CRANFIELD UNIVERSITY

Christopher James Turner

A Genetic Programming Based Business Process Mining Approach

School of Applied Sciences

PhD Thesis
Academic year: 2008-09

Supervisor: Dr. Ashutosh Tiwari
May 2009

CRANFIELD UNIVERSITY

School of Applied Sciences

PhD Thesis

Academic year: 2008-09

Christopher James Turner

A Genetic Programming Based Business Process Mining Approach

Supervisor: Dr. Ashutosh Tiwari

May 2009

This thesis is submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

Abstract

As business processes become ever more complex there is a need for companies to understand the processes they already have in place. To undertake this manually would be time consuming. The practice of process mining attempts to automatically construct the correct representation of a process based on a set of process execution logs.

The aim of this research is to develop a genetic programming based approach for business process mining. The focus of this research is on automated/semi automated business processes within the service industry (by semi automated it is meant that part of the process is manual and likely to be paper based). This is the first time a GP approach has been used in the practice of process mining. The graph based representation and fitness parsing used are also unique to the GP approach. A literature review and an industry survey have been undertaken as part of this research to establish the state-of-the-art in the research and practice of business process modelling and mining. It is observed that process execution logs exist in most service sector companies are not utilised for process mining.

The development of a new GP approach is documented along with a set of modifications required to enable accuracy in the mining of complex process constructs, semantics and noisy process execution logs. In the context of process mining accuracy refers to the ability of the mined model to reflect the contents of the event log on which it is based; neither over describing, including features that are not recorded in the log, or under describing, just including the most common features leaving out low frequency task edges, the contents of the event log. The complexity of processes, in terms of this thesis, involves the mining of parallel constructs, processes containing complex semantic constructs (And/XOR split and join points) and processes containing 20 or more tasks. The level of noise mined by the business process mining approach

includes event logs which have a small number of randomly selected tasks missing from a third of their structure. A novel graph representation for use with GP in the mining of business processes is presented along with a new way of parsing graph based individuals against process execution logs. The GP process mining approach has been validated with a range of tests drawn from literature and two case studies, provided by the industrial sponsor, utilising live process data. These tests and case studies provide a range of process constructs to fully test and stretch the GP process mining approach. An outlook is given into the future development of the GP process mining approach and process mining as a practice.

Acknowledgements

I would like to thank my supervisor Dr. Ashutosh Tiwari for his support and guidance in the completion of this research. In addition I would like to thank staff members of the industrial sponsor BT. I am also grateful to Dr. Jörn Mehnen for his technical advice on this research and I would like to say thanks to the staff of the Manufacturing Department for their support. Finally, I would like to thank my parents for their support and encouragement throughout the PhD.

Table of Contents

List of Figures

[Page is left intentionally blank]

List of Tables

List of Abbreviations

ARIS	Architecture of Integrated Information Systems
BEPL	Business Process Execution Language
BPM	Business Process Management
CIMOSA	Computer Integrated Manufacturing Open System Architecture
DFD	Data Flow Diagram
EPC	Event Process Chain
GA	Genetic Algorithm
GIM	GRAI Integrated Methodology
GP	Genetic Programming
GRAI	Graph with Results and Activities Interrelated Methodology
IDEF	Integrated Definition Modelling
InWoLvE	Inductive Workflow Learning via Examples
OMT	Object Modelling Technique
OOA	Object Oriented Analysis
OOD	Object Oriented Design
OOT	Object Oriented Technique
PAIS	Process Aware Information Systems

RAD	Role Activity Diagrams
RFID	Radio Frequency Identification
RID	Role Interaction Diagrams
SA-MXML	Semantically Annotated Mining Extensible Mark-up Language
SECI	Socialisation, Externalisation, Embodying and Connecting
SSADM	Structured Systems Analysis and Design Method
SSM	Soft Systems Methodology
UML	Unified Modelling Language
WIFA	Workflow Intuitive Formal Approach
XML	Extensible Mark-up Language

Publications

Accepted Publications (Journals/Book Chapter)

Turner, C. J. and Tiwari, A. (2009). An exploration of genetic process mining. In: Avineri, E., Köppen, M., Dahal, K., et al (eds.) *Applications of soft computing: Updating the state of the art*, Springer, Heidelberg, pp 199-208.

Tiwari, A., Turner, C.J., Majeed, B. (2008). A review of business process mining: State of the art and future trends, *Business Process Management Journal (BPMJ)*, 14 (1): pp 5-22.

Vergidis, K., Turner, C.J. and Tiwari, A. (2008). Business process perspectives: Theoretical developments vs. real-world practice, *International Journal of Production Economics (IJPE)*, 114(1): pp 91-104.

Tiwari, A, Turner, C.J, and Sackett, P. (2007). A framework for implementing cost and quality practices within manufacturing, *Journal of Manufacturing Technology Management (JMTM)*, 18 (6): pp 731-760.

Accepted Publications (Conferences)

Turner, C.J, and Tiwari, A, (2008). Process mining: A soft computing approach, In: *Multi-strand conference: Creating wealth through research and innovation (CMC 2008)*, Cranfield University, 6-7 May 2008, 6 pages.

Turner, C.J, and Tiwari, A. Mehnen, J, (2008). A genetic programming approach to business process mining, In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 12-16 July 2008, Atlanta, USA, pp 1307-1314.

Damour, L, Turner , C.J, Tiwari, A. (2007). Global strategic inventory planning, In: *Proceedings of the 5th International Conference on Manufacturing Research (ICMR-2007)*, 11th – 13th September, Leicester De Montfort University, UK, pp 229-233.

Turner, C..J and Tiwari, A. (2007). An experimental evaluation of genetic process mining, In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 7th – 11th July, University College London (UCL), London, pp 2268.

Turner, C.J, and Tiwari, A. (2007). An experimental evaluation of feedback loops in a business process mining genetic algorithm, In: *IEEE Congress on Evolutionary Computation (CEC)*, 25th – 28th September, Singapore, pp 2679–2686.

Turner, C.J, Vergidis, K, and Tiwari, A. (2007). Surveying business processes: The industry perspective, In: *Proceedings of the 5th International Conference on Manufacturing Research (ICMR-2007)*, 11th – 13th September, Leicester De Montfort University, UK, pp 313-317.

Sackett, P.J., Tiwari, A., Turner, C.J. and Salmon, R., (2006). Business benefit analysis for manufacturing enterprise projects: Towards generic tools, In: *Proceedings of 4th International Conference on Manufacturing Research (ICMR-2006)*, 5th - 7th September, Liverpool John Moores University, UK, pp 435 – 440.

Tiwari, A., Turner, C.J. and Majeed, B. (2006). Techniques for business process comparison and conformance: State of the art and future trends, In: *Proceedings of 4th International Conference on Manufacturing Research (ICMR-2006)*, 5th - 7th September, Liverpool John Moores University, UK, pp 441– 446.

Other Activities

The author of this thesis has been involved with a number of other professional activities while completing his PhD at Cranfield University –

- Special Session Co-Chair, on Soft Computing Applications in Production, Manufacturing and Design, in 12th Online World Conference On Soft Computing in Industrial Applications (WSC12) and member of the International Technical Programme Committee.
- The author has also been involved in a successful bid to win a 1 year 'follow-on fund' grant from the EPSRC to develop the concepts within this thesis into a commercial prototype.
- Paper reviewer and technical committee member of the IEEE World Congress on Computational Intelligence - Congress on Evolutionary Computation 2008.
- Paper reviewer and member of the international programme committee of the IEEE Congress on Evolutionary Computation 2007.

1. Introduction

In the complex modern business environment there is a need for managers to learn more about the processes they already have in place. There is also a need to know how those processes function in the real world. To undertake such a task manually would be extremely time consuming so the practice of process mining attempts to automatically reconstruct the correct representation of a process based on a set of process execution traces for automated or semi-automated processes. This thesis details the practice of process mining, and introduces the concept of process mining using a Genetic Programming (GP) approach.

1.1. Aim of Process Mining

The aim of process mining is to identify and extract process patterns from data logs to reconstruct an overall process flowchart (van der Aalst and Weijters, 2004). The data logs to be mined, more commonly known in the business process field as event logs, contain execution data for a live process. Such event logs may be hosted within Business Process Management (BPM) and other process related systems, owned by medium and large organisations, recording the task by task completion of computer assisted processes. In some organisations, event logs are hosted by Enterprise Resource Planning (ERP) systems.

Process mining as a practice adopts many of the techniques used for data mining (Cook and Wolf, 1998a). Data mining practice has been developed and adapted to create the business-process mining techniques that are now being used.

1.2. Genetic Programming

The approach of Genetic Programming (GP) is an evolutionary technique that uses the equivalents of the biological crossover and mutation operators to create successive populations of individuals. GP allows one to represent individuals in a variety of ways. In fact in many uses GP can be used to breed successive populations of computer programs, with each generation of programs bringing slightly different solutions to a problem in the form of new programs (Koza, 1992). There are three main types of representation in common use with GP –

Tree based representation

Linear representation

Graph based representation

This openness is in contrast to most other representations used by other forms of evolutionary techniques. For example classical Genetic Algorithms (GA) (Banzhaf et al., 1998) use a binary string to encode an individual. The representation used for this thesis is graph based, and offers a flexible way of representing candidate process model solutions. This flexibility offers advantages in terms of the way individual process models can be modified and their fitness assessed. It is true that both linear and tree representations could have been used though the mapping provided by a graph based representation provides a natural fit for the task of mining business processes. A GP approach negates the need to use an intermediary representation; when using a GA, for example, there is a need to translate to and from a binary string. In addition when implementing an approach with a binary representation a set of objects need to be created in order to represent individuals as graphs. The individuals created for a GP approach can be directly acted upon by the

operators (crossover and mutation) and fitness parsing (a method of providing a score to each individual showing how well that individual is able to describe the process traces recorded in the event log) and displayed as flowchart style graphs without need for translation. As will be seen in chapter 5 of this thesis, there are also advantages in that more intuitive fitness parsing methods may be used on an individual that is directly represented by a graph construct.

1.3. Parent EPSRC Project

This thesis is part of the wider intelli-process research project (Intelligent Decision Support for Process Re-design and Conformance, EPSRC project EP/C54899X), 2005-08. The intelli-process project aim was to provide new techniques for process mining and optimisation. The key objectives for this wider project included the investigation of automated techniques for the mining and optimisation of explicit models for complex business processes, establishing the state of the art in process mining approaches and gaining industry's current level of awareness of business process mining.

Two researchers were engaged in the research of the project; the author's contribution to the project was the development of a business process mining approach. A second researcher investigated the area of business process optimisation.

1.4. Problem Statement

Business process mining is a rapidly evolving area of research. While many techniques in this area (Herbst and Karagiannis, 2004; Schimm, 2004; Greco et al., 2005) have concentrated on the adaptation of existing approaches used in data

mining practice, only limited research has been carried out into evolutionary techniques and their application to process mining problems. The investigation of these non–deterministic techniques could hold the opportunity to improve the results obtainable for process mining practice. The aim of this research is to develop a genetic programming based approach for business process mining. The focus of this research is on automated/semi-automated business processes within the service industry.

1.5. Introduction to the Research

The design and management of business processes is a key factor in the successful operation of companies in an increasingly competitive global business environment. Most initiatives to improve processes within organisations are manual in their implementation; expensive and time-consuming due to the complex subjective nature of process re-design. Within service based organisations there is a need to facilitate automated mapping of current business processes in order to capture the knowledge elements required to partly automate the development of optimised process models; which staff may be monitored against to check for conformance to business practices.

1.6. Research Aim and Objectives

1.6.1. Research Aim

The aim of this research is to develop a genetic programming based approach for business process mining. The focus of this research is on automated/semi automated business processes within the service industry.

1.6.2. Research Objectives

The main objectives are to:

- Establish the state-of-the-art in research and practice in business process mining.
- Develop a genetic programming based approach for the mining of business processes.
- Refine the proposed approach for the mining of complex process constructs and semantics.
- Further enhance the proposed approach to mine process event logs containing noise.
- Validate the research results using a software prototype applied on a range of test problems and two real life case studies provided by the industrial sponsor.

1.6.3. Research Scope

Based on the aforementioned objectives the research scope is as follows –

- Domain: this research only looks at automated/semi-automated business processes within the service industry.

- Literature survey: there is a concentration on defining the notion of a business process and modelling techniques currently used to define such processes. The area of process mining and that of genetic programming are the main points of focus in this survey.

- Industry survey: research has been conducted into the service industries perceptions of business process modelling and management along with

the relatively new area of business process mining.

- Techniques: Genetic Programming (GP) is examined as part of this research due to its ability to represent complex business process designs.
- Development: the design and operation of the proposed business process mining approach, is discussed.
- Validation: the technique is evaluated with a range of test data from standard artificial tests and two real life case studies provided by the industrial sponsor.

1.7. Research Methodology

1.7.1. Problem Area

The area concerns the role that EC techniques, GP in particular, can bring to the efficient mining of complex business processes from process data, with and without the presence of noise.

1.7.2. Literature Review

The literature survey undertaken as part of this research has highlighted a number of areas for research with regard to current problems encountered in process mining. An evaluation of soft computing techniques has been carried out including an investigation of genetic programming. A survey of business process modelling techniques was included with this review of literature. The following steps were taken in the completion of this review: -

- Keyword identification – The main keywords regarding business process mining and the wider subject of business processes were sought, for example business process mining, business process modelling and business process management.

- Database search – A total of 250 papers, based on the keywords, were sought (databases used for this search included Google Scholar, Scopus and Science Direct from Elsevier).

- Paper filtering – Papers were filtered based on keyword areas and relevance (from a total of 130 papers identified from this stage).

- Paper analysis and classification – Key papers were identified at this stage (for the key papers relating to process mining see Table 1 section 2.4 chapter 2).

- Research gap – the research gap was identified from the analysis provided by the previous stage.

1.7.3. Research Focus Identification

Both the literature survey and industry survey have allowed the focus of this thesis to be narrowed down to the aforementioned aim and objectives. Current process mining problems and the identified strengths of evolutionary techniques such as Genetic Programming (GP) have shaped the research reported in this thesis. The strengths and limitations of current process mining techniques have also been added into the consideration for the focus of this thesis.

1.7.4. Industry Survey

An industry survey has been completed as part of this research in order to establish industries' perceptions of business processes. Feedback was sought on areas such as process modelling techniques knowledge, business process automation software in use and awareness of process mining. The survey has been conducted using a combination of face to face interviews (the questionnaire is available in Appendix D) and an online questionnaire (available in Appendix E). The following steps were required in the completion of the industry survey: -

- Participant selection – These were selected from a contact list of Cranfield University industry partners. Organisations in the survey should all be engaged in service sector activities.

- Questionnaire design – The literature review stage provided the material for the design of two questionnaires, one for administration by way of face to face interview and the other by a web form on the internet (containing short answer questions).

- Interview stage – This stage involved 5 face to face interviews with participants. The Online questionnaire was also completed by the respondents at this stage. In total the survey includes the responses from 25 participants in 23 organisations (such as banks, management consultancies and pubic sector organisations).

- Analysis of responses – The completed interview responses were collected along with the answers from the online questionnaire to form the completed industry survey.

The industry survey details and results are presented in chapter 4 of this thesis.

1.7.5. Development of a Process Mining Approach based on GP

A GP approach is developed for the practice of process mining. Existing techniques are evaluated and their strengths and weaknesses assessed in order to develop this new approach. The following stages are involved in the development of the GP approach: -

- Investigation into current process mining approaches – an examination of the way other approaches were developed.
- Identification of gaps in current approaches.
- Development of the proposed approach including algorithm formation based on gaps in current approaches
- Selection of software for completion of programming activities – including research into existing software resources available for process mining.
- Software development.
- Initial evaluation of the approach – use of existing test data sets to evaluate the approach.

1.7.6. Validation of the Approach

A range of artificial process data sets are used to validate the approach. Each data set contains a number of process constructs which vary in complexity, providing a balanced range of mining challenges for the approach. In addition a number of sets have been amended to allow for the addition of noise, that is a number of tasks in the process data are removed from a percentage of the process traces. This simulates real life process data from organisations which may be incomplete/incorrect. Two case studies provided by the industrial sponsor of this project are also used as part of

the validation. This tests the ability of the proposed GP approach to mine larger and more complex processes. The following steps are required for the completion of the validation of the GP approach: -

- Identify current test data sets from literature.
- Run the initial experiments using the test data sets.
- Review results from initial experiments with test data sets to analyse performance of the proposed GP approach and to determine optimum test parameters (such as maximum generations).
- Examine data sets provided by the industrial sponsor.
- Carry out testing using the data sets provided by the industrial sponsor and collate the results.

1.7.7. Contribution to Knowledge, Limitations and Future Research

The contribution to knowledge is established by reviewing the research outcomes against the aim, objectives and the research gap. An examination is also made of the current and future trends in the field of process mining, and their possible impact on the GP approach detailed in this thesis. The limitations of the GP approach are discussed and possible future modifications are outlined. Extensions to the approach to enable the solving of additional mining problems are also put forward. The methodological approach to be used for this research is summarised as the following research design shown in Figure 1.

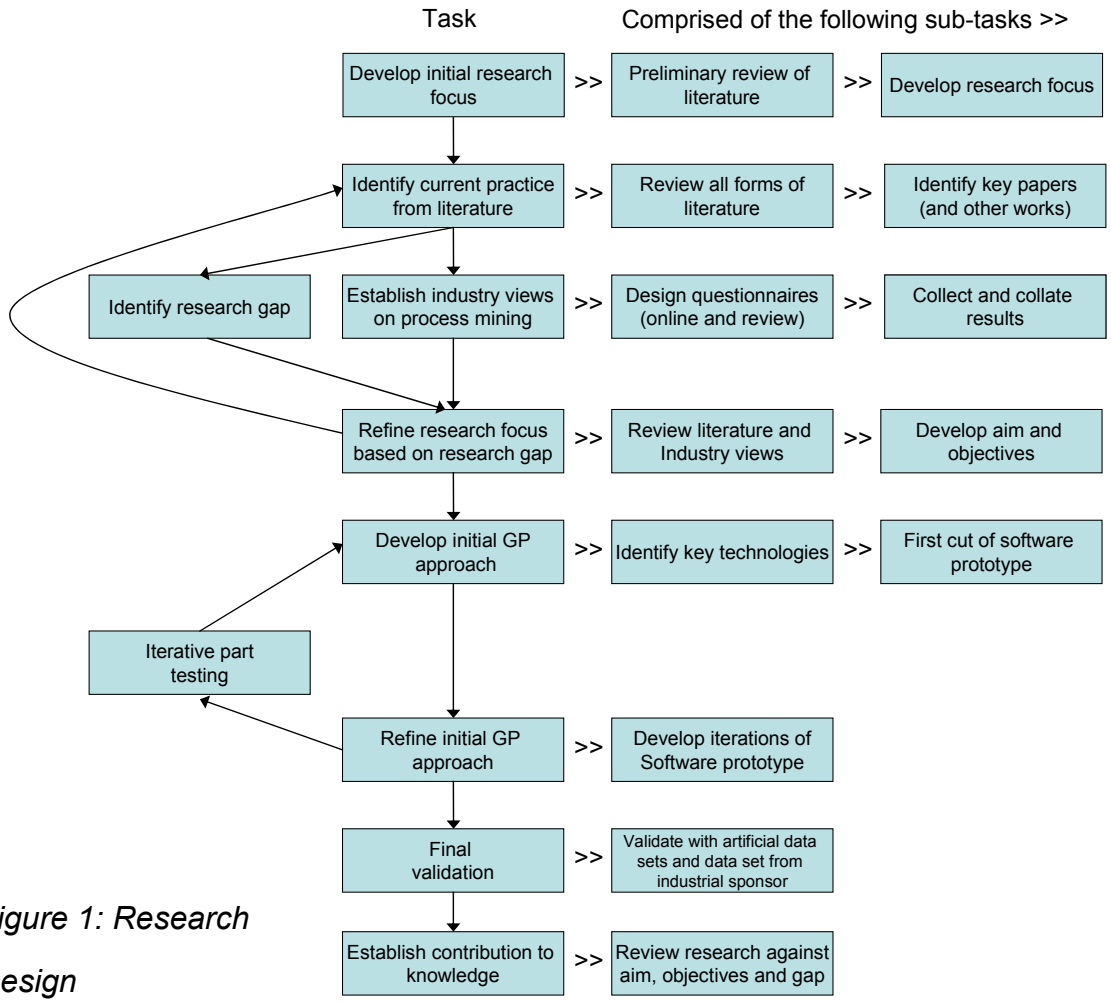


Figure 1: Research Design

The research design shows the main steps that must be completed in this research. As can be seen in Figure 1 the initial research focus is developed from the work carried out at the literature review stage. This focus is narrowed with a more in depth survey of the literature where key papers are identified and examined in detail. This literature review helps to identify questions that may be incorporated into an industry survey, to gain insight into company views and understanding of business processes and the practice of business process mining. A loop from the refinement of the research focus back to the literature review and industry survey is made as their findings may influence the focus of the research and encourage the further exploration of literature.

Once the research focus has been decided, the initial GP approach can be investigated and designed. A software prototype emerges from this initial development stage. An iterative testing loop is deemed necessary, in this research design, between the initial software prototype and the GP approach, refining the GP approach in order to add and test additional functionality. The validation process is then carried out in this design (using the GP approach hosted in the ProM process mining suite (van Dongen et al. (2005b)). This is followed by a review of the research gap, aim and objectives to determine the contribution to knowledge of this research.

1.8. Research Deliverables

The following have been completed as part of this thesis:-

- A literature and industry survey detailing the state-of-the-art in business process mining and modelling.
- An initial approach for process mining based on GP.
- A set of refinements for GP process mining to improve mining accuracy in terms of mining complex process constructs (that is constructs containing multiple parallel flows and long task sequences) and semantics (complex split and join points allowing edges to multiple tasks).
- A GP process mining prototype capable of mining data containing noise.
- A series of validation experiments to determine the value of the GP process mining approach, based on artificial data sets and two real case studies provided by the industrial sponsor.

1.9. Outline of Thesis

This thesis is comprised of the following chapters

Chapter 1

Chapter 1 gives an introduction to the research conducted for this thesis and the aim and objectives.

Chapter 2

Chapter 2 provides a review of the literature surrounding the subject area of process mining including such themes as:

- Defining a business process
- Representing business processes as models
- Introducing the main process mining techniques
- Process mining problems
- Outline of soft computing
- Soft computing and process mining
- Genetic programming

Chapter 3

The results of an industry survey on the subject of process modelling and mining are presented in this chapter.

Chapter 4

The concept of Genetic Programming and its application to the area of process mining is introduced in this chapter along with the business process model representation to be used as part of the GP approach.

Chapter 5

The further development of the GP process mining approach to mine complex process constructs along with noisy data logs is explained in this chapter.

Chapter 6

Chapter 6 outlines the experiments conducted with the GP approach to assess its ability to mine a range of process logs with and without the presence of noise.

Chapter 7

This chapter outlines additional enhancements that could be made to the GP approach to improve its performance along with a general outlook on the field of process mining. In addition a discussion of the findings and conclusions of this thesis is provided in this chapter.

1.10. Summary

The main research theme of this thesis has been introduced in this chapter. The areas of process mining and Genetic Programming have also been outlined. A summary of the chapters that comprise this thesis has also been provided. This chapter has also discussed the following:

- The aim of this research.
- The objectives to achieve the aim of the research.

- The scope of the research has been outlined.
- The methodology followed in the development of this research.

The aim of this research is to develop a genetic programming based approach for business process mining. The focus of this research is on automated/semi automated business processes within the service industry. The focus of this research is on business processes within the service industry. This is achieved through the development of a business process mining approach based on GP. Chapter 2 details the literature review undertaken in the completion of this research.

2. Literature Review

2.1. Introduction

A number of areas have been investigated in the course of this research in order to develop the Genetic Programming (GP) process mining approach. Research has been made into soft computing and evolutionary techniques and their strengths and weaknesses. Similar work has been conducted into existing process mining techniques and the process mining problems that they aim to solve. This review is a summary of those investigations. The process mining section of this chapter (section 2.4) has been published by the author in Tiwari et al. (2008).

2.2. Definitions of Business Process

An examination, and definition, of the term 'business process' is initially required in order to fully understand the central subject of this thesis. Havey (2005) helps to this end by providing guidance on the meaning of the word 'process' stating that the term process implies issues of work, movement and time, and suggesting that the actions of a process are performed over a time-period in order to journey towards or reach an objective. A generic definition of business process is given by Harmon (2003) as 'Any set of activities performed by a business that is initiated by an event, transforms information, materials, or business commitments, and produces an output'.

Harmon also mentions that the outputs of processes may be valued by customers or other processes. Processes are usually comprised of multiple tasks (each with their own inputs and outputs); a process with only one task is not usually

considered to be a process in its own right (Amaravadi and Lee, 2005). There is evidence that processes are often described at varying levels of abstraction; indeed many processes may be broken down into sub processes (Giaglis et al., 1996).

Lee (2005) points out that the term process is often confused with practice. Lee defines process as codified routines carried out by individuals with explicit knowledge of those routines, whereas practice refers to tacit knowledge of individuals being used to carry out a routine without codified guidance. In essence Lee argues that practice requires heuristic knowledge of a routine. In the view of the author of this thesis Nonaka's SECI (Socialisation, Externalisation, Combination, and Internalisation) model of knowledge conversion is relevant in the context of this discussion. The SECI model (shown in Figure 2) illustrates the process of converting tacit knowledge to explicit representations that can be shared and perhaps embodied in business processes. The dynamic perpetual nature of SECI is represented by a spiral at the centre of the model.

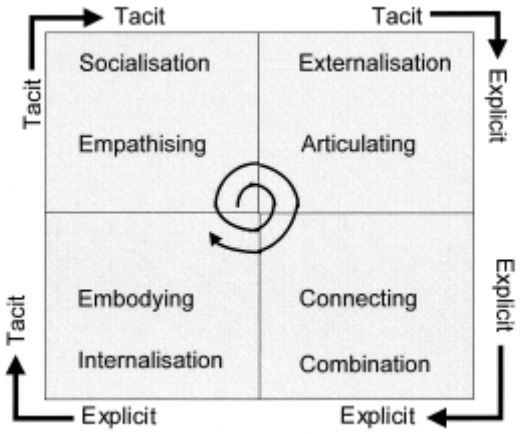


Figure 2: SECI Knowledge Conversion Model (Nonaka et al., 2000)

Examining the area of business process in greater detail a variety of definitions for this subject have been but forward over a number of years, as evidenced by Dietz

and Habling (2004). In the opinion of some authors (Dietz and Habling, 2004) the term business process has been misused, as some organisations may assume that the term business process means information process, and that information systems represent business processes in their entirety. Further it is argued that current definitions and organisational understanding of business processes, in general, do not recognise anything more than a purely mechanistic approach where set paths are followed based on predetermined choices. This argument will be explored in greater detail in the literature review section on modelling techniques of this thesis, where it is argued by some that the choice of technique employed by an organisation may differ depending on their understanding of the term business process. Dietz and Habling (2004) put forward a model that attempts to acknowledge and formalise the soft skills of human actors necessary for business processes to operate effectively.

Lindsay et al. (2003) also argue that current understanding of business processes encourages a mechanistic approach. These authors also state that much business process practice concentrates on what has happened in the past rather than being flexible enough in construction to adapt to current or future events.

Lindsay criticises Davenport (1993) in his definition of process as being a prescriptive approach paying little attention on how a process is performed. Davenport defines a process as ‘a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs, a structure for action’. To Davenport the structure of a process is paramount in how work is completed in an organisation.

Goal orientation is another aspect found in some business process definitions. Kueng and Kawalek (1997) assert that there is a need to have a goal in mind when designing any business process; and that recognition of process goals, and risks inherent in reaching a goal, must be translated into the resulting process. These

authors describe a method for recognising the importance of goals when designing processes, while acknowledging the subjective nature of goal setting (this work will be expanded on in section 2.3 of this literature review). Andersson (2005) also gives reference to goal orientation when designing processes along with the notion of business process patterns, which may be followed in the production of new processes.

Amaravadi and Lee (2005) describe differences between processes found in manufacturing and non manufacturing environments. Amaravadi and Lee go onto mention the work of Garvin (1998) and his classification of approaches to organisational processes. Garvin outlines three schools of thought on organisation based processes –

Work Process – A process is simply a set of steps that need to be followed in order, with clearly defined inputs and outputs. This viewpoint is usually held by those who believe in a mechanistic approach to process design and operation (authors such as Davenport (1993)).

Behavioural Process – A process is seen more in the context of the human actors involved in its design and operation. The way goals are set and problems solved by the human actors are deemed important factors in the design of processes.

Change Process – A process is seen as a dynamic and flexible entity. Such processes are designed to capture new, sometimes stochastic, behaviour as it occurs. A change process is most often used for companies undergoing rapid evolution.

Irani et al. (2000) point to the work of Paul et al. (1997) who highlight the inherent stochastic nature that characterises business processes. The work of Tumay (1996) also highlights the business environment that processes try to model as stochastic.

Nature. Tumay (1996) also notes that the entities that processes incorporate tend to be dynamic in nature (entities such as customers, products and orders). Reijers and Vanderfeesten (2004) assert that administrative (service) type processes tend to be more flexible than processes used in manufacturing, due to the 'virtual' processing of information in comparison with the physical processing of parts in manufacturing. Though it is also stated by these authors that the flexibility in non manufacturing processes is easily lost through poor process design, especially in respect to the definition of divisions between individual processes and size of the process dividends. Melao and Pidd (2000) describe a framework of four different views of business processes -

- Deterministic machines.
- Interacting feedback loops.
- Dynamic complex systems.
- Social constructs.

Melao and Pidd (2000) also comment on mechanistic approaches to business process design arguing such methods ignore both human and organisational issues in the pursuit of computationally 'correct' models. Business processes may also harbour intricate information flows which may be examined over a period of time. This belays the notion that static representations of processes, used by many mechanistic approaches, can explain the dynamic nature of a process's use. Further efforts to classify different types of process include the CIMOSA (Computer Integrated Manufacturing Open System Architecture) framework (Zelm et al., 1995). This framework divides processes into three broad categories: Manage, Operate and Support :-

- Manage – largely direction setting and organisational strategy
- Operate – Are focussed on the customer and delivering their needs
- Support – A framework of processes related to the administrative operation of the organisation and the support of manage and operate processes.

Van der Aalst and van Hee (2002) describe a business process as one 'focussed on the production of particular products' where the term product could mean a product or service. The authors go on to define three categories of processes, which are very similar to those put forward by the CIMOSA model –

- Primary – Concerned with the production of products or services for an organisation, customer focussed processes that generate income for an organisation.
- Secondary – Support primary processes and are usually to do with the administration and maintenance of the organisation.
- Tertiary – managerial processes that embody the organisational rules and regulations; these processes are used to coordinate the other two groups of processes.

(van der Aalst and van Hee, 2002).

Regev et al. (2005) cite another aspect of business processes in their ability to act as regulative forces within an organisation. The notion of business stability is outlined by the authors in their four principles of business homeostasis.

- A level of stability in a business in a dynamic and changing environment is evidence that a number of processes exit in the business that contribute to its stability.
- A business remains stable in a changing environment if its process that resist change become more effective.
- Business stability can be maintained through the coordinated use of its co-operating processes.
- To avoid over compensation automatic control of processes (or use of processes with an opposite effect) may be contemplated when the state that they change is inherently stable.

(Regev et al., 2005)

According to Regev et al. (2005) these four principles may be used to provide a business with a level of stability in a dynamic and rapidly changing environment.

In summary there is no general agreement on the definition of the term business process, though certain elements do reoccur in many definitions. These elements include the notion of workflow and the transformation of inputs to a process (Garvin, 1998). In terms of the current practice of process mining it is the author's opinion that the business process definition offered by Davenport (1993) (that processes are structural in nature with a beginning and an end, including a specific ordering to the activities within the process) is most relevant to the research presented in this thesis and will be used as the definition of business process here. For the purposes of this thesis the definition of Davenport (2003) gives a good fit to the operation and role of the GP approach, and is concise in nature. The criticism of Davenport's definition is that it is mechanistic and concentres on the past rather than how a process is performed (Lindsay, 2003). In the view of the author of this thesis Davenport's (definition describe how business processes can be mined at present. The types of

measures and parameters used by industry at present are largely structural and mechanistic. For a less mechanistic definition, such as that provided by Melao and Pidd (2000), to be relevant would require a change in that way processes are administrated and measured. This may be the case in the future, but for now (in the view of the author of this thesis) the definition provided by Davenport (1993) best describes the current application of a process mining tool when used in industry.

2.3. Representation of Business Process Models

This section of the literature review looks at a variety of business process modelling techniques that are now available. Sadiq and Orlowska (2000) offer a useful definition of a process model stating that it orders and defines aspects of a process, such as resources, data and tasks. Some authors use the term workflow to refer to processes (in particular the concept of Business Process Management (BPM)); though to other authors workflow refers only to the general flow of documents though an information system and is different from process, in that processes are normally automated and they may communicate with each other (Havey, 2005). The author Havey (2005) offers a discussion on the subtle differences said to exist by some commentators, between workflow and BPM. In the course of this thesis mention will only be made of the term process; this term can be taken to encompass workflow in the context of this research. Process models take the form of a directed graph showing the individual tasks that make up a particular process. Sadiq and Orlowska (2000) also define the term process instance as a particular instance of a process. An instance of a process may represent a certain path through a process model that is taken in the satisfaction a particular case (live execution of the process). For certain cases only a subset of tasks within the process model may need to be executed, and this sub set may change depending on the case in question

(Sadiq and Orlowska, 2000). Meyer (1995) gives guidance as to the composition and design of a method. This author argues that any method (including process modelling methods) is comprises of three main areas -

- Discipline – Syntax of and procedure for the use of the method.
- Use – How should the method be used (in a range of situations).
- Definition – The motivation and concepts behind the method.

Meyer provides a diagram to explain these three areas; this is shown in Figure 3.

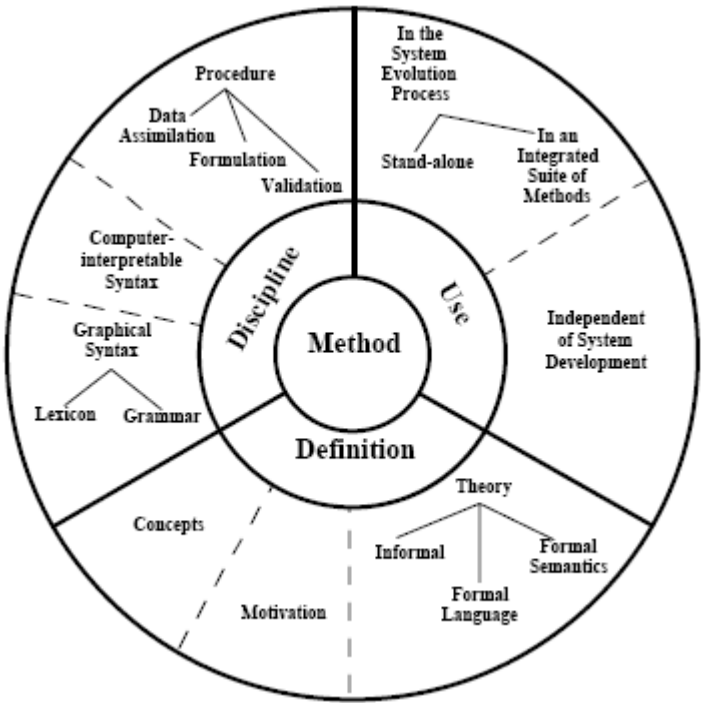
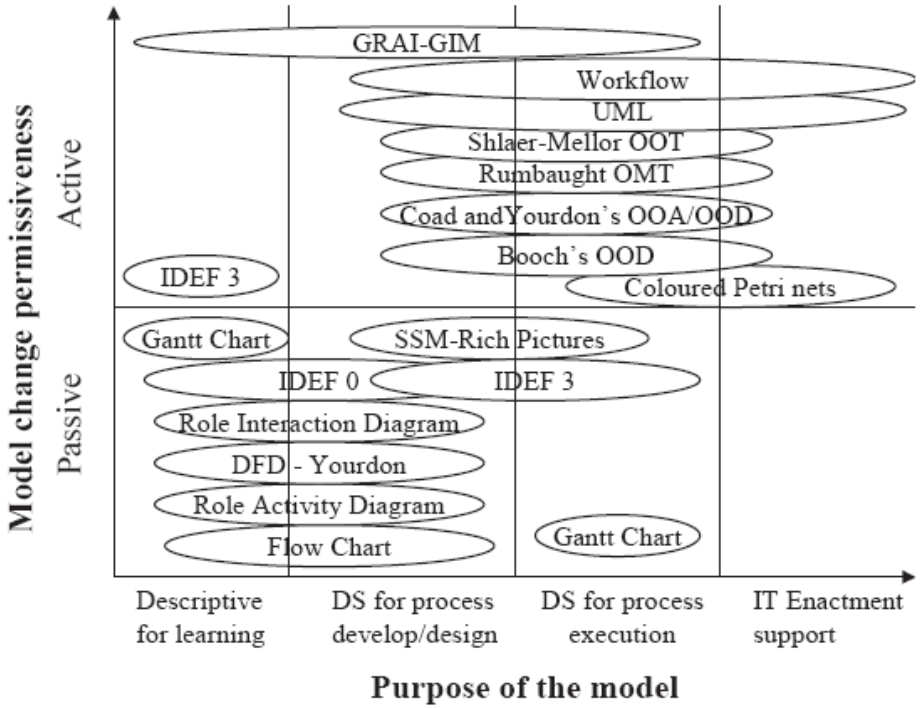


Figure 3: Anatomy of a Method (reproduced from Meyer et al. (1995))

Aguilar-Saven (2004) describes a process model as the means to analysing and understanding a business process. Aguilar-Saven mentions the work of Phalp and Shepperd (2000) who define two types of uses for process models. One use is in the software development world, another is in the task of redesigning business

processes. The work of Phalp and Shepperd promotes the use of simple diagrammatic notation in order to understand a process. Aguilar-Saven proposes a simple classification of business process models (shown in Figure 4). Models are compared on their ability to allow the model developer to make changes to a model as the process being modelled changes. A model that does not allow changes is said to be passive, one that does allow changes is said to be active.



Key to Acronyms

DS – Describe System	GRAI – Graph with Results and Activities Interrelated	GIM – GRAI Integrated Methodology	UML - Unified Modelling Language	OOT – Object Oriented Technique
OMT – Object Modelling Technique	OOA – Object Oriented Analysis	OOD – Object Oriented Design	IDEF – Integrated Definition Modelling	SSM – Soft Systems Methodology
DFD – Data Flow Diagram				

Figure 4: Classification Framework for Business Process Modelling Techniques (reproduced from Aguilar-Saven (2004))

Of the business process modelling techniques, Petri nets, UML and Workflow techniques are rated as being the most active modelling methods in that they can support real time changes, without the need of complete redesign, and in some cases allow for automated updating based on changes detected in business processes. This last point is especially poignant in the completion of this thesis. In addition to the classification shown in Figure 4 Aguilar-Saven (2004) describes a list of techniques that in their opinion constitute the main process modelling methods.

- Flow Chart Technique – A generic term for a diagram showing flows within a system.
- Data Flow Diagrams (DFD) – A descriptive modelling technique showing logical flow for structural analysis use.
- Role Activity Diagrams (RAD) – Provides a role based graphical representation of a process.
- Role Interaction Diagrams (RID) – Provides a graphical representation of a process based on a combination of activities and roles.
- Gantt Chart – A matrix representation of al the tasks and activities to be performed in a process over a certain timescale.
- Integrated Definition Modelling (IDEF) – A formal methodology for enterprise modelling (only IDEF – 0 and 3 are particularly relevant for process modelling)
- Coloured Petri Net – A formal (algorithmic) representation of a system in terms of a defined graphical notation set.

- Object Oriented Methods (including Unified Modelling Language (UML)) – Methods that allow the activities of a process to be modelled as (programmable) objects.

- Workflow technique – The automation of a business process where document flows through a computerised system are governed by a set of rules (various notations may be used such as forms of net based languages, based on Petri nets).

A number of generic modelling techniques, that have some relevance to the modelling of business processes, are described by Aguilar-Saven (2004).

- Computer Integrated Manufacturing Open System architecture (CIMOSA) – Object based enterprise integration methodology used to model processes to be programmed into an automated system.

- Structured Systems Analysis and Design Method (SSADM) – A systems based methodology used in the analysis and design stage of new information systems.

- Soft Systems Methodology (SSM) – Used to support and structure ideas about existing and new systems.

- Graph with Results and Activities Interrelated Methodology (GRAI) – Manufacturing based production management systems decision support methodology.

- Simulation – The abstract imitation of the behaviour of an existing or proposed system, for the purposes of further analysis.

Aguilar-Saven (2004) advises that these techniques, as they are not solely designed for the purposes of modelling business processes, are used as part of a wider initiative within an organisation. As an example SSADM is used to develop new information systems; its use for business process modelling is most relevant when the processes in question must be modelled as part of the information system implementation.

Of the techniques outlined by Aguilar-Saven (2004) it is the opinion of the author of this thesis that only the modelling techniques marked as active in the classification shown in Figure 4 need to be considered in more detail. This is due to the nature of the subject of this thesis in that any model(s) employed as part of this project must be able to react or be modified to reflect real-time changes in processes (the need for this will be expanded on in later sections of this literature review).

The modelling techniques that will be examined are :-

- GRAI.
- Workflow.
- UML.
- IDEF3.
- Object Oriented (OO) approaches.
- Petri nets (and coloured Petri nets).

GRAI

The GRAI method, originated at the University of Bordeaux over 25 years ago, is a set of analysis techniques designed to model the management structure of manufacturing organisations (Wainwright and Lee, 2000). Centres of decision making within an organisation are identified using this system along with information needed to make decisions. The technique is graphical in nature and requires a decomposition approach to the analysis of an organisation. There are three parts to the GRAI technique:

- Development of a conceptual model – provides a scheme to allow the practitioner to identify a model of the organisation which represents a desired end state.

- Provision of tools and representation rules – A GRAI grid provides a hierarchical and structural representation of the decision centres within an organisation and GRAI nets allow for the display of decisions taking place within decision centres and links between centres

- The GRAI application methodology provides a series of steps to take the practitioner through the GRAI analysis process. A potential pitfall with the GRAI methodology is its inherent complexity. Wainwright and Ridgway (1995) suggest that GRAI would benefit from data dictionary functionality so that individual elements of the model could be identified and analysed with greater ease.

Workflow

Havey (2005) states that workflow concerns the processing of a section of work as it passes through different operations. It also concerns the activities that must take place when an exception occurs in the processing of and the order of operations that must be performed on a section of work. The term workflow pre dates the introduction of automated processes (Havey, 2005), initially referring to paper based/manual transactions. Havey also states that the term workflow has now been superseded by Business Process Management (BPM) which incorporates the functionality of workflow systems. In addition Reijers (2003) notes that a workflow is a special type of business process that may be managed by a BPM system.

Business process management comprises of four steps -

- Design – The steps of a process need to be organised, this is usually achieved through a software based graphical interface with the steps represented in the form of a flowchart (known as a directed graph in BPM terminology).

- Implementation – Business process steps, once designed, are then implemented in a business process language, such as BPEL (Business Process Execution Language) which is a meta-language based on Extensible Mark-up Language (XML).

- Execution – A coded process can be mounted on a BPM server and executed in real-time.

- Controlling – applications exist on the BPM server that allow a level of control over executing processes, in case unhandled exceptions occur. Processes can also be monitored for performance

(Havey, 2005)

At any time during the four stages a visual representation of the process(es) being managed may be displayed (depending on the software tools being used).

Herbst and Karagiannis (1998) put forward the view that workflow models are limited in their ability to adapt to changes in the environment. These authors put forward the view that the integration of machine learning techniques would allow workflow systems to adapt to changing environments by analysing the data collected from previous process executions. Sheth (1997) provides a research agenda for the area of process automation and the active adaptation of workflow executions based on changes in the organisation.

Van der Aalst (1999) addresses a number of problems with automated workflow management, namely the provision of management information and the support of dynamic actions such as changes to active workflows.

Jablonski (2000) defines a workflow management system as an entity containing a number of workflows that in combination can comprise a discrete application system.

Wang et al. (2005b) propose a new workflow technique that, it is claimed by the authors, is simpler to apply than other techniques. The Workflow Intuitive Formal Approach (WIFA) workflow model allows processes to be broken down into a set of tasks that are related to ‘execution entities’ such as human workers or automated

operators (e.g. computer applications). Tsai et al. (2006) have developed a toolset and graphical modelling interface to take advantage of the WIFA workflow model.

UML

Unified Modelling Language (UML) is defined by Aguilar–Saven (2004) as a language used to define software aiding in the specification and construction of software objects through a set of graphical notation. The UML notation was developed at the Rational Software Company in the early 1990s and was accepted by the ratifying body, the Object management Group (OMG), in 1997 (Rumbaugh et al., 2005). The UML notation consists of a set of nine diagrams capable of defining different views of a system. UML is a more recent invention than other object oriented modelling techniques. In effect UML is an amalgamation of several pre-existing methods and notations. The language is derived from the work of three authors of object oriented modelling techniques; Booch, Jacobson and Rumbaugh (Quatrani, 2003). Both Rumbaugh's Object Modelling Technique (OMT) and Booch's Object Oriented Design (OOD) approach were used as the basis for UML (Rumbaugh et al., 2005).

OMT and OOD

The Object Modelling Technique (OMT) is another modelling system used to design information systems in an object oriented fashion (Wang et al., 1997). Three different design views are provided by this model; object, dynamic and functional. One of the major drawbacks of this technique concerns the overall integration of the three design views that are provided. Wang provides a formalisation of the dynamic aspect of OMT and attempts to integrate the object and dynamic aspects as well. In the opinion of Wang this formalised model of OMT can be used to produce software based systems that are consistent in several different dimensions.

OMT has largely been used by the software industry in the design of new systems. Rohollof (1996) introduces a way for OMT to be used in the task of modelling business processes. This author argues that OMT presents difficulties when trying to model an entire process as it is really designed to examine the operation of individual objects. Rohloff puts forward an extension to the OMT model, the OGM object oriented business process modelling approach (the OGM acronym is translated as object oriented business process modelling). This approach allows a process to be modelled at several levels of abstraction allowing a practitioner to examine a process at both macro and micro levels.

One interpretation of the Object Oriented Design (OOD) method was developed by Booch (Rumbaugh et al., 2005) as an iterative approach to software modelling. This process is illustrated in the following four steps -

- Identify the classes – identify all of the potential objects in a system and create corresponding classes for each object.
- Identify the semantics – define the responsibility of each class.
- Identify the relationships – between classes.
- Specify the interface and implementation – how will the software be developed and deployed.

