98.611	98.611	98.611	98.611	98.611
200	96.250	96.250	96.250	96.250	96.250
300	97.596	97.596	97.596	97.596	97.596
400	96.382	96.382	96.382	96.382	96.382
500	96.543	96.543	96.543	96.543	96.543
600	95.474	95.474	95.474	95.474	95.747

TABLE J2.2 – SOUNDNESS (%)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
300	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
400	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	0.000	0.000	20.000	20.000	20.000	20.000
200	20.000	0.000	0.000	20.000	20.000	20.000
300	20.000	0.000	0.000	0.000	20.000	20.000
400	0.000	0.000	0.000	0.000	0.000	0.000
500	0.000	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000	0.000

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000
300	0.000	0.000	0.000	0.000	0.000
400	0.000	0.000	0.000	0.000	0.000
500	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000

TABLE J2.3 - ARC TRAFFIC (transition number per transaction)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.89	0.89	0.56	1.11
200	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.89	0.56	0.56	1.11
300	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.56	0.56	1.11
400	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.56	0.56	1.11
500	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.56	0.56	1.11
600	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.56	0.56	1.11

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	1.11	1.11	1.11	1.11	1.11	1.11
200	1.11	1.11	1.11	1.11	1.11	1.11
300	1.11	1.11	1.11	1.11	1.11	1.11
400	1.11	1.11	1.11	1.11	1.11	1.11
500	1.11	1.11	1.11	1.11	1.11	1.11
600	1.11	1.11	1.11	1.11	1.11	1.11

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	1.11	1.11	1.11	1.11	1.11
200	1.11	1.11	1.11	1.11	1.11
300	1.11	1.11	1.11	1.11	1.11
400	1.11	1.11	1.11	1.11	1.11
500	1.11	1.11	1.11	1.11	1.11
600	1.11	1.11	1.11	1.11	1.11

TABLE J2.4 – TOTAL PROCESSING TIME (sec.)

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	304.52	299.41	311.89	302.32	304.40	304.52	304.38	304.82	304.06	305.42	310.40
200	319.68	334.15	320.80	317.47	321.96	319.68	320.13	318.03	314.02	313.97	341.42
300	325.61	332.17	330.06	330.51	330.76	325.61	331.98	335.79	335.10	333.17	328.86
400	347.75	360.21	357.94	358.39	361.47	347.75	355.66	353.56	347.20	347.10	352.51
500	350.66	378.69	377.64	376.98	377.38	350.66	371.21	380.36	382.38	404.08	360.87
600	322.98	397.30	395.79	397.84	404.67	322.98	479.45	412.52	402.87	414.25	381.25

|P|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	93.77	304.52	608.08	915.97	1226.97	1529.78
200	94.51	319.68	636.92	932.54	1252.50	1557.58
300	104.97	325.61	664.88	991.67	1341.87	1616.75
400	106.61	347.75	692.39	1027.67	1355.01	1685.10
500	108.79	350.66	760.50	1023.14	1380.84	1814.70
600	114.65	322.98	746.83	1115.22	1503.22	1847.64

|P|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	308.55	304.52	309.15	305.44	311.41
200	363.55	319.68	367.09	362.68	367.14
300	327.78	325.61	328.99	330.18	332.29
400	355.65	347.75	346.53	349.14	351.65
500	347.16	350.66	351.25	350.38	351.86
600	381.36	322.98	382.29	382.59	382.53

TABLE J2.5 – TOTAL SIZE (KB)

|P|	businessprocesses table	fromtochart table	transactionlog tuple number	initialfromtochart table	tallymark table	tcodes table	transactionlog table	TOTAL SIZE
100	2.1	2.3	836	2.4	2.2	2.7	78.9	90.6
200	2.1	2.3	1688	2.4	2.2	2.7	157.4	169.1
300	2.1	2.3	2532	2.3	2.2	2.7	227.6	239.2
400	2.1	2.3	3392	2.4	2.2	2.7	302.4	314.1
500	2.1	2.3	4228	2.3	2.2	2.7	382.3	393.9
600	2.1	2.3	5080	2.3	2.2	2.7	455.8	467.4

APPENDIX J3 Runtime Data (With respect to Number of Process Instances) Travel Management Business Process

TABLE J3.1 – COMPLETENESS (%)