(Lewis et al., 2004)

Both OOD and OMT methods have been criticised by Thurner (1997) as lacking proper semantic underpinnings leading to possible misunderstandings due to differing interpretations of the notation. According to Thurner (1997) a good semantic grounding is required for a method to be applicable to the task of modelling business processes; that is the syntax should reflect the semantics of the subject being

modelled minimising the risk of mistakes. Aspects of the OOD method, along with the OMT method have been used as a basis for the UML notation set.

Object Oriented Analysis (OOA) / Object Oriented Design (OOD)

Another interpretation of Object Oriented Design was developed by Coad and Yourdon (1991) along with the complementary Object Oriented Analysis method. OOD practice involves the design of objects concerned with the implementation of system design, whereas OOA involves the design of objects concerned with the operation of a system and helps to organise the thoughts of a system analyst. Coad and Yourdon recommend that OOA should be used at the system specification and analysis stage and then followed by OOD to implement the physical design of the system.

OOT

The Object Oriented Technology (OOT) method differs from other object oriented approaches by allowing software designers to develop object classes without the need to describe the top level of the system they are designing. This method is also said to support design reuse rather than just software reuse, an important factor when designing business processes (Wang and Leung, 2001). OOT promises faster code production, clearer program structure and an iterative approach to development. Wang and Leung (2001) have found these claims to be largely correct. In terms of OOT's use in business process modelling there is little literature available. One interesting contribution is Ren et al. (1997), who suggest that OOT should be seen not just as a software system modelling technique but as a way of thinking about the design of a wider information structure for an organisation.

Integrated Definition Modelling 3 (IDEF3)

This method is used to describe how a particular system operates by capturing descriptions of activities while 'in motion'. This method is observation based, representing how sequences of activities operate in a live environment. IDEF3 has been designed to promote reuse of recorded process information where possible. There are two dimensions to this method, a process and an object schematic (Meyer et al., 1995). The process schematic displays a process based view on how the activities being examined actually work in the organisation. The object schematic displays the set of objects involved, and their allowable state transitions, in the activities being examined (Aguilar-Saven, 2004). Kim (2003) mentions, as one of the strengths of IDEF3, its ability to record time based process behaviour. Work by Kim (2003) has identified the possibility of using IDEF3 and UML methods in unison to model enterprise information systems. It is the opinion of this author that IDEF3 can provide a business modelling front end to software system implementation diagrams provided by UML.

Petri Nets

Devised by Carl Adam Petri in 1962 (Murata, 1989), Petri Nets are a formal method of process modelling. They allow processes to be described as a graphical network of nodes. Tokens are used within the net to represent state within a process and aid in the active evaluation of processes (Havey, 2005). Van der Aalst and van Hee (2002) mention three extensions made to the basic Petri Net technique: -

- Coloured Petri Net extension.
- Time extension.
- Hierarchical extension. (van der Aalst and van Hee, 2002)

The coloured Petri Net allows for different attributes of a process instance to be represented, each attribute is represented by a different coloured token. Preconditions can be set on nodes (called transitions in Petri Nets) so that the nodes will only 'fire' for certain values or combinations of tokens (Jensen, 1987).

For time extended Petri Nets, tokens are given timestamps, and may only pass through net transitions at the time set by their timestamp. Transitions may also be set to fire after a certain time delay depending on the value of a received token (Murata, 1989).

The hierarchically extended Petri Net allows the nodes of a simple Petri Net to be 'exploded' to reveal a Petri Net of a sub-process. This is a similar approach to the one use by IDEF3 software tools, allowing units of graphs to be broken down into further levels of detail (van der Aalst and van Hee, 2002).

Van der Aalst and van Hee (2002) introduce the concept of workflow nets as a type of Petri Net tailored for workflow modelling. Van Dongen et al. (2005a) also mention the concept of Petri Net soundness which provides an indication of correctness for a Petri Net. Three levels of correctness are put forward: -

- Soundness.
- Generalised Soundness.
- Relaxed Soundness.

(van Dongen et al., 2005a)

Dehnert and Zimmerman (2005) discuss further the notions of soundness in relation to business process models, describing soundness as a measure of whether

a process once complete can be terminated without leaving dead transitions, deadlocks or live locks. Relaxed soundness was introduced as a way to represent a more realistic view of correctness. Unlike the measure of soundness, relaxed soundness does not demand that dead transitions, deadlocks or live locks should not be present as long as each task of the process is part of a correctly terminating sequence. The notion of generalised soundness has been put forward by van Hee et al. (2003), who state that generalised soundness can be achieved by the iterative refinement and combining of sound process models. The iterative and model combining aspects of this measure allow for the recognition of temporal, yet still correct, traces through a process model.

The notion of relaxed soundness has been used in the fitness approach of mining techniques such as Genetic Algorithm (GA) based process mining (van der Aalst et al., 2005).

Due to its formal mathematical definition the Petri net formalisation provides a suitable method for aggregating multiple process instances. Van Dongen and van der Aalst (2005a) propose such a technique, using event process chains as an intermediate stage between process instance acquisition and process aggregation. Juhas et al. (2005) discuss the design of an algorithm to decide if a particular scenario can be executed within a given Petri net.

Event Process Chain (EPC)

Although not part of the techniques outlined by Aguilar - Saven (2004), Event Process Chains provide a way of informally representing a business process (van Dongen and Jansen-Vullers, 2005). An EPC is comprised of three core elements: -

- Functions - a function represents a process activity within an EPC diagram as a box.

- Events – events (known as tasks in wider process mining literature) are used to link functions and describe states of process flow before and after the execution of events. They are represented as hexagons

- Connectors – functions and events are linked together by connectors. Connectors are represented by circles and act as gates containing control flow logic.

Van Dongen and Jansen-Vullers (2005) proceed to describe EPC modelling as a useful method for use in verifying high level models in order to minimise mistakes in the overall modelling exercise.

Neiger and Churilov (2004) describe an extended version of the EPC called the e-EPC. This development of the event process chain method, used in the Architecture of Integrated Information Systems (ARIS) House of Business Engineering Software (Scheer et al., 2004), allows for the inclusion of additional objects such as resources and information relevant to a process. In addition the process model may be linked to an organisational hierarchy chart. Neiger and Churilov (2004) also put forward a framework for use with e-EPCs that recognises goal orientation in the development of business processes.

2.4. Process Mining

The need for companies to learn more about how their processes operate in the real world is a major driver behind the development and increasing use of process mining techniques. The practice of business process mining derives from the field of data mining. Data mining refers to the extraction of knowledge from large data sets though identification of patterns within the data (Witten and Frank, 2005). Data mining practice has been developed and adapted to create the business process mining

techniques that are now being used to mine data logs containing process execution data to reconstruct actual business processes. In addition customised algorithms (and algorithms borrowed from other fields of computing) have been developed specifically to address the needs of process mining specialists. Business process mining techniques use execution logs of business processes. These are typically hosted within business process management (BPM) systems (for example, ARIS), though they may also be accessible though other process-related systems installed within a company (van der Aalst et al., 2003a).

Agrawal et al. (1998) were early pioneers of process mining. Their algorithmic approach to process mining allowed the construction of process flow graphs from execution logs of a workflow application. The discipline of process mining also has its roots in the work of Cook and Wolf (1998a) who attempted to discover software process models from the data contained in event logs. In terms of business process mining van der Aalst et al. (2003a) state that almost any transactional information system can provide suitable data. Van der Aalst et al. (2003b) identify two broad types of workflow meta models. These are graph- and block-orientated models, each with their own language and graphical representation. Aguilar-Saven (2004) adds net-based languages to this definition (with block-oriented models/languages being grouped under the term workflow languages). Van der Aalst et al. (2003b) do not make this distinction between net and graph models describing net-based models, such as Petri Nets as a form of graph-oriented model. The most common form of graph oriented meta-model is the directed graph. Agrawal et al. (1998) were one of the first to use directed graphs in process mining. These authors describe a number of constructs involved in the actual graph. Activities, usually enclosed in boxes or circles, are referred to as vertices and the arrows between the activities, that indicate the direction of flow, are known as edges. Some workflow meta-models may be used to define workflow models as well as act as a language for the display of

process-mining activities. An example of this is provided by Herbst and Karagiannis (2004) and their Inductive Workflow Learning via Examples (InWoLvE) workflow mining system. It is often useful to be able to define an “ideal” workflow template so that mined process models may be compared against it for conformance purposes.

Cook and Wolf’s (1998a) work, while not directly related to that of business process discovery, did examine the use of three statistical analysis methods for use in mining tasks:-

- RNet – A statistical method that examines past behaviour to describe a process state (Das and Mozer, 1994).
- Ktail – This algorithmic method examines future behaviour to describe a potential current state (Feldman, 1972).
- Markov (Markovian approach) – A hybrid statistical and algorithmic approach that looks at both past and future behaviour to define a potential current state (Cook and Wolf, 1998a).

It was the opinion of Cook and Wolf (1998a) that the Markov and Ktail approaches promised the most in terms of process discovery, allowing a process analyst to introduce existing knowledge in order to refine the results.

In later work Cook et al. (2004) extend their findings to allow for the discovery of concurrent models of system behaviour in workflow systems. Concurrent workflows are characterised by simultaneous threads of process execution.

Currently there are a number of techniques that may be used to perform mining of business processes: –

- Genetic Algorithms – Algorithms designed around the process of Darwinian

natural selection.

- General Algorithmic Approach – Custom techniques designed for mining processes developed by individual authors.
- Markovian Approach – An algorithm that examines past and future behaviour to define a potential current state.
- Neural Network – Models the human mind in its ability to 'learn' and then identify patterns in data.
- Cluster Analysis – Divides a group of solutions into homogenous sub groups.

A number of papers exist that detail these techniques. Table 1 highlights the main references concerned with the application of these techniques to business process mining. The compilation of Table 1 has been made possible through the consultation of the following online search tools: Google Scholar, Scopus, Science Direct (Elsevier), Springer Online.

Figure 4 graphs the number of papers found in the area of process mining techniques (based on the papers in Table 1). It is interesting to note that most authors in this area have devised their own algorithmic approach to process mining, 39 percent of the papers in total. Around 12 percent of papers detail the use of soft computing techniques when applied to process mining tasks (papers include those on Genetic Algorithms (GA), Neural Networks and Fuzzy Logic). Petri Net modelling seems to be the most popular notation method with 29 percent of papers detailing its use in the modelling of business processes.

There is a concentration, in the more recent papers listed in Table 1, on the use of non deterministic techniques for process mining. Papers from 2006 and 2007 concentrate on the use of clustering (Greco et al., 2006; Gunther and van der Aalst,

2007). Use is also made of the theory of regions applied to process mining (van der Aalst et al., 2006) and genetic algorithms (Alves de Medeiros et al., 2007c). The use of such non deterministic techniques for process mining is justified by many of these authors in their citing of issues such as noise, the presence of duplicate tasks in the data and event logs that contain low frequency task transitions, where the use of deterministic approaches may produce poor results.

Table 1: Main Techniques Used for Process Mining

Papers	Process Mining Techniques						Modelling Technique Used	
	Data Mining Techniques	Genetic Algorithms	Markovian Approach	Cluster Analysis	Neural Networks	Other Approaches	Event-Driven Process Chains	Petri Nets
van der Aalst et al. (2005a)		x					x	x
van der Aalst et al. (2005b)						x		x
Zhang et al. (2003)						x		
Herbst and Karagiannis (2004)	x							
van der Aalst and Alves de Medeiros (2005)						x		
van der Aalst et al. (2004)						x		x
Schimm (2004)	x			x				
Schimm (2003)	x							
Greco et al. (2005)	x			x				
Weijters and van der Aalst (2001)						x		x
Alves de Medeiros et al. (2004a)		x						x
Agrawal et al. (1998)	x							
Adams et al. (2001)				x				
van Dongen et al. (2005b)						x		
Dustdar et al. (2004)						x		x
van Dongen and van der Aalst (2005a)						x	x	x
van Dongen et al. (2005a)						x	x	x
Maruster et al. (2002a)						x		x
Weijters and van der Aalst (2003)						x		x

	Data Mining Techniques	Genetic Algorithms	Markovian Approach	Cluster Analysis	Neural Networks	Other Approaches	Event- Driven Process Chains	Petri Nets
Maruster et al. (2002b)						x		x
van Dongen and van der Aalst (2004b)						x	x	x
Alves de Medeiros et al. (2003)						x		x
Gaaloul and Godart (2005)						x		
Hammori et al. (2004)	x							
Golani and Pinter (2003)						x		
Cook and Wolf (1998a)			x		x			
Hwang and Yang (2002)	x							
Cook et al. (2004)						x		x
Hwang et al. (2004)						x		
Alves de Medeiros et al. (2004b)						x		x
Chen and Yun (2003)						x		
Greco et al. (2004)				x				
van der Aalst et al. (2002)						x		x
Mannila and Rusakov (2001)			x					
Herbst and Karagiannis (1998)			x					
Cook and Wolf (1998b)						x		
van der Aalst (2005)						x		x
van der Aalst and Song (2004)						x		x
Alves de Medeiros et al. (2005)		x						x
van Dongen and van der Aalst (2004a)						x		x
Alves de Medeiros et al. (2004c)						x		x

	Data Mining Techniques	Genetic Algorithms	Markovian Approach	Cluster Analysis	Neural Networks	Other Approaches	Event- Driven Process Chains	Petri Nets
van Dongen and van der Aalst (2005b)						x		
Rozinat and van der Aalst (2006)						x		x
van der Aalst et al. (2006)						x		x
Li and Feng (2007)			x					
Gunther and van der Aalst (2007)				x		x		x
Lamma et al. (2007)						x		
Greco et al. (2006)				x		x		
Alves de Medeiros et al. (2007a)				x		x		x
Ferreira et al. (2007)				x				
Nikovski and Baba (2007)						x		
Alves de Medeiros et al. (2007c)		x						x

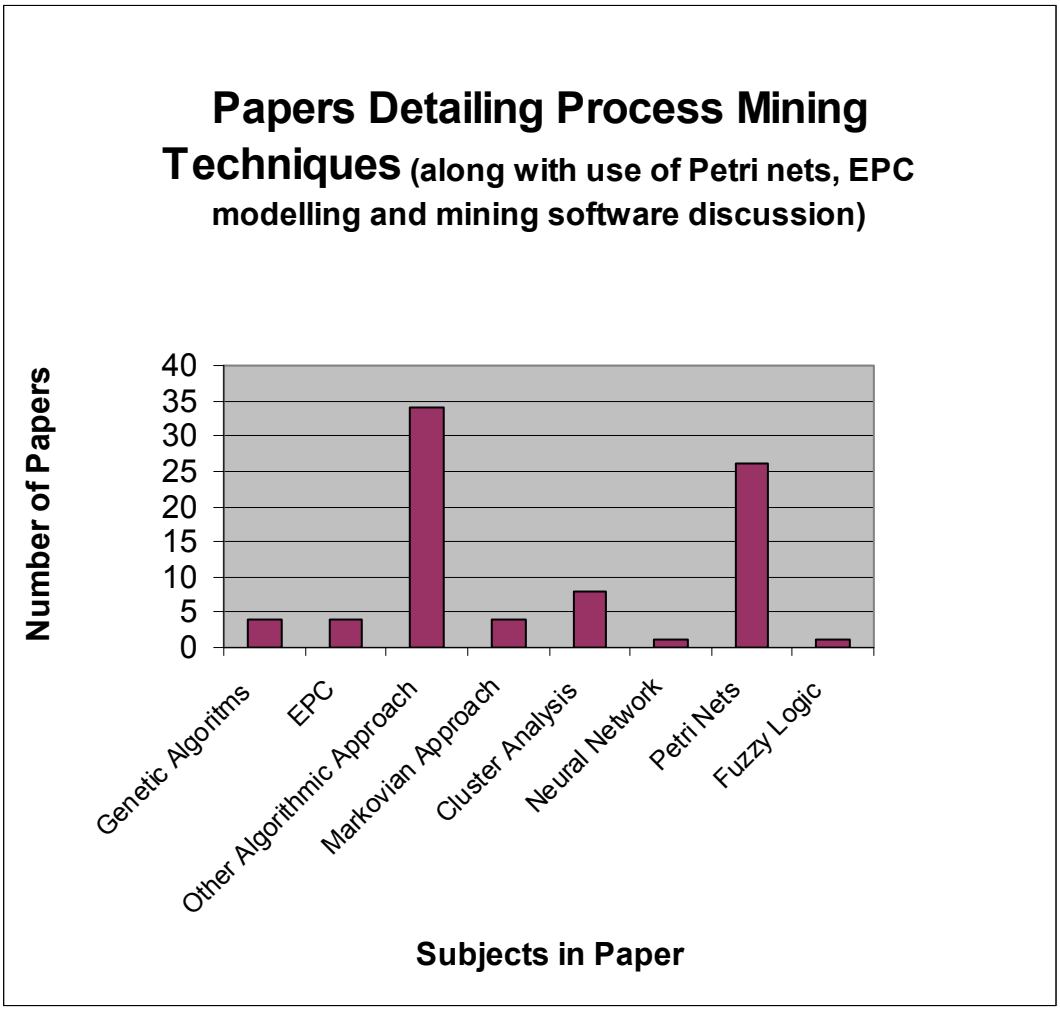


Figure 5: Papers Detailing Process Mining Techniques (based on Table 1)

Much of the work reported in the area of process mining deals with the problem of discovering a process model from a set of process instances. In this vein, such discovery, using a directed graph, a finite state machine or a Petri net for representing process instances, aims at discovering a process model that best describes the set of process instances (van der Aalst et al., 2003b). However, it may not always be possible to assume the existence of a single process model to which all process instances comply; here the goal must be to discover a set of frequently occurring temporal patterns. An example of the work in this area is the paper of Hwang et al. (2004) who model frequently occurring temporal patterns from event-based data. Dong and Li (1999) describe a method of identifying emerging patterns from data. While their

work does not consider business processes it may nevertheless be applied to this domain, identifying changes in the way processes are being carried out in a live environment. Other notable papers in this area include Hwang and Yang (2002) who present an algorithm that they claim can produce process flow graphs in a more detailed and efficient manner, an algorithm which builds upon the process-mining activities of Agrawal et al. (1998). A number of software tools exist for process mining. Herbst and Karagiannis (2004) describe their InWoLvE workflow mining tool and detail a number of process mining algorithms incorporated in the software. The tool deals with four classes of workflow models :-

- Injective Sequential
- Injective Parallel
- Non - injective Sequential
- Non - injective Parallel

The term injective relates to the uniqueness of the activity nodes of the process model. An injective problem class means that each node of the process is unique (there are no repetitions of nodes in the process model). A sequential problem class (process model) does not contain any splits or joins (that is AND or XOR split and join constructs), unlike a parallel problem class. The ability of this tool to mine process models with non unique nodes is an advantage. The mining approach taken by this tool is inductive, using rules and success criteria to enable the identification of processes from data. The data to be mined must be in the form of an ASCII string showing tasks within each process traces. The visualisation of graphs with this tool is through a custom block form notation called ADONIS™ (ADONIS is a registered trademark of BOC GmbH, Vienna, Austria). The InWolvE tool has a limited amount of success in the mining of simple loops (mainly short loops and self referential loops). Duplicate tasks can be mined with this tool as it has a pre processing filter that is used to differentiate multiple tasks sharing the same name before mining takes place. Complex parallel processes may also be mined with this approach.

The ProM framework, put forward by van Dongen et al. (2005b) is a tool to support process mining activities. This software provides auxiliary 'housekeeping' functions such as xml file handling, graphical user interface and process flowchart display capabilities. This suite is designed to host process mining approaches as plug-in units of code; ProM does not perform process mining itself. In terms of soft computing techniques authors such as Alves de Medeiros et al. (2005) have explored the use of evolutionary computing techniques, in the form of genetic algorithms, in the task of process mining. In terms of neural network techniques only Cook and Wolf (1998a) have explored this area in relation to process discovery. It can be seen from Figure 6 that the number of papers published in the area of process mining has increased significantly in the past few years (peaking in 2004). Of particular interest is the recent publication of papers concerning the use of soft computing techniques in process mining. Many of the process-mining techniques employed by researchers involve the inclusion of heuristics at some point in order to initiate and "fine tune" the identification of processes within data. Alves de Medeiros et al. (2003) argue that there are problems in using heuristics in that very complex process constructs cannot be handled correctly by algorithms employing heuristic techniques. Instead these authors encourage the examination and use of the "α" algorithm which has been designed to work, without heuristics, for a defined set of conditions and process constructs (van der Aalst et al., 2004). It is said by van der Aalst et al. (2004) if a process can be described as a Workflow net (WF-Net) it can be mined by this technique (WF-Nets are a type of Petri net used to describe process models, described in van der Aalst et al. (2004)). Though, the technique cannot mine loops or process data logs containing noise. The approach is available as a plug-in for the ProM process mining suite (with ProM providing the graph visualisation functionality and XML parser required to interpret event logs). The algorithm has, more recently, been modified by Alves de Medeiros et al. (2004b) to cope with mining problems such as short loops though this version of the algorithm still cannot mine noisy data. Mining problems such as these are discussed in more detail in a later section of this chapter.

At present, both genetic algorithms and neural networks have been used to perform process-mining tasks. Cook and Wolf (1998a) conducted some work with a neural network-based method called RNet.

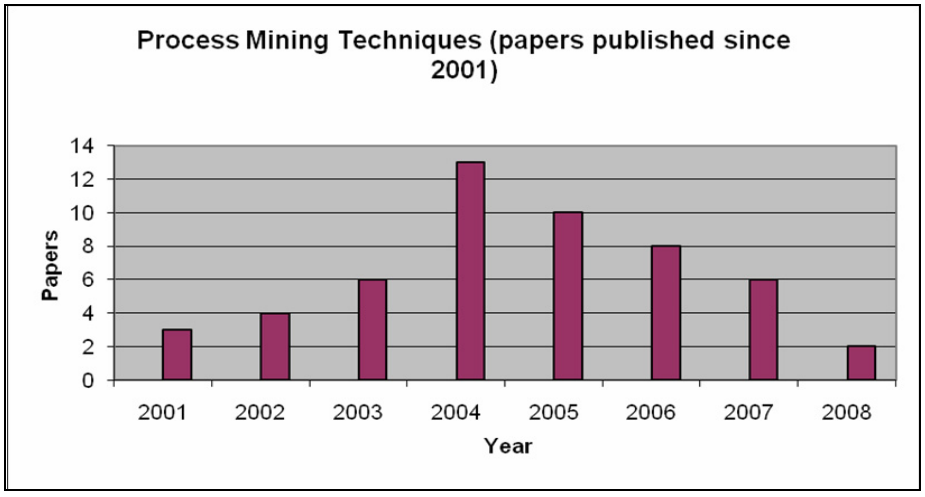


Figure 6: Papers Published in the Area of Process Mining Since 2001

Such a method "learns" how to recognise process patterns in data through a feedback mechanism; learned errors are passed back in the system to correct the method. Cook and Wolf (1998a) state that at the time of their research this method was still in the early stages of development. Genetic algorithms have been used more recently in process-mining activities. Alves de Medeiros et al. (2004a) have investigated the use of genetic algorithms in the area of process mining, discovering their applicability in the mining of noisy event logs. This technique allows for process patterns to be represented as chromosome strings.

Van der Aalst et al. (2005a) claim that process-mining problems such as hidden tasks and non-free choice constructs can also be addressed by genetic algorithms. The technique used by these authors makes use of causal matrices to represent the relations between tasks of a process. The matrices are designed to show if there is a causal relationship between an activity and the other tasks of a process and display the expected output for each activity. Alves de Medeiros et al. (2005) point out that more research in the genetic algorithm approach is necessary to ensure that the process models mined by using this approach always reflect the behaviour found in process

logs, as current algorithms tend to allow extra behaviour not found in logs. Table 1 details a number of papers that address process mining problems using soft computing techniques. As can be seen from this table a number of papers employ a genetic algorithm approach too partially or fully solve current process mining problems. However, many process mining problems are still solved using methods derived from data mining.

2.5. Process Mining Problems

The main issues still encountered in the area of business process mining are outlined by van der Aalst and Weijters (2004) as: -

- Noise – Logged data may be incorrect or incomplete providing problems when data is being mined.
- Hidden Tasks – Tasks that must exist but cannot be found in the data.
- Duplicate Tasks – Two process nodes may refer to the same task.
- Non free choice constructs – Are controlled choices that depend on choices made in other part of the process model.
- Mining loops – A process may be executed several times; loops may be simple involving one or more tasks or are more complex.
- Different perspectives – Process tasks may be appended with additional information for mining purposes.
- Delta Analysis – Comparison of process model and reference model to check for similarity/disparity.
- Visualising Results – The results of process mining may be presented in graphical form in terms of a management panel.

- Heterogeneous Results – Access to information systems based on different platforms.
- Concurrent Processes – The mining of processes occurring at the same time.
- Local/Global search – Local strategies restrict the search space and are less complex, global strategies are complicated but have a better chance of finding the optimal solution.
- Process re-discovery – The selection of a mining algorithm which can rediscover a class of process models from a complete workflow log.

(van der Aalst and Weijters, 2004)

In the opinion of the author of this thesis the skew-ness of the data in an event log could be included in the above list as an additional problem. Being able to detect if the data is skewed in a particular direction (or pattern) may help in areas such as process anomaly detection (such as detecting instances of fraud and staff adherence to a set process). The dynamism of the processes being recorded may also be presented as a problem encountered in process mining. Certain processes may be highly dynamic, changing on a regular basis. In such a case a process mining approach would need to be able to adapt to this providing multiple results showing the 'evolution' of the process. To date Hwang et al. (2004) are the only authors who have attempted the mining of complete temporal process models.

Unlike the field of data mining, most process mining techniques do not make use of data cleansing routines before actual mining takes place. The only pre condition for most mining techniques is that the event log should be in a format readable by the mining technique (in most cases this means the event log should be presented in some variety of XML or a proprietary format such as the MXML format used by the mining techniques hosted in the ProM process mining suite (van Dongen et al., 2005). The mining technique of Herbst and Karagiannis (2004) utilises a basic form of cleansing in

that duplicate tasks are filtered out of an event log before mining begins.

A number of papers exist that attempt to address many of the most common process mining problems. A cross section of the most important papers is shown in Table 2.

Table 2: Papers Detailing Process Mining Problems

	Noise	Hidden Tasks	Duplicate Tasks	Non Free Choice Constructs	Mining Loops	Different Perspectives	Delta Analysis	Visualising Results	Heterogenous Data Sources	Concurrent Processes	Local Global Search	Process Rediscovery
van der Aalst (2005)							x					
Greco et al. (2004)											x	
Cook and Wolf (1998b)										x		
van der Aalst et al.. (2002)				x								x
van Dongen et al. (2005)								x				
Weijters and van der Aalst (2003)	x				x					x		x
Golani and Pinter (2003)										x		
Cook and Wolf (1998a)	x							x				
Alves de Medeiros et al. (2004c)					x			x		x		
Zhang et al. (2003)										x		
Alves de Medeiros et al. (2004b)	x				x							
Gaaloul and Godart (2005)	x									x		
Alves de Medeiros et al., 2003					x							
van Dongen and van der Aalst (2004a)								x				
Alves de Medeiros et al. (2004a)					x							
van Dongen and van der Aalst (2005a)	x											
Agrawal et al. (1998)	x		x									
Alves de Medeiros et al. (2005)	x	x		x							x	
Schimm, 2003					x							

	Noise	Hidden Tasks	Duplicate Tasks	Non Free Choice Constructs	Mining Loops	Different Perspectives	Delta Analysis	Visualising Results	Heterogenous Data Sources	Concurrent Processes	Local Global Search	Process Rediscovery
Weijters and van der Aalst (2001)	x				x					x		
van Dongen and van der Aalst (2005b)									x			
Hammori et al. (2004)								x				
Cook et al. (2004)	x									x		
Hwang and Yang (2002)	x											
van der Aalst et al.. (2005a)	x	x	x	x	x			x		x	x	
van der Aalst and Alves de Medeiros (2005)	x	x	x		x							
Herbst and Karagiannis (2004)			x		x			x				
Dustdar et al. (2004)								x				
van der Aalst and Song (2004)						x						
Cook and Wolf (1998b)										x		
Schimm (2004)										x		
Rozinat and van der Aalst (2006)	x	x	x		x	x						
van der Aalst et al.. (2006)			x	x	x					x		
Li et al. (2008)	x									x		
Gunther and van der Aalst (2007)					x							x
Lamma et al. (2007)					x			x		x		
Greco et al. (2006)	x			x	x					x		
Alves de Medeiros et al. (2007c)	x	x		x	x					x		

	Noise	Hidden Tasks	Duplicate Tasks	Non Free Choice Constructs	Mining Loops	Different Perspectives	Delta Analysis	Visualising Results	Heterogenous Data Sources	Concurrent Processes	Local Global Search	Process Rediscovery
Ferreira et al. (2007)	x											x
Nikovski and Baba (2007)										x		
Li and Feng (2007)	x									x		
Alves de Medeiros et al. (2007a)	x				x					x		

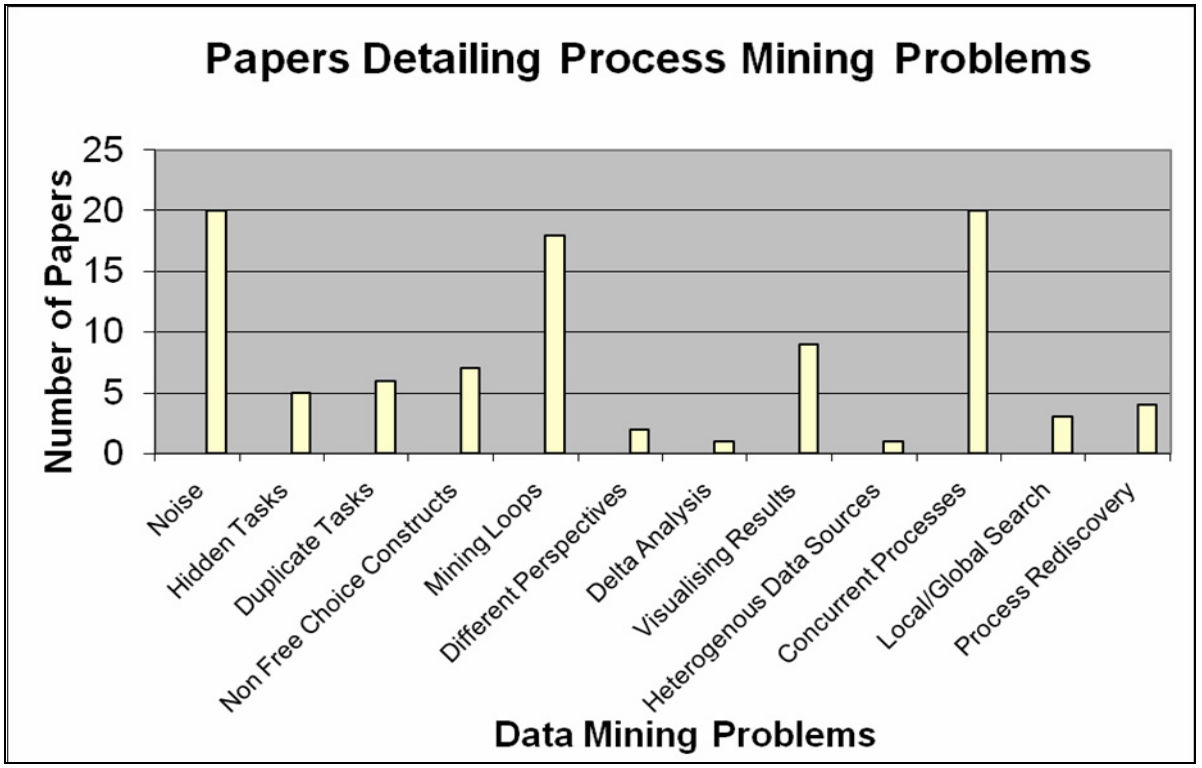


Figure 7: Papers Detailing Process Mining Problems

Figure 7 details the papers available that address process mining problems (based on the papers in Table 2). One of the most regularly tackled problems is that of noise, with 12 papers detailing its mitigation. Mining loops and concurrent processes are also dealt with by a substantial amount of papers. Problems of heterogeneous data sources and mining perspectives are not so well researched with only two percent of papers each. There has been an increase in the number of papers addressing process mining problems in recent years. Many of these papers involve the use of heuristic techniques to solve mining loop and noise problems. The problem of noise in process data has been described by van der Aalst and Weijters (2004) as the presence of “incorrectly logged information” within a process log. Van der Aalst and Weijters go on to mention the value of algorithms that may discern exception from normal process execution traces.

Agrawal et al. (1998) were one of the first authors to address the problem of noise in event logs (event logs containing incomplete or inaccurate process traces are considered to be noisy) , applying an algorithmic approach to this issue. Cook and Wolf (1998a) also examine a range of software process discovery algorithms that can deal with noise. Weijters and van der Aalst (2003) employ heuristics in their mining approach and are able to mine processes in the presence of noise. A set of rules determine the precedence between tasks and overall task sequences. Later work by Alves de Medeiros et al. (2005) employs genetic algorithms and heuristics to address mining problems involving noise and other issues. The visualisation of process mining results is another issue that has been addressed by several authors. The issue concerns the display of mining results to the user, especially the graphical representation of mined processes by models. The clustering approaches of Greco et al. (2006) and Alves de Medeiros et al. (2007a) rely on the heuristics of Weijters and van der Aalst (2003) to perform noise free process mining.

Cook and Wolf (1998a) were among the first to address process mining visualisation issues through the use of a graphical interface. They provide a method of displaying process models derived from mined process data. One of the most recent

developments in the visualisation of business processes and the practice of process mining is outlined by van Dongen and van der Aalst (2005a). The ProM framework is a tool to support process-mining activities (van Dongen and van der Aalst, 2005a). The framework is a “pluggable” software tool able to support many sub tools ranging from graphical process display tools to mining algorithms and analysis tools. The framework encourages a unified approach to be adopted in the design of process mining software so users are presented with a generic interface capable of hosting a range of process mining and modelling activities. Work on mining loops includes the contributions of Alves de Medeiros et al. (2004b) who addressed the mining of short loops from event logs. This work is complemented by a paper on the mining of mobile (telecommunications) systems by Alves de Medeiros et al. (2004c) which also deals with mining loops. The work of van der Aalst et al. (2006) on transition systems and regions also deals with loops. Their approach deals with loops in two ways. One by creating separate transitions for each loop possibility based on their occurrences in the log, and the second by merging transitions that have the same output task.

The mining of concurrent processes has been explored by Schimm (2004) who concentrates on the task of mining exact workflows from event-based data. Cook et al. (2004) examine the problem of mining concurrent workflows in greater depth. These authors have developed algorithms to detect the presence of concurrent behaviour in event logs. Four metrics are used to detect concurrency in process logs in the work of Cook and Wolf (1998b): -

- Entropy
- Event Type Counts
- Deciding Causality
- Periodicity

Entropy gives a measure of the amount of information contained in an event sequence and the number of occurrences of tasks within a process. It is possible to look for particular entropy values that may suggest a concurrent branching of a

process in the event data. Event type counts are used to determine the likelihood of a fork occurring joining, branches of a process together. The count of event types following a fork should be the same as the count at the fork occurrence. This metric is often used alongside the entropy measure for process branching. Causality of tasks must be decided when examining a concurrent process in order to decide if certain tasks are sequentially causally related in the presence of multiple threads of execution within a process. A summation of the event frequencies is used as part of this metric. The fourth metric, periodicity, examines the probability of an event sequence occurring, by examining the period of occurrence of event sequences within the event log. In the mining technique of van der Aalst et al. (2006), potential concurrent tasks, not observed in the event logs, are added in order to mine complex parallel constructs.

The issue of process rediscovery is fundamental to process mining and is addressed by van der Aalst et al. (2002). These authors explore a class of business processes, which in their view are proven to be rediscovered from workflow logs by their approach (the α algorithm). Process event sequences can be identified from a process log by looking for sequences of tasks and measuring the frequency of these event sequences in a log. The mining algorithm outlined by these authors cannot model transitions that share an input place. The algorithm is also not able to detect implicit tasks in a process.

One area in particular that has scope for further investigation is that of process mining perspectives. Van der Aalst and Song (2004) have explored the mining of business processes from a social perspective, where staff interactions in business processes can be mapped out. This involves appending extra information to process event logs. It is possible that qualitative information relating to certain parts of a business process (or set of processes) may be captured in event logs for analysis. Few papers have been published detailing the comparison of mined business process models with “ideal” models. Van der Aalst (2005) outlines a node mapping technique for such a delta analysis task.

In such methods syntactical differences between an ideal process model and a model of an actual process are highlighted, though van der Aalst (2005) also describes a way of taking into account behavioural differences. The method put forward examines two models from the point of view of behaviour inheritance, and is able to construct a model based on all available executions of a process showing only the tasks that are common to both processes. The problem of duplicate activity instances within process data could also benefit from further investigation. Herbst and Karagiannis (2004) describe a method for removal of repeated nodes using counters. Their technique differs from Agrawal's (1998) approach in that mining parameters can be set to force a high degree of specialisation in mined nodes. However, the authors concede that further techniques need to be considered in order to totally eliminate this problem.

Table 3 displays a list of common mining problems and shows which techniques can be used to address them. Genetic algorithm (Alves de Medeiros et al., 2005), clustering (Greco et al., 2006) and transition/region (van der Aalst et al., 2006) approaches are used to address many mining problems, though data mining-based and custom approaches are still relevant for the most common problems such as noise and mining loops. Neural network and Markovian-based techniques (Cook and Wolf, 1998a; Li and Feng, 2007) have been used to solve problems of noise in mining data though their wider application has not been explored in great depth. Many problems such as delta analysis (van der Aalst, 2005) and mining perspectives (van der Aalst and Song, 2004; Rozinat and van der Aalst, 2006) require a custom built algorithm. While many of the process mining problems can be addressed by a combination of modified data mining approaches and custom built algorithms there is no single approach that can address all of the problems encountered in process mining; many custom process mining algorithms exist though they tend to solve only one or two problems. The approaches with the widest application to mining problems are that of genetic algorithms, transition/region technique and clustering algorithms. In Table 3 the approaches listed under the Data Mining Based heading are known to be largely based on existing or modified approaches more commonly used for data mining tasks.

Table 3: Papers matched to common process mining problems

	Data Mining Based	Neural Networks	Genetic Algorithms	Markovian Approach	Other Approaches
Noise	Agrawal et al. (1998) Hwang and Yang (2002)	Cook and Wolf (1998a)	Alves de Medeiros et al. (2005) Alves de Medeiros et al. (2007c) van der Aalst et al. (2005a)	Cook and Wolf (1998a) Li and Feng (2007)	Gaaloul and Godart (2005) Cook et al.. (2004) Weijters and van der Aalst (2003) Weijters and van der Aalst (2001) van der Aalst and Alves de Medeiros (2005) Greco et al. (2006) Ferreira et al. (2007) Alves de Medeiros et al. (2007a)
Hidden Tasks			Alves de Medeiros et al. (2005) Alves de Medeiros et al. (2007c)		Rozinat and van der Aalst (2006)
Duplicate Tasks	Herbst and Karagiannis (2004) Agrawal et al. (1998)		Alves de Medeiros et al. (2005) Alves de Medeiros et al. (2007c) van der Aalst et al. (2005a)		van Dongen and van der Aalst (2005a) Rozinat and van der Aalst (2006) van der Aalst, et al. (2006)
Non Free Choice Constructs			Alves de Medeiros et al. (2005) Alves de Medeiros et al. (2007c) van der Aalst et al. (2005a)		van der Aalst et al. (2002) Weijters and van der Aalst (2001) Greco et al. (2006) van der Aalst et al. (2006)
Mining Loops	Herbst and Karagiannis (2004) Schimm (2003) Schimm (2004)		Alves de Medeiros et al. (2005) Alves de Medeiros et al. (2007c) van der Aalst et al. (2005a) Alves de Medeiros et al. (2004a)		Weijters and van der Aalst (2001) Weijters and van der Aalst (2003) Alves de Medeiros et al. (2004b) Alves de Medeiros et al. (2004a) van der Aalst and Alves de Medeiros (2005) Gunther and van der Aalst (2007) Lamma et al. (2007) Greco et al. (2006) van der Aalst, et al. (2006) Alves de Medeiros et al. (2007a)
Different Perspectives					van der Aalst and Song, 2004 Rozinat and van der Aalst, 2006
Delta Analysis					van der Aalst, 2005
Visualising Results	Herbst and Karagiannis (2004) Hammori et al. (2004)		Alves de Medeiros, et al. (2005) Alves de Medeiros, et al. (2007c) van der Aalst et al. (2005a)		Dustdar et al. (2004) Alves de Medeiros et al. (2004b) van Dongen and van der Aalst (2004a) Alves de Medeiros et al. (2004b) van Dongen et al. (2005b) Lamma et al. (2007)
Heterogeneous Data Sources					van Dongen and van der Aalst, 2005b
Concurrent processes	Schimm (2004)		Alves de Medeiros, et al. (2005) Alves de Medeiros, et al. (2007c) van der Aalst et al. (2005a)	Li and Feng (2007)	Golani and Pinter (2003) Alves de Medeiros et al. (2004b) Weijters and van der Aalst (2001) Lamma et al. (2007) Greco et al. (2006) Nikovski and Baba (2007) van der Aalst et al. (2006) Alves de Medeiros et al. (2007a)
Local/Global Search			van der Aalst et al.. (2005a)		Greco et al. (2004) Greco et al. (2006)
Process Re-discovery					van der Aalst et al.. (2002) Weijters and van der Aalst (2003) Gunther and van der Aalst (2007) Ferreira et al. (2007)

(Tiwari et al., 2008)

2.6. Discussion on Computational Intelligence Techniques in Process Mining

2.6.1. Outline of Computational Intelligence (CI)

Soft computing comprises a number of techniques that may be used to mimic human based reasoning in the solving of complex problems. There are three main techniques used in the field of soft computing :-

- Fuzzy Logic
- Neural Networks
- Evolutionary Computing

Fuzzy logic concerns the calibration of vagueness through the use of a fuzzy set (a set with fuzzy boundaries). An element belongs to a fuzzy set with a degree of membership. An element can represent a certain decision or answer which may be selected (Negnevitsky, 2005). The use of linguistic variables is important in defining fuzzy sets. Such a variable is normally used in the definition of fuzzy rules such as the following rule defined in relation to a car - *If speed is slow then stopping distance is short*. Included with linguistic variables is the concept of fuzzy set qualifiers called hedges. Hedges modify the shape of fuzzy sets by introducing terms that quantify and add detail to fuzzy rules (Negnevitsky, 2005, Zadeh, 1973).

Fuzzy inference is the process of mapping so called crisp inputs (numerical values) into outputs using fuzzy sets. In general the following four steps are employed in the process of fuzzy inference (when using the common Mamdani inference style) (Negnevitsky, 2005; Cox, 2005):

- Fuzzification – Determine the degree the crisp input belongs to each fuzzy set being considered.

- Rule Evaluation – Apply the fuzzy rule sets and obtain a single value for the fuzzified inputs for each rule.
- Aggregation of the Rule Outputs – The outputs of all of the rules are combined to a single fuzzy set.
- Defuzzification – Obtain a single crisp value from the fuzzy set.

(Negnevitsky, 2005; Cox, 2005)

An examination of fuzzy soft computing techniques is relevant to this thesis as such techniques are relevant to process mining activities when used in combination with Neural networks and potentially in the area of process conformance.

A neural network has been described as a model of reasoning based on the human brain. A neural network is capable of learning through a feedback mechanism. A neural net is composed of many neurons (processing units) linked together, with each link containing a numerical weight. A neural network can adjust each of the weights to reflect the conditions found in its environment. A training phase is used to update the weights in a network to reflect a corrected view of the environment. Outputs from a network are checked for correctness and errors are then fed back into the network to effect suitable changes in the weights of certain links. Neural networks are potentially useful in the task of process mining, learning to identify certain process patterns from noisy data (Negnevitsky, 2005).

Evolutionary computing in the form of genetic algorithms has been used to select appropriate solutions to problems based on the process of natural selection. Problem variables are represented as chromosome strings (usually binary number strings). A well defined problem must be present along with a set of candidate solutions for a genetic algorithm to proceed successfully in the selection of an overall solution. Other concepts from biological evolution have been integrated into the genetic algorithm paradigm. The concept of crossover is represented by exchanging parts of two selected chromosome strings. In addition the concept of mutation is represented by

changing a single digit in the chromosome string. Once an initial population of solution chromosomes has been created the following steps are executed in an iterative fashion: –

- Calculate the fitness of each chromosome
- Select a pair of chromosomes for mating
- Perform a crossover
- Perform a mutation
- Place resulting chromosomes in new population
- Replace the current population with the new population

(Negnevitsky, 2005; Goldberg, 1989)

Normally this process is repeated for a pre-defined number of 'generations', after which the best chromosomes are selected and examined. Only the fittest chromosomes should survive this process meaning that every generation is producing a population that has a higher overall fitness than that of the previous generation. Genetic algorithms are only just starting to be applied in the task of process mining.

2.6.2. Soft Computing Techniques in Process Mining

At this present time, both genetic algorithms and neural networks have been used to perform process mining tasks. Cook and Wolf (1998a) conducted some work with a neural network based method called RNet. Such a method 'learns' how to recognise process patterns in data through a feedback mechanism, learning errors are passed back in the system to correct the method. Cook and Wolf (1998a) state at the time of their research this method was still in the early stages of development.

Genetic algorithms have been used more recently in process mining activities. De Medeiros et al. (2004a) has investigated this use of genetic algorithms in the area of

process mining, discovering their applicability in the mining of noisy event logs. This technique allows for process patterns to be represented as chromosome strings. Van der Aalst et al. (2005a) claim that process mining problems such as hidden tasks and non free choice constructs can also be addressed by genetic algorithms. Hidden tasks are tasks that are not recorded in the event log but are still part of the process being mined, and their presence may be inferred by certain mining algorithms. Non free choice constructs men that a choice between two tasks is not made at a directly preceding task but at a task at another part of the process (van der Aalst and Weijters, 2004). The technique used by these authors makes use of causal matrices to represent the relations between tasks of a process. The matrices are designed to show if there is a causal relationship between an activity and the other tasks of a process and display the expected output for each activity.

Alves de Medeiros et al. (2005) point out that more research in the genetic algorithm approach is necessary to ensure that the process models mined by using this approach always reflect the behaviour found in process logs, as current algorithms tend to allow extra behaviour not found in logs. Heuristic techniques such as those of Weijters and van der Aalst (2001) have shown promise in the field of process mining, along with clustering and region mining. In the author's opinion it is the largely unexplored area of soft computing and evolutionary techniques that holds the most potential for further research.

2.6.3. Genetic Programming

According to Banzhaf et al. (1998), the work of Smith (1980) was instrumental in the development of the precursors to Genetic Programming (GP), working on a variant of a classifier system that acted on a population of variable length programs. Though it is Koza (1994) who is credited with the inception of GP as a technique in its own right (Banzhaf et al., 1998). Koza (1994) puts forward the argument that genetic programming extends the concept of the genetic algorithm so that populations of computer programs can be generated over time.

Koza also states that there are 'five major preparatory steps' that need to be taken account of when attempting to develop a GP application:

- Set of terminals
- Set of primitive functions
- Fitness measure
- Parameters for controlling the run
- Termination criterion

The set of terminals refers to possible inputs to each of the computer programs in a generation. The function set could be any mathematical, arithmetic, or logical function that performs an action. A fitness measure is required to assess how well a computer program will solve a certain problem or a set of problems. Criteria such as the population size and the maximum number of generations that GP must produce are both valid controlling parameters for this technique. The termination criterion for a GP may well be a certain fitness value or a pre defined number of generations that must be produced by the approach.

Although the way in which individuals may be represented in GP is open and flexible, there are three common modes:

- Tree Based Representation.
    - This is a way of representing an instruction set in a tree formation. Instructions are contained in nodes and joined together by branches. Each branch has an IF/THEN execution instruction attached to it. A path may be followed down a particular set of braches in the parsing/running of a particular tree program.
- Linear Representation.

- Linear representation is a chain of instructions that are processed in a left to right or top to bottom fashion.

- Graph Based Representation.

  - Graph based individuals may be processed in a top down fashion assuming a directed graph. Like a tree structure there may be splits in the path of processing (branches in tree based GP) but these splits must come together at a join at some point in the graph. Each graph has a single start and end point. Each graph contains tasks (which may contain instructions) that are linked together by edges. Such edges may be processed together in which case they are in an AND relationship or executed on an OR/XOR choice basis.

(Banzhaf et al., 1998)

Genetic Programming is generally known for the generation of either tree structures or linear programming code (Koza, 1992) (Daida et al. (2005) give guidance on the visualisation of tree structures). However, GP has been used for generating graphs for Artificial Neural Networks (Hornby, 2006), bond graphs (Wang, et al., 2005) electronic circuits (Miller and Banzhaf, 2003), and algorithm structures (Shirakawa et al., 2007). Genetic Network Programming (Hirasawa et al., 2001) deals with the evolution of graph structures in which the numbers of nodes and their functional behaviour is fixed. Graph structured program evolution (GRAPE) as introduced by Shirakawa et al. (2007) is a GP technique using graph structures for generating computer programs with branches and loops. GRAPE uses a linear string of integers as its genotype, the graphs, that represent program flow, must therefore be built from this intermediate representation. This technique is able to avoid the problem of bloat (common in tree based GP where an individual may grow in an uncontrolled fashion (Banzhaf et al., 1998) as a fixed length is defined for the string of integers that is used to represent individuals. Problems such as list retrieval and the Fibonacci sequence have been solved using this approach. The GRAPE technique is similar to another form of graph based GP namely Cartesian GP. The work of Miller (1999) introduces

Cartesian Genetic Programming as a graph based approach (work that is used later on in Miller and Banzhaf (2006) to evolve electronic circuits). This technique also uses a linear string of integers to represent individuals from which graphs may be built. Most Cartesian GP applications rely on mutation rather than recombination to provide variety in a population (Miller and Harding, 2008). According to Koza (1992) there are seven ways to create a new population of individuals in a GP application, the two most important are reproduction and crossover (known as the primary operators (Koza, 1992)). The remaining five operators (known as secondary or optional operators (Koza, 1992)) are:

- Mutation.
- Permutation
- Editing
- Encapsulation
- Decimation

Reproduction involves the copying of individuals, from a preceding population, directly into a new population, without modification by crossover or mutation (Banzhaf et al., 1998). Only a percentage of individuals from an existing population are crossed over or mutated to produce altered individuals for a new population; many individuals are just placed in the new population. With an elitist strategy a number of the fittest individuals from an existing population are copied over directly to a new population without modification. Crossover is a way of swapping one part of an individual with another. Two parent individuals are selected from the population. A common crossover point, found in the structure of both individuals, is selected at random. Material below the crossover point in both individuals is then swapped. In the case of linear GP, linear sections of code are swapped between individuals. The result is the production of two new offspring that may be placed in a new population. A fitness measure is used to score each individual in a population. Ideally only two individuals (the parents), who

score highly, in terms of fitness, are selected for crossover. For crossover between two graph based individuals Banzhaf et al. (1998) relay the work of Teller (1996) who states that graph individuals must be divided up into node sets with labelled edges. Fragments, sections of each graph, can then be identified and swapped between individuals.

Mutation is another way of altering an existing individual to create a new individual. Banzhaf et al. (1998) describe the way mutation works in tree based GP as the random selection and replacement of a subsection (sub-tree) of that tree with a randomly generated subsection (sub-tree). Mutation occurs on an individual basis and does not involve the swapping of material between individuals. Mutation should occur at a much lower level than crossover due to its ability to introduce radically different individuals.

Permutation, according to Koza (1992), acts as an inversion function reversing a number of characters between two points in an individual (the origin of the operator stems from the GA research area). Permutation just act on a single individual at one time (asexual recombination). This operator can be used on high fitness individuals to alter their genetic material to find new combinations that may increase the fitness of those individuals (Koza, 1992).

The editing operator allows for the application of certain predefined rules to each individual in a population. Like permutation it is a form of asexual reproduction involving one individual at a time (Koza, 1992). Koza (1992) illustrates the concept of editing in terms of the modification of tree based GP individuals, where the instructions that make up the tree based program are edited in line with a given rule set. The frequency with which the rule set is applied within the GP application (such as every generation for example) can be controlled by a parameter set at the start of a run.

The encapsulation operator is described by Koza (1992) as a method of identifying useful sub-trees in individuals (when using a tree based representation with GP). A point in an individual is selected at random. The sub-tree located at that point is removed and a function is created in the individual to allow a reference to be made to the extracted sub-tree. A mutation operator can then be use to insert a function in an

individual that gives a reference to the sub-tree. This form of code preservation allows for the definition of 'building blocks' which may be used in the creation of new generations (Koza, 1992). Encapsulation is not affected by the use of crossover (which can be quite a destructive operator) in that the tree to be preserved is abstracted from the individual.

The decimation operator is used when the overall fitness of a population is very low and a more expedient method than provided by crossover is required to try and raise the average fitness of individuals (Koza, 1992). This operator allows a percentage of the population to be preserved while the rest are destroyed. Koza (1992) gives an example of this operators use in that with 10% of the population to be preserved ten times the population is required at the start of the run than the population level required at the end.

The quality of the result from a GP run may be evaluated in a number of ways. The fitness level and the level of complexity in an individual may be used as a measure (Banzhaf et al., 1998). In addition to this Banzhaf (1998) refers to Iba et al. (1995) and their work on Minimum Description Length (MDL) fitness function for use with tree based GP applications. MDL limits the size trees can grow to and provides a more effective way to measure fitness imposing a trade-off between the amount of errors in an individual and the level of detail it contains (Iba et al., 1995). The reproductive cycle of the GP approach is illustrated below in Figure 8.

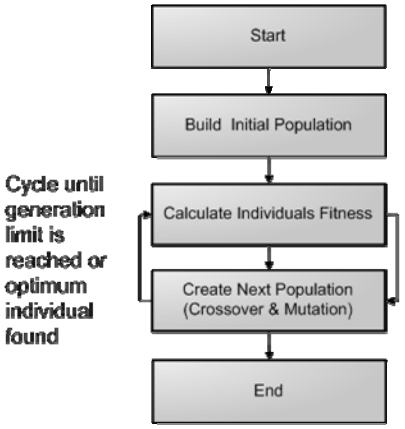


Figure 8: The GP Reproductive Cycle

Some of the most recent developments in GP include the use of hyper-heuristics (a hyper-heuristic acts on sets of heuristics that are either provided as is or within a meta-heuristic technique, such as GP (Ozcan et al., 2008)) to evolve Pareto optimal heuristic rules for the solution of the knapsack problem (Kumar et al., 2008). Other developments include the use of context sensitive crossover in GP. This technique has been demonstrated in tree based GP by Majeed et al. (2007a) and involves the identification of the best crossover point in tow given individuals. In effect multiple possible crossover points are tested for their fitness and the point with the highest fitness is selected as the crossover point (Majeed, et al., 2007a). Similar work on context sensitive mutation has also been carried out by Majeed et al. (2007b). Context sensitive mutation operates by replacing a sub tree in an individual with a subtree thought to be beneficial; selected from a repository of such subtrees (Majeed et al., 2007b).

2.7. Future Areas of Research for Process Mining

The latest process mining techniques are looking towards the goal of being able to mine not just more complex processes but to gain more understanding on the interaction of the processes in terms of the parameters that may flow between the tasks and the wider implications of these flows (Alves de Medeiros, et al., 2007b). There is also a concentration on breaking down processes into groups of features which may be modelled as subgraphs (Greco et al., 2006). Evolutionary techniques still have many areas left to explore, in terms of their applicability to process mining practice (a claim that is backed by the contents of this chapter). Much of the new research in process mining still concentrates on the use of the Petri net metaphor as a representation vehicle for process mining (van der Aalst et al., 2009). This thesis has demonstrated that parsing and representation techniques need not be based on Petri net theory, and more intuitive graph representations may be safely employed in their place. The work of van Dongen and Jansen-Vullers (2005), on Event Process Chains

(EPC), also points to the use of process modelling formats that may have the potential to gain wider acceptance in industry than that received so far by Petri nets.

One recent development in process management, that is sure to drive up the demand for process mining techniques, is the emergence of Process Aware Information Systems. The authors van der Aalst and van Hee (2002) describe the concept of Process Aware Information Systems (PAIS) as “a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models” (Dumas et al., 2005). Examples of PAIS systems include collaboration software, project management software and application integration frameworks such as Microsoft BizTalk (the PAIS definition also includes ERP systems).

Process mining techniques are now available for mining the latest generation of adaptive PAIS systems. With such systems it becomes possible to edit and change the structure and operation of live running processes (Li et al., 2008). This results in the mining of a number of variations of the same process depending on when changes to a process have been made. The work of Li et al. (2008) aims at producing a generic process model while taking into account the variants created by live modifications to the process. In addition, according to Gunther et al. (2008) hardware systems are also capable of running PAIS like systems that can record tasks to data logs. Such equipment includes Radio Frequency Identification (RFID) scanners, high end photocopiers, and point of sale machines in shops. All of these systems are capable of providing real time data streams to a process mining application.

Conformance of process models to a template is another area receiving much research attention. The area of process disparity is only really investigated in terms of errors that must be removed from a correct process model rather than an area of interest for process mining practice.

The work of van der Aalst and Song (2004) is relevant here as the authors try to mine social networks using process mining practice. In terms of the event log, organisational parameters (such as the name of the person performing a task) are

recorded along with normal task data. This may remain a specialised event log format as major software vendors may not wish to offer functionality which may be considered controversial or threatening by some potential customers.

Peleg (2007) puts forward the view that the outcomes of the execution of processes should be included. The work of this author is in the area of healthcare and gives the example of a patient visiting a doctor about an ear infection. The process recorded the set of visits the patients required for this condition, included the actions and medicines taken and the outcomes on the progress through the illness. This work gives a good indication on the need for future process management and process aware information systems to be flexible in the number of parameters that can be included in event log data. Perhaps the most likely future for event logs would be for flexibility in their development, allowing end users to define the parameters that they wish to record, with the output available as an Extensible Mark-up Language (XML) based stream that is translatable by future process mining approaches.

Currently there is research going on into the nature of event logs and ways of interpreting the data that they record. At present most event logs from major process management software vendors such as SAP and ARIS, record only information such as task names with timestamps. In the future a wide range of parameters may also be recorded along with the task names. Though, in the work of Alves de Medeiros et al. (2007b) the concept of ontologies is introduced whereby a set of meanings can be attached to event log labels so enabling forms of mining such as -

- Organisational Mining – Drawing out relationships between tasks based on task name and/or additional information attached to tasks
- Hierarchical Mining – The aggregation of tasks would be possible if ontological information allowing for the identification of task subsets were available (Alves de Medeiros, et al., 2007b).

Alves de Medeiros (2007b) puts forward the Semantically Annotated Mining Extensible Markup Language (SA-MXML) format for event logs which allows logs to be

annotated with semantic information (information that would aid a process mining approach to mine the event log in a more intelligent fashion). It will be interesting to see if such a metadata standard for business processes gains wider acceptance from industry in the near future.

2.8. Research Gap

In terms of process mining, the use of soft computing techniques is gathering pace. Over the last few years, a number of authors have started to develop the use of genetic algorithms in the pursuit of better process mining results. This approach has shown promise in the area of noise reduction.

Many problems encountered by process mining practitioners have been partly or fully addressed over the past few years. Though there is a possibility to further investigate the role of evolutionary techniques in the mining of processes. This may yield a greater level of accuracy when mining complex process data and mining event logs that contain noise.

Genetic Programming (GP) has not been utilised in the mining of business processes. GP offers a level of flexibility, that is not available with alternative techniques (evolutionary or otherwise), in that actual process model graphs may be acted upon directly by the GP without the need for an intermediary representation. This holds advantages both in the programmatic terms and in the evaluation of process model graphs (known as individuals in GP).

The future development of process mining as a practice holds many opportunities. Evolutionary techniques, such as GP, have much to offer this maturing area of research. The aim of this research is to develop a genetic programming approach for process mining. The focus of this research is on automated/semi automated business processes within the service industry (see section 1.4 of chapter 1). Areas such as the mining of complex process constructs and semantics and the presence of noise in event logs could all benefit from a GP approach. These areas, among others, will form

the focus of this research and are detailed as objectives in section 1.6 of chapter 1. In addition the GP approach will be validated using a variety of test problems and two real life case studies provided by the industrial sponsor.

2.9. Summary

This literature review has examined the following areas concerned with this research –

- Business Process Definitions
- Representations of Business Processes
- Process Mining Techniques
- Process Mining Problems
- Future Areas for Research in Process Mining
- An Outline of Soft Computing
- Soft Computing Techniques and Process Mining
- Genetic Programming

The findings of the literature review have helped to shape the focus of this research. Chapter 3 details the Industry survey undertaken for this research.

3. Industry Survey

3.1. Introduction

A decade after interest was first taken in business processes, by academics and practitioners, there are still different perceptions on how an organisation can benefit by adopting a business process perspective. A range of bespoke business process management applications is available on the market and an extensive body of literature now exists on business processes. However, there is no comprehensive understanding as of yet regarding the understanding of business processes within the service industry. In order to better understand the perceptions of industry on business process automation and mining techniques a survey was undertaken for the wider Intelli-Process project (Intelligent Decision Support for Process Re-design and Conformance, EPSRC project EP/C54899X/1), of which this research is part. The survey was jointly carried out with a fellow researcher working on the Intelli-Process project, but the results were analysed separately. The fellow researcher focussed on business process improvement and optimisation. The author focussed on the following relevant sections –

- Business process mining
- Business process software tool requirements

Both authors developed questions for the business process perception and modelling section of the survey as this was of mutual relevance. Parts of this industry survey have been published by the author in Turner et al. (2007) and Vergidis et al. (2008).

3.2. Survey Methodology

The survey involved the participation of 25 respondents (in 23 organisations) working in service industry sectors such as finance, public sector and consultancy; Table 4 details the type of organisations involved in the survey. The organisational

perception of business process management and modelling has been captured by this survey. The industry views about process automation and mining have also been sought.

Table 4: Types of Organisations Participating in the Survey

Organisation Type	Respondents
Finance & Banking	7
University & Public Sector	7
Consultancy	3
Other Service-Based Organisations	8

The survey was conducted using two types of questionnaire:-

1) Face-to-face interview questionnaire.

2) Online questionnaire (containing mostly multiple choice questions).

It was decided early on that the best form of data capture for this survey would be provided by questionnaires. This was due to the semi-qualitative nature of the subject being researched and the need to be consistent and precise in the questions. In terms of the interviews 4 company visits were arranged. Prior to each visit background information on the project was sent to each company. In addition, before each interview details of the project were given again in the form of a short presentation. The transcripts from each interview were then collated and analysed. A range of question types were included in the interviews, along with closed short answer and multiple choice questions a number of free response boxes were built in to capture views and opinions expressed by interviewees. In terms of the internet delivered questionnaire most questions were closed short answer or multiple choice in nature, due to the

nature of the delivery method. Table 5 displays the responses by questionnaire type showing a wider range of respondents to the internet based questionnaire.

Table 5: Responses by Questionnaire Type

	Organisations	No. of Respondents
Face-to-face interview questionnaire	3	5
On-line questionnaire	20	20

In total five face-to-face interview questionnaires were completed, containing a mixture of open and closed questions. This included a multiple-choice questionnaire which repeated some key questions for confirmation. The on-line questionnaire allowed respondents to answer a range of multiple choice and short answer questions via a website based fill-in form. The online questionnaire, while lacking in some areas of detail, was able to deliver a more quantitative view of the overall trends in the service industry with reference to business process mining activities. Both the interview and online questionnaires were structured in a similar way. The relevant sections of the questionnaires to this thesis are detailed in Table 6.

The following steps were required in the completion of the industry survey: -

- Participant selection – These were selected from a contact list of Cranfield University industry partners. Organisations in the survey should all be engaged in service sector activities.

- Questionnaire design – The literature review stage provided the material for the design of two questionnaires, one for administration by way of face-to-face interview and the other by a web form on the internet (containing short answer questions).

- Interview stage – This stage involved 5 face-to-face interviews with participants. The Online questionnaire was also completed by the respondents at this stage. In total the survey includes the responses from 25 participants in 20 organisations (such as banks, management consultancies and pubic sector organisations).

- Analysis of responses – The completed interview responses were collected along with the answers from the online questionnaire to form the completed industry survey.

It was noted during the piloting of the interview questionnaire that the wording of several questions needed to be changed. A question on enterprise software was changed so that several common products were mentioned (such as SAP). This clarification was required to ensure the interviewees knew what was meant by the term process management software. In addition the first question of the interview questionnaire, on the perception of business process, was originally closed, asking for the interviewees responses to a number of business process definitions. The opening up of this question allowed for a wider discussion of perceptions on this area. The development of the online survey followed on from the first questionnaire. This survey required the implementation of the learning points from the first questionnaire and further piloting due to the unsupervised nature of its completion. In particular the business process perception question of the interview questionnaire was divided into two closed multiple choice question for the online version (asking for the selection of a phrase that best describes corporate business process practice and the selection of a definition of business process).

Room was given in the online survey for free responses if the given choices were not seen as adequate by respondees. Question 7 on business process review methodology and question 8 on the review steps, from the interview questionnaire, were combined with Question 7 (of the interview questionnaire) in final version of the online survey as piloting suggested that no methodology or structured review was in use (question 4 of the online survey). This was borne out by the responses gained from

the internet questionnaire that confirmed that the organisations that undertook reviews of business processes on a regular basis completed only qualitative reviews. Both questionnaires were piloted by three people from a service based organisation and a bank.

Table 6: Relevant Questionnaire Sections for Both Interview and Online Surveys

Questionnaire Section	Description
Business process perception and modelling	This section asked about the perceived notion and understanding of business processes within an organisation. Further questions asked about the nature of the processes, modelling of processes and common process constructs.
Business process mining	This section asked about the level of automation inherent in an organisation's implementation of its business processes. This section also enquired if the organisation undertook any process mining activities; and if so, what software did they use to mine process data.
Industry requirements for a software tool	This final section recorded the organisation's needs with regard to an integrated process mining, optimisation and conformance tool.

In the development of both questionnaires the advice of Robson (2002) on the subject of questionnaire design and interview technique was beneficial. Also Brace (2004) provided guidance as to the format of individual questions. The questionnaire for the interview survey is included in Appendix D and the online survey is in Appendix E.

3.3. Survey Responses

3.3.1. How the Service Industry Perceives Business Processes

The first question asked 'what is the current practice within a respondent's organisation?'. Figure 9 demonstrates the answers of the 25 respondents. The results are encouraging as 88% recognise the existence of business processes within their organisations, another 64% observe cross-functional co-operation thus highlighting

the need to facilitate the co-operation between different departments within an organisation. This question seeks to see how different organisations treat their business processes. The results for the next question, which asks about the level of business process knowledge within an organisation, are illustrated in Figure 10.

3.3.2. Who has the Knowledge about the Process Flow Within an Organisation

This question seeks to see how different organisations treat their business processes. Although 44% claim that there is a specific process owner responsible for each business process, another 32% state that the process knowledge is shared among the main participants of the process. In the first case, it means that the organisation is built around business processes, having employees responsible for the complete process flow. In the second case, there is a loose process knowledge shared among the participants. This demonstrates a lack of understanding and central co-ordination in business processes. The 8% of respondents that agreed that no-one has explicit knowledge about the complete process flow were the same respondents who identified that departments in their organisation worked independently of each other.

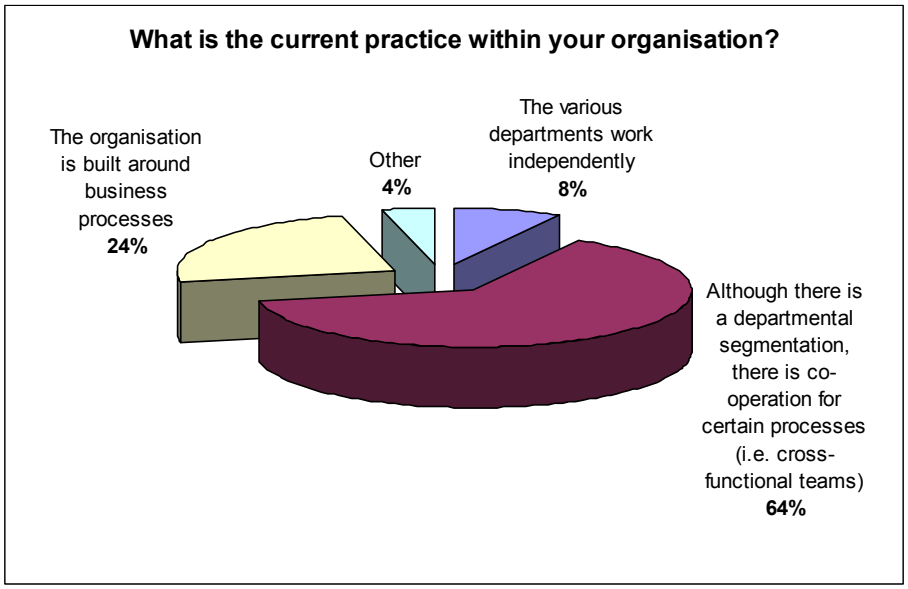


Figure 9: Current Practice within the Organisation

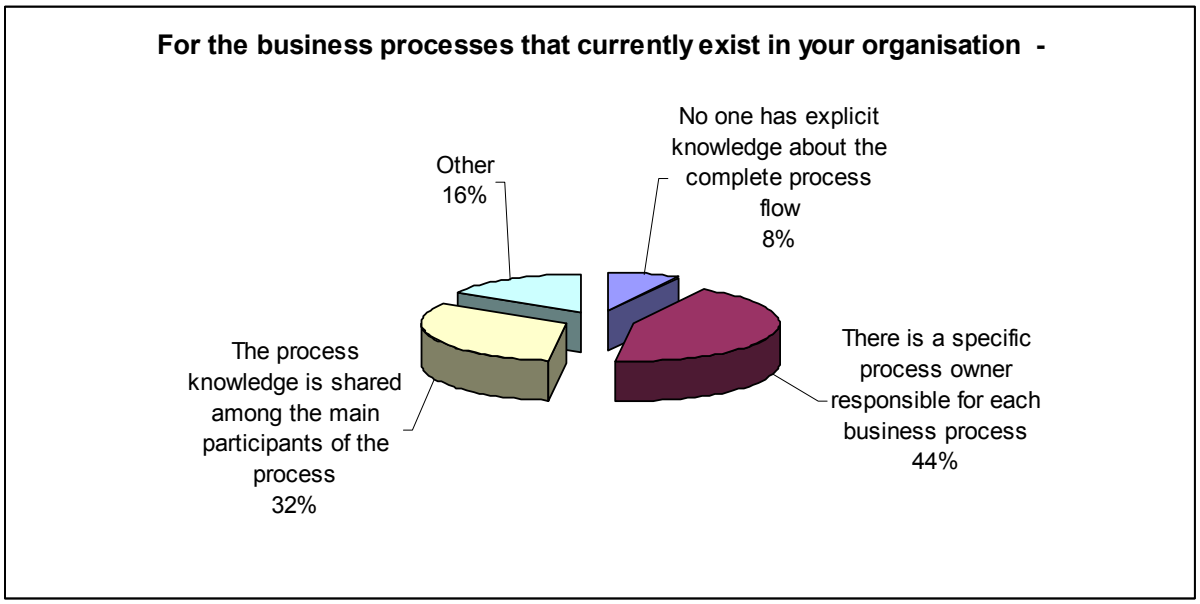


Figure 10: Business Processes Currently Existing in the Organisation

3.3.3. Business Process Modelling Techniques

Figure 11 illustrates the most popular process modelling techniques used within organisations. For each technique a 5-scale ranking (from 'common practice' to 'not used at all') was used. The results demonstrated that the majority of participants use simple flowcharts with informal notation, as 76% responded that these flowcharts are 'common practice' or 'frequently used' and 24% responded 'sometimes used'. Integrated Definition Modelling (IDEF) was frequently used by only 8% of respondents with 32% of respondents using the modelling technique occasionally. For Petri nets only 32% of respondents had used the technique on a one off basis, the technique was not in regular use within industry.

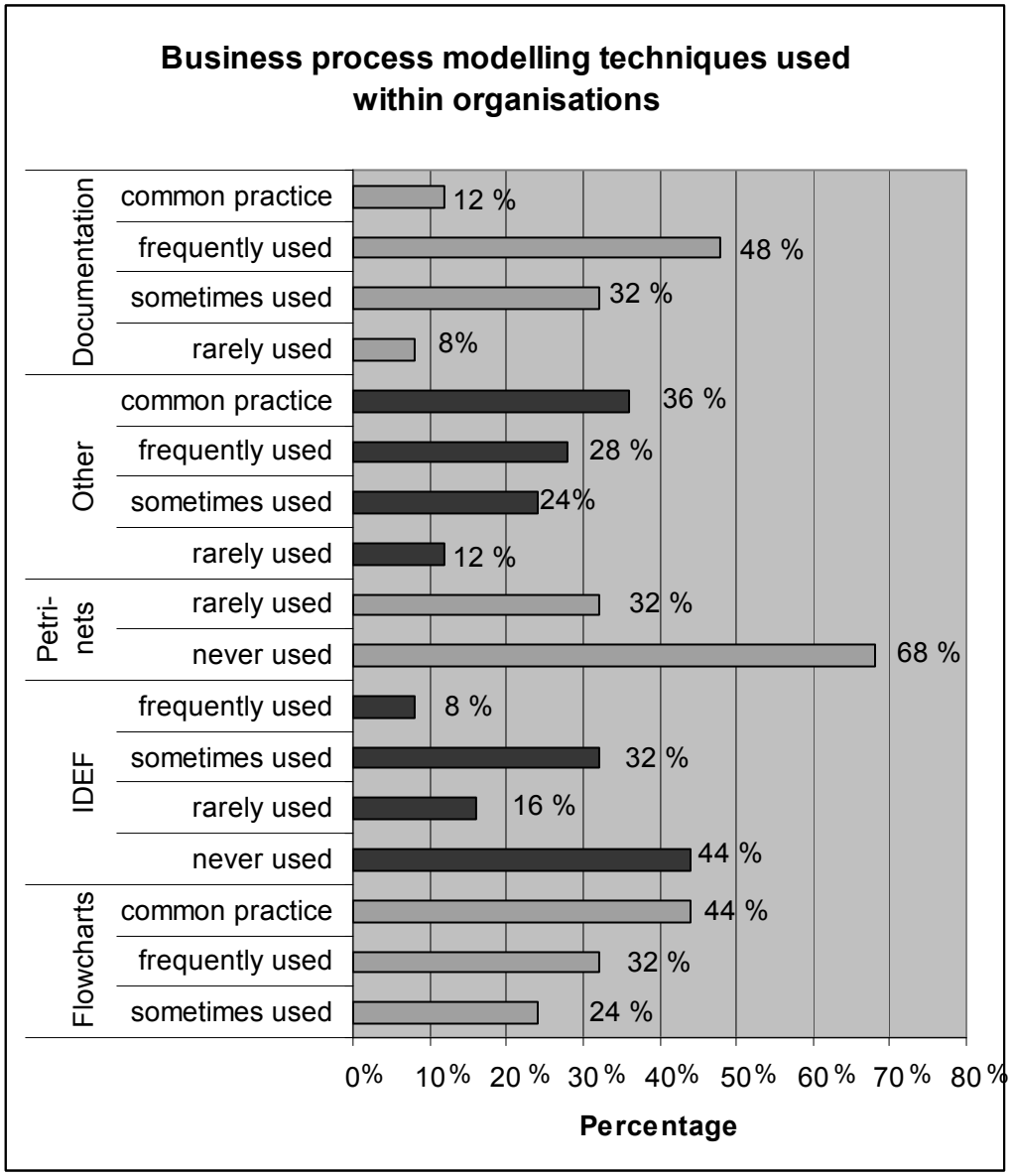


Figure 11: Business Process Modelling Techniques Used Within Organisations

It is interesting to note that only 12% declared that it is common practice to maintain documentation about a process and another 80% claimed that processes are documented on a frequent or occasional basis.

3.3.4. Business Process Patterns

In the questionnaire section on business process patterns, respondents were asked to select from a predefined list all the patterns that appear in their business processes. A Process pattern can be thought of as a specific arrangement of activities to solve a problem (Havey, 2005). The business process patterns that were listed were: sequential flow, parallel flow, decision point and feedback loop. As can be seen from Figure 12 this survey confirmed that organisations do have complex constructs within their processes which may require an automated mining technique.

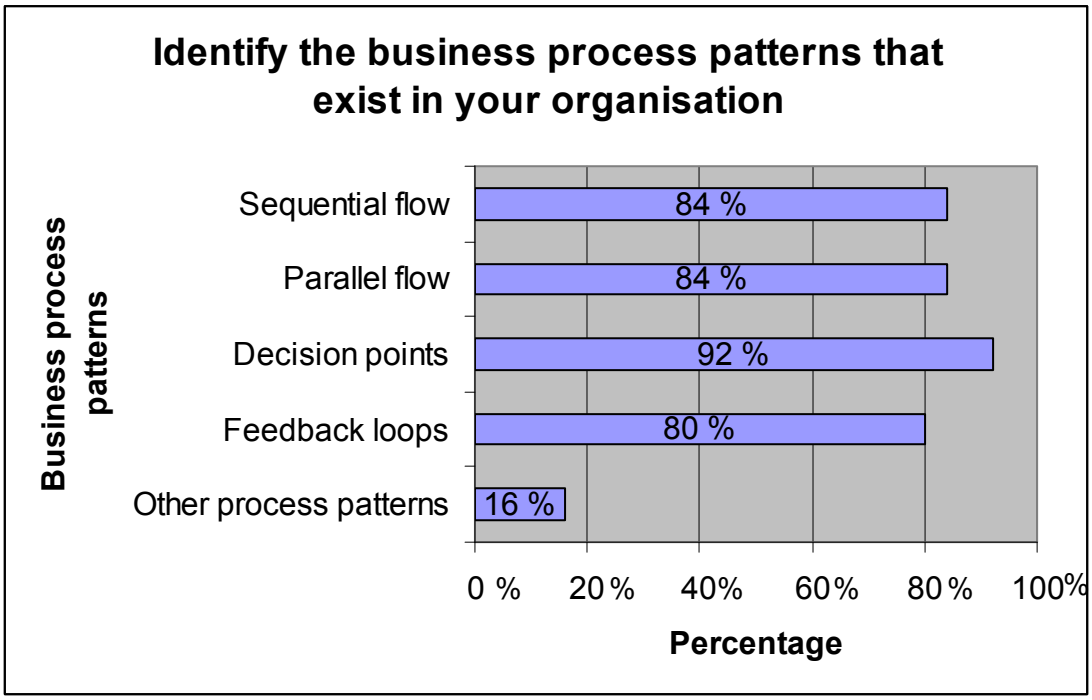


Figure 12: Business Process Patterns Existing in the Organisation

3.3.5. Business Process Mining

As can be seen from Table 7 the overall average percentage of automated processes was identified as 66% for banking and finance organisations which was the highest for any sector. The lowest level of automation was to be found in the university and public sector with only 30% of processes automated, followed by the consultancy sector at 33%. Figure 13 demonstrates the respondents' answers when asked about the use of business process management software within their organisations. Analysis

of the results reveals that 50% of the respondents said that their organisation possessed a specific business process management software suite (as an example 15% are using SAP and 11% are using ARIS out of the listed packages which also included Tibco), 23% use other software packages not listed within the choices (e.g. Oracle Financials, Sophia, Intelicorp and Nimbus). The other half of the respondents claimed that either their organisation does not use or they are not aware of a business process suite being used within their company.

Table 7: Percentage of Automated Processes in Different Industry Sectors

Sector	Percentage of Automated Processes
Finance and banking	66%
University & public sector	30%
Other service based organisations	57%
Consultancy	33%

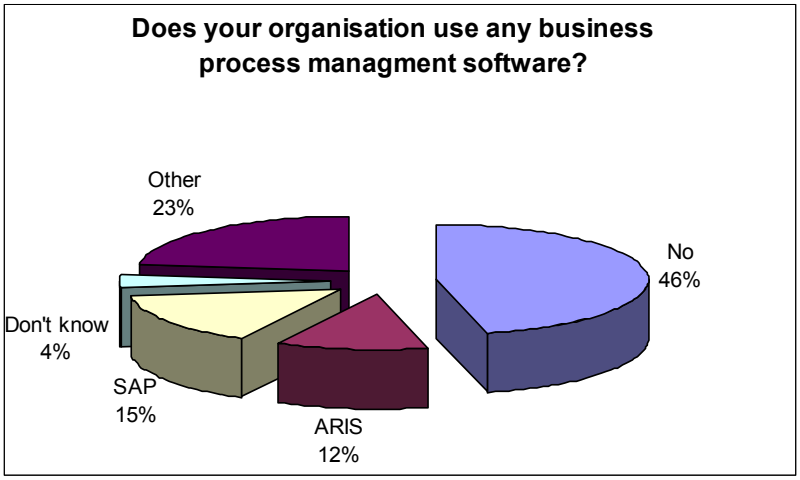


Figure 13: Use of Business Process Management Software (based on the answers given by the 25 respondents)

Several respondents mentioned that the process management suite at their organisation was custom built consisting of legacy databases and other in-house software based systems. Only 12% of respondents stated that their organisation

undertook process mining activities. This is despite 60% of respondents confirming that their organisation did retain records of past process execution. For those respondents whose organisation did conduct process mining the actual task of mining was performed by in-house developed software rather than a specific process mining tool. From the findings of this survey it may be concluded that the market for business process mining software has yet to mature.

3.3.6. Industry Requirements for Business Process Mining Software

Most respondents of this survey (80%) require a software tool with the ability to mine process models from stored logs of past process executions, Figure 14. This suggests that the market penetration of existing process mining tools in the service industry is still limited. Commercial process mining tools do exist, such as Pallas Athena and the process miner provided with ARIS, though the market penetration is still limited. Existing tools are basic at the moment and the range of process constructs that can be mined is limited. The GP approach detailed in this research should provide a more flexible commercial process mining tool than that currently available. It is also likely that an increasing use of process management software in the future could also lead to increased awareness of the need for a formal process mining technique.

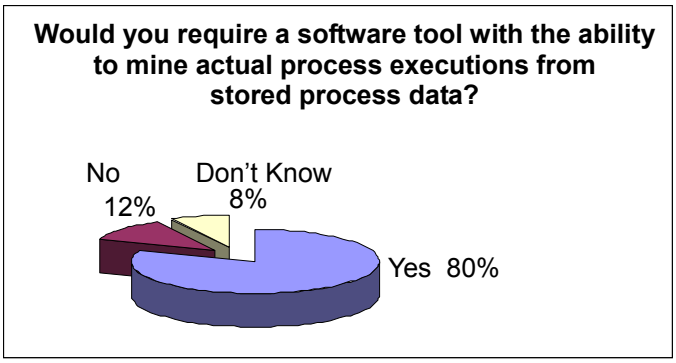


Figure 14: Process Mining Software Tool Needs (based on 25 respondents)

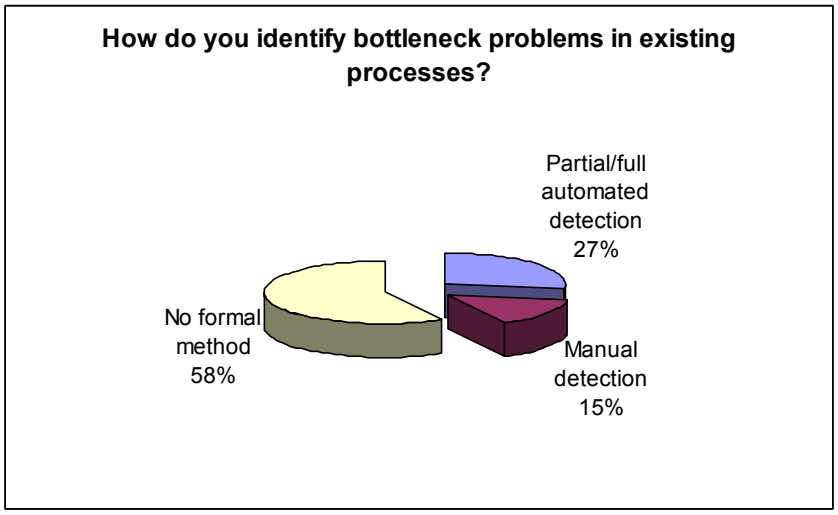


Figure 15: Bottleneck Identification in Existing Processes

The survey also showed (in Figure 15) that 58% respondents said that their organisation did not have a formal method for undertaking a bottleneck problem identification survey (that is a way of finding out where delays and/or problems are occurring within a business process). It was mentioned by a number of respondents that bottleneck identification functionality could be a useful addition to a process mining software tool. It was also noted that some respondents would like a visual editor in order to analyse business processes.

3.4. Survey Discussion

The overall impression from the survey is that although practitioners are aware of business processes, their organisations do not provide support in areas such as process mining, modelling and analysis. This survey has highlighted a general lack of process co-ordination and consistent modelling. Although there is a relatively high level of business process automation in the service industry, with more than 50% level of process automation in some sectors of the industry, only 12% of respondents undertake some form of process mining. However the potential to conduct process mining in most organisations does exist as a large amount of data exists from past process executions recorded from automated and semi-automated processes.

Business Process Management (BPM) software exists in a large number of organisations along with some form of process monitoring.

It can be concluded that business process reviews take place in the majority of respondents' work places, though they are largely manual and time consuming. There are, however, few formal approaches adopted to identify 'bottleneck' conditions in existing processes. The industry mostly uses basic techniques such as flowcharts for capturing and modelling business processes and applies manual efforts in business processes analysis and improvement. In essence organisations do not tend to be organised around business processes. A need exists for process modelling and mining tools in the service industry. However it must be recognised that the existing software-based process automation applications (such as process management tools) in organisations lack in these aspects. It is likely that an increasing use of process management software in the future could also lead to increased awareness of the need for a formal process mining technique.

3.5. Summary

This chapter has outlined the main findings from the industry survey undertaken on business process knowledge and perceptions in industry. This Included views on –

- Organisational business process knowledge.
- Business process modelling techniques.
- Business process automation.
- Need for business process mining.

The following chapter will outline the initial development of Genetic Programming (GP) approach to process mining which is the core deliverable of this research

4. Proposed Genetic Programming (GP) Based Business Process Mining Approach

4.1. Introduction

The concept of Genetic Programming (GP) has already been introduced in the literature review section of this thesis. In summary, the concept has the following key attributes –

- An evolutionary technique utilising crossover and mutation to generate new individuals.
- Open representation, can use graphs as individuals in a population.
- A non-deterministic technique allowing wide and complex problem spaces to be searched effectively (Banzhaf et al., 1998).

This chapter sets out the initial design of the approach with regard to process mining along with the results of some early evaluation tests of the technique. The descriptions of the approach and initial results, shown later in this chapter, have been published by the author in Turner and Tiwari (2007a) and Turner et al. (2008).

4.2. Representing Individuals in GP

As was mentioned in the literature review chapter of this thesis the representation of individuals within GP is fairly open. This is in contrast to Genetic Algorithms where each individual must be reduced to a binary a real chromosome string for it to be manipulated by the algorithm. A GP technique is particularly beneficial in the practice of process mining in that the representation of individuals need not be abstracted to a lower level representation. In GP, individuals may be represented and modified by evolutionary operators, as graph objects. The use of graph structures in combination with genetic programming is not a new concept (Ashlock et al., 1999). Genetic

Programming is generally known for the generation of either tree structures or linear programming code (Koza, 1992). However, GP has been used for generating graphs for Artificial Neural Networks (Hornby, 2006), bond graphs (Wang et al., 2005a), electronic circuits (Miller and Banzhaf, 2003), and algorithm structures (Shirakawa et al., 2007). The work of Miller (1999) introduces Cartesian Genetic Programming as a graph based approach (work that is used later on in Miller and Banzhaf (2006) to evolve electronic circuits). This is true though it must be noted that the representation used is a linear string of integers rather than an actual graph. Therefore the genetic operators work at the level of the string rather than on any other type of abstraction, graph based or otherwise. While GP individuals tend to benefit from crossover and mutation operators, Miller (1999) points out for Cartesian Genetic Programming crossover has little or no effect, relying much more on mutation instead to provide variety in a population.

4.3. Representation Used in the Proposed GP Approach

The GP process mining approach presented in this thesis utilises a directed graph structure, provided by the JGraphT software Application Programming Interface (API) (JGraphT, 2009), to represent individuals (individual process models). JGraphT provides a fully functional directed graph object that may be integrated with a Java based software which makes this a unique offering in the open source Java software world (JGraphT, 2009). This abstraction allows a better fit to the practice of process mining in that individuals' fitness may be evaluated more efficiently due to the fact that there is no need for an intermediary representation (as explained in section 5.4.3.2). A guide to the terms used in this and following chapters is presented in Table 7. In a GP approach, an individual is effectively an executable program (Banzhaf et al., 1998). In a JGraphT object, a graph is represented as a set of task nodes and edges linking task nodes. Each task from a process is programmatically represented as a task node within JGraphT. The JGraphT Java software is open-source. This has allowed the author to modify its operation to suit the practice of process mining. The author has

redesigned the task node of JGraphT to accommodate an input set object and an output set object. The input set contains edges to a task (edges are in effect links between tasks) from other tasks, similarly the output set holds edges from a task to other tasks. The input and output set objects can in turn accept multiple subsets containing edge nodes (a task node is shown in Figure 16). A subset may contain edges that are either in an XOR relationship or an AND relationship. Each input and output set may contain zero, one or more XOR subsets and zero or one AND subset (in effect due to the nature of the relationship all AND edges are mandatory in their execution so can be grouped into one set). In programmatic terms Edge nodes (grouped In subsets and held at a task node in ether an input or output set) are objects that contain three pieces of information, the name of the task node they are held in, the name of the task node they link to or from and an indication of the semantic relationship they are in with other edge nodes in that edge sub set (AND or XOR). An example of an edge node is shown in Figure 16.

Table 8: Key terms used in the following chapters

Task node	A process task is represented as a task node within a process model and programmatically within the GP approach
Edge	An edge, in graph theory, is a link between two tasks (or vertices) in a particular graph
Edge node	An edge node refers to an edge between two task nodes and is a programmatic construct within the GP approach
Edge node sub-set	An edge node sub-set contains a set of edges that are in an AND or an XOR relationship. An edge node sub-set is a programmatic construct within the GP approach

Input/Output set	A task node contains both an input and an output set. These sets contain edge-node subsets and represent the edges linking to and from a task node (and therefore a task node's links or edges to other task nodes). Both sets are programmatic constructs within the GP approach
Process trace	A record of a single enactment of a process recording the tasks in the sequence that were executed in (essentially the path taken through the process in that enactment)
Process model/graph	A process model is an overall view of a process (an aggregation of individual process traces. It is effectively a flowchart and maybe represented by a variety of graph notations such as Petri Nets and Event-driven Process Chains (EPC) (hence in this thesis the terms process graph and process model are synonymous)
Event log	A record of the order in which a set of tasks (comprising a process) have been enacted on multiple occasions
Task	A task is a unit of work or stage within a process (sometimes referred to as an activity in some literature) (Havey, 2005)

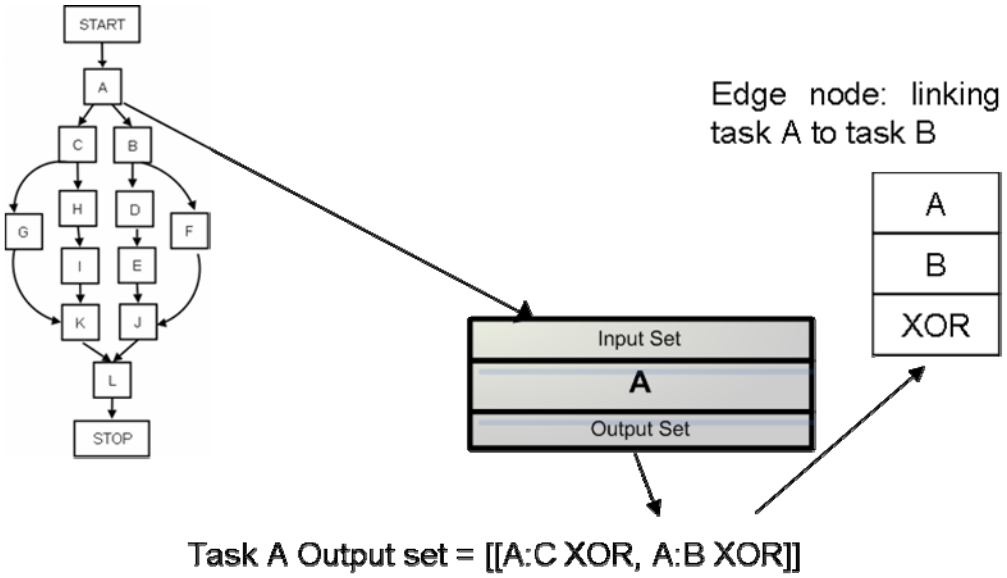


Figure 16: A Task Node from a Process Model Showing the Input and Output Sets

Multiple XOR subsets may exist in each input and output set and may contain one or more XOR edges (as shown in Figure 16). Each JGraphT individual can be interpreted by another open-source Java software object set, called JGraph (2009), to allow for it to be displayed on screen to a user as a visual flowchart representation. This representation displays the tasks and a single edge between tasks that are linked together. Each edge node in an edge node sub-set (see Table 17 for a guide to the terms used here) has the form shown in Figure 16. In the output set in task A (shown in Figure 16) a task node A:B would indicate that an edge is present linking task A and task B (assuming that there is a edge node A:B in the input set of task B). Each edge node has an additional marking which denotes whether it is in an XOR relationship (in an XOR subset) or an AND relationship (therefore in the AND subset). An AND relationship means that all of the tasks at the end of the edges in that relationship must be executed. An XOR relationship means that just one of the tasks at the end of an edge in the relationship needs to be executed.