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	97.368	95.000	92.500	92.500	85.000	97.368	87.500	87.500	87.500	62.500	90.476
200	93.902	85.185	83.951	77.632	76.543	93.902	79.310	77.011	72.840	19.540	92.593
300	92.373	88.393	86.441	84.746	81.356	92.373	77.966	77.966	76.271	34.146	96.610
400	89.189	89.189	85.135	85.135	80.405	89.189	82.581	78.065	67.105	38.065	89.865
500	88.235	88.235	86.096	86.096	86.096	88.235	83.243	75.936	57.754	37.433	89.189
600	89.732	84.753	84.753	84.753	84.753	89.732	82.960	75.336	56.250	36.607	91.518

|PI|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	93.333	97.368	100.000	100.000	100.000	100.000
200	91.822	93.902	97.500	100.000	100.000	100.000
300	89.717	92.373	96.552	100.000	100.000	100.000
400	86.536	89.189	94.521	91.304	97.368	100.000
500	86.319	88.235	94.382	93.103	97.826	100.000
600	88.005	89.732	94.444	93.333	94.545	100.000

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	97.368	97.368	97.368	97.368	97.368
200	93.902	93.902	93.902	93.902	93.902
300	92.373	92.373	92.373	92.373	92.373
400	89.189	89.189	89.189	89.189	89.189
500	88.235	88.235	88.235	88.235	88.235
600	89.732	89.732	89.732	89.732	89.732

TABLE J3.2 – SOUNDNESS (%)

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	0.000	0.000	25.000	14.286	0.000	0.000	16.667	16.667	16.667	0.000	11.111
200	0.000	0.000	0.000	16.667	0.000	0.000	0.000	0.000	0.000	0.000	8.333
300	0.000	12.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	7.692
400	12.500	0.000	0.000	0.000	0.000	12.500	0.000	0.000	0.000	0.000	16.667
500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	8.333
600	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	7.692

|PI|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	0.000	0.000	0.000	55.556	33.333	55.556
200	0.000	0.000	33.333	50.000	16.667	58.333
300	0.000	0.000	11.111	11.111	22.222	44.444
400	0.000	12.500	0.000	25.000	12.500	25.000
500	0.000	0.000	0.000	14.286	0.000	14.286
600	0.000	0.000	0.000	12.500	12.500	25.000

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000
300	0.000	0.000	0.000	0.000	0.000
400	12.500	12.500	12.500	12.500	12.500
500	0.000	0.000	0.000	0.000	0.000
600	0.000	0.000	0.000	0.000	0.000

TABLE J3.3 – ARC TRAFFIC (transition number per transaction)

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	1.29	1.29	1.14	1.00	0.57	1.29	0.86	0.86	0.86	0.43	1.29
200	1.71	1.14	1.00	0.86	0.71	1.71	1.14	1.00	0.57	0.14	1.71
300	1.29	1.14	1.00	0.86	0.71	1.29	0.71	0.71	0.57	0.29	1.86
400	1.14	1.00	0.86	0.86	0.71	1.14	0.86	0.71	0.57	0.29	1.71
500	1.00	1.00	0.86	0.86	0.86	1.00	0.86	0.71	0.43	0.29	1.71
600	1.14	0.86	0.86	0.86	0.86	1.14	0.86	0.71	0.43	0.29	1.86

|PI|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	1.29	1.29	1.29	1.29	1.29	1.29
200	1.71	1.71	1.71	1.71	1.71	1.71
300	1.43	1.29	1.29	1.29	1.29	1.29
400	1.14	1.14	1.14	1.14	1.14	1.14
500	1.00	1.00	1.00	1.00	1.00	1.00
600	1.14	1.14	1.14	1.14	1.14	1.14

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	1.29	1.29	1.29	1.29	1.29
200	1.71	1.71	1.71	1.71	1.71
300	1.29	1.29	1.29	1.29	1.29
400	1.14	1.14	1.14	1.14	1.14
500	1.00	1.00	1.00	1.00	1.00
600	1.14	1.14	1.14	1.14	1.14

TABLE J3.4 – TOTAL PROCESSING TIME (sec.)

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
100	7.67	8.37	8.04	8.05	7.70	7.67	7.72	7.92	7.76	7.79	7.88
200	12.30	11.96	12.21	12.07	12.06	12.30	11.98	12.50	12.32	12.01	12.03
300	17.68	17.36	17.60	17.34	17.38	17.68	17.29	17.45	17.76	17.18	17.48
400	24.65	25.13	24.47	24.55	24.64	24.65	24.83	24.48	24.43	25.08	24.42
500	32.70	32.99	32.70	32.73	32.79	32.70	32.79	33.20	32.77	32.80	32.81
600	43.29	43.68	44.23	44.10	44.68	43.29	43.56	43.57	43.60	44.10	44.00

|PI|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
100	2.83	7.67	14.90	22.24	28.98	36.14
200	4.49	12.30	23.26	34.15	45.66	56.67
300	5.84	17.68	34.60	49.67	65.90	83.78
400	8.36	24.65	48.00	70.94	92.23	114.74
500	10.93	32.70	63.38	93.25	124.81	154.63
600	14.64	43.29	83.89	124.59	164.13	203.95

|PI|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
100	7.88	7.67	7.65	7.74	7.68
200	12.28	12.30	11.95	11.88	11.88
300	17.19	17.68	17.02	17.28	17.58
400	24.49	32.70	24.47	24.24	24.35
500	32.38	32.70	32.57	33.72	32.57
600	43.80	43.29	43.02	43.58	43.39