As mentioned earlier, each graph can be programmatically accessed in a generic way (via the JGraphT Application Programming Interface (API)) in order to render the structural changes to graphs required by the GP operators. This representation differs from that used by a range of existing process mining approaches in that it is an actual

graph representation that is being acted on directly not an intermediate form. In effect there is no requirement with the GP approach to represent an individual as an abstract object or data representation which would require interpretation and conversion. The approaches of Alves de Medeiros (2006) and Weijters and van der Aalst (2001) utilise a representation called a causal matrix where the tasks are numbered and the relationship between them represented in a binary form utilising a Java based matrix object. Table 8 shows the causal matrix and Figure 17 is the process that is described by Table 8. In Table 8 the brackets inside the outer bracket set indicate subsets of tasks. Subset with more than one task indicates that the contained tasks are in an XOR relationship, single tasks are in an AND relationship. All of the subsets within a bracket set and in an AND relationship i.e. one task must execute from each of the subsets. This approach has the advantage of providing fairly compact individuals (in programmatic terms) though their means of manipulation is abstract and non intuitive as an intermediate representation is required in order to model an actual directed process graph.

The use of a graph representation offers a greater level of flexibility than with the causal matrix as it would be possible to extend the GP representation to model normal OR constructs (something that is not readily possible with the causal matrix approach). The parsing of individuals is also more straightforward with the GP approach as it is possible to maintain the state of an individual when parsing (also something that cannot be achieved with the token passing parsing of the causal matrix). This fundamental difference between the GP approach and the techniques of Alves de Medeiros (2006) and Weijters and van der Aalst (2001) will be demonstrated further in chapter 6 where innovations in the way individuals are parsed, altered and repaired, will be put forward. It was decided early on in this research project that a new alternative means of representation was required as part of an overall approach to provide a more flexible and streamlined means of mining processes.

Task	Input Task	Output Task
A	{}	{{F,B,E},{E,C},{G}}
B	{{A}}	{{D}}
C	{{A}}	{{D}}
D	{{F,B,E},{E,C},{G}}	{}
E	{{A}}	{{D}}
F	{{A}}	{{D}}
G	{{A}}	{{D}}

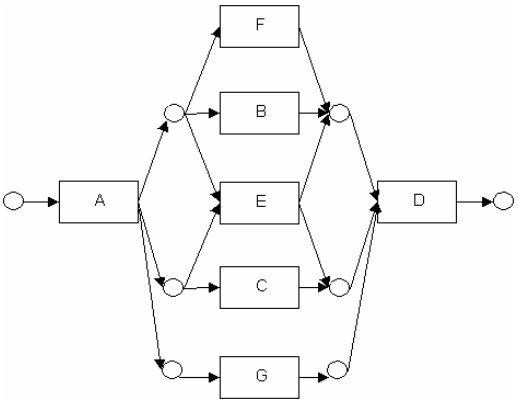


Figure 17: A Causal Matrix and the Process it Represents (reproduced from van der Aalst et al., (2005a))

4.4. Main Steps of the Proposed GP Business Process Mining Approach

This section provides an overview of the main steps in the proposed GP process mining approach. The main steps are shown in Figure 18. In Figure 18 the GP process miner 'reads-in' process data in the form of an Extensible Mark-up Language (XML) based event log. Event log is the name given to data logs that contain process execution data for a live process. As mentioned before in Table 8 an event log is a record of the order in which a set of tasks (comprising a process) have been enacted on multiple occasions. Such event logs may be hosted within Business Process Management (BPM) and workflow systems, owned by medium and large organisations, recording the task by task completion of computer assisted processes. In some organisations event logs may be hosted by Enterprise Resource Planning (ERP) systems (van der Aalst et al., 2003b). The second step involves the calculation of dependency relations between process tasks, in effect the frequency with which each of the process tasks in a trace from the event log precedes a given task from the event log (used to workout a rough precedence order in the tasks). From the dependency relations an initial population of individuals can be built from a random selection of the

relation frequencies. Each trace in the event log is then parsed (compared) to each individual in a population. A fitness score is then assigned to each individual based on how well it compares to the event log. A new generation of individuals is produced by copying over the two fittest individuals of the current population. Crossover and mutation operators are then used to alter a number of randomly selected individuals from the first generation for inclusion in the new population. The approach terminates after a certain number of generations have been produced or an individual with a fitness of 1.0 (perfect match to the event log) has been found.

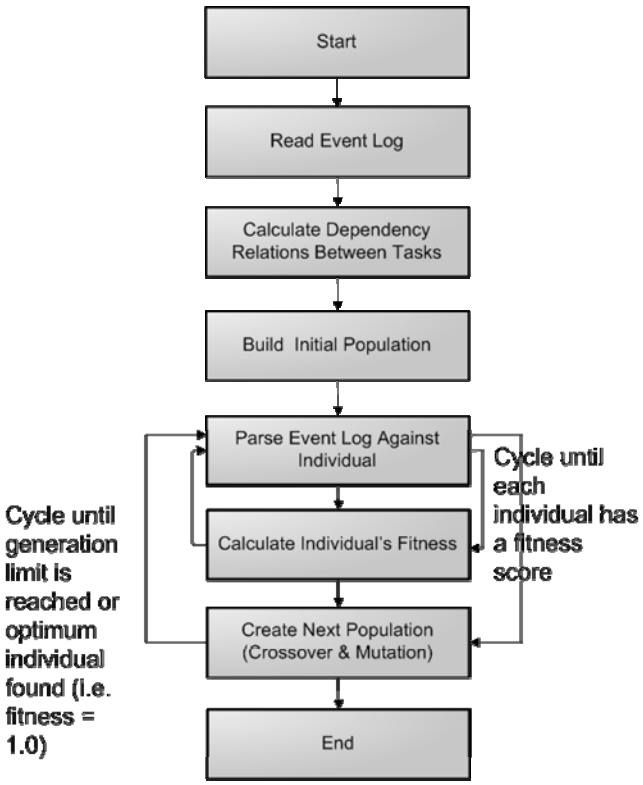


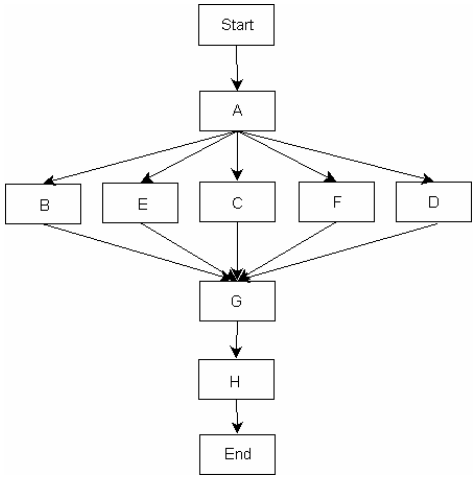
Figure 18: The Main Steps of the GP Process Mining Approach

The XML format, known as Mining Extensible Mark-up Language (MXML), used to describe the event logs that the GP approach will mine is the one used by the ProM process mining suite (van Dongen et al., 2005b). This format is open and allows for third parties to mark-up their process data into a form that is readable by the ProM process mining suite. The GP approach is hosted in the ProM process mining suite. The ProM suite is an open source platform for the development of new process mining

approaches. The suite was selected because it provides functionality to read event logs (in an MXML format) and to handle files, useful for developing a new process mining approach with Java. An example of the MXML format is given in Figure 19; this XML 'language' provides only a basic description of the tasks and trace numbers recorded in an event log (it is not a Turing complete language in any sense). The MXML in Figure 19 represents a single trace from the event log of a process, the complete model of which is shown in Figure 20.

```
<ProcessInstance id="0"
description=""><AuditTrailEntry>

<WorkflowModelElement>Start</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>A</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>B</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>E</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>C</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>F</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>D</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>G</WorkflowModelElement>
</AuditTrailEntry>AuditTrailEntry>

<WorkflowModelElement>H</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>End</WorkflowModelElement>
</AuditTrailEntry></ProcessInstance>
```

Figure 19: An Example of the MXML Event Log Format (showing a trace from the Process Model in Figure 19)



Start - A - B - E - C - F - D - G - H - End

Start - A - D - B - E - C - F - G - H - End

Start - A - E - C - F - D - B - G - H - End

Start - A - C - F - D - B - E - G - H - End

Start - A - F - D - B - E - C - G - H - End

……..

Figure 20: A Process Graph and a Number of Valid Traces Through It (graph provided by Alves de Medeiros (2006))

It can be seen in Figure 19 that each task is referred to as a workflow model element and as a separate audit trail entry. One process instance is known as a process trace and describes the path of execution through a process, from the start task to the end task. An event log will normally contain many hundreds of process traces, ideally with each trace describing a slightly different path through a process. Below the process model in Figure 20 a number of traces can be seen that describe possible valid paths through that process model.

4.4.1. Read Event Log and Calculate Dependency Relations

The first step in process mining approach (shown in Figure 18) involves reading each trace from the event log in a sequential order. Once the traces are read, the dependency relations between individual process tasks are calculated from the event log as a whole. The concept behind this is that the precedence between individual tasks can be worked out in a statistical manner giving a basic set of probable relations between tasks. The dependency relation measure is based on five basic rules.

For each task in a given event log with A and B as two tasks:-

1. A→B the strength of the relation between task A and another task B.
2. A>B the frequency of A where it is directly followed by B.
3. B<A the frequency of A where it is directly preceded by B.
4. B<<<A the frequency of A directly or indirectly preceded by another task B
5. A>>>B the frequency of A directly or indirectly followed by another task B

(Weijters and van der Aalst, 2003)

The rules outlined before determine the potential dependencies between tasks and may be modelled as a dependency/frequency table. The graph information is derived from task relations that are above a certain frequency threshold. An example of part of a dependency/frequency table is shown in Table 9. In Table 9 the task A from the event log that created the model in Figure 20 is examined in terms of its relations to task B. The notation X2 = B and X1 = A, the notation #X2 represents the frequency of a given task. As can be seen in Table 9 task A has no dependency relations and task B always follows task A. In the complete table for task A the frequency relations for tasks C to H would also be shown and a separate table would be calculated for each of the tasks A to H.

Table 9: Dependency/Frequency Relations Table for Tasks A and B

	#X2	**X2<X1**	**X1>X2**	**X1<<<X2**	**X2>>>X1**	**X1 → X2**
A	1020	0	0	0	0	0
B	1020	0	1020	0	1020	1020

(Weijters and van der Aalst, 2003)

Individuals (in the form of graph objects) are built randomly from a set of dependency/frequency tables in a task by task fashion. In equation 1 if D is the dependency relation, L is an event log and X is the set of tasks in that event log. Each task in X is compared to each of the other tasks in X (X(X1,X2)). Also in equation 1 if a randomly selected number r (between 0.0 and 1.0) is less than the dependency relation for a given task relationship (X1,X2) then an edge between the two tasks is created (Alves de Medeiros, 2006). This mechanism is used to encourage the production of viable edges between tasks based on the evidence given in the event log.

$$r < D(X1, X2, L)^p \quad (1)$$

4.4.2. Build Initial Population

The dependency relation concept, described in the previous step, was developed by van der Aalst et al. (2002) and is also implemented in the work of Weijters et al.

(2003). Alves de Medeiros (2006) has used this approach to develop a random first population of individuals for the GA approach. A modified version of the GA approach is also employed by the GP process mining technique. The approach has been modified to work with the graph representation used by the GP approach rather than the causal matrix (the GA utilises). This involved the development of a semi-stochastic graph building object which utilised the initial population function of Alves de Medeiros (2006) to infer possible relations between tasks. In the work of Alves de Medeiros (2006) individuals are created to be compatible with the causal matrix representation employed by the ProM process mining software. The GP approach translates the initial population created for the GA approach into JGraphT individuals. In effect individuals in the population created for the GA approach are in the form of a causal matrix which may be interpreted and converted into a set of tasks and edge nodes that can be represented in a JGraphT directed graph object. Custom built Java code is used to do this and to create edge nodes for input and output sets of each task node to be represented in an individual (JGraphT directed graph object). This initialisation approach provided a set of rough heuristically built process models that exhibited some common structures within the process log. It was decided that the selected initialisation approach was necessary as the mining of an initial population built with a completely random view of an event log would be needlessly time consuming and challenging for the GP approach.

4.4.3. Fitness Parsing

Once the initial population has been created, each individual needs to be scored in terms of its ability to accurately reflect the traces in the event log. This scoring is known as fitness, and a measure of fitness is required along with a method assessing an individual against this measure. In order to assess the fitness of an individual the approach of Alves de Medeiros (2006) compares each individual in a population to every trace in the event log on which the individual is based. In this way an individual can be assessed on how accurately it reflects the event log. To do this, a technique known as parsing is employed to compare the task sequence (the task names in a

sequence) contained in each trace (the series of tasks that executed during one enactment of a process) recorded in the event log with the process model contained within an individual (the trace is compared with the graph in the individual to test if its task sequence is feasible).

Parsing reads a process trace on a task by task basis and checks if –

- The task exists in an individual.
- The task can be reached from a preceding task via a task edge.
- The task edges between a task and a preceding task are present (i.e. is the task sequence possible and what type of relationship is the linking task in, either XOR or AND).

As mentioned earlier in this thesis, the links between process tasks are known as edges. Edges are represented by edge nodes in the input and output sets of a task. For example the edge node B:D in the output set of task B and the edge node B:D in the input set of task D indicate the presence of an edge between tasks B and D (the input set of task D can be seen in Figure 21). Edge nodes are placed in subsets, a subset with a single edge node indicates that the tasks in that edge are either in an AND relationship (an example is provided in Figure 21 where both subset in the input and output sets of task D contain a single edge node). Subsets that contain multiple edge nodes indicate a number of edges and therefore tasks that are in an XOR relationship.

In terms of parsing a task can be enabled if all the preceding AND relationship tasks that link to its input set can be enabled. Any XOR relationship edges that appear in the input set of the task can also be enabled if they are not already enabled (by another task in the XOR relationship).

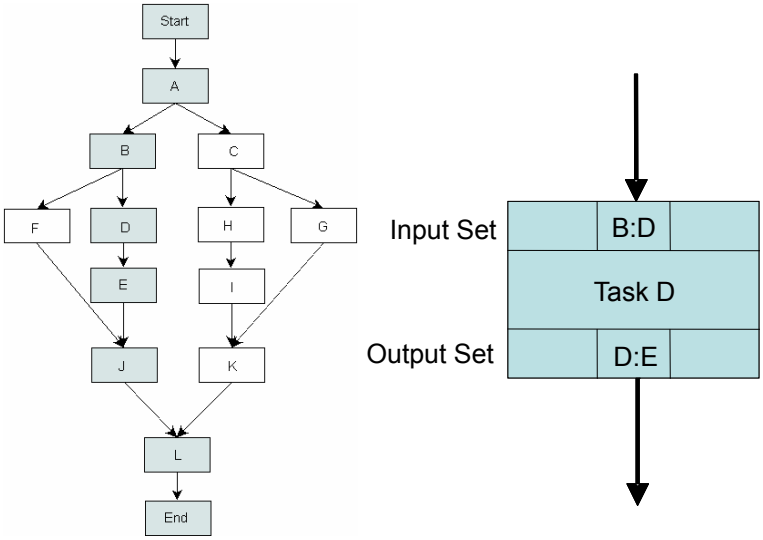


Figure 21: Task Sets and Edge Node Subsets for a Task D, With a Valid Path Shown Through the Process on the Left

Parsing Used by the GA Technique

In the work of Alves de Medeiros (2006) the parsing process is carried out by feeding tokens through an intermediate construct mapped from an individual represented by the causal matrix construct provided by the ProM process mining software. In effect each task has an input set and an output set which, in turn, contain pointers to other tasks that the task should be linked to and from. Parsing proceeds by reading tasks from an event log trace and passing tokens through the input and output sets of each task in the individual.

Element being parsed	Individual's current marking	
A, D, E, F, G, H	A| B = 0, C = 0, D = 0 B| H = 0 C| H = 0 D| E = 0, F = 0	E| G = 0 F| G = 0 G| H = 0 Start = 1 End = 0
A, D, E, F, G, H	A| B = 1, C = 1, D = 1 B| H = 0 C| H = 0 D| E = 0, F = 0	E| G = 0 F| G = 0 G| H = 0 Start = 0 End = 0
A, D, E, F, G, H	A| B = 0, C = 0, D = 0 B| H = 0 C| H = 0 D| E = 1, F = 1	E| G = 0 F| G = 0 G| H = 0 Start = 0 End = 0
A, D, E, F, G, H	A| B = 0, C = 0, D = 0 B| H = 0 C| H = 0 D| E = 0, F = 1	E| G = 1 F| G = 0 G| H = 0 Start = 0 End = 0
A, D, E, F, G, H	A| B = 0, C = 0, D = 0 B| H = 0 C| H = 0 D| E = 0, F = 0	E| G = 1 F| G = 0 G| H = 0 Start = 0 End = 0
A, D, E, F, G, H	A| B = 0, C = 0, D = 0 B| H = 0 C| H = 0 D| E = 0, F = 0	E| G = 0 F| G = 0 G| H = 1 Start = 0 End = 0
A, D, E, F, G, H	A| B = 0, C = 0, D = 0 B| H = 0 C| H = 0 D| E = 0, F = 0	E| G = 0 F| G = 0 G| H = 0 Start = 0 End = 1

Figure 22: Parsing Sequence for Trace A, D, E, F, G, H (which may be traced through the process shown in Figure 23)(reproduced from Alves de Medeiros et al. (2004a))

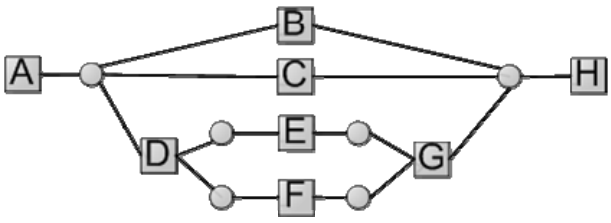


Figure 23: A Process Graph (reproduced from Alves de Medeiros et al. (2004a)

When a task in the process trace can be matched to a task in the individual and the task is not a start task (which does not have any input sets) all of the input subsets are marked if the tasks that they denote can be reached by following back the edges that terminate at that task (as process graphs are directed the output of one task must link to the input of another task). In effect any tasks that are linked by edges from a given task B, that is have tasks that link to task B, are marked (unless marked already). For tasks that cannot be linked back to previous tasks (excluding the start task) an extra token is added. This type of parsing relies on passing tokens through an individual as if it is a Petri net, in fact many of the rules for this parsing are based on those required for the firing of transitions within Petri nets (van der Aalst et al., 2005a). Figure 22 demonstrates the parsing of a trace A,D,E,F,G,H in the process model shown in Figure 23. In Figure 22 only the output sets are shown as these are the sets that are marked. A marking is indicated by the presence of a 1 and the progression of parsing through the sets in Figure 22 is shown by highlighting the sets under consideration in light grey. For a more detailed explanation of this technique see Alves de Medeiros et al. (2004a).

The number of tokens used in the parsing of an individual changes depending on how far through the individual the point of execution is (the markings in Figure 22 represent tokens). When a task is fired one token is consumed from each of the input subsets and one token produced for each of its output subsets, though as mentioned before tokens may be added when there is no reachable inputs to a task (van der Aalst et al., 2005a). When parsing a particular individual tokens may be left behind. This indicates that output subsets of a task have been marked but the tasks that they link to are not present in the event log trace just parsed. In this case the model is potentially allowing for extra behaviour not recorded in the event log; and the left over tokens are an indicator of this. Further details on this form of parsing can be found in (van der Aalst et al., 2005a).

Parsing Used by the Proposed GP Approach

The parsing process used by the GP approach uses a different approach and marking schema than that used by Alves de Medeiros et al. (2004a). The parsing used

by the GP approach is not based on Petri net firing rules (unlike the GA approach of Alves de Medeiros et al. (2004a)), and no tokens are passed through an intermediate representation of an individual (causal matrix) (though the notion of a token is still used to act as a punishment at the fitness stage for incorrect sections of a process model). Instead, edge subsets in an individual can be marked leaving an audit trail of parsing through an individual. This can offer the advantage of state maintenance, allowing for the current state of parsing in an individual to be known (in comparison to Alves de Medeiros et al. (2004a)). In total 3 markings are used –

Marking 1 – Edge is not enabled (tasks at both ends of the edge are not enabled)

Marking 2 – Edge is possibly enabled (the task that the edge is linking from is enabled)

Marking 3 – Edge is enabled (the tasks at either end of the edge are enabled)

A number of parsing rules have been used with the GP parser (shown in Table 10) and the parsing cycle is shown in Figure 24. It is important to note that only the edge nodes in subsets in the output set of a task are actually marked (see Figure 21 and Figure 16 for an example of an edge nodes and edge node subsets). Parsing takes place in a directed fashion as all trace tasks are arranged in order of execution with each trace possessing a generic start and end task (the start and end tasks are added to each trace in the event log before mining begins). During parsing every event log trace must be parsed against every individual in a population. In Figure 24 it can be seen that parsing is initiated by reading the first trace from the event log. In the next step (Figure 24) the first individual in the population is selected. The first task is then read from the trace. If the task exists in the individual all of the output subsets of that task (in the individual) are marked as 2.0 possibly enabled. The ‘possibly enabled’ marking means that any of the tasks that are marked this way could be enabled if selected. For example in an XOR split in the output of a task only one task linked to by this split construct can be enabled once a selection has been made.

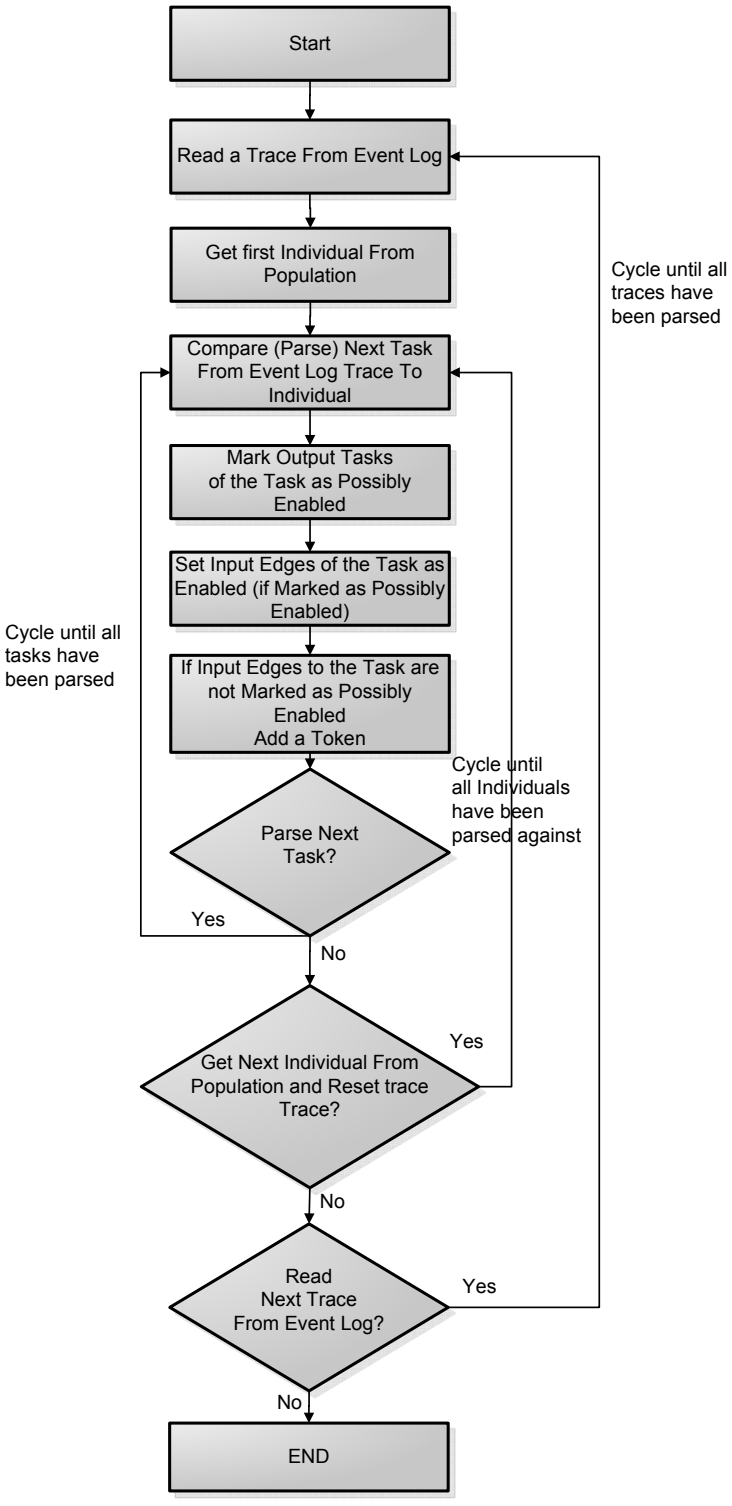


Figure 24: Parsing Cycle of the GP Approach

Table 10: GP Fitness Parsing Rules

Rule	**Reason**
If incoming XOR edge node is not enabled then leave	An incoming XOR edge node that is not enabled could mean that another edge node in that XOR subset is enabled or will be enabled in the future
Add a token for any incoming AND edge nodes that are not marked as possibly enabled	A task cannot execute unless all of its mandatory inputs (AND edge nodes) are marked as enabled
Mark all incoming AND edge nodes, that are currently marked as possibly enabled, as enabled	In order to register that a task has fired correctly, all mandatory AND edge nodes and XOR edge nodes that are marked as possibly enabled must be marked as enabled
When an incoming edge node is marked as possibly enabled, visit the other task and mark the linking edge node as enabled and mark the other edge nodes in that XOR subset as not enabled	Only one edge node of an XOR subset can be marked as enabled; all other edge nodes must be marked as not enabled
If a task in the trace does not exist in the individual, register the task as missing and add a token (the notion of token is used as a punishment score at the fitness calculation stage)	A missing task is a serious omission from an individual
If an XOR edge (or edges) is the only input into a task and another edge node in that edge node's XOR subset is already marked as enabled, add a token	An XOR can only have one edge node marked as enabled
Additional rules (after a trace has been parsed against an individual)	**Reason**
Check each XOR edge subset; if all edge nodes in a subset are still marked as possibly enabled, add a token	Any possibly enabled XOR edge node subset that has not been enabled, after the parsing of a trace has completed, indicates a potential error; a token must be added as punishment for this
Check each AND edge node; if still marked as possibly enabled, add a token	Any possibly enabled AND edge node that has not been enabled indicates a potential error; a token must be added as punishment for this

If the task does not appear in the individual a token is added (counted as a missing token in the fitness calculation, see 5th rule down in Table 10). For a task to be successfully parsed against an individual the input subsets to that task are assessed for the presence of AND edges (edges that have subsets at either end containing just one edge node). Such edges must be marked as 2.0 (possibly enabled) for the parser to progress to the next task. If all the AND incoming task edges are set to 2.0 (possibly enabled) they can then be set to 3.0 (enabled) to show that the preceding task and edges in the individual match the trace being parsed. If this is not the case a token is added for each non-enabled AND edge (counted as a missing token in the fitness calculation, see 2nd rule down in Table 10).

An incoming XOR edge even when set to 1.0 does not necessarily indicate a problem, as another branch of the XOR may be set to 3.0 or the XOR is not involved in that particular process trace (see 1st rule in Table 10). However, if an XOR set is the only input into a task and an edge in that edges XOR subset is already marked as 'enabled' a token is added as an XOR can only have one edge marked (see 6th rule down in Table 10). As can be seen from Figure 24 every task in the trace is parsed in a similar fashion. When a trace has been parsed against an individual, all edge subsets are examined to check that one edge has been set to 'enabled' or all edges are 'not enabled'. If all the edges of an XOR set are found to be marked as 'possibly enabled' a single token is added (counted as an extra token left behind in the fitness calculation, see 7th rule down in Table 10). Any AND edges that are still marked as 'possibly enabled' a single token is also added (counted as an extra token left behind in the fitness calculation, see 8th rule down in Table 10). Both missing tokens and tokens left behind reduce the fitness score for an individual.

Once every task in one trace has been parsed the next individual in the population is selected and the same trace is parsed against it (shown in Figure 24). Only when the last individual has been parsed can the next trace in the event log be read and parsed against the population. This cycle continues until every trace has been parsed against the population.

4.4.4. Fitness Calculation

In order to assess the fitness of an individual in the GP approach a fitness algorithm is required. Equation 2 details the fitness function of Alves de Medeiros (2006) that is used in the experiments with the GP mining approach. Once the entire event log has been parsed against an individual equation 2 is used to give that individual a fitness score. It must be noted that there is no data cleansing routine applied to the event log before mining. In addition the process logs must contain at least 300 traces (this number has been confirmed from initial experimentation).

$$Fitness(L, In, In[]) = PFComplete(L, Ind) - \kappa * PFPrecise(L, In, In[]) \quad (2)$$

In equation 2, L is an event log and In an individual (a process model represented by a graph object). The notation $In[]$ represents a generation (or population) of individuals (process graphs). Each individual in a generation is being measured for its fitness , so *Fitness* can be calculated with a completeness measure (represented by the $PFComplete$ function of equation 2) and a preciseness measure (represented by the $PFprecise$ function of equation 2) in its ability to reflect the event log process traces.

An individual is both complete, if all of the traces in the event log can be parsed against it without error, and precise, if it does not contain any additional edges or tasks that are not recorded in the event log; the fitness for such individual would be 1.0 (Alves de Medeiros et al., 2004a). The value κ, shown in equation 2, acts as a punishment weighting for the amount of extra behaviour found in an individual (its value is in the range [0,1]). Extra behaviour refers to additional edges present in an individual but not expressed in the event log. As mentioned earlier in the parsing section when a trace cannot be parsed against an individual correctly, either because a task has not been enabled or missing inputs have disabled a task, that individual's fitness is reduced by counting the added tokens.

In equation 3 the *PFComplete* function of equation 2 is detailed along with the fitness reduction measure. In equation 3 the *allParsedActivities* function is a count of the amount of tasks that were parsed against an individual. The value from this function is then reduced by the punishment function. The value resulting from *allParsedActivities* minus punishment is then divided by *numActivitiesLog* which is a count of all the tasks in the event log. When mandatory inputs to a task are missing; a token is added for each missing input which acts as a penalty. If there are tokens left behind (known as extra tokens left behind in equation 3) when a trace has been fully parsed (that is edge node subsets with edge nodes still marked as possibly enabled as none of the tasks that they refer to have been enabled) these tokens are also counted and used to reduce the fitness score for that individual. For the missing tokens calculation the number of missing tokens (represented by *allMissingTokens*) for an individual is divided by the number of traces in the event log (*numTracesLog*) minus the number of traces in the event log that have missing tokens (*numTracesMissingTokens*). The second part of this calculation divides the total amount of tokens left behind (represented by *allExtraTokensLeftBehind*), for an individual, by the number of traces in the event log (*numTracesLog*) minus the number of traces in the event log that have tokens left behind (*numTracesExtraTokensLeftBehind*). The missing tokens and extra tokens left behind parts of the equation are then added together to give a punishment value for an individual.

$$PFComplete(L, In) = \frac{allParsedActivities\,(L, In) - punishment\,(L, In)}{numActivitiesLog(L)}$$

where

$$punishment\,(L, In) = \frac{allMissingTokens\,(L, In)}{numTracesLog(L) - numTracesMissingTokens\,(L, In) + 1} + \frac{allExtraTokensLeftBehind\,(L, In)}{numTracesLog(L) - numTracesExtraTokensLeftBehind\,(L, In) + 1} \quad (3)$$

In terms of preciseness the individual should not contain behaviour (expressed as tasks and edges) that is not detailed in the event log traces. The measure $PFprecise$ in equation 2, and shown in full in equation 4, provides an approximate measure of the amount of extra behaviour (additional edges not recorded in the event log but present in the mined process) an individual allows for in comparison to other individuals in the generation (Alves de Medeiros, 2006). In equation 4 the number of tasks that could be enabled for an individual (represented by $allEnabledActivities$) is divided by the maximum number of enabled tasks that individuals in that population had. According to Alves de Medeiros (2006) on average individuals with a higher amount of enabled tasks tend to be more likely to express extra behaviour. The amount of enabled tasks in an individual is divided by the maximum value obtained for enabled tasks that the generation of individuals had after parsing all of the event log traces.

$$PFprecise(L, In, In[]) = \frac{allEnabledActivities(L, In)}{max(allEnabledActivities(L, In[]))} \quad (4)$$

Once each individual in the initial population has a fitness score, individuals may be selectively modified by crossover and mutation operators. An elite of two individuals is used in this crossover. This means that for every new generation created the two fittest individuals are copied directly over (unchanged) to the new generation. In order to create the remainder of the new generation a five individual tournament selection function is used. Tournament selection was chosen above alternatives, such as roulette wheel, due to it's to perform an even balanced selection with a minimal of bias in large populations (Cox, 2005). With tournament selection five individuals are selected at random from the existing population and the two fittest are selected. These are the parent individuals that may undergo modification through crossover/mutation (depending on a predefined rate expressed as a percentage of the existing population) or just included in an unchanged state in the new population. This selection process continues until a predefined number of individuals have been inserted into the new population. Once $PFcomplete$ and $PFprecise$ have been calculated the value κ, shown in equation 2, acts as a punishment weighting for the amount of extra behaviour found

in an individual. The value of κ may be larger than 0 and smaller or equal to 1. A value of 0.025 for κ is recommended by Alves de Medeiros (2006) through empirical research.

4.4.5. Crossover Operator

The crossover operator in use in the GP approach has been motivated by the work of Alves de Medeiros (2006) and modified for use with the GP representation. In the work of Alves de Medeiros (2006) duplicate tasks are allowed in the crossover operation, this is not the case with the GP approach as each edge node in a subset must be unique. Unlike many GP implementations (particularly tree based GP) the size of the individuals are not changed by any of the operators only the structure is varied. This is due to the fact that genotypes of different lengths are not required as the number of nodes for a particular process is already known (by reading the initial entry in the event log).

A task, which exists in both individuals, is selected at random as the crossover point. The input and output edge node subsets are then split at a randomly chosen swap point. In Figure 24, example input subsets for two parent tasks are shown. The letters in the subsets refer to the names of process tasks that link to the task in the form of an edge. The operator for each subset defines the edge relationships. In Figure 24, assume the swap point is after the second subset in each parent. The crossover cycle is repeated for both input and output sets of the selected crossover task. As mentioned earlier a pre-selected crossover rate determines how often crossover will take place in a generation.

parentTask1 = AND(A:B), AND(A:E), XOR(A:B, A;C, A:D)
parentTask2 = XOR(A:B, A:E), AND(A:C), AND(A:B), AND(A:D)

Randomly select a crossover point in parentTask1 and another crossover point in parentTask2

swapset1 = XOR(A:B, A:C, A:D), remainder1 = AND(A;B), AND(A:E)
swapset2 = AND(A:B), AND(A:D), remainder2 =XOR(A:B, A:E), AND(A:C)
swapset2 + remainder 1
swapset1 + remainder 2

If crossoverNumber > crossover rate

For each subset in swapset 1 select, with equal

probability, one of the following two crossover methods –

For each subset in swapset1 do either -

1. Add subset as a new subset in remainder2
XOR(A:B, A:E), XOR(A:B, A:C, A:D)

2. Select a subset in remainder2 and remove the edges that are in common between that subset and the subset from swapset 1
XOR(A:B, A:E, A:C, A:D)

Repeat for the combination swapset2 & remainder1

Figure 25: Outline of GP Crossover (Adapted from Alves de Medeiros (2006) for use with the GP Graph Representation)

In Figure 25 the output set of task A in two separate individuals is represented as parentTask1 and parentTask2. parentTask1 contains three subsets, two AND subsets (each containing a single edge node) and one XOR subset (containing three edge nodes). A swap point is selected at random in parentTask1. The subsets after the crossover point are removed (the crossover point is subset two) leaving remainder1. The removed subsets become swapset1 (swapset1 contains XOR(A:B, A:C, A:D) the third subset of parentTask1). In a similar fashion a swap point is selected at random in parentTask2 (as the second subset) meaning the last two subsets are removed to create swapset2 (AND(A:B) , AND(A:D)). The remaining subsets (XOR(A:B, A:E), AND(A:C)) become remainder2. Crossover between parentTask1 and parentTask2 will only proceed if a randomly selected number (crossoverNumber) is greater than the crossover rate. Assuming crossoverNumber is higher than the crossover rate the subsets in swapset1 could be combined with remainder2 in the following ways –

- swapset1 subsets could be added as new subsets in remainder2
- the edge nodes of swapset1 subsets could be added to existing subsets in remainder2

How each subset is combined from swapset1 to remainder2 is decided at random. This process is then repeated for swapset1 and remainder2. Once the input sets between

parentTask1 and parentTask2 have been crossed over this sequence (shown in Figure 25) is repeated for the output sets.

In the crossover approach of Alves de Medeiros (2006), duplicate tasks are allowed in process models, this however is not the case with the GP approach so when node edges are added to existing subsets a check is made to see if a similar node edge already exists (and if it does the node edge is not added).

A step by step outline of GP crossover is given below (for an explanation of the terms used see Table 8): –

1. Use tournament selection to select the two fittest individuals from five individuals randomly selected from the population.
2. If the individuals are identical go to step 11(do nothing, the two individuals are left unchanged).
3. Randomly select a task to be the crossover point in both individuals (this task must be the same for both individuals).
4. Create an empty edge node set called set 1 and copy in the input set of selected task in the first individual.
5. Create an empty edge node set called set 2 and copy in the input set of selected task in the second individual.
6. Select a swap point to crossover in set 1 (based on the random selection of a subset in set 1).
7. Select a swap point to crossover in set 2.
8. Swap over the subsets between sets 1 and 2 (from the swap point subset to the end of the set).

9. For both sets - add the swapped subsets, set by set, to the other set in one of the following two ways (to be selected at random).
    a. Add the swapped subset as a new subset.
    b. Randomly select a subset and merge the swapped subset with removing any common edge nodes.
10. Repeat steps 3 to 8 but use the OUTPUT sets instead of the INPUT sets.
11. Return the two individuals that result from the crossover.

In the GA approach of Alves de Medeiros (2006) after the crossover and mutation operators have run, a repair routine is executed to make sure that only viable changes are made to individuals. The repair routines in the GP approach are complemented by the inherent error checking provided by JGraphT (Website for JGraphT, 2009) graph objects. The repair routines are streamlined due to the fact that the GP graph representation can be modified directly without using an intermediate abstract construct as with the approach of Alves de Medeiros (2006).

The crossover operation is also illustrated in Figure 26 where crossover occurs at task D meaning that changes at task D, where input and output subsets are swapped between parent 1 and 2, produce the two offspring shown directly below the parents. In effect by swapping input and output subsets at task D between individuals, after repair, gives the two offspring shown in Figure 26. It is the two offspring from this crossover that are then included in the new population, though they may first be subject to mutation, on a random basis, before inclusion in the new population (this operator is discussion in the following section of this chapter).

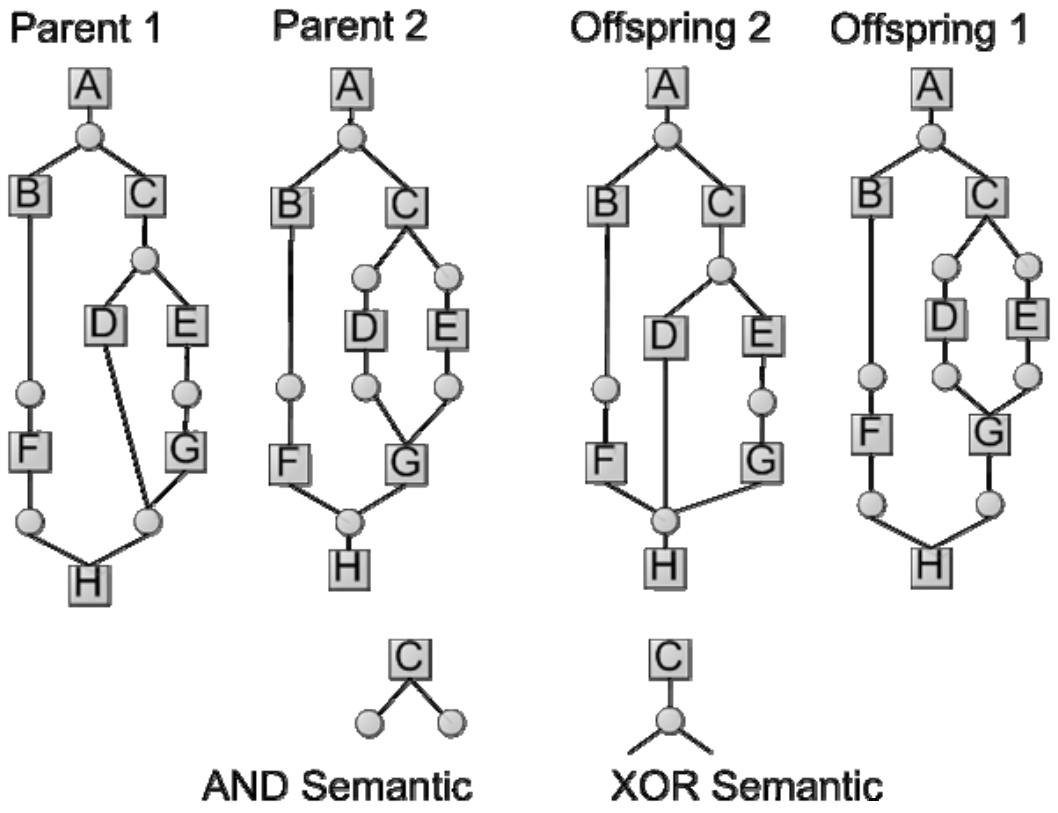


Figure 26: An Illustration of GP Crossover

4.4.6. Mutation Operator

The second operator used by the GP approach to modify individuals for inclusion in a new population is mutation. Once a pair of individuals has been selected for modification by crossover (or not) they may be selected for mutation (again depends on a predefined rate expressed as a percentage of the existing population). Each individual in the pair is mutated separately. For each task within an individual, a random number (between [0,1]) is generated, if this number is less than the mutation rate, mutation will take place on the subsets of the input and output sets of that task. There are three different ways that the input and output subsets for a randomly selected task can be mutated. An edge node may be added to a randomly selected subset or an edge node may be randomly selected from the same subset (selected at random) for deletion. Alternatively edge nodes of all subsets within a task input set may be

redistributed randomly. This is repeated for the output set of the same task. Figure 27 shows the mutation procedure for the input set of a task A.

If mutationNumber > mutation rate
Input set task A = AND(A:B), AND(A:E), XOR(A:B, A:C, A:D)

Randomly select one of the four mutation operators below to mutate the input set for task A

1. Choose a subset and add an edge node to the subset
(edge node pointing at a task selected from the complete set of available tasks in the individual)
AND(A:B), AND(A:E), XOR(A:E, A:F), XOR(A:B, A:C, A:D)
Task F is added as a new edge node to the AND subset AND(A:E) to create XOR(A:E, A:F)

2. Add a new subset (create a new AND subset)
AND(A:B), AND(A:E), AND(A:F), XOR(A:B, A:C, A:D)
Task F is added as an edge node within a new subset

3. Remove an edge node from a subset
AND(A:B), AND(A:E), XOR(A:B, A:D)
Task C is removed, the edge node for task C is removed

4. Random redistribution of edge nodes between subsets
AND(A:C), XOR(A:B, A:E, A:D)

Figure 27: An outline of GP Mutation (adapted from Alves de Medeiros (2006))

In Figure 27 the output set of task A has three subsets (AND(A:B), AND(A:E), XOR(A:B, A:C, A:D)). If a randomly chosen number mutationNumber is greater than the mutation rate value then mutation will occur. At random one of the following four options will be used to mutate the input set –

- Add a new edge node to an existing subset.
- Add a new subset (and edge node) to the output set.
- Remove an edge node from a subset in the output set.
- Randomly redistribute edge nodes between existing subsets in the output set.

A new edge node can be created by selecting, at random, a task that exists in the individual e.g. the edge node A:F; where task F is selected at random and the edge node A:F is created. Once the output set of task A has been mutated the input set can then be mutated following the procedure shown in Figure 27.

The mutation operator may be summarised in the following steps:-

For every task in an individual:-

1. Generate a random number; if the number is less than the mutation rate the task will be mutated.
2. Randomly select a method of mutation for the input set of the task from the following -
    a. Randomly select a subset and add a random edge node to the set.
    b. Add a new AND subset.
    c. Remove an edge node from a subset at random.
    d. Redistribute the edge nodes between subsets.
3. Repeat step 2 for the output set of the task.
4. If the task is the last in the individual, return the individual.

In Figure 28, it can be seen that mutation can change the relationship of edge nodes from XOR to AND. The mutation operator for the GP approach has again been adapted from the work of Alves de Medeiros (2006) and modified for use with the graph representation. As with the GP crossover operator duplicate tasks are not allowed in the mutation operation, each edge node in a subset must be unique. Figure 28 gives an illustration of mutation, where a mutation at task A turns an XOR split into an AND split.

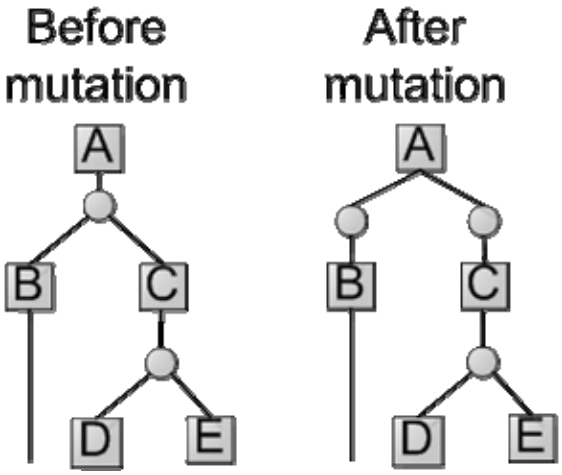


Figure 28: An Illustration of GP Mutation at Task A

This cycle of fitness assessment and modification of individuals is repeated for generation after generation usually until a 'fittest' individual has been found or a certain number of generations have been produced. Both crossover and mutation operators are designed to produce feasible (properly connected) individuals (after a repair routine has run).

4.5. Initial Experiments with the GP Process Mining Approach

4.5.1. Experimental Set-up

When using GP it is a customary to set out the key parameters to be used in the application of the technique in a Koza table (1992). The Koza table for this GP process mining application is shown in Table 11.

Koza (1992) states that this tableau can broadly describe a set of steps that needs to be identified before progressing with a GP project. The objective of the GP process mining approach is to obtain a graph that best describes the process data contained in the event log being mined. The terminal set contains, in effect, the 'state variables' of the system the GP approach is working with.

Table 11: Koza Tableau for the GP Approach

Parameters	Values
Objective	To mine a best fit process from the given event log data.
Terminal Set	The process task marked as STOP terminates the parsing of the process it is contained in. Similarly the START task initiates the parsing of a process. Each task in a process can also be thought of as a terminal (e.g. tasks A, B, C….).
Function Set	AND, XOR, Node Marking.
Fitness Cases	Event log traces to be parsed against each individual.
Raw Fitness	Number of correctly parsed traces for each individual.
Standardised Fitness	Number of correctly parsed traces for each individual with a penalty for each missing task and for non enabled tasks calculated against the values for the generation as a whole (see Table 1 for parsing rules and equation 1 for the fitness measure).
Termination criterion	Fitness of 1.0 or maximum number of generations chosen (e.g. 1000 generations, the level used in the GA approach of Alves de Medeiros, 2006)).
Selection	Tournament section, five individuals.
Initialisation Method	Parse first event log trace against first individual (of the initial population created by calculating dependency relations in the event log – see section 5.4.1 for a description of this).

So in the case of the GP process miner the start and stop tasks are used as initiation and termination criterion for the parsing of individual process traces. Similarly the function set consists of the AND/XOR marking of the individual edges. The fitness

cases are in effect the measure by which each individual is scored and comprise the set of event log traces that each individual is parsed against. In the case of the GP process mining approach the raw fitness is the number of correctly parsed traces for an individual. The raw fitness is a broad fitness measure that is used in a GP application, as Koza puts it “the measurement of fitness that is stated in the natural terminology of the problem itself” (Koza, 1992). The standardised fitness, when used in optimisation problems, is in general to find the highest numerical value for fitness. In the case of process mining where the value closest to 1.0 or the value 1.0 itself indicates the optimal individual. In this situation the larger value of raw fitness is considered good and standardised fitness is computed from raw fitness. In the computation of standardised fitness, in the case of the GP process miner, penalties for each missing task and for non enabled tasks, calculated against the values for the generation as a whole, are derived as an elaboration on the raw fitness.

The termination criterion is either when an individual reaches the fitness of 1.0 or a set number of generations has been reached, in this case the set number is 1000 generations. As mentioned earlier, the selection is by a five individual tournament. The two fittest individuals from the tournament are put forward for modification and inclusion in a subsequent generation. The GP process mining approach is initiated by the parsing of the first trace from the event log against an individual from the initial generation.

In addition to the parameters included in the Koza tableau a number of additional parameters are given

Generations 1000
Population size 100
Elitism rate 0.02
Crossover rate 0.8
Mutation rate 0.2
κ (extra behaviour fitness reduction) 0.025

The values are recommended through the empirical work of Alves de Medeiros

(2006). Of particular note from these settings is the κ extra behaviour fitness reduction. This is the rate by which an individual will have their fitness reduced by, if by parsing it is found that additional behaviour, not expressed in the event log, is allowed by the individual. Additional behaviour may take the form of links between tasks and changing semantics such as the swapping of XOR edge nodes to AND edge nodes (the term semantics is used in relation to edge nodes and refers to the type of relationship an edge is in, which may be AND or XOR (Alves de Medeiros, 2006)). The parameter set presented here has been the subject of additional systematic experimentation for conformation of its suitability for use with the GP technique (especially the value used for κ).

The GP code is written in Java and hosted in the ProM process mining framework (van Dongen et al., 2005b). The experiments were carried out on a standard desktop PC running Windows XP with 500MB of RAM and a Pentium 4 3GHz processor.

Two sets of test processes, in the form of event logs, were mined. The event logs are artificial; though they do model real life process constructs (Alves de Medeiros, 2006). All of the event logs are balanced (contain a representative amount of traces for each path through a process). The process data sets are shown in Table 12. The data sets are provided by Alves de Medeiros (2006) through van Dongen et al. (2005b). In this set of initial tests there was a concentration on the correct mining of process structures rather than both structures and semantics. Two important structural aspects were noted in the mining of the processes. The term sequence, in this context, refers to the complexity of the flow of tasks in terms of process constructs and the amount of tasks involved in a process. Parallelism is a particular type of process construct where a number of tasks that are linked back to the same task may be executed simultaneously. In Table 12 the terms Low Medium and High have been used to describe the complexity of certain structural aspects of mined process models. The exact level of complexity of a process model is difficult to establish in empirical terms. In light of this the author has decided to use the three aforementioned complexity gradings which allow for a relative comparison between the process data sets used in

this thesis.

In terms of the size of the event logs used both process data 2 and 3 contained 300 traces. Process data 2 contained ten tasks and process data 3 ten tasks.

Table 12: Process Test Data Sets for the Experiments

	Sequences	**Parallelism**
Process data 2	Low	Low
Process data 3	Medium	High

This type of construct is particularly challenging to mine correctly in that it is difficult to ascertain from a linear event log which tasks are in parallel. Normally an event log with a large enough number of process traces indicating as many of the possible correct paths as possible is required for successful mining results. In terms of the size of the event logs used both process data 2 and 3 contained 300 traces. Process data 2 described a process containing ten tasks and process data 3 was also comprised of multiple occurrences of ten tasks.

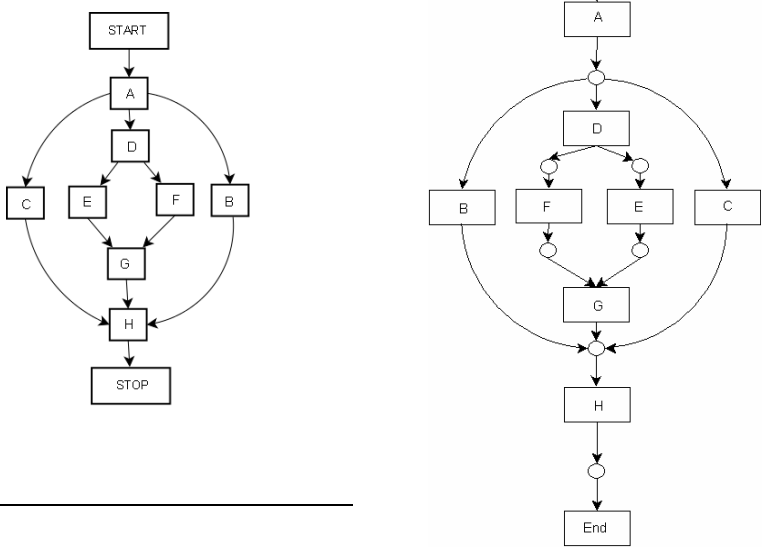


Figure 29: Correct Templates for Process Data 2 Structure View (left) Semantics View (right)

Process data 2 offers a medium level of complexity in terms of parallel process execution (the structure and semantics are detailed in Figure 29). Of particular note are the three parallel tasks D, B and C. These three tasks provide the main area for potential mining errors in both structure and semantics. The AND split and join construct at the centre of the process can also be misinterpreted (split at task D and join at task G), especially as all three branches from task A split as XOR and then rejoin at task H in another XOR semantic construct. The correct 'templates' for process data 2 are shown in Figure 29.

Parallel processes are amongst the most difficult constructs to mine successfully (van der Aalst and Weijters, 2004). Process data 3 (the correct templates for process data 3 are shown in Figure 30) provides a challenging structure with AND spit/join semantics and five parallel tasks.

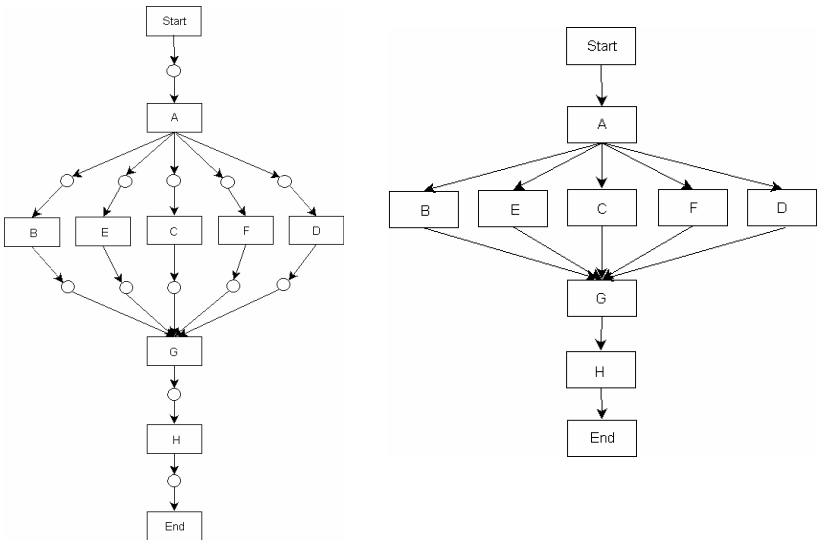


Figure 30: Correct Templates for Process Data 3 Structure View (Right) Semantics View (Left)

4.5.2. Experimental Results

In the completion of the experimentation and as a comparison the three processes mined by the GP approach the five tests were also carried out with the GA of Alves de

Medeiros (2006) using crossover and mutation operators. Overall 10 runs were conducted with each of the three process data sets for the GP and GA techniques, with each run taking approximately 10 minutes to complete. The mining outcomes presented in this section are the typical results obtained from 10 runs of each process data performed with the GA and GP approaches (the best and typical results are the same for all process data presented here). In this context the term typical refers to the result that was reliably achieved on the greatest number of occasions.

The structure for process data 2 could be mined correctly by both GP and GA techniques and the mined process is shown in Figure 30 (on the left for GP and right for GA, best and typical results are the same for both GP and GA results). However, it was noted that the GP technique did add an extra incorrect link between a middle task and an end task on two out of ten runs. Semantics are not presented in the results here as at this stage of development there was a concentration the mining of process structures. The semantics for the GP were on the whole correct in terms of mining the correct semantics for each split and join task for process data 2, though additional incorrect subsets of edges were also included in the results. The GA could mine the semantics for this process without error. The data set for process data 2 contained low to medium complexity levels of task sequences and parallel splits and joins.

For the data set 3 the GP produced two or three superfluous links from task A on all runs (shown in Figure 31 on the left, task A has two edges to tasks B, E and F) of the test though the general structure of the process was still well defined. The GA consistently produced extra links between the parallel tasks making it difficult to ascertain the tasks that were in parallel on several runs. The best mined structure for the GP and GA approaches are shown in Figure 31 (on the left for GP and right for GA in Figure 31, best and typical results for both GP and GA are the same as presented in Figure 31).

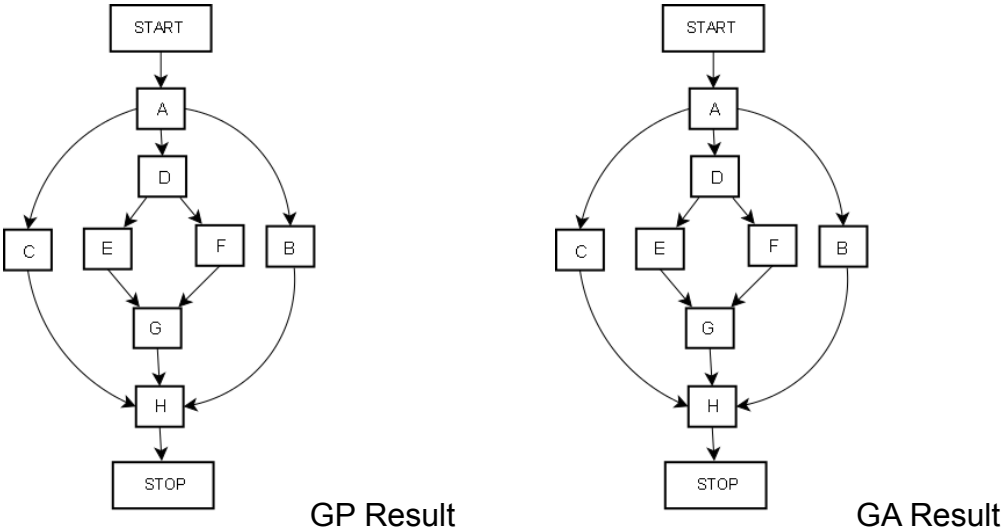


Figure 31: GP and GA Process Mining Results for Data Set 2

The semantics for process data 3 could not be mined correctly by either the GP or GA mining approaches with multiple subsets of tasks generated for the parallel spilt and join tasks of the mined process models produced from both the approaches.

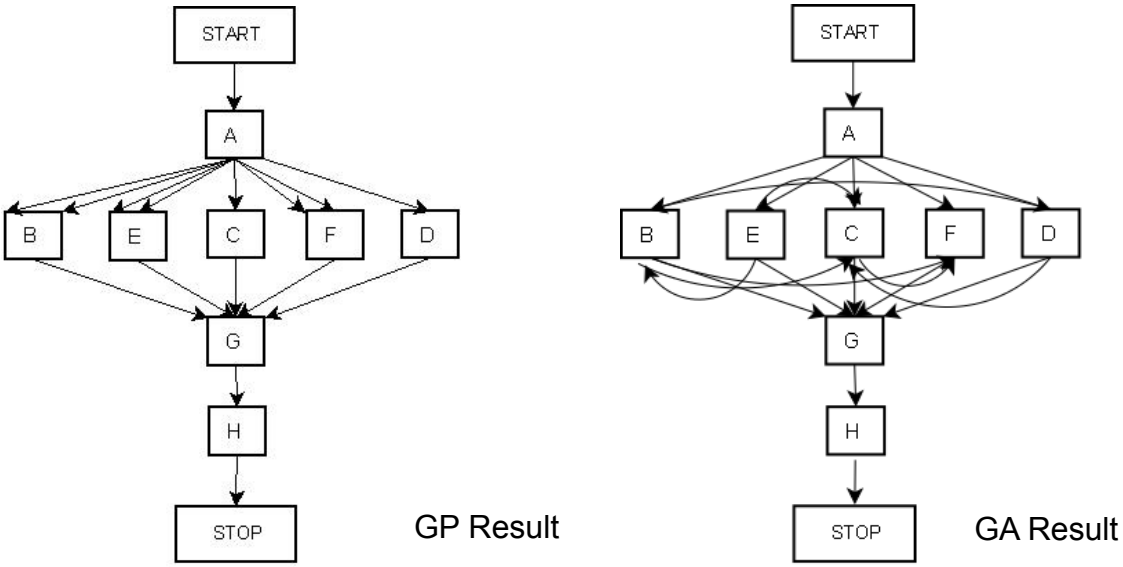


Figure 32: GP and GA Process Mining Results for Data Set 3

The experiments showed that the mining of process models with low to medium levels of complexity, such as process data 2, was possible with the GP approach. The structure of process data 2 was mined correctly by the GP approach though the semantics did contain some errors at the split and join tasks. The mining of the structure of process data 3, while slightly incorrect, was an improvement of the result obtained by the GA approach (shown in Figure 32). A one or two additional edges were added to the graph from task with the GP approach which was eventually traced to the error handling in the creating of the graphs. The reason for the GA's poor performance on this test is most likely due to its fitness parsing approach not being able to determine if a connecting edge from a neighbouring parallel task that has been enabled or not (hence the multiple edges between the five central tasks that are meant to be in parallel). The semantics for both the GA and GP approaches contained errors for the mining of this data set. In terms of the GP the additional edges produced between task A and some parallel tasks could have been caused by the production of duplicate edge node sub sets. It must be noted that the experiments undertaken at this stage were preliminary and designed to explore the use of GP, graph representations and parsing techniques as a viable method of mining business processes. However the results obtained do demonstrate the GP technique's ability to mine correct process structures with a level of consistency.

In initial experiments conducted with the GP process mining approach, it was found at this early stage of development that the removal of the mutation operator (retaining just crossover as the only operator) produced individuals with a higher level of fitness. Experiments were also conducted with the mutation operator set at a reduced rate though this still led to poor individuals at the end of a process mining run. It was noticed that there were problems in the variety of edge node sets being created. It was noted that due to the requirement of the representation to pre determine whether a subset is of type AND or type XOR it was not possible to create single XOR edge node subsets. This had a detrimental effect on only on the semantics but also, for process data 3, the structure. Although the single XOR issue was known by the author at the outset it was hypothesised that the GP approach would be able to overcome this

by relaxing the rules on the linking of AND edge nodes (being flexible about the real AND/XOR nature of single edge nodes at fitness parsing time). However experimental results were no better than those presented in this chapter where no relaxing of the fitness measure for single edge nodes has been allowed. This is one issue that is addressed in the following chapter. In additional experimentation process data set 2 was adapted to contain a low level of noise. The result showed that the mining of both structure and semantics for this data set, when mined by the GP approach, deteriorated significantly. At this early stage of development the GP approach was not able to mine noisy data sets.

Crossover was also used in a specialised way for this set of experiments. After experimentation it was found that using just one of the two crossover techniques (adding subsets from one task as new subsets in an input or output set of another task) produced the best results. This was due to non-destructive nature of this crossover technique, in that existing subsets were not altered. The crossover operator was only allowed to add new subsets within the input and output sets of a task (crossover is detailed in Figures 25 & 26 In section 4.4.5).

4.6. Summary

This chapter has explored the use of GP as a process mining technique and the use of a graph structure as a method of representing individuals. The main steps of the GP process mining approach have been outlined along with the fitness parsing approach utilised. At this stage the GP approach utilises a simple method of crossover to mine low to medium complexity constructs from data, though the initial experimentation demonstrates the potential of this approach, especially in dealing with highly parallel processes. The parameters for the GP approach have been given and a Koza tableau (1992) has set out the approach in formal GP terms. The following chapter details the development and refinement of the proposed GP approach in order to correctly mine not just the structure of more complex processes but the semantics as well. The mining of process data containing noise is also explored in the following chapter along with refinements aimed at the mining of complex parallel constructs.

5. Extension of the Proposed Business Process Mining Approach for Mining Noisy Process Logs and Complex Constructs

5.1. Introduction

In the previous chapter it was shown that the GP process mining approach had been used to successfully mine the structures of two low to medium complexity processes. It was also demonstrated that the technique may also address the mining of more complex process structures such as parallel tasks. This chapter details how the GP approach was further developed to be able to address the following areas: –

- Mine a wider range of process structures.
- Mine AND/XOR semantics for the split and join points within a process.
- Mine processes in the presence of noise.
- Mine more complex processes and constructs such as parallelism.

This chapter outlines modifications made to the representation and parsing along with a set of functions to deal with the above areas. This chapter concludes with a set of preliminary evaluation experiments.

5.2. Limitations of the Initial GP Approach

While achieving some success with the mining of low to medium complexity process structures the GP approach does have limitations in the mining of more complex constructs and in the correct identification of semantics (finding out if a split or join within a process is AND or XOR). In order to address these limitations it is

necessary to review the representation and the parsing used within the technique. The initial GP approach was not able to mine process data containing noise.

With more complex processes, incorrect links were more prevalent, especially between split and join points. It was also noticed that the tasks that were meant to be in parallel were occasionally linked together (this experimentation is detailed in chapter 4).

The following sections detail the work conducted to address the limitations and the results of a number of preliminary experiments are presented.

5.3. Modification to the Graph Representation

From the experimentation as described in chapter 4 it was found that there were problems in the correct mining of edge semantics. The problems concerned the following areas –

- Production of duplicate edge node subsets when not required.
- Mismatches between input edge nodes of one task and output edge nodes of linked tasks, for example –
    - Subset containing a single AND edge node in one task being linked to a task with only an XOR subset available (an AND edge node must link to another AND edge node as an edge between tasks must be either AND or XOR).
    - XOR subset being linked to an AND subset containing an AND edge node (an XOR edge node must link to another XOR edge node as an edge between tasks must be either AND or XOR)
- XOR subsets containing a large amount of edge choices (over describing the data)

In the current implementation of the GP process mining approach, single edges are interpreted as AND edges. This causes a problem when trying to mine edges where only one end is in a single subset and the other is in an XOR subset. This particular edge problem is encountered in the work of van Dongen and van der Aalst (2005a). In the current GP such an edge would be incorrectly interpreted as an AND when it should be an XOR. Therefore, in the current representation of the GP it would be necessary to create the notion of an XOR set that contains just one edge (in Figure 33 an output set for task A is shown, containing three subsets, with one of the two XOR subsets containing a single edge node). This however creates an additional problem in that it is not possible to know which type of subset a single edge should be in at task node creation time. This is due to the fact that the creation of a single edge node connector on one task would require knowledge on the creation of the requisite connector in the other linking task plus knowledge about other edge sub sets in that linking task (in order to find out which subset to link to if more than one option is possible).

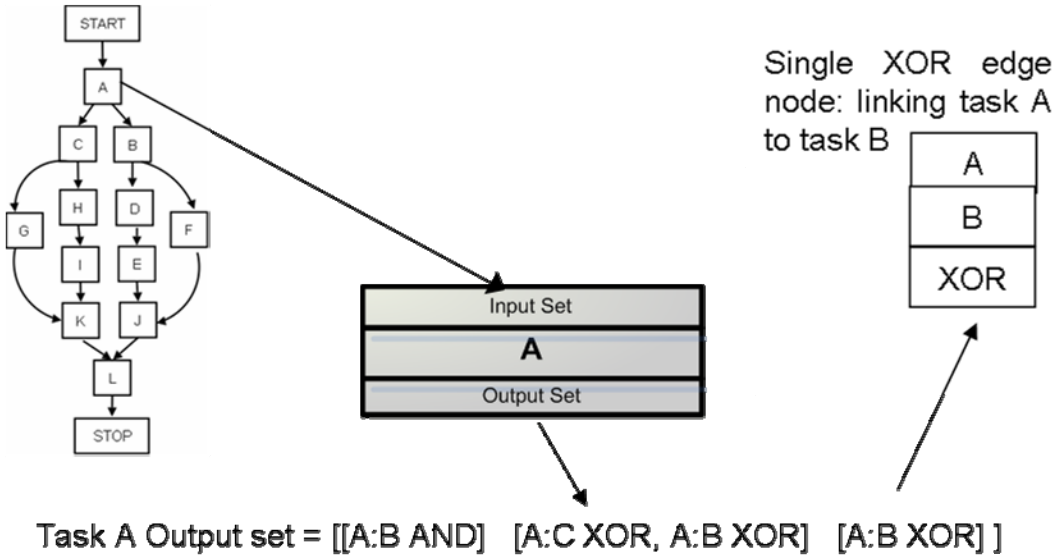


Figure 33: The Output Set of Task A Showing Three Subsets

A number of options were explored in trying to solve this issue and allow the GP approach to correctly identify single AND/XOR edges –

1. Use of heuristics.
2. Enumeration.
3. Semantic abstraction (the notion of edge connectors).
4. Removal of edge semantics from subsets.

(1) Heuristics were considered as an option though few existing techniques deal with edges that have predefined semantics (already in strict AND/XOR subsets as shown in Figure 33). The work of Weijters and van der Aalst (2001, 2003) utilises the causal matrix which does not explicitly identify the semantics of single edges, but builds up graph abstractions from frequency/dependency matrices, and utilises a set of heuristic rules for the further refinement of process models. Utilising an approach similar to this with pre-determined edge semantics would require the development of additional heuristics.

(2) Enumeration is proposed as an alternative solution (by the author of this thesis) and is the next option to be discussed here. This would involve the calculation of all the semantic options for a structure, creating a new graph structure for each different edge subset. Alternatively the input and output subsets could be enumerated for all alternative linking options. This would require the possible creation of single edge node subsets marked as XOR. The former of these enumeration options is computationally expensive as each of the graph options would need to be parsed against each trace in the event log (taking several hours of processing time). The different permutations of the semantic options would need to be calculated, especially for larger processes. For the later option again enumerating the linking options to work out the semantics of single edges could be expensive in terms of computing resources.

(3) The approach of van Dongen (2005a) (the semantic abstraction option) takes a different approach to edge semantics defining AND, XOR and OR connectors for edges (the connectors define the relationships of the edges that they connect). Where a connector has only one incoming edge but multiple outgoing edges or one outgoing edge and multiple incoming edges the connector is removed and replaced with an edge. This has the effect of making the edge become part of a semantic by default when the resulting graph is built. Such an approach is interesting but would be difficult to implement with the current GP representation which does not support the abstraction of semantics into connectors.

(4) The final option (again suggested by the author of this thesis) requires the removal of edge semantics from the representation. This approach is similar to that required for utilising the causal matrix representation (the causal matrix is described in van der Aalst et al. (2005a) and provided through (van Dongen et al., 2005b)), in that no semantics are attached to edges when they are represented in a matrix. When the edge semantics are removed there is no way to identify the semantics of edge subsets that contain a single edge (they may be AND or XOR depending on the subset they are linked to in the linking task) until fitness parsing. However subsets that contain more than one edge are automatically XOR subsets. The semantic (AND or XOR type) of the single edge is therefore determined at the fitness parsing stage. In terms of the GP approach semantics from the subsets in the task nodes can be achieved through the stochastic nature of the fitness parsing, randomly determining the semantics of single edges.

Of all the options the one of removing the semantic edges is in the author's opinion the best option (advantages and disadvantages of each option are set out in Table 13). From the initial experimentation described in chapter 4 and the discussion above, it is clear that assigning semantics to edges is not the most practical of options.

At the outset it was envisaged that the matching of 'semantically labelled' edges between tasks was possible by use of an evolutionary technique and beneficial to the accuracy of the mining results. However in the context of an application which must return a result within a finite amount of generations this is not possible.

After consideration the option of edge semantic removal was employed in the revision of the GP representation. This option is computationally less expensive than the alternative options discussed above. It also required modest changes to the representation, which was not the case with the other options. The selection of this approach has also meant that changes to a number of rules were required in the fitness parsing stage along with the way that parsing operates. The revised subsets are shown in Figure 34. The parsing changes that were required will be detailed in a later section of this chapter.

Table 13: Options for Addressing the Single Edge Node Semantic Determination Problem

Technique	Advantages	Disadvantages
Use of heuristics	Would not require changes to the current GP representation	Few existing examples, much work would be involved in designing and testing different rule combinations
Enumeration	High probability of success	Computationally expensive
Semantic abstraction	High probability of success – solves semantic pre-determination problem	Would require extensive changes to the current GP representation
Removal of edge semantics	Probability of success as GP would be free to determine semantics of single edges	Changes to representation would be required

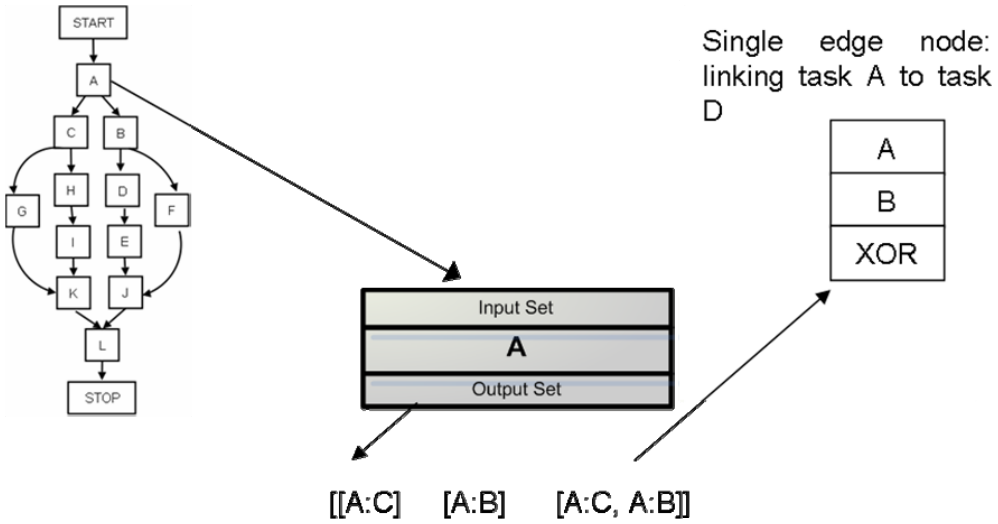


Figure 34: Revised GP edge subsets (with semantics removed)

5.4. Modification to the Fitness Parsing

As mentioned in the last section changes to the representation used by the GP process mining approach mean that the fitness parsing must also change. A number of revisions have been made to the original rule set (shown as Table 8 in chapter 4 section 4.4.3); these are reflected in the revised rule set shown in Table 14.

Table 14: Revised GP Parsing Rules

Rule	Reason
For each task select the first subset and randomly select one edge node in the subset as the first edge node to enable	The random selection of edge nodes will give the best chance of enabling a task whilst parsing all of the process traces in the event log (if the task subsets were selected at random the parsing would be different for each individual leading to the possibility of unreliable results)
If the edge node selected to be enabled in the subset cannot be enabled add a missing task token	The subset cannot be enabled as all corresponding subsets (containing the selected edge node) are already enabled or the task node the selected edge node is referring to does not exist

When enabling a subset containing multiple edge nodes enable the corresponding selected edge node and set all of the other edge nodes in the subset to not enabled	A subset can only be enabled by a single edge node
If a task in the trace does not exist in the individual, register the task as missing	A missing task is a serious omission from an individual
Additional Rules (after a trace has been parsed against an individual)	**Reason**
Check each single edge node subset; if all edge nodes in a set are still marked as possibly enabled add a token	Any possibly enabled subset that has not been enabled indicates a potential error; a token must be added as a punishment for this
Check each multiple edge node subset, if still marked as 'possibly enabled' add a token	Any possibly enabled sub set that has not been enabled indicates a potential error; a token must be added as a punishment for this

In this rule set, the approach to enabling XOR sets (subsets containing more than one edge node) remains the same as does the missing token and tokens left behind allocations. It is in the selection of edges and subset marking that there are significant differences (allowing for the fact that the notion of the AND set has been removed, as a single set may be AND or XOR).

With this type of parsing, there is no notion of AND/XOR so looking to match up edges from one task output set with another input set is not practical. Instead a stochastic method of matching is required. After much experimentation, the following methods were used for the marking of subsets -

1. Select edge node from the first input subset at random and mark first output subset in the linking task that contains an edge node back to the input task.
2. Select edge node from the first input subset at random and identify all of the output subsets in the linking task that contain an edge node back to the

input task and mark one of those subsets at random.

3. Select first edge node in the first input subset and identify all of the output subsets in the linking task that contain an edge node back to the input task and mark one of those subsets at random.

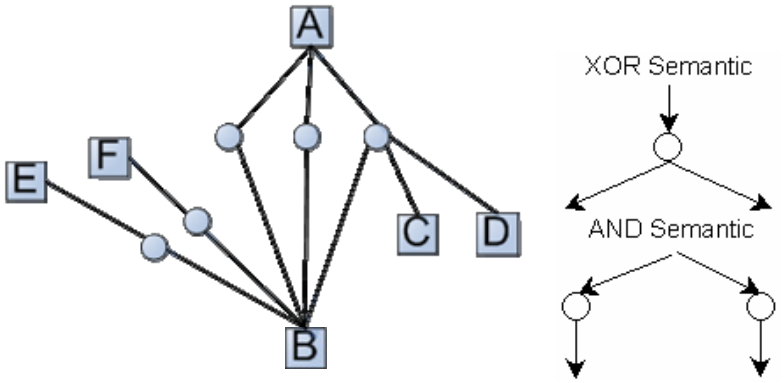
4. Select first edge node in the first input subset and mark first output subset in the linking task that contains an edge node back to the input task.

Marking Option Results

Of these four marking options, the fourth one produced the poorest results. Individuals resulting from this parsing included either extra links or had missing links between low and high frequency tasks. It is likely that the lack of any random selection in the way individuals are parsed and therefore linked together is more likely to lead to poor individuals. In a similar way, a high level of randomness leads to almost equally poor individuals. This is reflected in the results obtained from the second marking option which produced individuals with poor fitness. This is due to the fact that an individual is never parsed with any consistency; one fitness evaluation for an individual may produce a different score to a fitness evaluation for that individual in a subsequent generation even if the individual has not been modified. This leaves options one and three. For option three the individuals produced were better than those produced by two and four, though there were still a number of errors in the task links. This option did produce better semantics than options two and four and slightly better than the results obtained for the GP described in chapter 4. The best result overall was obtained by using option one where the input subset edge is selected at random and the first available output subset containing an edge to the input task is always selected. In this way a limited amount of random linking is allowed. This means that a greater number of linking alternatives between tasks can be explored by the GP. In all of the options discussed above the selection of subsets was sequential in that the first subset was selected first and the second and so on.

Preliminary experiments were also undertaken concerning the random selection of subsets in the input set of a task. However the results were very poor and no further experimentation in this direction was pursued. In addition to the modified rule set already mentioned for the fitness parsing used in the GP approach, an extra rule has been added in terms of the marking of subsets. When an output subset that contains an edge back to the input task is to be marked all subsets that contain that same edge are also marked. This rule helps to improve the process structure by reducing the prevalence of incorrect edge linkages between tasks in the GP approach. This rule is shown in Figure 36, where edge A:B is to be marked in task A. Figure 35 shows the graph view of the edges between tasks A and B.

Figure 35: Process Graph View of Revised parsing rule for GP approach (with legend for semantics on the right)



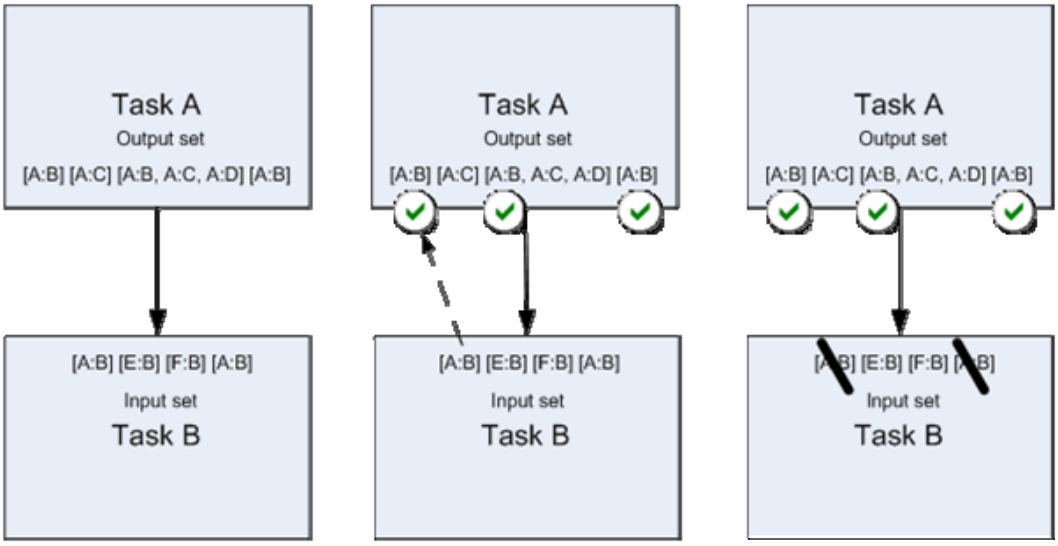


Figure 36: Revised parsing rule for the GP approach

As can be seen in the second illustration in Figure 36 A:B appears three times so all three subsets are marked (shown with ticks in Figure 36 illustration 2). Notice that the first subset containing A:B in the output set of task A is marked first (denoted by the dashed arrow in Figure 36 illustration 2), followed by the other two subsets that contain A:B. In the third illustration both subsets that contain the edge node A:B are treated as parsed edge nodes (shown as a strike-through in Figure 36 illustration 3). In Figure 36 the black arrows denote the presence of at least one edge between two tasks and the direction of flow.

5.5. Crossover in the Revised GP Approach

In the modified GP approach the use of crossover has been expanded from the crossover method where subsets are swapped between tasks. The changes made to the representation and parsing have allowed the use of one additional function, where subsets of one parent task can be merged with the subsets of another task. For details on the crossover function see chapter 4 section 4.4.5 where both of these crossover functions are described.

The addition of the extra function, in a limited number of evaluation tests, did not reduce the fitness of individuals, though its impact in evolving fitter individuals is inconclusive.

5.6. Mutation in the Revised GP Approach

In the GP approach as described in chapter 4 mutation was not used as an operator. This was due to its effect in significantly reducing the fitness of individuals and introducing errors in mined models, even when the rate of mutation was reduced. For the modified GP approach mutation was reintroduced. The mutation operator is described in chapter 4 section 4.4.6.

With this operator there are three ways in which an individual may be mutated. The first way is to add a task edge to a randomly selected subset. This involves randomly selecting a task from the complete set of tasks that make up the process. The second way is to remove an edge at random from a randomly selected subset. The third option involves the redistribution of edge nodes between subsets for an entire input or output set (the addition of an AND subset, as described in chapter 4 section 4.4.6, is no longer possible due to the removal of edge node semantics).

With this operator, once the input set of a task has been mutated the output set of the same task is also mutated (see chapter 4 section 4.4.6 for more details). From initial experiments, using the mutation operator with the crossover operator, the second and third mutation options helped to produce fitter individuals in terms of structure. It is likely that the removal of pre-determined semantics for single edge node subsets has allowed for the successful reintroduction of mutation, as the fitness routine is now fee to explore a wider range of linking options concerning such edge nodes. The addition of the first mutation option, however, reduced the fitness slightly. In the unmodified version of the GP approach, described in chapter 4, the mutation operator reduced the fitness of individuals no matter what the combination of mutation options. In the modified approach only the first mutation option reduces the fitness of individuals. This is likely to be due to that fact that any task, regardless if where it is in the process, can be selected as a task to be linked to or from. In the next section it is shown how an additional heuristic has been used to improve the performance of the first mutation operator (to increase the probability of selecting a viable task to link to), and its applicability to other areas of the GP approach.

5.7. Additional Heuristics

In the further development of the GP approach, exploration has been made into additional heuristic techniques that may improve the mining results. In particular more use can be made of the event log as a heuristic in itself. In many ways the event log contains the best guide for the correct mining of processes. In the heuristic technique

employed here the event log is used as an additional guide when selecting tasks to link to when performing mutation activities. In mutation the operator can make use of two functions devised by the author when selecting a task to add as a new edge node –

- A function called getLaterTask which is used to select a task that appears later on in a task sequence, for a process, than a given task
- A function called getEarlierTask which is used to select a task that appears earlier in a task sequence, for a process, than a given task

The mutation operator has three different types of mutation that it may use to modify an individual (these mutation types are derived, and adapted for GP, from the work of Alves de Medeiros (2006)). Of those mutation types the addOneTask function can make use of the get later and get earlier task functions in order to randomly select tasks that are more likely to improve the individual. In general when an edge node is being added to an input subset of a task then it is likely that it should ideally point to an earlier task than the task whose input set it is to be connected to. Similarly when adding an edge node to an output subset of a task it should point to a task that is later in the task sequence. The general sequence of tasks can be inferred from examination of a sample of event log traces.

In the case of both the getLaterTask and getEarlierTask functions ten event log traces are extracted from the event log and one trace is selected at random to act as the heuristic. The pseudo code for the getLaterTask function is shown in Figure 37.

getLaterTask(compareTask){

randomly select one trace

Select tasks after compareTask from trace and remove End task – put in collection

Extract laterTask from collection at random

```
    if  laterTask = EndTask
        laterTask = null;
    end if
    if laterTask = null
        fullTaskSet = get set of all tasks
        select from fullTaskSet
        if selectedTask = EndTask or compareTask then loop
        laterTask = select from fullTaskSet again
        end loop
    return laterTask
}
```

Figure 37: GetLaterTask Function Pseudo Code

From Figure 37 it can be seen that the getLaterTask function identifies tasks that are later on in the event log trace than the compareTask that is fed into the function (effectively the compareTask is the point in the trace at which the function should look beyond in order to identify later tasks). In Figure 37 a trace from the event log is selected at random. A given task, called the compareTask, is located in the trace and all of the tasks that appear after it are put into a collection from which a task is randomly selected (making sure that the End task is removed, as this is an artificial task that is dealt with separately). If no task can be selected normal selection takes place where all tasks in the process (bar the comparison task itself) can be selected from (this is the pseudo code from '*if laterTask = null*').

Once the later tasks have been identified the compareTask and the End task, if they appear, are removed from the collection of later tasks (the End task is literally the last task in the event log trace, effectively an artificial end point inserted into each event log trace along with an artificial start task). If a later task has not been selected by the function (because there are no later tasks) then a task will be selected at random from the full set of tasks (fullTaskSet in Figure 37) for that process.

The getEarlierTask and getLaterTask functions increase the chances of the mutation operator (through its addOneTask function) to improve an individual by encouraging task linkages that are more likely to be correct based on the task order discovered from an actual event log trace. The application of these heuristics is carried out with an inherent level of stochastic behaviour. This is in-keeping with the operation of evolutionary techniques of which GP is a member.

The concept of using the event log as an additional heuristic has also been employed in other areas of the GP approach. In the repair functions that are called after the GP operators have run, the getEarlierTask and getLaterTask functions may be called as part of the repair routine. An example is where a mutation has left a disconnected task; the task may be reconnected to a later task using the later task function. The mutation option, DeleteOneTask, that deletes a task (detailed in chapter 4 section 4.4.6) also utilises an additional output of the heuristic function of the fitness parsing. As will be shown in the next section the GP parsing is able to identify certain edges as being incorrect and add them to a list of edges to be deleted. This list can be used by the mutation operator to, at random, occasionally delete edge nodes that are known to be incorrect instead of selecting edge nodes at random for deletion.

In later sections of this chapter it will also be shown how the mining of highly parallel processes can also benefit from heuristics.

5.8. Further Improvement of GP Parsing Using Heuristics

As mentioned in the last section, the use of the event log as a heuristic can be extended to parsing. It is sometimes the case in parsing that a task may link back to a task (accept an edge as input) that is actually later on in the task sequence as recorded in the event log. This may occur due to the random nature in which evolutionary techniques such as GP create new individuals. It is perhaps beneficial to be able to decide when a task is being linked to out of sequence. For this reason the isTaskEarlier function has been devised to check if a task being linked back to is actually earlier on in the task sequence (so one task is not being linked back to a task that is actually later on in the task sequence). The pseudo code for this function is shown below in Figure 38.

```
isTaskEarlier(task1, task2){
    randomly select one trace
    is task1 earlier than task2
        compare task1.position with task2.position
    if  true exit function returning true value
    else if false
        add task2 edge to list for deletion
    exit function returning false value
}
```

Figure 38: isTaskEarlier Function Pseudo Code

In the pseudo code above, a trace from the event log is selected at random and used in the comparison of two tasks. In Figure 38 *if task2 is not earlier than task1* (the

output task of an edge is later in the task sequence than the input task) then a value of false is returned to the calling function. The calling function, in the parsing code, will then add a missing task token and continue with parsing. Also when a false is returned from this function, the edge linking task1 to task2 is added to a list of tasks for deletion. This list is used by the mutation operator as an additional heuristic allowing it to delete an edge known to be wrong. A value of true returned from the isTaskEarlier function will ensure that the normal parsing continues without the need to add a missing task.

This heuristic along with the changes to the GP approach outlined in this chapter so far have had a positive effect on the fitness of the individuals. An additional effect has been noted in that the amount of generations needed to arrive at a fittest individual has been reduced from 1000 to just 200 (after preliminary experimentation at settings ranging from 100 to 1000 generations at intervals of 50 generations). As will be demonstrated in the experimentation section of this chapter, even quite complex process can be mined successfully in just 200 generations.

5.9. Improvement in the Mining of Parallel Processes

One of the most challenging process constructs to mine is that of parallelism. As mentioned in chapter 4, defining which tasks are in parallel from a linear event log is difficult. The accuracy of the mining result can depend on the quality of the event log being mined. In terms of the GP most of the errors experienced in the mining of parallel processes occur with processes that have a split and/or join with more than three parallel edges. While the GP approach has shown some promise with complex parallel processes, as evidenced in the experiments section in chapter 4, the structure mined is still not correct. There are also problems in the mining of the semantics for such parallel processes.

In order to aid the mining of complex parallel structures an additional function has been added to the GP approach. In previous sections of this chapter it has been shown how the event log can be used as a heuristic aid for the GP operators and parsing. In a

similar way the initial population of the GP can also be used as a heuristic to improve the mining of complex parallel structures.

Once the initial GP population has been created, the parallel process heuristic code is called (the parallel process heuristic has been devised by the author). This code has the following steps –

- Select 10 individuals from the population.
- Identify which graphs have 4 or more parallel edges.
- Identify which tasks are likely to be in parallel.
- Return parallel task set.

```
GPParallelProcessHeuristic(population){
Select 10 fittest individuals from population
For each individual {
    Identify fanOut tasks with more than four edges
    Identify fanIn tasks with more than four edges
}
For all fanOut  tasks
    If task appears in 4 or more individuals put in FanOutTaskSet
For all fanInTasks
    If task appears in 4 or more individuals put in FanOutTaskSet
Identify parallel tasks from FanOutTaskSet
```

```
Identify parallel tasks To FanInTaskSet

Return parallel task sets

}
```

Figure 39: Parallel Process Heuristic Pseudo Code

As can be seen from the steps above, and in more detail in the pseudo code shown Figure 39, ten individual graphs are selected from the initial population for further analysis (in effect these are the 10 fittest individuals from the initial population of 100, as the individuals are selected after the first fitness parsing). The tasks of the graphs are then examined for edges to multiple tasks. This is completed by identifying fanOut tasks (shown in Figure 39), where an output set of a task has edges to four or more distinct tasks and fanIn tasks (shown in Figure 39), where an input set of a task receives edges from 4 or more tasks. The value of 4 edges was decided from preliminary investigations into the results obtained from a range of parallel processes that varied in complexity. From this analysis the tasks that are likely to be in parallel can be identified. If a task appears in a certain number of graphs as a parallel task then it is included in the parallel task set. When fanIn and fanOut tasks have been identified from the previous analysis stage further analysis can be carried out to define which task edges are in parallel (shown in Figure 39 as Identify parallel tasks from FanOutTaskSet and FanInTaskSet, respectively). This is achieved by looking for common edge nodes in the same task drawn from multiple individuals. If the same edge node from the same task appears in more than 50% of the individuals examined then this task is identified as a parallel task. In this way parallel edge sets emanating from and converging to certain tasks can be identified. This allows the parsing to randomly ignore edges that link between tasks that should be in parallel. In addition the parallel task linking edges are added to a list of edges that may be deleted by the mutation operator. The values used in this heuristic have been established by the author through experimentation.