TABLE J3.5 – TOTAL SIZE (KB)

|PI|	businessprocesses table	fromtochart table	transactionlog tuple number	initialfromtochart table	tallymark table	tcodes table	transactionlog table	TOTAL SIZE
100	2.1	2.3	512	2.4	2.2	2.7	45.0	56.7
200	2.1	2.4	1026	2.4	2.2	2.7	88.1	99.9
300	2.1	2.4	1476	2.4	2.2	2.7	129.7	141.5
400	2.1	2.4	1933	2.4	2.2	2.7	169.5	181.3
500	2.1	2.4	2383	2.4	2.2	2.7	204.1	215.9
600	2.1	2.4	2855	2.4	2.2	2.7	247.5	259.3

APPENDIX K Runtime Data (With respect to Number of Activity Types)

TABLE K.1 – COMPLETENESS (%)

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
7	88.235	88.235	86.096	86.096	86.096	88.235	83.243	75.936	57.754	37.433	89.189
8	86.364	86.364	86.364	86.364	86.364	86.364	86.364	86.364	72.995	72.995	86.364
9	96.543	96.543	96.543	96.543	96.543	96.543	96.543	96.543	62.500	62.500	96.543

|AT|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
7	86.319	88.235	94.382	93.103	97.826	100.000
8	86.640	86.364	88.889	89.381	93.151	94.828
9	94.792	96.543	96.875	97.656	97.115	98.611

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
7	88.235	88.235	88.235	88.235	88.235
8	86.364	86.364	86.364	86.364	86.364
9	96.543	96.543	96.543	96.543	96.543

TABLE K.2 – SOUNDNESS (%)

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	8.333
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

|AT|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
7	0.000	0.000	0.000	14.286	0.000	14.286
8	0.000	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000

TABLE K.3 – ARC TRAFFIC (transition number per transaction)

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
7	1.00	1.00	0.86	0.86	0.86	1.00	0.86	0.71	0.43	0.29	1.71
8	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.75	0.75	0.88
9	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.56	0.56	1.11

|AT|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
7	1.00	1.00	1.00	1.00	1.00	1.00
8	0.88	0.88	0.88	0.88	0.88	0.88
9	1.11	1.11	1.11	1.11	1.11	1.11

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
7	1.00	1.00	1.00	1.00	1.00
8	0.88	0.88	0.88	0.88	0.88
9	1.11	1.11	1.11	1.11	1.11

TABLE K.4 – TOTAL PROCESSING TIME (sec.)

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.125 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.150 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.175 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.200 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.275 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.450 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.625 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.800 CV with 10-fold backtracking pp=2	Modified Lift CV with 10-fold backtracking pp=2
7	32.70	32.99	32.70	32.73	32.79	32.70	32.79	33.20	32.77	32.80	32.81
8	80.34	78.33	80.49	78.54	81.53	80.34	78.17	79.20	79.18	79.89	80.44
9	350.66	378.69	377.64	376.98	377.38	350.66	371.21	380.36	382.38	404.08	360.87

|AT|	MST=0.100 and MCT=0.100 Holdout backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 20-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 30-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 40-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 50-fold backtracking pp=2
7	10.93	32.70	63.38	93.25	124.81	154.63
8	25.89	80.34	163.65	242.53	327.95	409.24
9	108.79	350.66	760.50	1023.14	1380.84	1814.70

|AT|	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=1	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=2	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=3	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=4	MST=0.100 and MCT=0.100 CV with 10-fold backtracking pp=5
7	32.38	32.70	32.57	33.72	32.57
8	80.74	80.34	83.54	80.64	80.56
9	347.16	350.66	351.25	350.38	351.86

TABLE K.5 – TOTAL SIZE (KB)

|AT|	businessprocesses table	fromtochart table	transactionlog tuple number	initialfromtochart table	tallymark table	toodes table	transactionlog table	TOTAL SIZE
7	2.1	2.4	2383	2.4	2.2	2.7	204.1	215.9
8	2.1	2.3	4240	2.3	2.2	2.9	237.1	248.9
9	2.1	2.3	4228	2.3	2.2	2.7	382.3	393.9