There is however one limitation of this technique. In some process models there will be a limited number of links between parallel edges that are correct, which may then be ignored by the fitness parsing and removed though the mutation operator. However, if these edges are frequent enough the mutation operator alone will not be able to remove a large enough number of such edges from the population as a whole and the edges would still be reflected in the fittest individual.

5.10. Dealing with Noise in Event Logs

Evolutionary techniques, such as GP, have been used in applications where noise is a factor (Koza, 1992). Banzhaf et al. (1998) cite the work of Pei et al. (1995) as a demonstration of GP's resilience to noise in data. It is partly with this knowledge that GP was explored and selected as a method for performing business process mining. As mentioned in the literature review, noise in event logs it still an active topic in process mining research. Van der Aalst and Weijters (2004) describe noise, in the context of business processes, as logged data that may be incorrect or incomplete providing problems when the data is being mined. The GP approach as detailed in chapter 4 was not at a level of development to justify the mining of noisy data, which adds an additional layer of complexity to any event log. However the modified GP approach, outlined in this chapter, does provide a potential way of mining such noisy event logs. The GP approach is suited to the mining of noisy data due to the following features –

- The use of stochastic operators such as mutation and crossover.
- The use of event log based heuristics.
- The fitness parsing approach.

The GP operators lend themselves to the mining of noisy data in that they can actually create new representations of the data while preserving the most frequently occurring features from the event log. The crossover feature will swap subsets

between two fit individuals, so preserving 'genetic material' with high fitness scores. The mutation operator will act upon subsets introducing and removing subset edges at random. This operator allows for the exploration of new process task linkages which may exist in the event log but perhaps only at low frequencies. The combination of operators potentially gives a good balance in finding high frequency correct process structures and semantics along with some low frequency edges despite the presence of noise. Both operators allow for a global search as they act on individuals which should reflect the event log in its entirety. This is different from local search techniques which are based on the incremental build up of models based on locally available information (van der Aalst and Weijters, 2004). The event log based heuristics, described in earlier sections of this chapter, potentially offer a certain amount of resilience to noise. The event log offers the best guide to the correct sequence of process tasks. Intelligent use of the event log, as a way of aiding the mutation operator and operator repair functions to make positive task selections, potentially offers the best way of avoiding poor fitness task choices (perhaps caused by noise in the event log). Some of the task selections made by the mutation operator will be incorrect due to noise, though the balance of the selections should be correct.

The fitness parsing of the GP approach should also be able to cope with noise in the individuals presented. The marking system used is able to continue when encountering parsing problems, such as missing tasks and missing edges. The use of stochastic operators in the GP approach is inherently noisy anyway. The isTaskEarlier function (section 5.8) is another way of removing a certain type of noise from the individuals. With this function, when a task being linked to should actually be earlier in the task sequence, the edge is recorded for potential removal by parsing and a missing task token is added. The general concept of parsing every event trace against each individual gives a good average score for the individual, assuming that a percentage of the traces are incorrect due to noise. The GP approach has been evaluated in terms of its ability to cope with noise in event logs and the results of the experiments are presented in the validation chapter (chapter 6).

5.11. Initial Experiments with the Modified GP Approach

5.11.1. Experimental Set-up

As a preliminary evaluation of the modified GP process mining approach a number of experiments have been completed. The experiments focused on mining the following aspects –

- Process structures.
- Process semantics.
- Complex parallel process structures (how many tasks are in parallel and the length of the parallel structure).

In terms of the process structures there was a need to check that the ability of the GP approach, as detailed in chapter 4, had not been altered by the modifications (detailed in this chapter). Positive results were achieved from the mining of structures with the unmodified version of the GP approach. In many process mining approaches the achievement of successful mining result in one mining aspect can often degrade the results obtainable for the other aspects, so a balance must be achieved between the various mining goals.

The mining of the process semantics was poor in the original version of the GP approach, and was therefore a priority area for the modified GP approach. The correct identification of the semantics was also an important requirement for the improved modelling of the split and join points within the mined process models.

The mining of parallel structures by the original GP approach was fairly accurate, though there were a few mistakes in terms of additional edges being added to mined models. The semantics of parallel processes were, however, poor with many duplicate

subsets holding multiple edges. The parallel process test of chapter 4 demonstrated the tendency of the mined model to over describe the event log, especially in terms of the semantics. Two sets of test processes, in the form of event logs, were mined. The event logs are artificial as opposed to being real process data, though they do model real life process constructs (Alves de Medeiros, 2006). All of the event logs are balanced (contain a representative amount of traces for each path through a process). The process data sets are shown in Table 15. The data sets are provided by Alves de Medeiros (2006) through van Dongen et al. (2005b). The same experiment parameters, environment and settings as used in chapter 4 (page 140) were used in the completion of these experiments. Each process data set was mined a total of ten times with the best and typical results presented in this section.

Table 15: Process Test Data Sets for Preliminary Experiments

	Sequences	**Semantics**	**Parallelism**
Process data 3	Medium	Medium	High
Process data 5	High	Medium	Low

A number of initial experiments were also carried out in the presence of noise with positive results. A formal evaluation of the GP approaches ability to mine process logs with noise will be presented in chapter 6.

5.11.2. Experimental Results

The structure and semantic views of the mined process is shown in Figure 40 along with a guide to the semantics. From a qualitative (visual) assessment of the resulting mined models, it is clear that both the structure and the semantics for this process model could be mined successfully. It is likely that the removal of edge semantics and the heuristics, outlined earlier in this chapter, contributed to this improvement in mining performance.

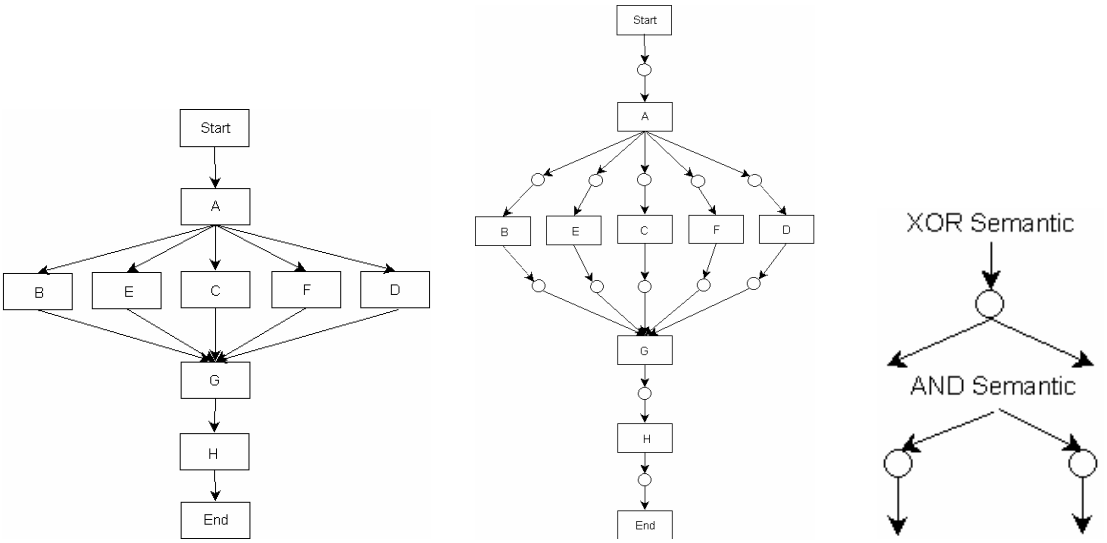
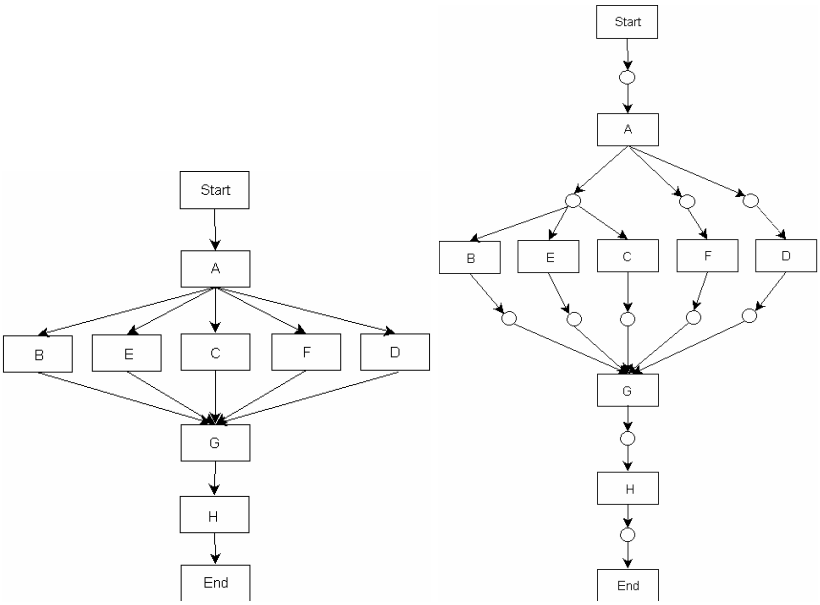


Figure 40: Best GP Result for Process Data 3 With Parallel Code Structure View (Left) Semantics View (Right) (and Semantic Guide)

Figure 41: Typical GP Result for Process Data 3 With Parallel Code Structure View (Left) Semantics View (Right)



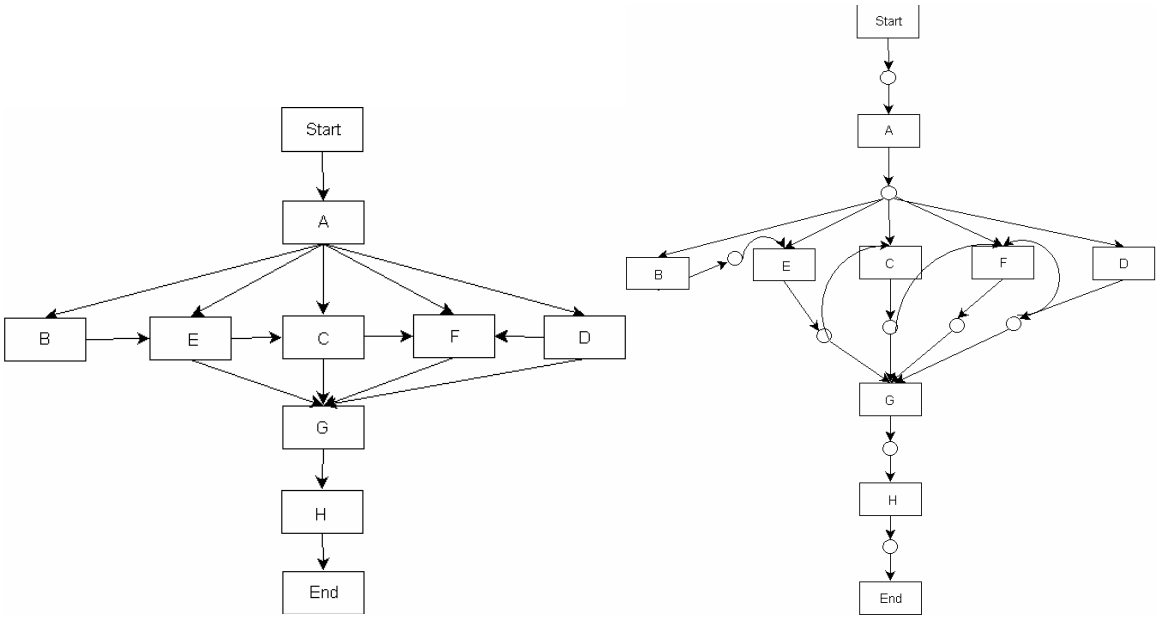


Figure 42: Best GA Result for Process Data 3 Structure View (Left) Semantics View (Right)

A parallel process function was produced (detailed in chapter 5 section 5.9) to aid the mining of such complex parallel processes. The results for the GP approach presented here show the effect of mining without the additional code and with the code. The semantics of this process were all in the form of AND split/joins.

When mining without the parallel process function (detailed in chapter 5 section 5.9) the structure of this process could be mined correctly only once and the semantics could not be mined at all. In the ten tests conducted errors included mistakes in edges between tasks, for example tasks G and H were being linked in the incorrect order by one or two parallel tasks in the process. The errors encountered in the semantics included additional edge nodes in sets turning correct single edge node sets into XOR sets (sets containing multiple edge nodes). Additionally some edges were missing from task edge sets and some present in the sets of incorrect tasks, so creating the incorrect task edge linking observed in the process structure. The ten tests were run again with

the additional parallel code and an improvement in the results was noted. With the extra parallel functionality, the structure could be mined correctly on eight out of ten runs. The semantics improved only slightly, on three runs where the structure could be mined correctly, the semantics could also be resolved (the best mined result for the GP approach is shown in Figure 40 and typical mined result is shown in Figure 41). The GA approach was not able to mine the structure or semantics of this process correctly on any of the ten runs. The errors included additional edges between the five parallel tasks, incorrect semantics XOR construct instead of AND edges from task A and missing edges joining from the parallel tasks to task G (the best mined result for the GA approach is shown in Figure 42).

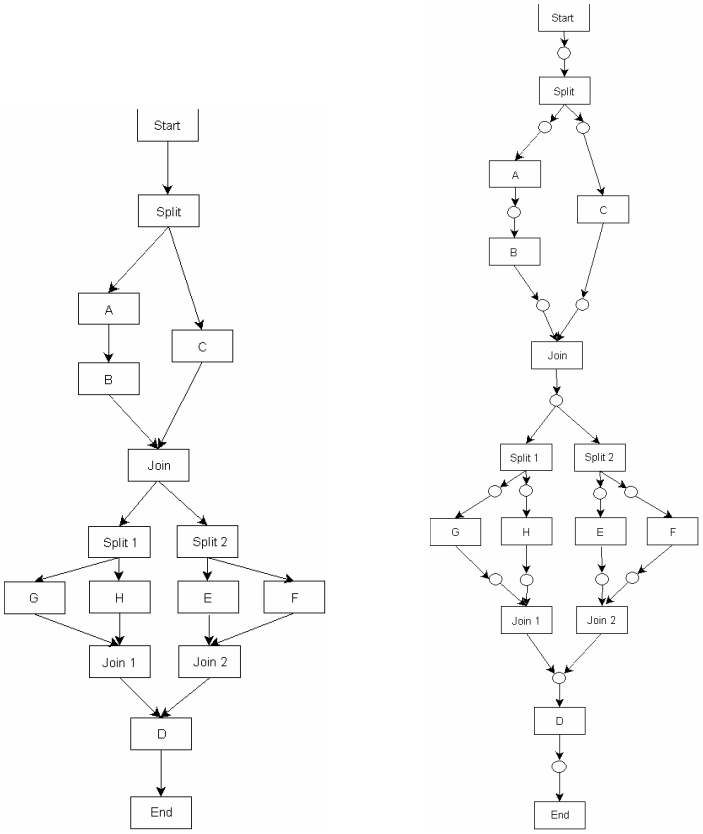


Figure 43: Best GP Result for Process Data 5 Structure View (Left) Semantics View (Right)

Figure 44: Typical GP Result for Process Data 5 Structure View (Left) Semantics View (Right)

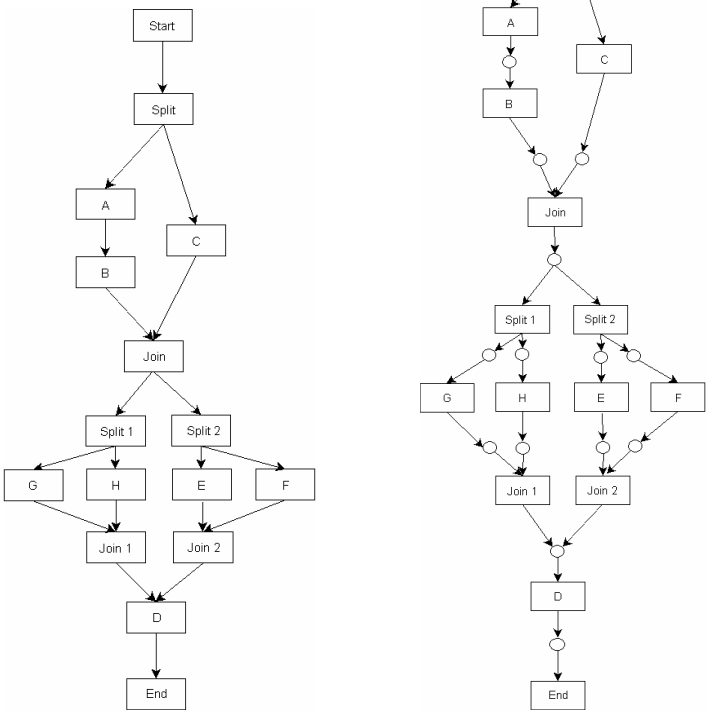
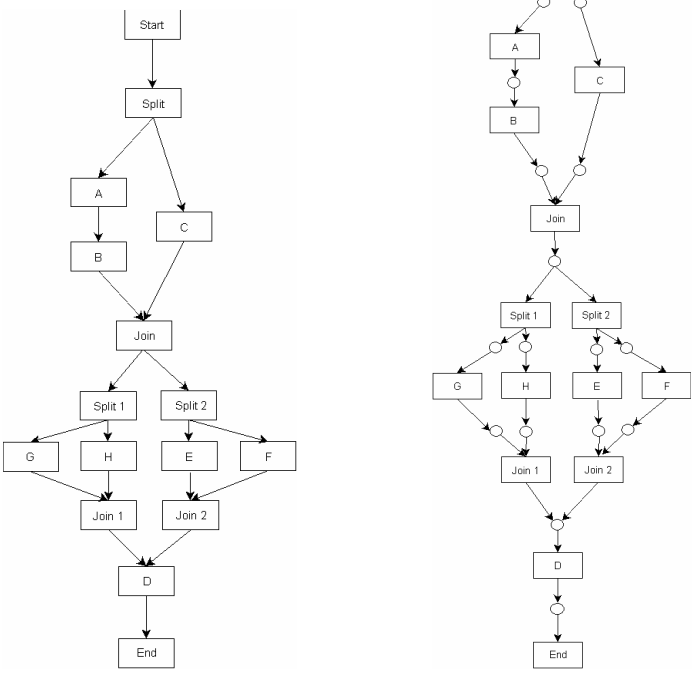


Figure 45: Best GA Result for Process Data 5 Structure View (Left) Semantics View (Right)



Process data 5 could be mined correctly on seven of the ten runs with the GP approach. Errors with this process included an incorrect link between tasks B and D on several occasions, instead of a link from B to the 'Join' task. One type of semantic error was noticed in several of the runs in that a correct XOR semantic appeared twice. One run produced a poor lower structure including three incorrect links between tasks (the best mined result for the GP approach is shown in Figure 43 and typical mined result is shown in Figure 44). The GA approach could mine both the structure and semantics correctly on nine out of ten runs, with one run producing an incorrect linear structure at the top of the process (the best mined result for the GA approach is shown in Figure 45).

Process data 3 was one of the more difficult processes to mine due to its highly parallel nature (with two tasks having a split/join of five edges). Parallel constructs are among the most difficult process constructs to mine correctly. The event log for the process gave 106 distinct traces outlining the level of detail required to successfully mine such features. The extra function for identifying complex parallel structures was necessary for the correct mining of the structure for this process. The structural mistakes on two runs with the extra function included incorrect linkages by passing the parallel join of one task. Errors in the semantics included incorrect creation of XOR edge sets in the semantics of the parallel split/join tasks. Again the fitness function may not be reducing the appearance of additional subsets allowing the process model to over describe the data and permit a greater level of activity than present in the event log. The parallel code helped in reducing the likelihood of parallel edges linking to each other, so allowing the GP approach to concentrate on the correct linking of tasks in the process structure overall. It is likely that the mining of this process benefited from the event log heuristics (detailed in section 5.7) in mining of this process as selection of viable tasks to link to is vital to the identification of viable task edges. In addition the use of the parallel process function detailed in section 5.9 benefited the mining of process data 3, in the correct identification of edges that are parallel. The GA approach failed to mine the structure or the semantics correctly on any of the runs.

The event log for process data 5 also provided a challenge due to its structure. The process could be mined correctly on seven occasions. Extra erroneous links were detected in the incorrectly mined structures along with duplicated AND/XOR edge sets. Again this is an example of over describing, effectively allowing for XOR subset node edges to be duplicated into new subsets for a task (allowing an AND relationship between node edges that is not supported by the event log). This over describing of the event log points towards the possibility of the fitness function not punishing large sets enough (sets containing a large amount of subsets). The GA approach could mine the structure and semantics of this process correctly on nine out of ten occasions.

Although the experimentation featured in this chapter is preliminary, it does point towards the improved mining capability of the modified GP business process mining approach. Further experimentation is required to build a more complete picture of the process mining capabilities of this mining approach.

5.12. Summary

This chapter has explored the GP approach and the various modifications that have been developed in order to refine its ability to mine business processes. Modifications to both the graph based representation and fitness parsing have been put forward and implemented. There has also been an investigation into the use of the event log as an additional heuristic, providing guidance to the mutation function, fitness parsing and operator repair functions. It has also been shown that the initial population of individuals may be used as an aid to the mining of complex parallel processes, by working out which tasks are most likely to be in parallel. Due to the modifications detailed in this chapter it has been possible to use the full implementation of both the crossover and mutation operators. This has contributed to the improved mining capability (in terms of the mining accuracy and complexity of process that may be mined) provided by the modified GP approach in comparison to the un-modified GP approach (as detailed in chapter 4).

The experimentation in this chapter is of course preliminary, though it does point towards an improvement of the results obtained by the GP approach in chapter 4. The following chapter will formally validate the GP approach as put forward in this chapter; utilising a range of test data, both artificial and obtained from live processes (provided by the industrial sponsor of this research). A wider variety of process constructs will be mined by the GP approach as a test of its flexibility. A justification has been put forward in this chapter that the GP approach is capable of mining event logs that contain noise. The use of heuristics, outlined in this chapter, is a development which should help in the selection of beneficial tasks by the mutation and crossover operators. The use of GP in itself will also aid the mining of problem event logs (containing noise), due to the stochastic search capability presented by the approach. The ability to mine noisy process data will be formally tested in the following chapter.

6. Validation of the Proposed Business Process Mining Approach

Over the last two chapters, the specification and development of the proposed Genetic Programming (GP) process mining approach have been analysed. The initial GP approach has been outlined along with its achievements and limitations. Further development of the GP approach into a technique that can mine structures and semantics has been put forward. The contribution towards mining noisy event logs has also been explained along with the potential to mine the structures of complex parallel tasks. In order to evaluate the approach in detail, a series of experiments are introduced as the validation of the GP process mining approach.

6.1. Introduction

The purpose of the experiments is to test the ability of the approach to mine a wide range of processes, containing many common and more complex constructs found in the business processes implemented by organisations. In order to do this, ten test data sets, in the form of process event logs, are used to ascertain the strengths and limitations of the GP approach.

In total 8 of the tests are artificially created to test the mining of a range of process constructs. These tests are drawn from the work of (Alves de Medeiros, 2006) and (Herbst, 2001). The tests were selected from a number of available tests as they cover a range of process constructs and do not contain constructs that cannot be mined by the GP approach (such as mining loops and duplicate tasks). In addition two actual business processes are given, provided by the industry sponsor of this thesis. Each of the ten process data sets have been mined ten times to give reliable feedback on the performance of the approach. This chapter will present the best result achieved for each process data set (when the mined process is compared to its template, in the case of the artificially created data sets) and the typical result (in this context the term typical refers to the result that was reliably achieved on the greatest number of

occasions). In addition three of the data sets have been filtered to allow different levels of artificially created noise. These sets are then mined multiple times to ascertain the resilience of the GP mining approach to mine different process constructs in the presence of noise. None of the process data sets using tin this experimentation have been cleansed or filtered in anyway before mining.

The GP process mining results are presented in this chapter for all ten processes with graphs that detail the mined structure of each process and the semantics that have been mined for that process structure.

6.2. Validation Methodology

The validation results are presented in two parts-

- The results for balanced and complete event logs.
- The results for event logs containing different levels of noise (that is event logs containing missing tasks and tasks recorded in the wrong order of precedence).

The event logs used for the first eight test data sets are all balanced and complete which means that they contain an equal number of transitions though a process detailing the most common paths through that process (a path through a process is shown in Figure 21 in chapter 4).

Table 16: Process Data Sets to be Mined

Data Set	**Sequences**	**Semantics**	**Parallelism**
(1) A12 (Alves de Medeiros, 2006)	Medium	Medium	Low
(2) A8 (see chapter 4) (Alves de Medeiros, 2006)	Low	Low	Low
(7) Bn1 (Alves de Medeiros, 2006)	High	Low	Low

(4) 6p45 (Herbst, 2001)	Low	Low	Low
(5) 6p41 (see chapter 5) (Herbst, 2001)	High	Medium	Low
(6) Choice (Alves de Medeiros, 2006)	High	High	Low
(3) Parallel 5 (see chapters 4 & 5) (Alves de Medeiros, 2006)	Medium	Medium	High
(8) A7 (Alves de Medeiros, 2006)	Medium	High	High
(9) BT Data	High	Medium	Medium
(10) BT Data(2)	High	Medium	Medium

The ten process data sets that have been used in this research are detailed in Table 16 (please note that process data 2 is detailed in chapter 4, process data 5 in chapter 5 and process data 3 is featured in chapters 4 and 5, rather than in this chapter). The first eight data sets have been artificially created for the purpose of testing process mining algorithms by Herbst (2001) for data sets 4 and 5, and by Alves de Medeiros (2006) for the remainder of the sets. The tests are distributed with the ProM software (van Dongen et al., 2005b). The final two event logs are case study data sets from BT (the industry sponsor for this research) and are drawn from real life processes operating within the organisation. The event logs have been selected to test a wide range of process constructs. In addition data sets 2, 3, and 6 have also been mined in the presence of noise.

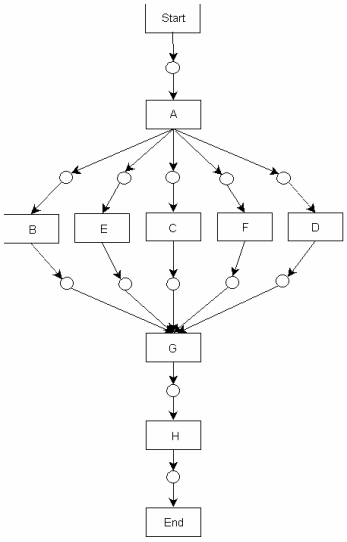
In general three broad areas are being tested in terms of process features: the process mining approach's ability to mine sequences, semantics (the term semantics is used in relation to edge nodes and refers to the type of relationship an edge is in, which may be AND or XOR (Alves de Medeiros, 2006)) and parallel structures. Working out the general order and precedence of tasks in a process is critical in process mining activities. The sequence of tasks in a process may become complex for many reasons, such as a long chain of tasks or precedence between different tasks. A complex sequence will always offer more of a challenge for a process mining approach. A specific form of complex sequence is provided by parallel task constructs.

Tasks in parallel (capable of executing simultaneously) pose a significant challenge as it is likely that not all of the possible routes through a process are present making such structures difficult to ascertain (van der Aalst and Weijters, 2004).

Figure 46 gives an example of a trace from a process log in the XML format that is used by the GP process mining approach (read in by the ProM software (van Dongen et al., 2005b) for the GP approach). In addition Figure 47 shows the process that is, in part, described by the trace in Figure 46. The bottom of Figure 47 shows a number of valid paths that may be taken through the process graph (which may be present as trace in the event log for the process).

```
<ProcessInstance id="0"
description=""><AuditTrailEntry>

<WorkflowModelElement>Start</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>A</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>B</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>E</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>C</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>F</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>D</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>G</WorkflowModelElement>
</AuditTrailEntry>AuditTrailEntry>

<WorkflowModelElement>H</WorkflowModelElement>
</AuditTrailEntry><AuditTrailEntry>

<WorkflowModelElement>End</WorkflowModelElement>
</AuditTrailEntry></ProcessInstance>
```

Figure 46: An Example of the MXML Event Log Format (showing a trace from the Process Model in Figure 47)



Start - A - B - E - C - F - D - G - H - End

Start - A - D - B - E - C - F - G - H - End

Start - A - E - C - F - D - B - G - H - End

Start - A - C - F - D - B - E - G - H - End

Start - A - F - D - B - E - C - G - H - End

……..

Figure 47: A Process Graph and a Number of Valid Traces Through It

The remainder of this section details the processes to be mined in the validation of the GP process mining approach. For each data set the correct process template is shown (with the exception of process data 9 and 10 as no templates for these data sets exist).

Process Data 1

This process offers a medium combination of sequences and semantics. As such it offers a reasonable test for the process mining approach. The correct template for this process is shown in Figure 48 in the structure form and in Figure 49 in the semantic form. A particular challenge with this process structure is the mining of the AND split and join constructs on the left of the process (split at task B and join at task J) combined with the opposite XOR split/join on the right (split at task C and join at task K).

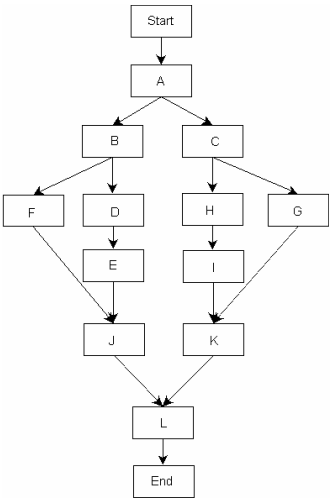


Figure 48: Process Data 1 Structure View

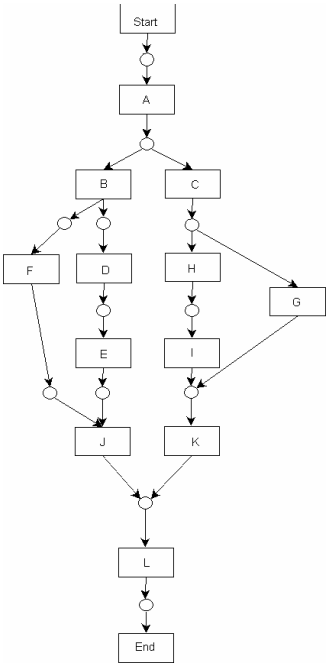


Figure 49: Process Data 1 Semantic View

Process Data 7

Processes that exhibit long chains of tasks are represented in this particular data set. The process consists of a simple split/join followed by a chain of 17 tasks, another

simple spit/join and then a series of 8 tasks. The semantics are, however, easier to mine being simple XOR constructs (the correct templates for process data 7 are shown in Figure 50).

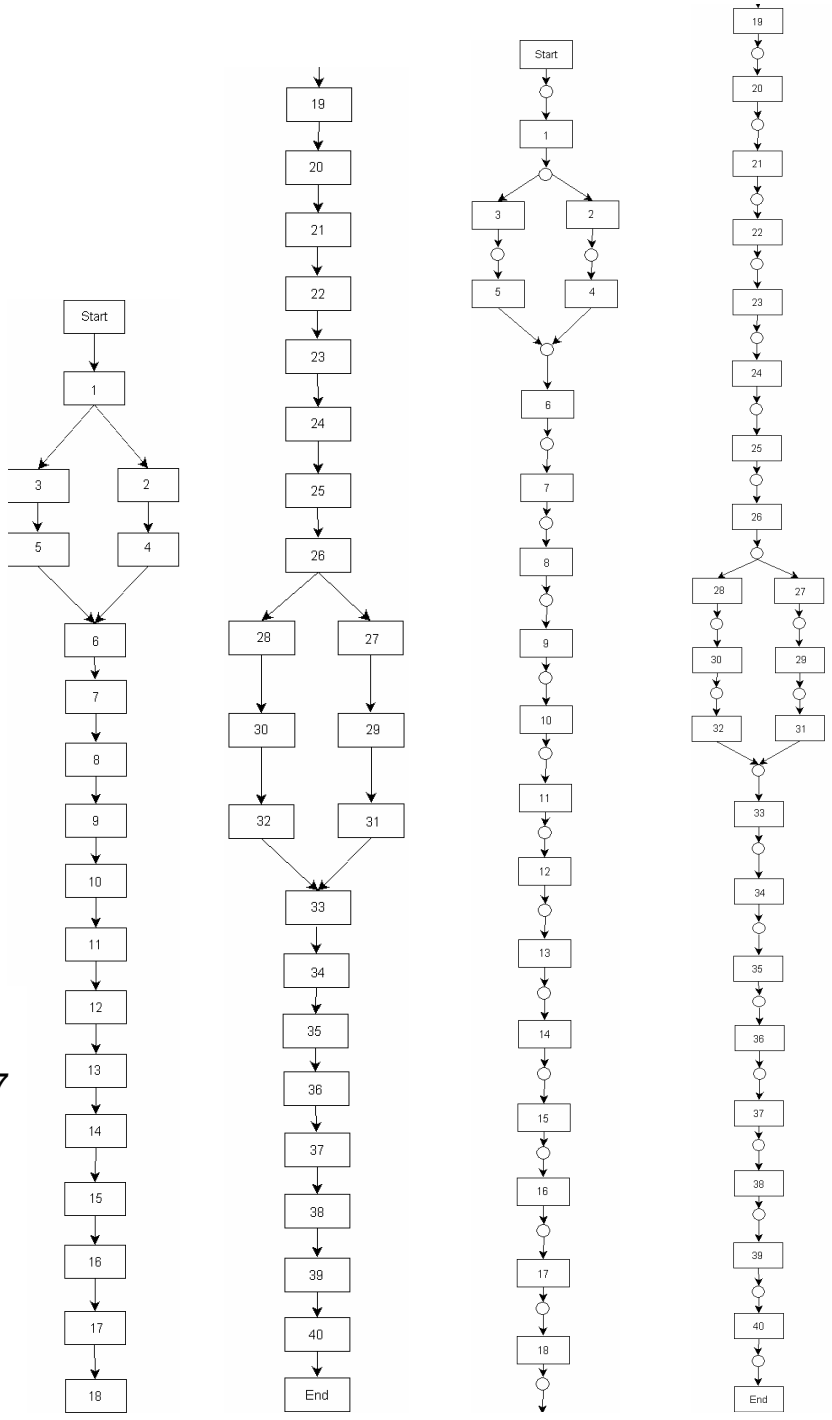


Figure 50: Process Data 7 Structure View (Left) and Semantics View (Right)

Process Data 4

The event log here provides a test of the GP approaches' ability to mine parallel constructs. As with the other data sets presented in this chapter; the process traces for this test data set have all been appended with artificial Start and End tasks to aid mining. The correct templates for process data 4 are shown in Figure 51.

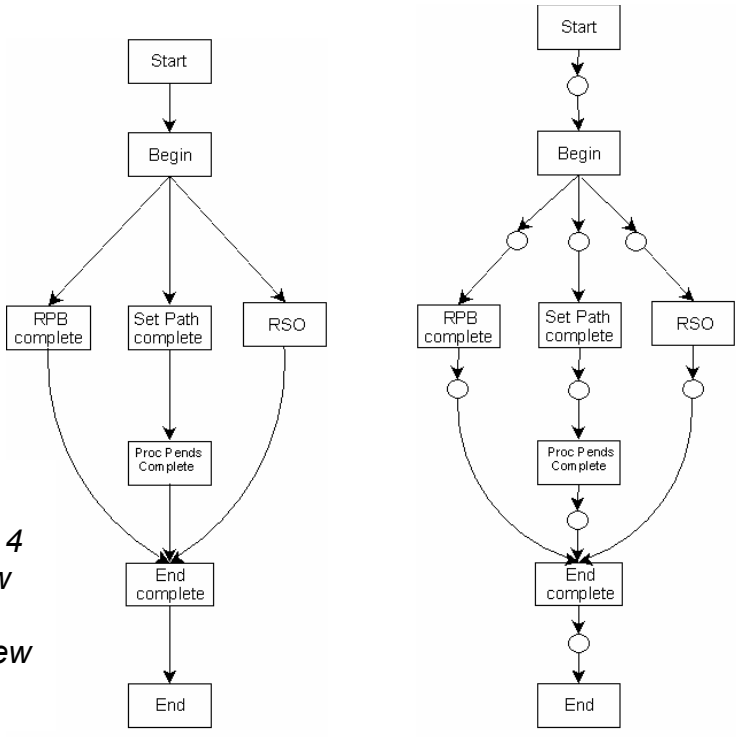


Figure 51: Process Data 4 Structure View (Left) and Semantics View (Right)

Process Data 5

This process contains one of the more challenging task sequences used in this experiment set, containing a longer sequence of tasks compared to process data 4, and a parallel construct. The correct templates for process data 5 are shown in Figure 52. The semantics include several split/join points within the process; along with the arrangement of tasks in the lower section. This process provides further challenge for the GP approach.

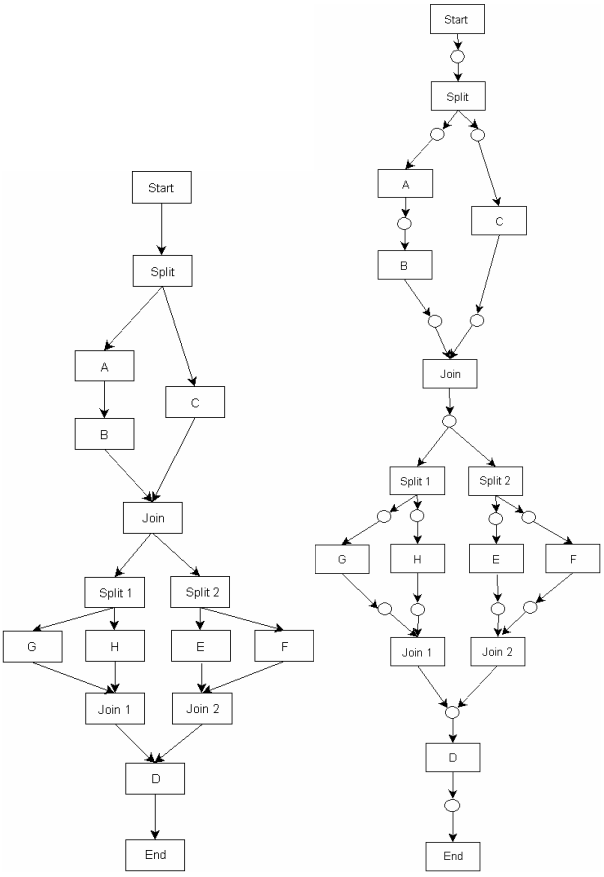


Figure 52: Process Data 5 Structure View (Left) and Semantics View (Right)

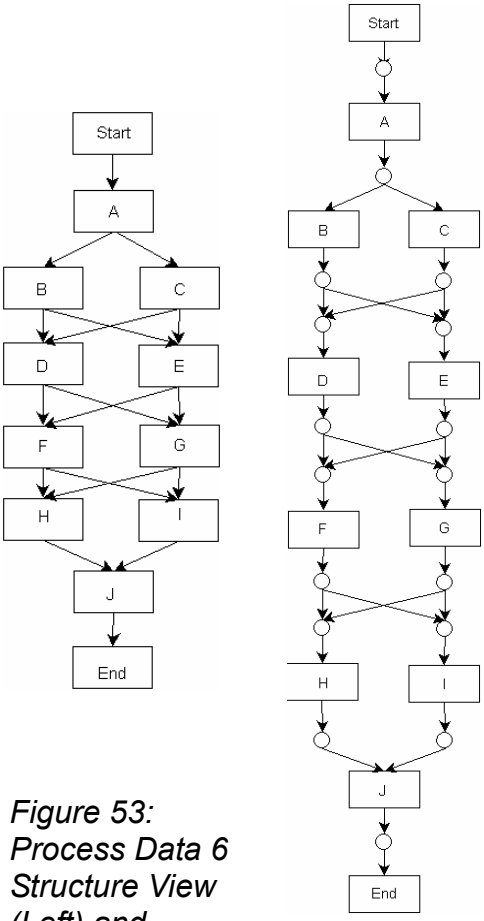


Figure 53: Process Data 6 Structure View (Left) and Semantics View

Process Data 6

The structure of process data 6 makes this a challenging process to mine (the correct templates for this process are shown in Figure 53). The simple parallel appearance is complicated by the splits and joins between the two process task chains.

Process Data 8

This data set provides another parallel structure that from a structure point of view looks identical to that of process data 7. However, the semantics are a mix of AND and XOR split/joins. This is the most difficult event log to mine compared to the presented data sets (the correct templates for this process are shown in Figure 54).

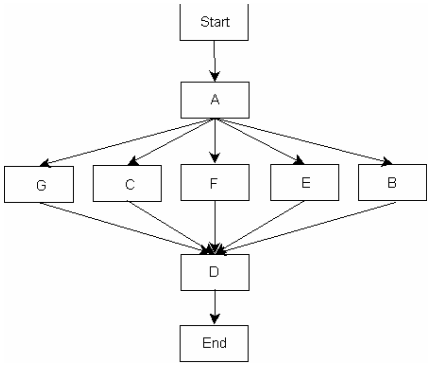
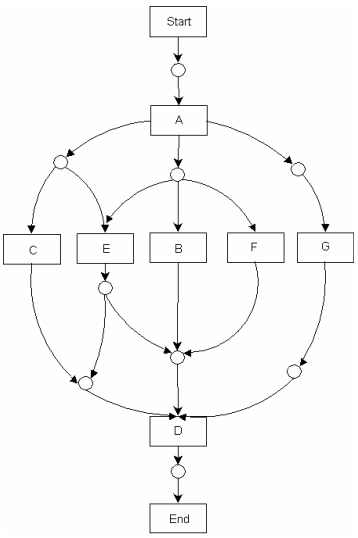


Figure 54: Process Data 8 Structure View (Left) and Semantics View (Right)



Process Data 9 and 10

The data sets 9 and 10 comprise live processes taken from the industrial sponsor of this project (British Telecom (BT)). The event log (process data 9) exhibits a long chain of tasks expanding into a parallel section containing over 20 tasks and involving complex split/join semantics. This set of data contains around 38 tasks in total. The second set of BT Data (process data 10) contains around 40 tasks in total (the resulting mined process diagrams are shown in section 6.4). The processes describe the generic administrative steps a telephone engineer must initiate after attending to a fault for a customer. Process data 9 describes an unsuccessful resolution of a fault and process data 10 describes a successful attempt to rectify a problem. A sample of the event for process data 9 is shown in Figure 55.

<ProcessInstanceid="309124"><Data></Data>
<AuditTrailEntry><WorkflowModelElement>SSSTTTAAARRRTTT</WorkflowModelElement><EventType>complete
</EventType><Timestamp>2007-10-03T07:01:19.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>start</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-03T07:01:19.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>CreateRequired</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-03T07:01:19.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>CreateRequired</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-03T07:01:20.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>CreateRequested</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-03T07:01:20.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>CreateRequested</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-03T07:01:50.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>CreatePending</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-03T07:01:50.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>CreatePending</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-03T07:02:54.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>Open</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-03T07:02:54.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>Open</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-05T07:45:04.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>PPONR</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-05T07:45:04.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>PPONR</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-05T09:46:21.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>Open</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-05T09:46:21.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>Open</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-06T10:02:36.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>PPONR</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-06T10:02:36.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>PPONR</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-06T14:03:07.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>ClearRequested</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-06T14:03:07.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>ClearRequested</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-09T14:04:03.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>Closed</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-09T14:04:03.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>Closed</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-09T14:04:03.000</Timestamp></AuditTrailEntry><Data></Data>
<AuditTrailEntry><WorkflowModelElement>end</WorkflowModelElement><EventType>start</EventType>
<Timestamp>2007-10-09T14:04:03.000</Timestamp></AuditTrailEntry>
<AuditTrailEntry><WorkflowModelElement>end</WorkflowModelElement><EventType>complete</EventType>
<Timestamp>2007-10-09T14:04:03.00</Timestamp></AuditTrailEntry></ProcessInstance>

Figure 55: A Single Trace for Process Data 9

6.3. Experimental Set-up

The experiments detailed in this section have all been carried out using the ProM process mining software suite (van Dongen et al., 2005b) using the proposed GP process mining approach and, for comparison purposes, the GA approach of Alves de Medeiros (2006). This environment provides a facility for adding different levels of noise to a pre existing balanced event log and XML file handling capabilities while also hosing the GP process mining code. The ProM tool does not perform process mining itself but acts as a host for plug-in mining techniques. The output from the GP process

mining approach is available in two forms. The first is a structural view of a process, showing which tasks in a process have a link between them. This is provided in graphical form via a JGraph (JGraph, 2009) diagram. The second output details individual edges between tasks and their semantics (if they are in an AND or an XOR relationship). This is provided in a textual format, which for the purposes of clarity has been manually translated into a Petri net graph. The following parameters have been used with the GP approach –

Population size 100
Maximum generations 200
Elitism rate 0.02
Crossover rate 0.8
Mutation rate 0.2
κ (extra behaviour fitness reduction) 0.025

For the GA approach the following parameters were used as suggested by Alves de Medeiros (2006) –

Population size 100
Maximum generations 1000
Elitism rate 0.02
Crossover rate 0.8
Mutation rate 0.2
κ (extra behaviour fitness reduction) 0.025

From initial experimentation, it was found that the above parameters provided the best and most robust results (against small variations in the data sets) with the GP approach. Parameters of particular note are the crossover rate, which is set to crossover 80% of individuals in the population per generation, and the extra behaviour fitness reduction κ. The κ measure reduces the fitness of individuals based on the amount of extra links found between tasks that are not recorded in the event log (individuals that are over described). Initial experimentation has established the optimum population as 100 (agreeing with the work of Alves de Medeiros (2006)).

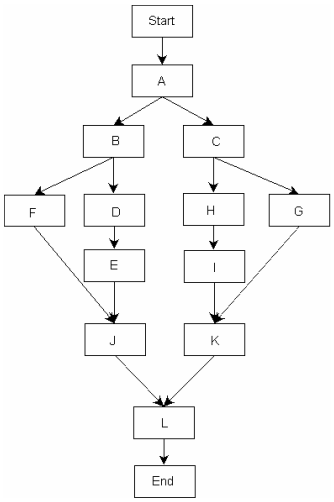
However, the maximum generations limit has been reduced from 1000 as recommended in the work of Alves de Medeiros (2006), to just 200. It is likely that the use of the event log heuristics detailed in chapter 6 has allowed for the reduction. Additional experimentation was used to determine the 200 generation value; experimentation that also established that, for the 10 processes mined in this chapter, there is was further improvement in the results obtained when mining to 500 and 1000 generations. Elitism copies over the best two individuals into the next generation without alteration. The experiments have been carried out on a Windows XP PC.

6.4. The Results: Mining Noise Free Event Logs

The process mining logs detailed in the previous section (and summarised in Table 16) have been mined in their given form. The following pages detail the findings from this set of experiments. The mining results in this research have been obtained without the use of a template (showing the correct structure and semantics for a process).

Process Data 1

This data set provides a number of real life process constructs. For the GP approach both the structure and the semantics for this process could be mined without fault on all ten occasions (the best mined result for the GP approach is shown in Figure 56 and the typical mined result in Figure 57).



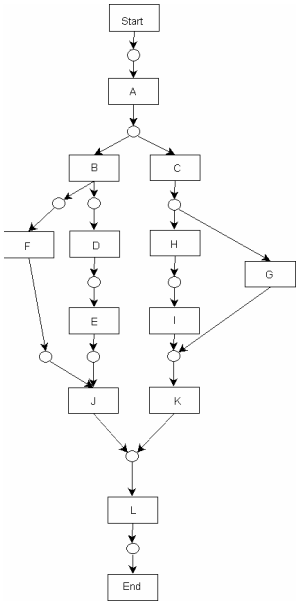
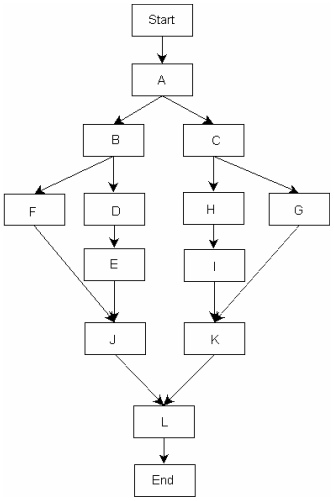


Figure 56: Best GP Result for Process Data 1 Structure View (Left) and Semantics View (Right)



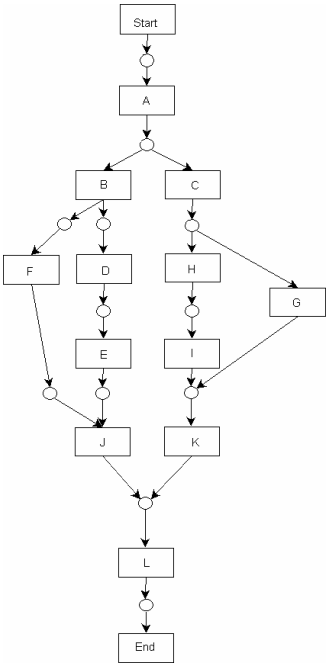


Figure 57: Typical GP Result for Process Data 1 Structure View (Left) and Semantics View (Right)

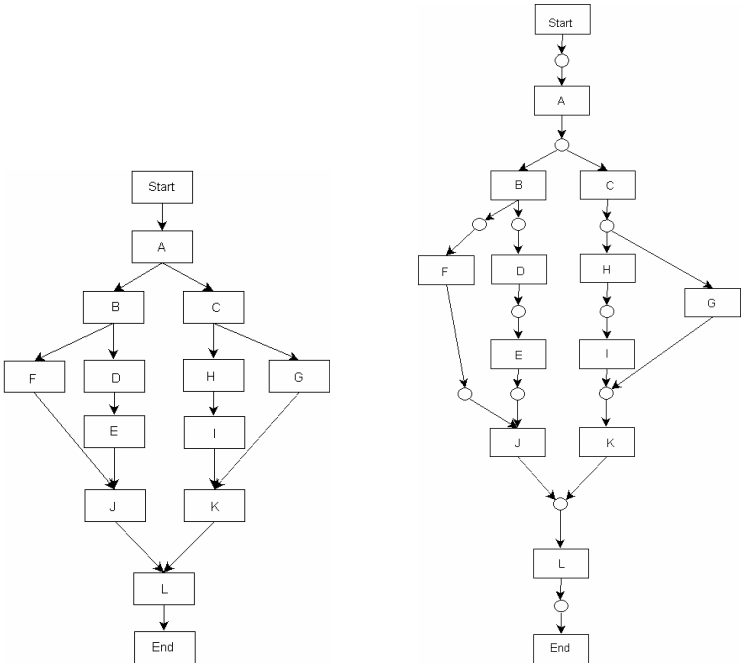


Figure 58: Best GA Result for Process Data 1 Structure View (Left) and Semantics View (Right)

The structure and semantics of this process could also be rendered correctly by the GA approach (the best mined result for the GA approach is shown in Figure 58).

Process Data 7

This process contains 40 tasks, the highest number of tasks in any of the artificially created data sets presented here. With the GP approach the process structure could be mined correctly on all ten runs along with the identification of the appropriate semantics (the best mined result for the GP approach is shown in Figure 59 and typical mined result is shown in Figure 60). The GA approach could also mine both structure and semantics correctly for all 10 runs (the best mined result for the GA approach is shown in Figure 61).

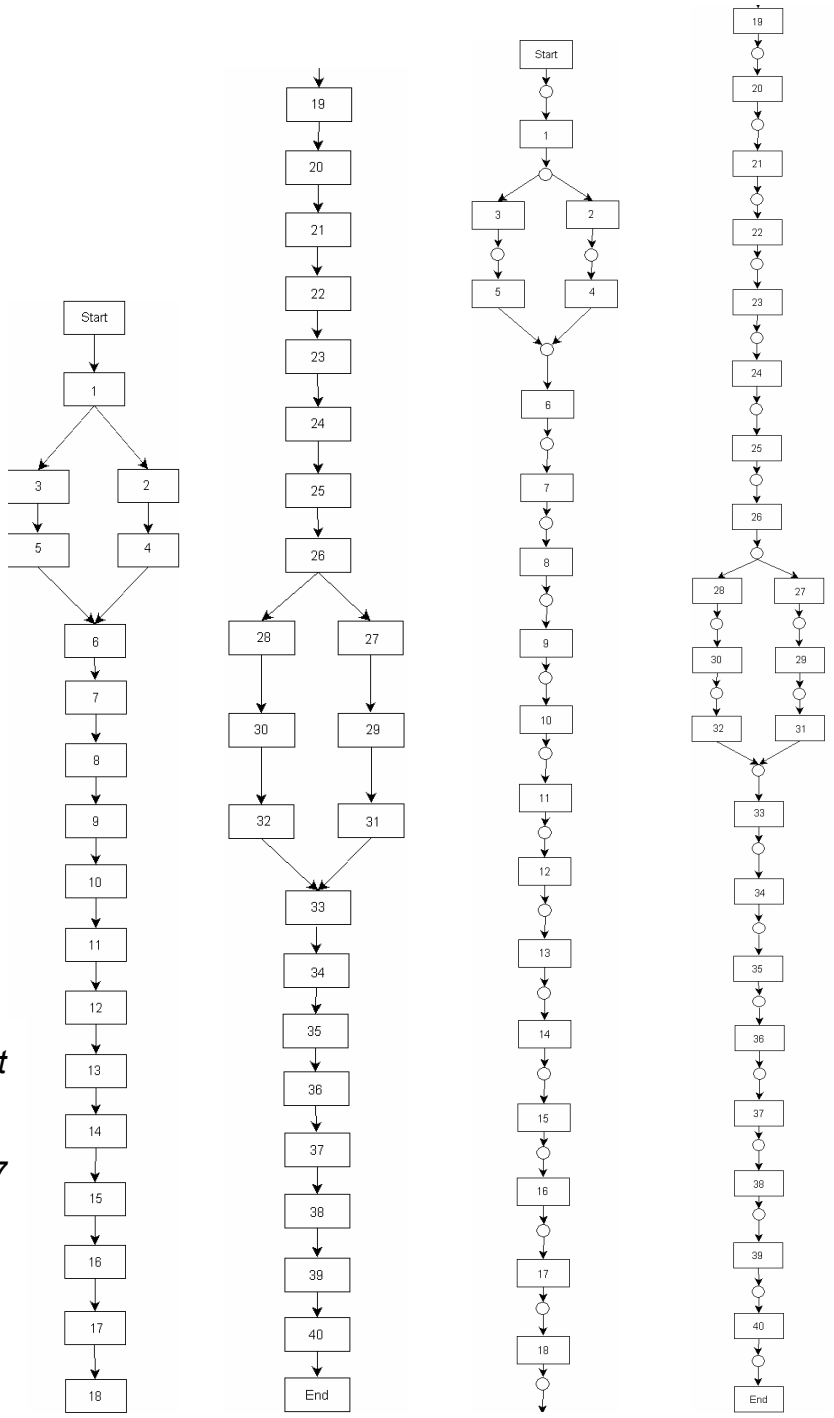


Figure 59: Best GP Result for Process Data 7 Structure View (Left) and Semantics View (Right)

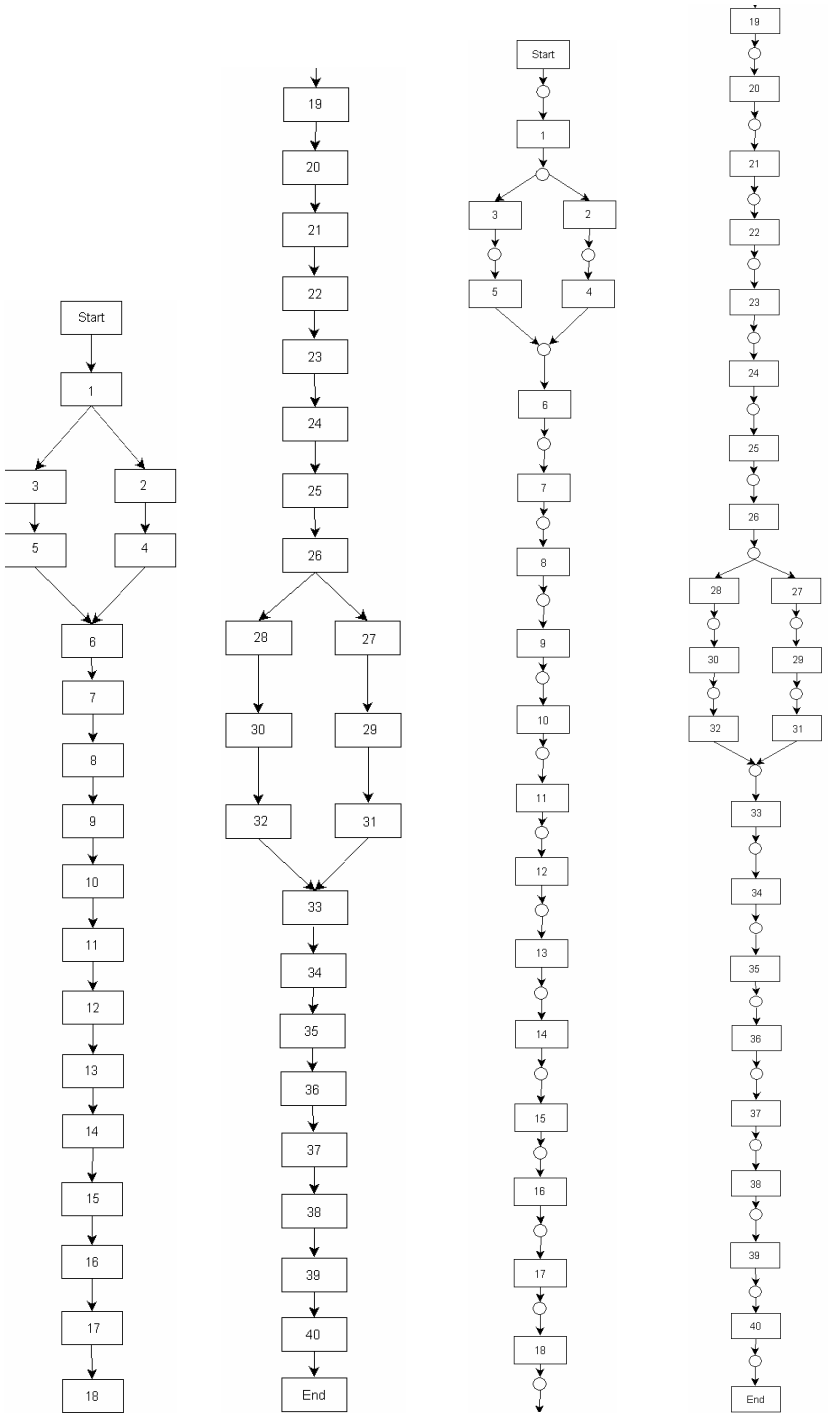


Figure 60: Typical GP Result for Process Data 7 Structure View (Left) and Semantics View (Right)

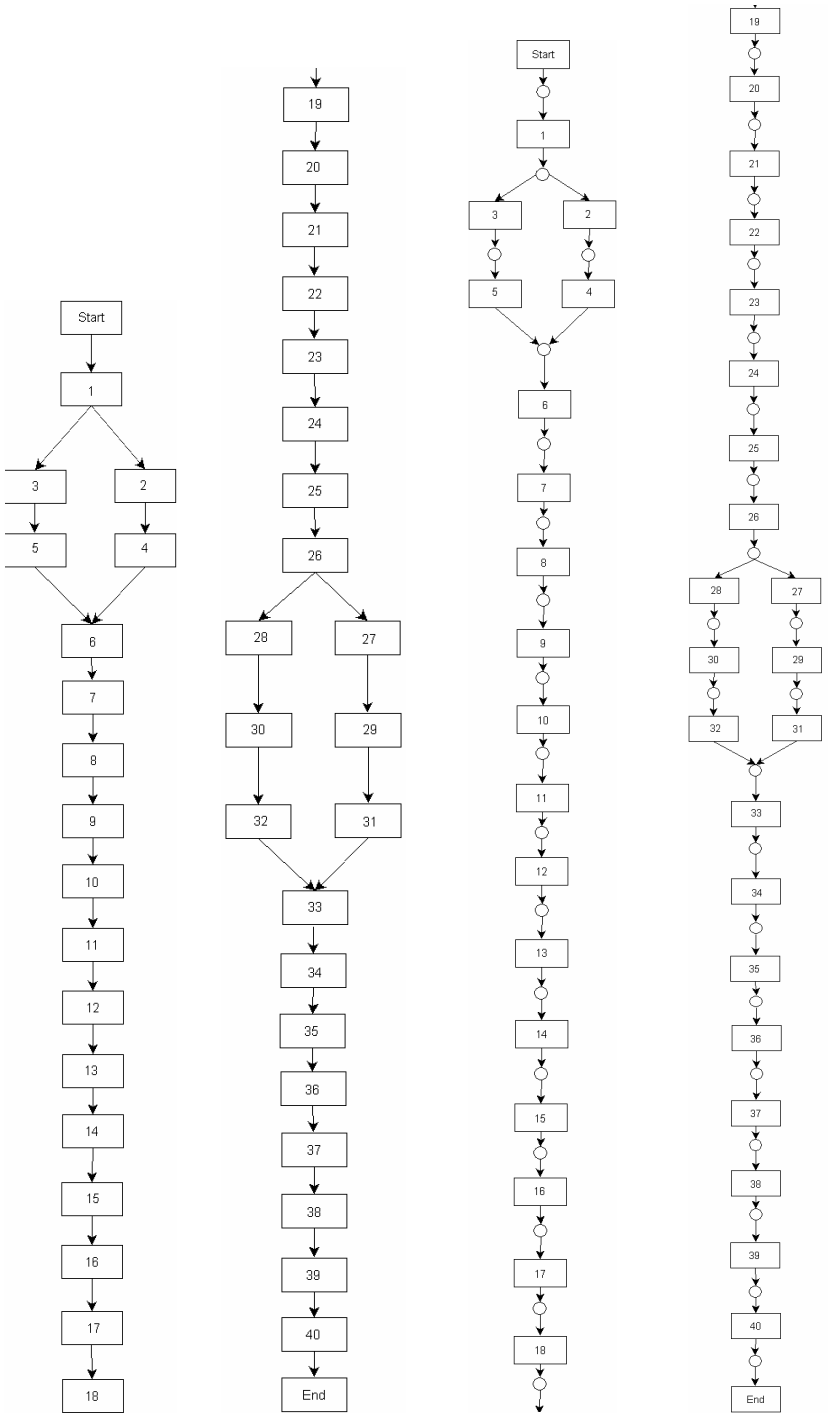


Figure 61: Best GA Result for Process Data 7 Structure View (Left) and Semantics View (Right)

Process Data 4

The results for the GP approach show that on ten invocations the structure could be mined correctly. However on four occasions the semantics displayed one or two errors (the best mined result for the GP is shown in Figure 62 and typical mined result is shown in Figure 63). The GA approach could mine both structure and semantics correctly for all 10 runs (the best mined result for the GA approach is shown in Figure 64).

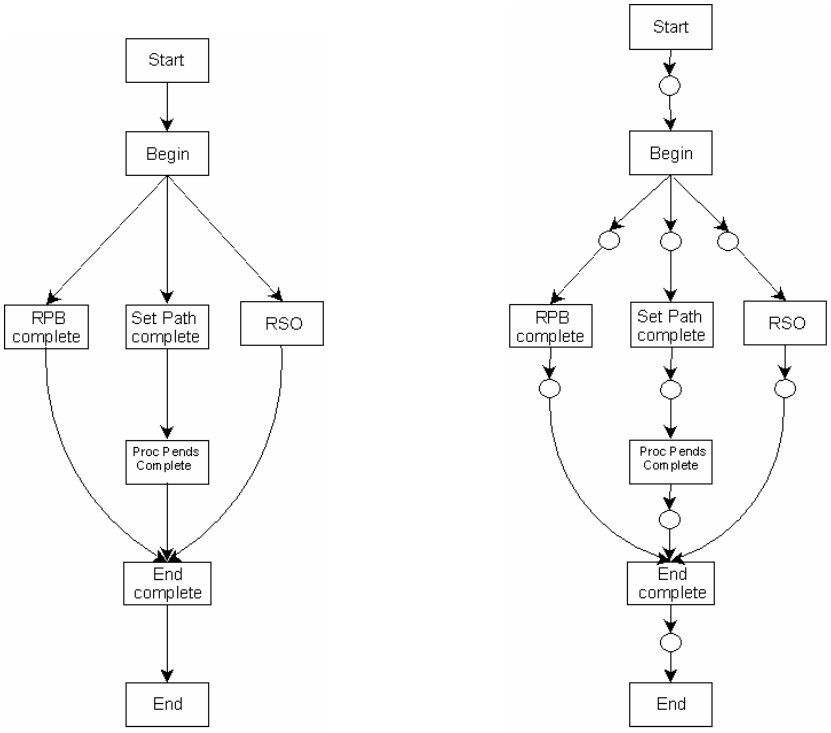


Figure 62: Best GP Result for Process Data 4 Structure View (Left) and Semantics View (Right)

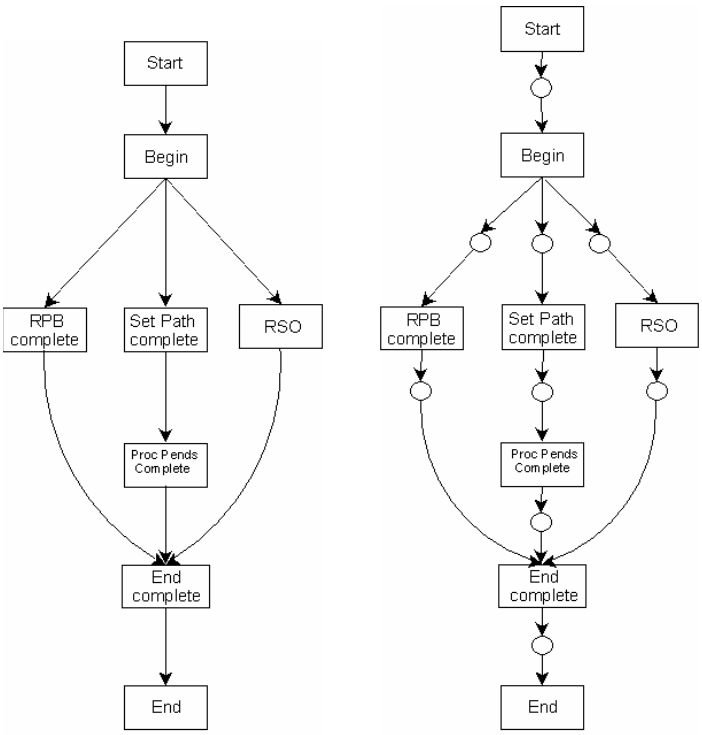


Figure 63: Typical GP Result for Process Data 4 Structure View (Left) and Semantics View (Right)

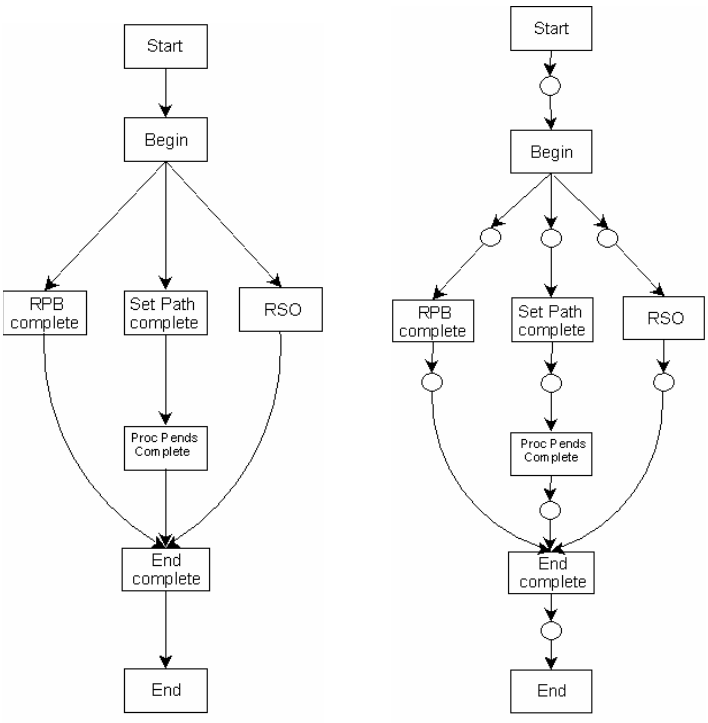


Figure 64: Best GA Result for Process Data 4 Structure View (Left) and Semantics View (Right)

Process Data 6

The structure could be successfully mined on seven out of ten occasions using the GP approach, along with the correct rendering of the semantics (the best mined result for the GP approach is shown in Figure 65 and typical mined result is shown in Figure 66). The GA approach mined this process correctly eight out of ten times (for both structure and semantics, the best mined result for the GA approach is shown in Figure 67).

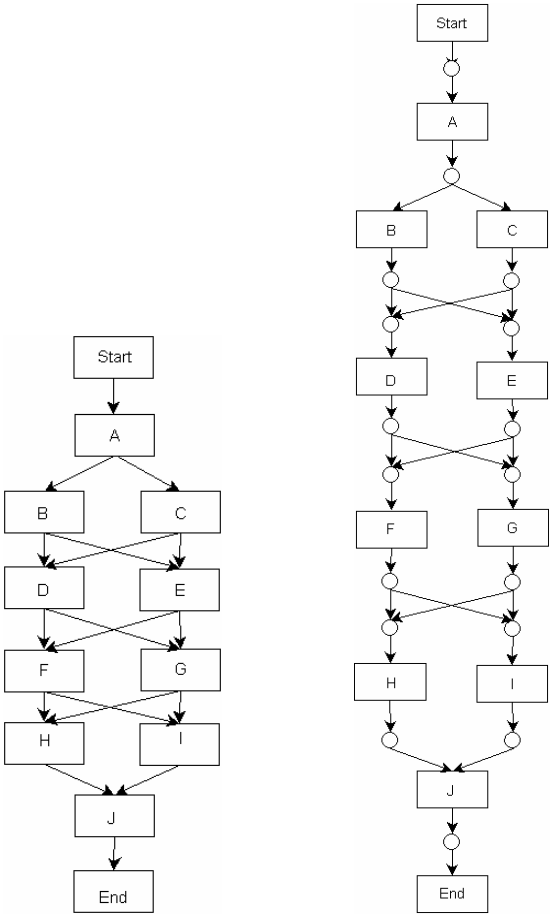


Figure 65: Process Data 6 Structure View (Left) Semantics View (Right)

Figure 66: Typical GP Result for Process Data 6 Structure View (Left) Semantics View (Right)

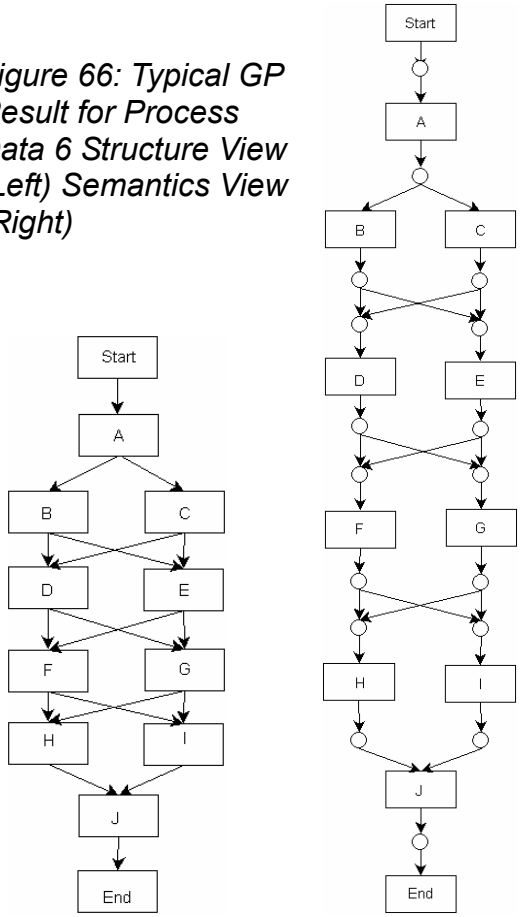
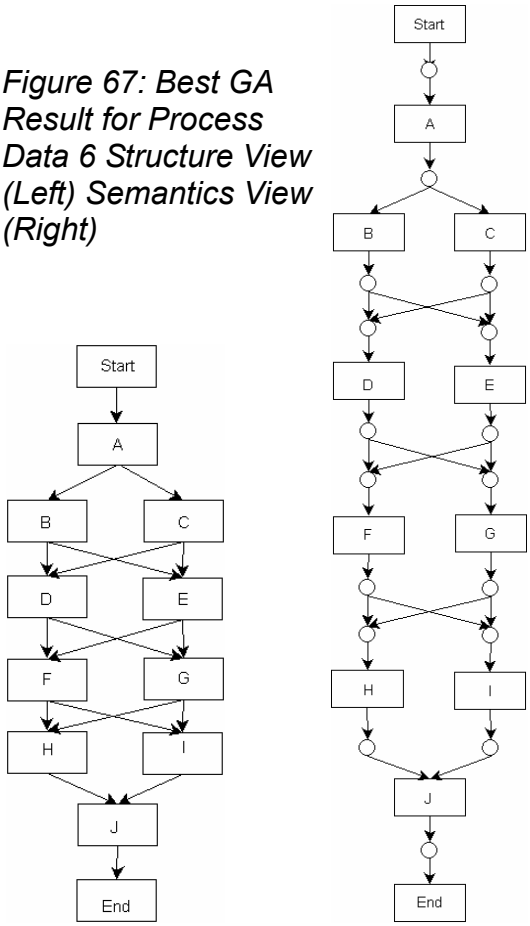


Figure 67: Best GA Result for Process Data 6 Structure View (Left) Semantics View (Right)



Process Data 8

This process has a similar structure to process data 7, though the semantics are different. The first ten runs with the GP approach for this process data set were carried out without the additional parallel code. The results showed that the structure could be mined correctly on three occasions; the semantics were incorrect for all ten runs.

The next ten runs (using the additional parallel code) for the data set demonstrated an improvement in the mining of the structure with seven out of ten runs producing correct representations of the process (the best mined result for the GP approach is shown in Figure 68 and typical mined result is shown in the same figure). The semantics view is not presented here as it could not be mined correctly in any of the runs. The GA approach could not mine the structure or the semantics correctly for this

process on any of the ten runs (the best mined result for the GA approach is shown in Figure 69). The semantic view mined by the GA is not presented here as it could not be mined correctly in any of the runs.

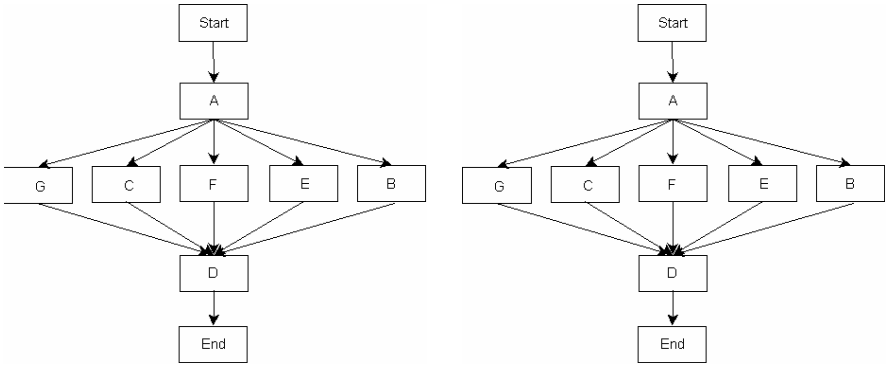


Figure 68: Best GP Result for Process Data 8 Structure View (Left) Typical GP Result Structure View (Right)

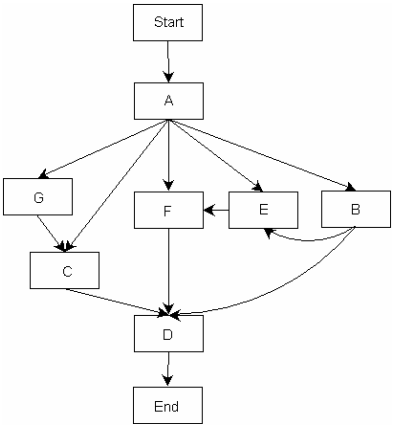


Figure 69: Best GA Result for Process Data 8 Structure View

Process Data 9

Process data 9 and 10 are provided by the industrial sponsor of this project BT. Both processes describe the generic administrative steps a telephone engineer must initiate after attending to a fault for a customer. Process data 9 describes an unsuccessful resolution of a fault. This process is comprised of 38 tasks and includes a parallel structure and complex task sequences. Ten invocations were undertaken on this data set with the GP approach and a typical result is displayed in Figure 70.

In Figure 70 the linear sequence of tasks refers to the opening of an engineering report. The mid section of the process on the left details the steps taken to resolve an issue though no booking of a follow on appointment is required. The mid section of the

process on the right details the closing of the engineering report without recording the steps taken. The GA could not mine this process at all and did not return a result.

Process Data 10

Process data 10 describes a successful resolution of a fault by a telephone engineer. Like process data 9, ten invocations were undertaken with the GP approach on this data set and the typical result is displayed in Figure 71. This process is comprised of 40 tasks and includes a parallel structure and complex task sequences. The structure could be mined in a logical fashion for the majority of the runs with only two runs producing spaghetti structures. In Figure 71 the linear sequence of tasks refers to the opening of an engineering report. The mid section of the process on the left details the steps taken to resolve an issue and possible booking of a follow on appointment. The mid section of the process on the right details the closing of the engineering report without recording steps taken. The GA could not mine this process at all and did not return a result. Please see section 6.5 for a detailed discussion of the results presented in this section.

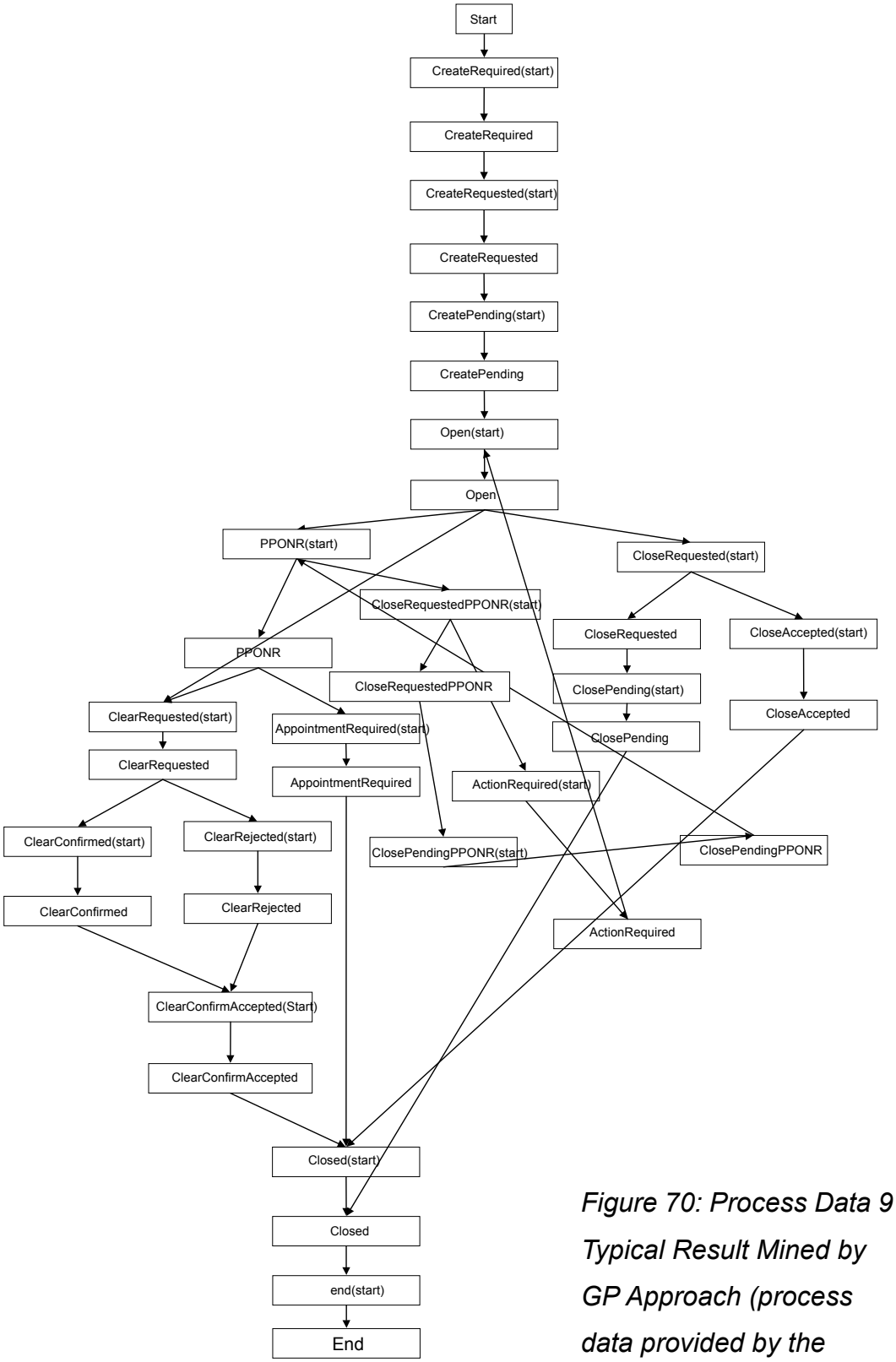


Figure 70: Process Data 9 Typical Result Mined by GP Approach (process data provided by the Industrial Sponsor)

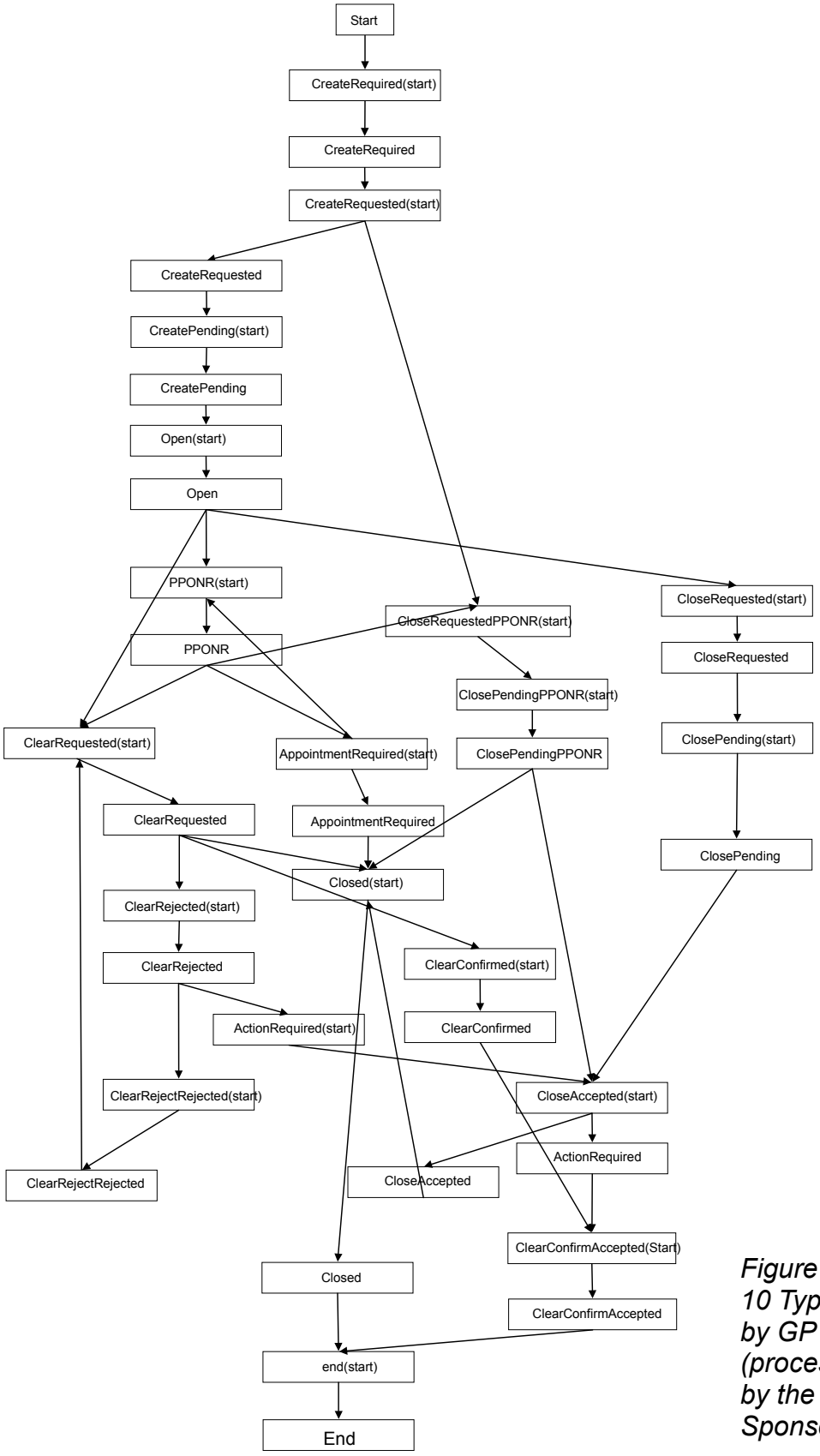


Figure 71: Process Data 10 Typical Result Mined by GP Approach (process data provided by the Industrial Sponsor)

6.5. Discussion of the Results: Mining Noise Free Event Logs

The results in section 6.4 demonstrate that the GP approach can mine a range of process constructs from noise free event logs.

Process Data 1

This data set presented a combination of AND/XOR semantics and a simple parallel construct. The process could be mined correctly on all 10 runs of the GP process miner. The AND semantics on the split at task B and the join at task J were both rendered properly along with the XOR split and join connecting task G to the process The correct identification of the semantics by the approach provided the greatest challenge in this test, which was met by the approach on all test invocations. This shows the approach has an ability to mine semantics as well as structures from process data. The GA approach could also mine the structure and semantics of this process correctly on all ten runs.

Process Data 2

Process data 2 could also be mined correctly on all test invocations. The most challenging aspect of this process is the correct identification of the inner split/join construct. Again the GP approach is proved to be able to mine simple structures with semantics. Both structure and semantics could be mined correctly on each of the tests performed, including the parallel construct containing the tasks B, D and C. The GA approach was also able to correctly resolve both structure and semantics on all runs with this process data set.

Process Data 7

The process contained in this data set tested the GP approach's ability with long task sequences. The semantics were simple in this particular test comprising of two split/join parallel constructs. However, the process as a whole contained around 40

tasks testing the approach's ability to mine larger process sequences. With the GP approach he process structure could be mined correctly on all ten runs along with the identification of the appropriate semantics. Linear processes are often easier to mine than more complex process constructs. The GA approach was able to mine correct results on all of the ten runs in terms of both structure and semantics.

Process Data 4

This event log provided a challenge in that the number of unique process log traces was very limited. While the structure of this process could be mined correctly on all test invocations by the GP approach, errors in the semantics were found on four test invocations. An example of an error is when two XOR sets have been created at task 'Begin' when actually all three sets should be single AND sets. This semantic construct causes the mined model to over describe the process data allowing extra behaviour not recorded in the event log. The models mined on the four occasions over described one split or join construct each. In all occasions this involved the appearance of one or more additional single edge node subsets in the input or output sets of at least one task causing the mined model to demonstrate semantics not present in the event log. This result demonstrates the challenge that parallel process constructs can present. It is possible that the fitness measure for the GP approach may need to be adjusted to reduce the occasional tendency of the approach to produce duplicate and/or unnecessary edge node subsets. The GA approach could mine this process correctly, on all ten runs, for structure and semantics.

Process Data 6

Again a complex structure and multiple task linkages provided a mining challenge for the GP approach. The structure of the process for this data set included a number of XOR split/join constructs between several task pairs. Both structure and semantics were correctly identified on seven out of ten test invocations. The errors on this process included missing links between certain process tasks, for example between the tasks E and G. This was also reflected in the semantics. The event log for this process contained 15 distinct traces, giving a balanced view. No duplication was

present in the edge sets though certain links were missing on the test invocations that did not provide correct mined models. It is possible that due to the stochastic nature of the GP approach the missing links were removed in the evolutionary process and therefore not present in the fittest individual. It is unlikely that this process could have been mined correctly without the removal of the edge semantics from the GP representation (detailed in chapter 5 section 5.3); the original GP approach would not have been able to correctly resolve the single XOR edge nodes (incorrectly rendering them as AND edges) The GA approach could mine the structure and semantics of this process correctly on eight out of ten occasions, with mistakes focussing on missing edges between tasks D and G

Process Data 8

Another parallel process is provided by process data 8. This is the most complex parallel test in this set of experiments. The joins are complex in comparison to the previous data set and coupled with the parallel structure make this process the most difficult of all to mine compared to the other artificial data sets. The structure is similar to process data 7 though the semantics are far more complex. The semantics present a mix of AND and XOR split/join constructs. Like process data 7, tests were carried out with and without the parallel process function. From the results, it can be seen that the parallel function allows the structure to be mined correctly on seven out of ten occasions versus just three out of ten without the function. Without the parallel function the structural errors included parallel tasks being linked in the incorrect order. An example is the task G being linked to by another of the parallel tasks. The semantics also showed that additional edges were being added to the sets incorrectly (without the parallel function). With the parallel function the structural linking errors, involving the linking of one parallel edge to another, noted from the previous ten runs were much reduced. The semantics showed only a marginal improvement. The edge sets contained additional edges allowing for a greater level of behaviour than described in the event log.

Modelling parallel spilt/join points that contain over 3 parallel edges is a difficult problem in process mining due to the interleaved nature of such constructs when mined from an event log (van der Aalst et al., 2003b). The semantics, being a combination of AND/XOR edge sets that split and join in an unconventional way, could not be mined correctly with or without the use of the function. This is a common occurrence for most process mining algorithms when tested with this process data set. It is therefore of benefit to use the parallel function with processes containing four or more parallel edges. Chapter 5 section 5.9 gives more detail on the parallel function designed for use in the GP approach. The GA approach failed to mine the structure or the semantics correctly on any of the runs. Errors in the GA approaches' mining of this process included additional edges between the parallel tasks and tasks A and G resulting in spaghetti process structures

Process Data 9 and 10

The process data sets 9 and 10 were provided by the industrial sponsor of this project. The data was thought to be noise free with a reasonable representation of traces containing the most frequent paths through the process. No overall template was documented within the industrial sponsor for either process. Both processes describe the generic administrative steps a telephone engineer must initiate after attending to a fault for a customer.

The structure of process data 9 is comprised of a linear sequence of tasks followed by a split at the open task. The parallel structure below this task is comprised of several more splits which eventually join up at the closed (start) task. The process describes an unsuccessful resolution of a fault.

In order to obtain feedback on this process, two senior managers from the sponsor organisation, involved in the management/maintenance of the two processes, were given a questionnaire to complete (shown in Appendix C). Overall the feedback from the sponsor organisation confirmed that the flow through the process and the structure was logical and largely correct. Comments about the process indicated that the structure had been mined correctly, including the main parallel construct. None of the

respondents noted tasks or constructs that they knew to be missing. The semantics of the process were not included as part of the assessment as on all occasions they were very poor. It was noted by the author that several duplicate edge sub sets were present in the sets of many of the split and join point tasks. One invocation of the test produced a spaghetti type structure (where additional erroneous edges are added to the mined model making it difficult to discern the correct flow).

Process data 10 describes a successful resolution of a fault by a telephone engineer. This data set displays almost the same task set as process data 9 but in a different order with a linear set of task followed by a split at the CreateRequested(start) task. Another split at the 'Open' task, forms a parallel structure below it. Again the feedback from the organisation was positive about the correctness of the mined process. However the author noted that the semantics did suffer in a similar way as reported for process data 9, with duplicate sets. The semantics of the process were very poor on al ten runs and are not included as part of the assessment. The GA could not mine this process at all and did not return a result.

From the questionnaire results (shown in Appendix C) it was clear that all tasks and constructs, that were expected, were present and in the correct order. The respondents also noted how useful the experience was of actually seeing the whole process presented as a graph.

The mining of the structures of both process data 9 and 10 shows that the GP approach can deal with more complex real life processes and large data sets but that more research is needed in order to correctly mine the semantics of such data sets. Again this points towards further refinement of the fitness function with regard to duplicate edge sets and over described semantics. It is interesting to note that both process data 9 and 10 produced some 'spaghetti' process models that sometimes result from the mining of complex processes. Such 'spaghetti' process models contain, in this case, additional arcs between tasks that do not exist in the event log and make understanding of the overall process more difficult. The GA approach did not return result for either of these data sets.

6.6. Experimental Results: Convergence graphs

In addition to the validation results given so far four processes have been examined in further detail. The aim of this analysis is to derive, if possible, a convergence point between the highest fitness achieved and the average population. To do this the highest fitness level achieved and the average fitness level scores are taken at intervals of 20 generations for a total of 200 generations for each of the three process data sets shown in Table 17. Each of the four process data sets are then mined a total of 10 times each (with different seed values) to produce a set of average values. The concept behind convergence concerns the reliability and repeatability of the results obtained from an evolutionary based technique (Bäck 1996), the nearer to convergence the more reliable the result.

Table 17: Process Data Sets to be Used in the Convergence Tests

Data Set	**Sequences**	**Semantics**	**Parallelism**
(1) A12 (Alves de Medeiros, 2006)	Medium	Medium	Low
(2) A8 (see chapter 4) (Alves de Medeiros, 2006)	Low	Low	Low
(6) Choice (Alves de Medeiros, 2006)	High	High	Low
(3) Parallel 5 (see chapters 4 & 5) (Alves de Medeiros, 2006)	Medium	Medium	High

According to Koza (1992) convergence is when a population becomes homogenous around the fittest individual. Though, in GP Koza (1992) states that convergence is ‘unlikely’ as the crossover operator will always introduce enough variety into the population to avoid a state of homogeneity. In the results presented here it will be seen how long its takes the GP approach to produce a population that is close to convergence. The mining parameters used for these three data sets remain the same as in the experiments detailed earlier in section 6.3 of this thesis.

Process data 1 was mined first and the resulting convergence graph is shown in Figure 72.

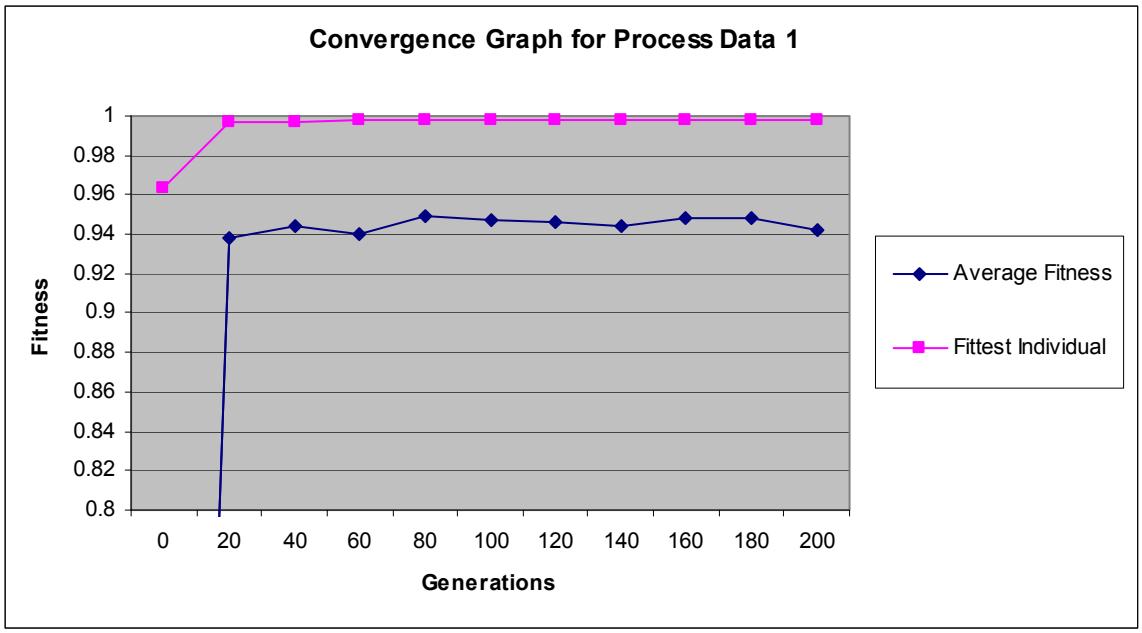


Figure 72: Convergence Graph for Process Data 1

From Figure 72 it is possible to see that the fittest individual was on average produced before generation 20. In fact with process data 1 the fittest individual was produced on average at generation 55 (with the heuristics producing an individual of fitness 0.78 in the initial population). It is interesting to note that the average fitness stays above 0.93 for the duration of the test beyond 20 generations despite the continued use of crossover and mutation operators. Figure 73 shows the result for process data 2. Again, like process data 1, this process is fairly straightforward for the GP mining approach. The fittest individual was produced on average at generation 46 (with the heuristics producing an individual of fitness 0.64 on average in the initial population). The average fitness measure, like process data 1, stays above 0.91 for the duration of the test. In common with process data 1 the average and fittest measures are fairly close (and relatively stable over the duration of the runs, though there is no actual convergence.

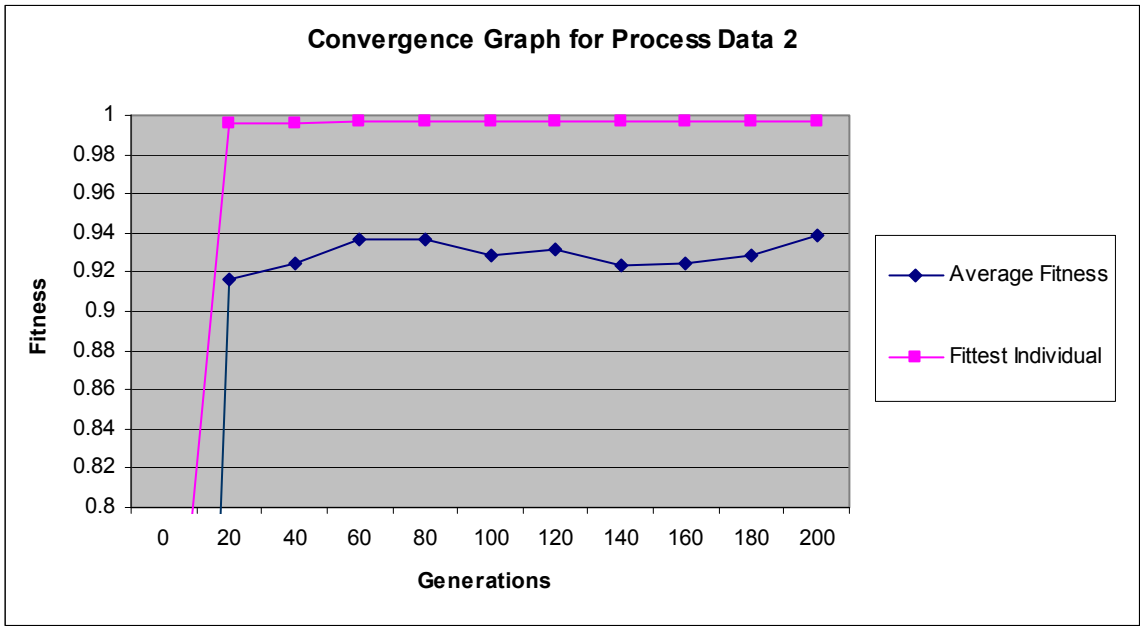


Figure 73: Convergence Graph for Process Data 2

Process data 6, shown in Figure 74, demonstrates a smaller gap between average fitness and the fittest individual. The average fitness measure over the course of the test remains above 0.93. This result is interesting in that the fittest individual is found over half way through the run (at 104 generations, starting from a fittest individual of 0.69 average fitness in the initial population). This process is more complex than 1 and 2 so perhaps it would have been expected that the gap between average and fittest should be larger.

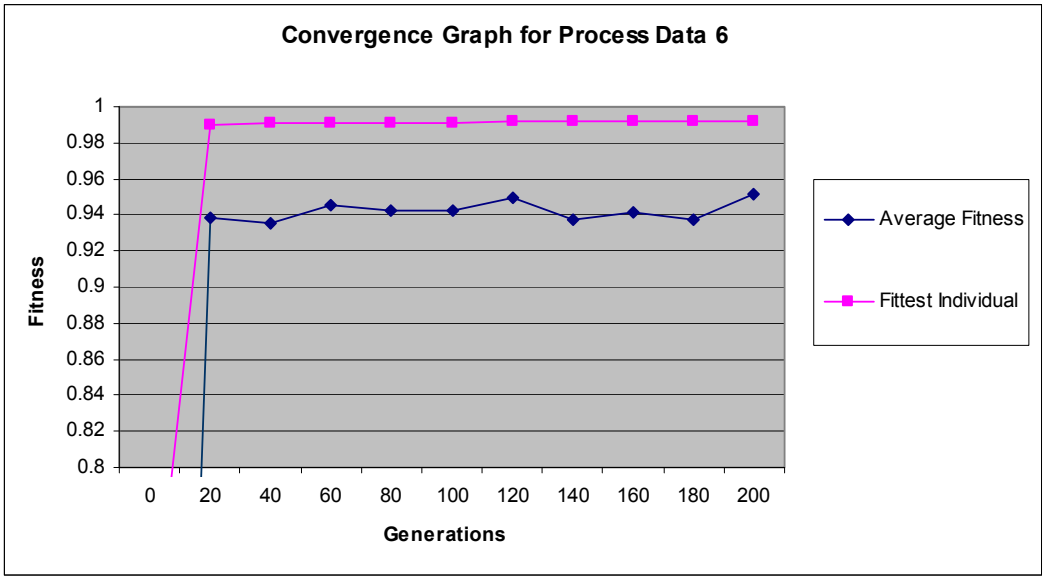


Figure 74: Convergence Graph for Process Data 6

The challenge provided by process data 3 is the greatest of the tests in this section. Figure 75 demonstrates a much larger gap between average and fittest measures. With this test the fittest individual is not found on average until generation 152 (with the initial population containing a fittest individual with an average score of only 0.17).

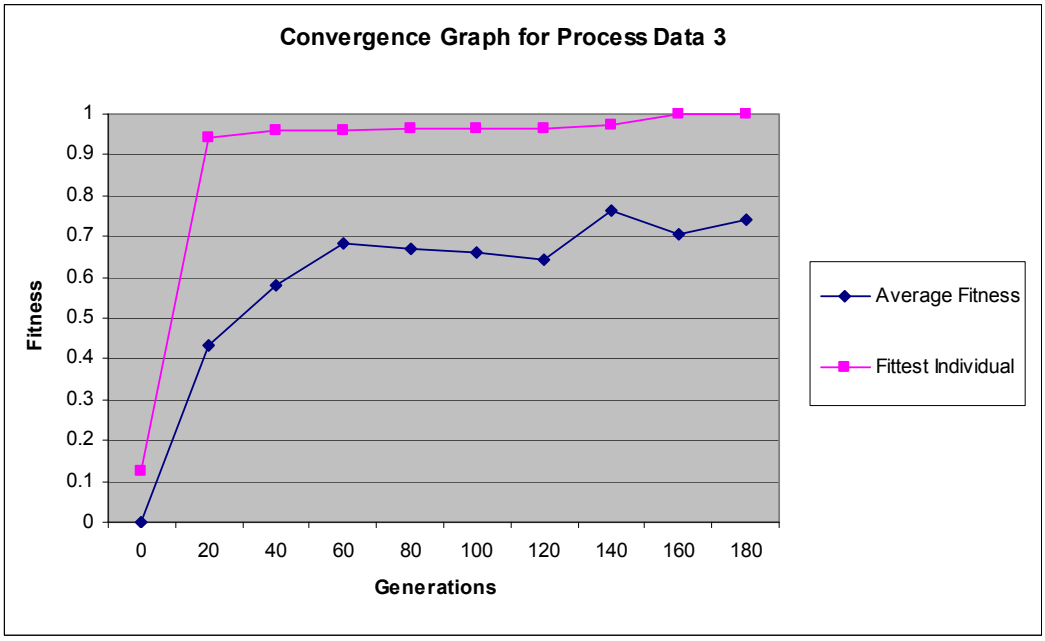


Figure 75: Convergence Graph for Process Data 3

Although convergence was not achieved in any of these tests, as postulated by Koza (1992), the gap between average and fittest was minimal and fairly stable throughout the run for all but the most complex data set. Towards the latter stages of the mining of each data set the average is either stable or rising. It was also noted that for each of the tests the fittest individual measure showed very little variance between runs. As expected for a stochastic technique such as GP the average fitness measure for each test showed a greater amount of variance within the constituent runs. These tests underline the reliability of the GP process mining approach.

6.7. Experimental Results: Mining Event Logs in the Presence of Noise

In this section, three process data sets (shown in Table 18) have been mined under simulated noisy conditions. The presence of noise in event logs creates a number of problems in process mining. Noise can take the form of missing tasks and tasks recorded in the wrong order of precedence in the log. Such problems can lead to the creation of erroneous edges between tasks, and missing edges, in a mined process model. Ideally a mining approach must have a level of noise resistance.

Each of the data sets, used in the validation presented in this section, were filtered and changed to add a percentage of noise. Two types of noise filter were used to remove tasks from the process at random. The data sets were specially prepared with 100 traces; with each trace used 3 times (this reduces the amount of parsing the fitness function has to do so reducing the amount of time the algorithm takes to produce results). The three data sets were selected to provide a range of process constructs the structure of which may be distorted with the introduction of noise. Noise free versions all three data sets were provided. Process data 2 and 6 was provided by Alves de Medeiros (2006) and process data 4 by Herbst (2001). The tests are distributed with the ProM software (van Dongen et al., 2005b).

Table 18: Noisy Process Data Sets to be Mined

Data Set	Sequences	Semantics	Parallelism
(2) A8 (Alves de Medeiros, 2006)	Low	Low	Low
(4) 6p45 (Herbst, 2001)	Low	Low	Low
(6) Choice (Alves de Medeiros, 2006)	High	High	Low

The two filter types worked in the following way -

- The 'remove task from top filter' removed a number of tasks at random from the top third of the process.
- The 'remove task from middle filter' removed a number of tasks, at random, from the middle third of the process.

A level of 5% noise was selected for the tests meaning that 5% of the traces were altered by the noise filter. The 5% noise setting was decided upon as a reasonable test for the GP approach based on consultations with the industrial sponsor on expected noise levels in real process data. The filters are provided as part of the ProM process mining software (van Dongen et al., 2005b). At most the filters removed one third of the top or middle of a process. The filters were selected after consultation with the industrial sponsor on what types of noise were most likely to be present in their process data.

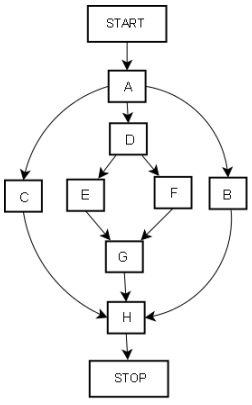
Process Data 2

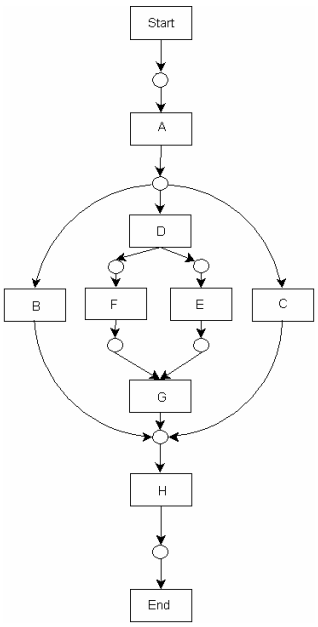
This data set was mined in the presence of 5% noise with the 'remove from top' task filter. Out of five runs, the structure and semantics could be mined correctly on four occasions for the GP approach (the best GP result is shown in Figure 76 and the

typical GP result is shown in Figure 77). On one occasion an extra arc was present between two tasks. The GA approach could mine both structure and semantics correctly on all five occasions (the best GA result is shown in Figure 78).

With the 'remove from middle' task filter at 5%, acting on the middle of the process, the structure could not be mined correctly with incorrect links between tasks and incorrect semantics (the best GP result is shown in Figure 79, and the typical GP result is shown in Figure 80). The GA approach could mine both structure and semantics correctly on all five occasions (the best GA result is shown in Figure 81). The correct template for this process is shown in section 4.5.1 chapter 4.

Figure 76: Best GP Result for Process Data 2 with 'Remove From Top Filter' Structure View (Left) Semantics View (Right)





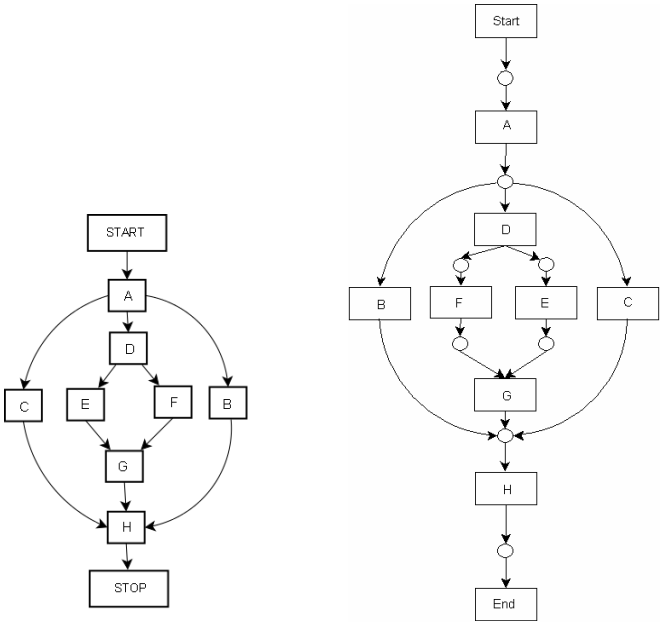


Figure 77: Typical GP Result for Process Data 2 with 'Remove From Top Filter' Structure View (Left) Semantics View (Right)

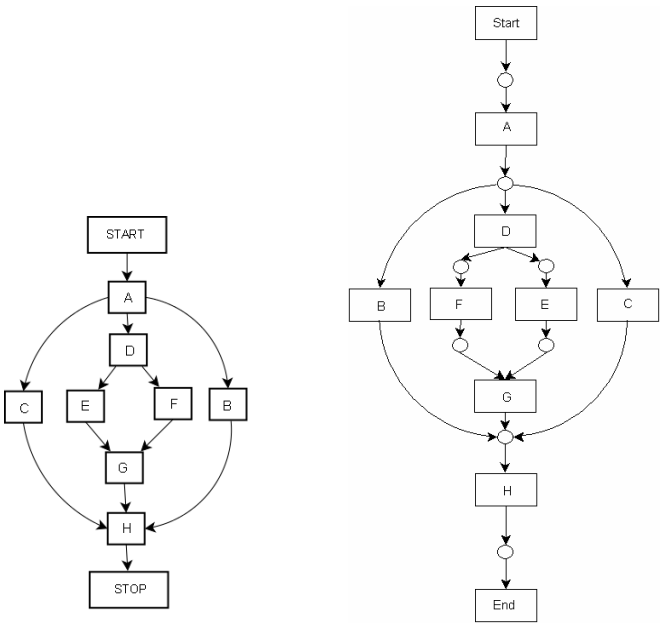


Figure 78: Best GA Result for Process Data 2 with 'Remove From Top Filter' Structure View (Left) Semantics View (Right)

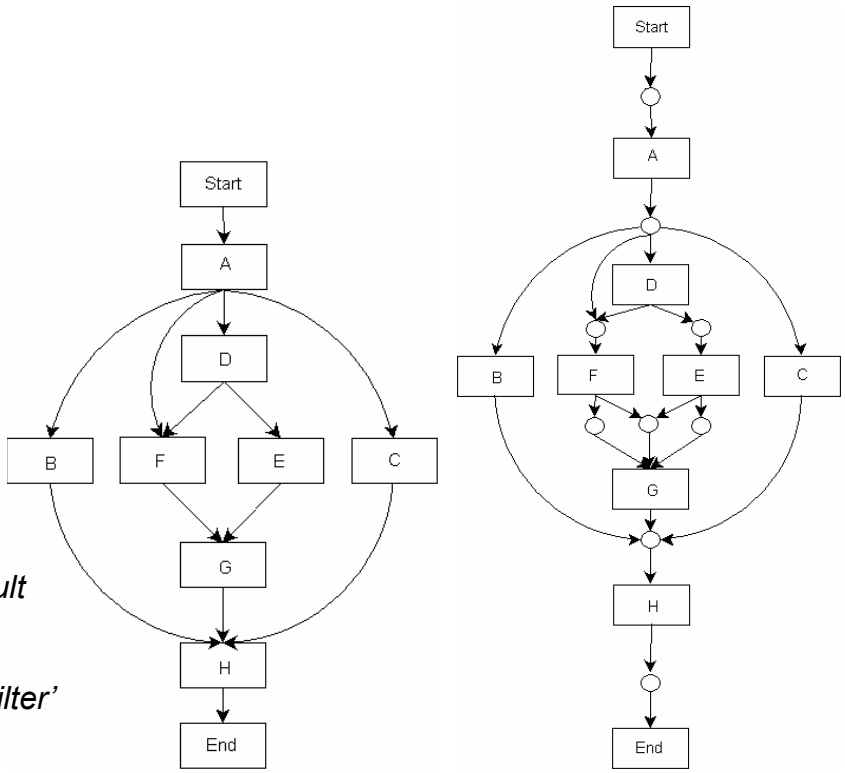


Figure 79: Best GP Result for Process Data 2 with 'Remove From Middle Filter' Structure View (Left) Semantics View (Right)

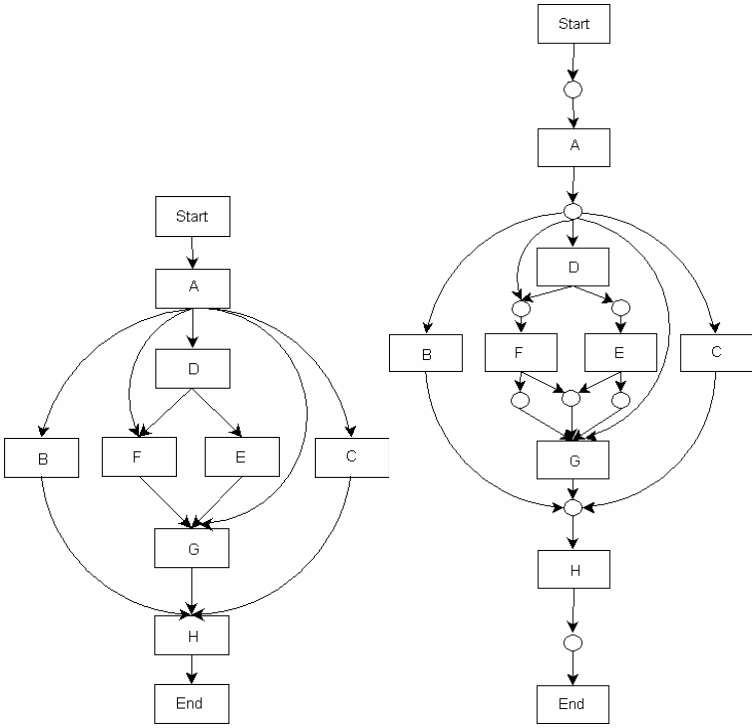
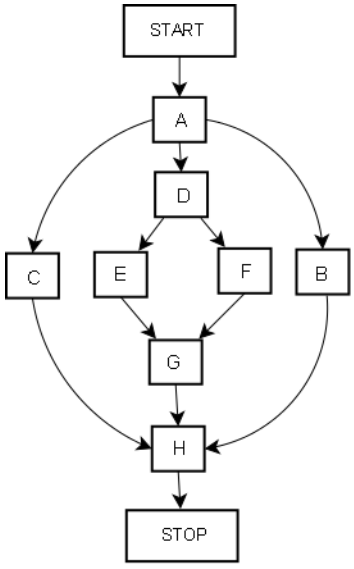


Figure 80: Typical GP Result for Process Data 2 with 'Remove From Middle Filter' Structure View (Left) Semantics View (Right)



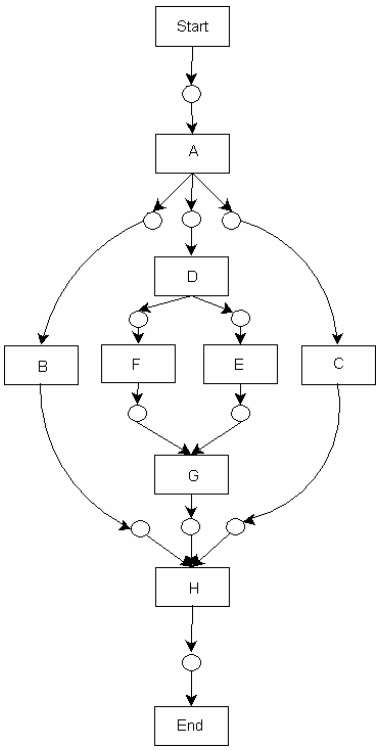


Figure 81: Best GA Result for Process Data 2 with 'Remove From Middle Filter' Structure View (Left) Semantics View (Right)

Process Data 4

Data set 4 was also mined in the presence of 5% noise with the 'remove from top' task filter. The structure for this process could be mined correctly by the GP approach on all 5 runs (the best result for the GP approach is shown in Figure 82 and the typical result is shown in Figure 83). The semantics for this process could also be resolved correctly on 4 occasions. On one run duplicate sets were produced. The GA could mine both the structure and semantics correctly on all five runs (the best result for the GA approach is shown in Figure 84). The 'remove from middle' task filter set at 5% noise allowed only 3 out of five process structures to be mined correctly. The semantics were slightly incorrect (the best GP mining result is shown in Figure 85 and the typical result is shown in Figure 86). The GA could mine both the structure and semantics correctly on all five runs (the best result for the GA approach is shown in Figure 87).

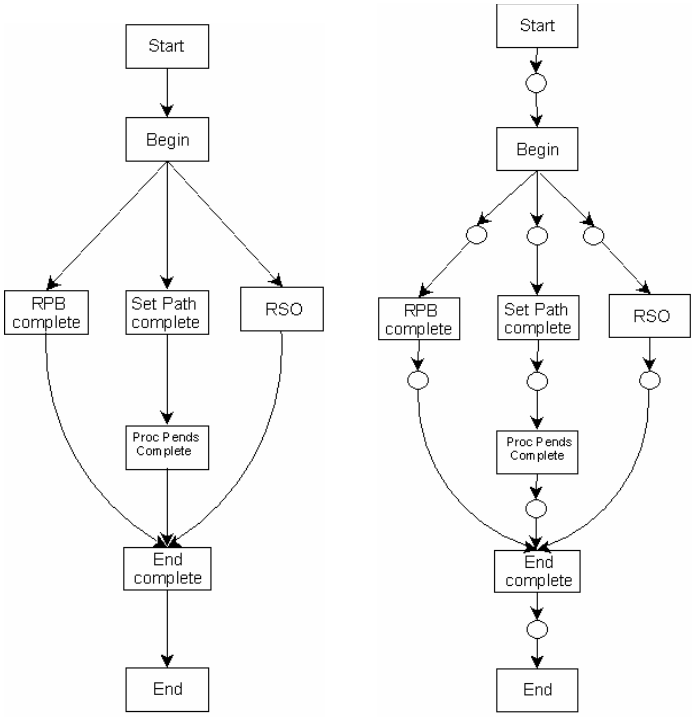


Figure 82: Best GP Result for Process Data 4 with 'Remove From Top' filter Structure View (Left) Semantics View (Right)

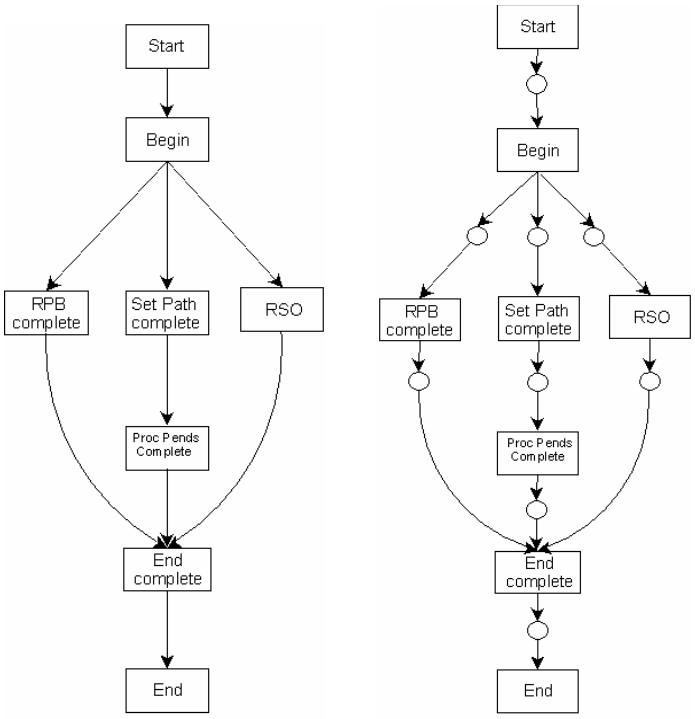


Figure 83: Typical GP Result for Process Data 4 with 'Remove From Top' filter Structure View (Left) Semantics View (Right)

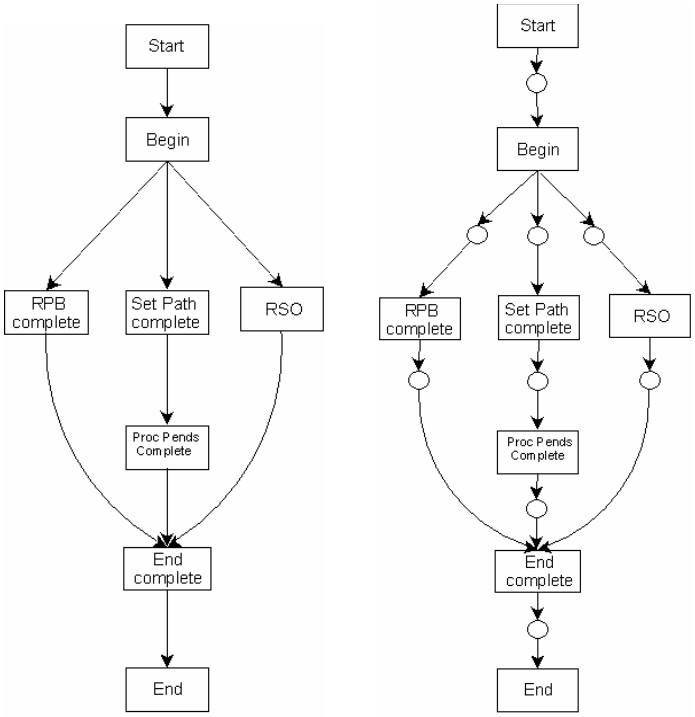


Figure 84: Best GA Result for Process Data 4 with 'Remove From Top' Filter Structure View (Left) Semantics View (Right)

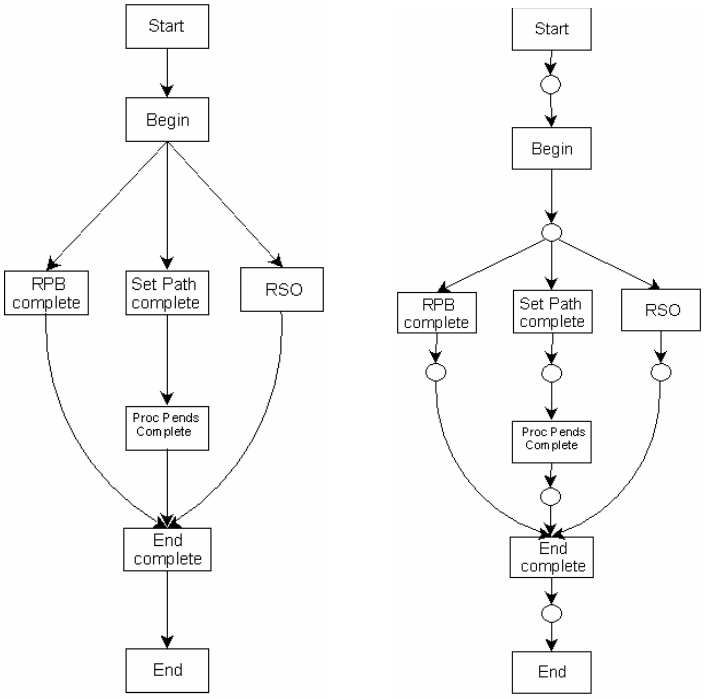


Figure 85: Best GP Result for Process Data 4 with 'Remove From Middle' Filter Structure View (Left) Semantics View (Right)

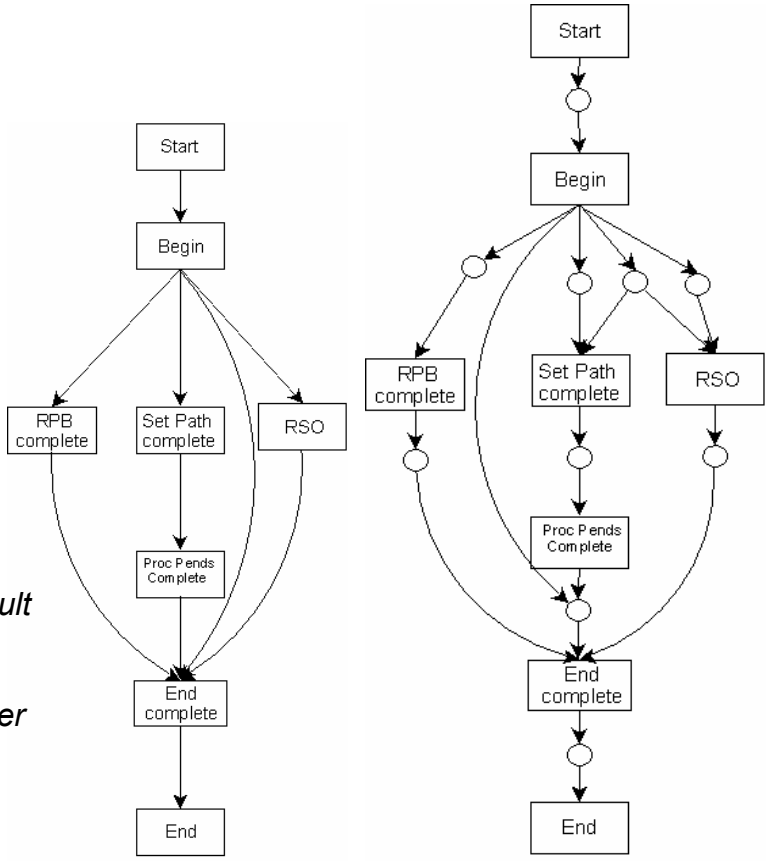


Figure 86: Typical GP Result for Process Data 4 with 'Remove From Middle' Filter Structure View (Left) Semantics View (Right)

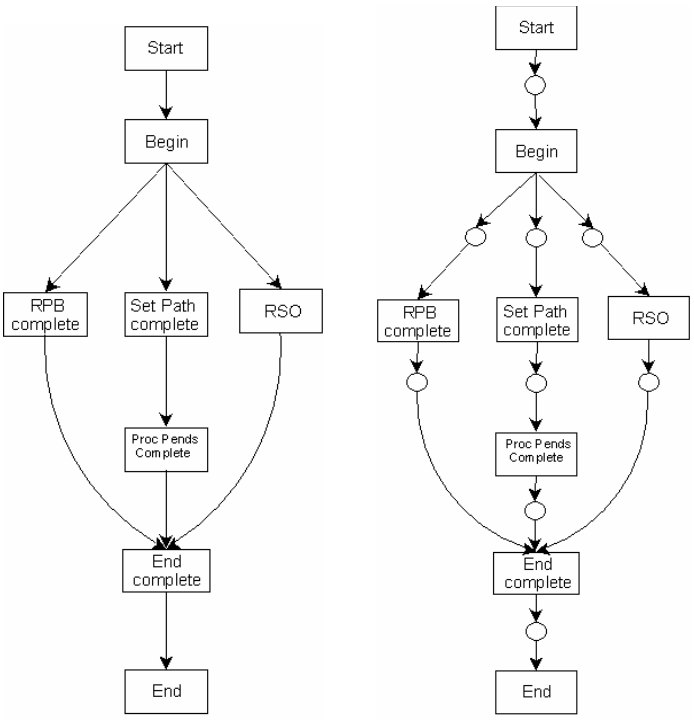


Figure 87: Best GA Result for Process Data 4 with 'Remove From Middle' Filter Structure View (Left) Semantics View (Right)

Process Data 6

This data set was mined in the presence of 5% noise with the 'remove from top' task filter. The structure of data set 6 could be mined successfully on two out of five runs with the GP approach. Another two runs showed some missing edges. The semantics showed a few errors on each run, with an additional edge being added to one or two edge sets and edges missing from sets (the best result for the GP approach is shown in Figure 88 and the typical result is shown in Figure 89). The GA could mine both the structure and semantics correctly on all five runs (the best result for the GA approach is shown in Figure 90).

With 5% noise and the 'remove from middle' task filter, the structure and semantics for this process could not be mined correctly by the GP approach (the best result for the GP approach is shown in Figure 91 and the typical result is shown in Figure 92). The main errors were missing links between tasks and in the semantics missing edges from sets. The GA could not mine both the structure and semantics correctly on any of the five runs (the best result for the GA approach is shown in Figure 93).

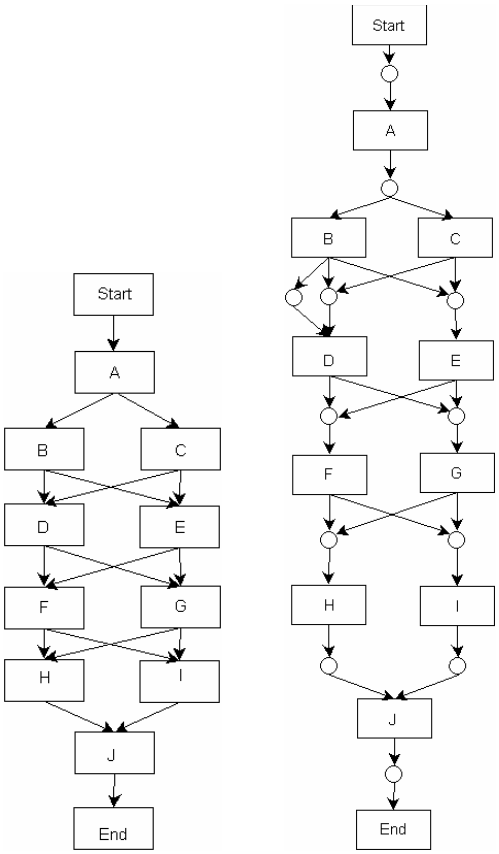


Figure 88: Best GP Result for Process Data 6 with 'Remove From Top Filter Structure View (Left) Semantics View (Right)

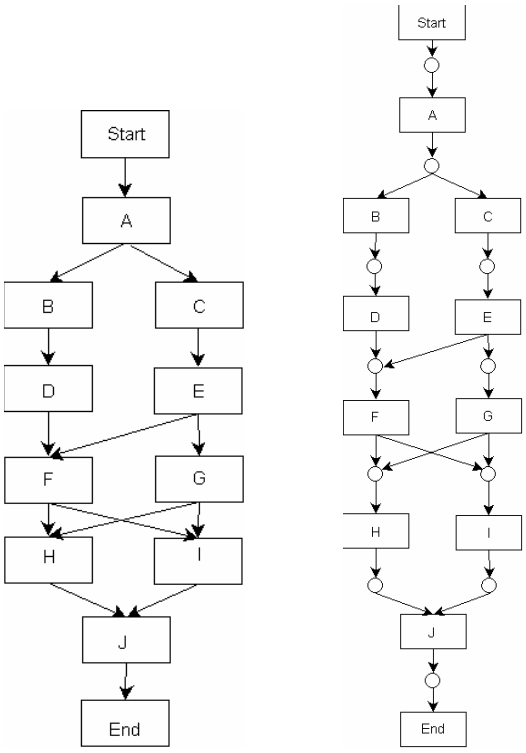


Figure 89: Typical GP Result for Process Data 6 with 'Remove From Top' Filter Structure View (Left) Semantics View (Right)

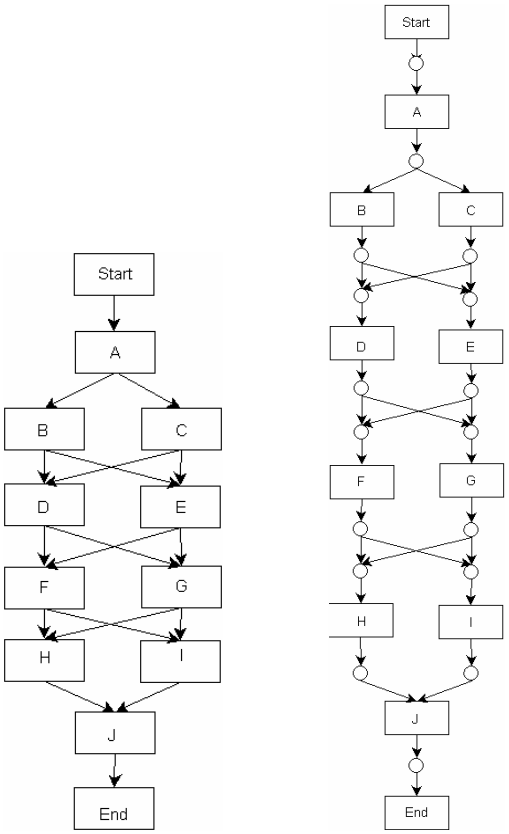


Figure 90: Best GA Result for Process Data 6 with 'Remove From Top' Filter Structure View (Left) Semantics View (Right)

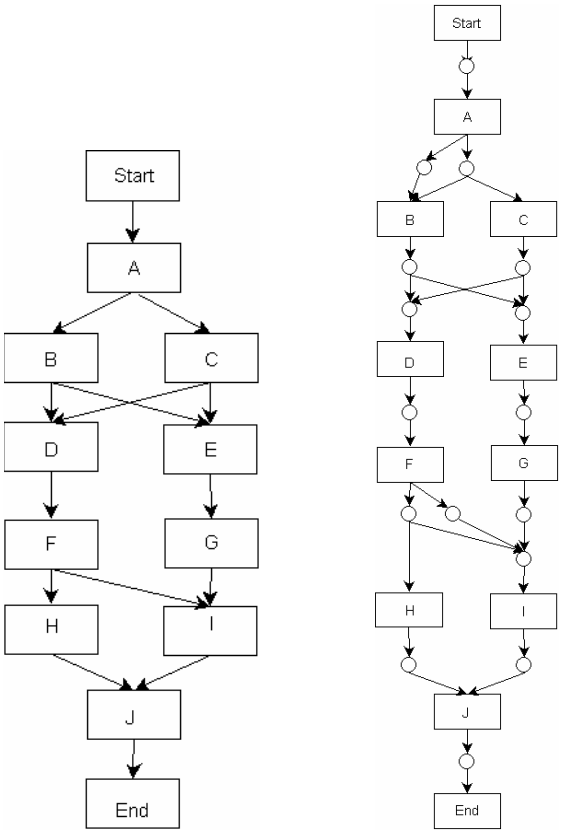


Figure 91: Best GP Result for Process Data 6 with 'Remove From Middle' Filter Structure View (Left) Semantics View (Right)

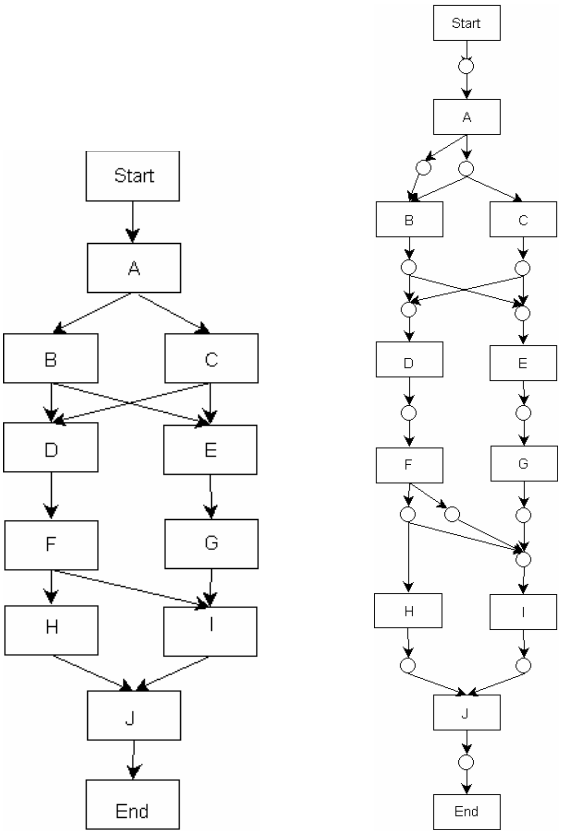


Figure 92: Typical GP Result for Process Data 6 with 'Remove From Middle' Filter Structure View (Left) Semantics View (Right)

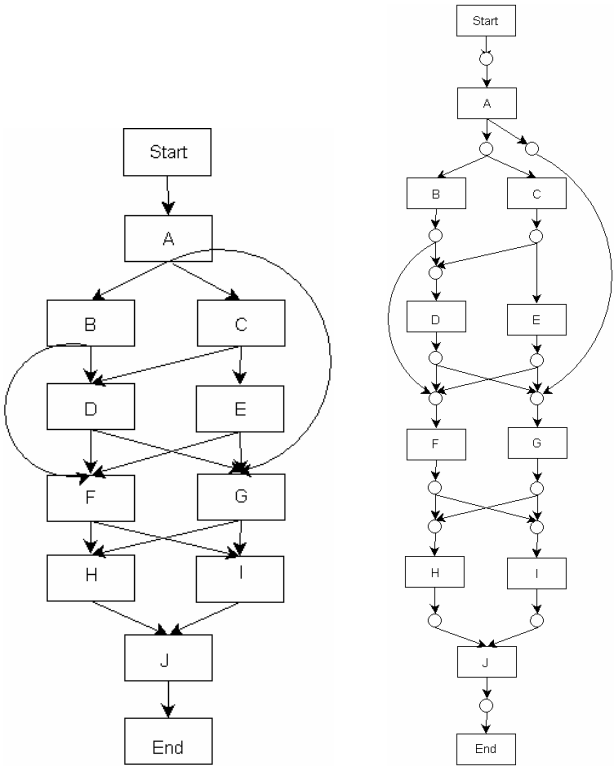


Figure 93: Best GA Result for Process Data 6 with 'Remove From Middle' Filter Structure View (Left) Semantics View (Right)

6.8. Discussion of the Results: Mining Event Logs in the Presence of Noise

The results show that the GP approach has a tolerance to low levels of noise (5%) in the data when mining structures. However, the mining of the semantics tends to suffer more when noise is introduced.

Process Data 2

The mining of this data set showed that removing tasks from the top of the process created fewer problems than when the tasks were removed from the middle of the process. The central split/join construct semantics of the process was error prone especially when tasks were removed from the middle. Split/join points do require a good representation of traces to be mined correctly, though the fitness function should still be able to lead the GP approach to complete individuals. The incomplete individuals made up the majority of the population and the evolutionary process did not repair enough individuals to produce a correct result on a number of runs.

This data set was mined in the presence of 5% noise with the ‘remove from top’ task filter. This filter only removes at most a third of the tasks from a process trace (acting on the top, middle or bottom of the trace, on 5% of the process traces). Out of five runs, the structure and semantics could be mined correctly on four occasions.

With the ‘remove from middle’ task filter at 5%, acting on the middle of the process, the structure could not be mined correctly with incorrect links between tasks and incorrect semantics. A common error was an extra link between task A and a task in the central AND split/join construct. The main errors in the semantics concerned the duplicate task sets and incorrect edges in sets

The GA approach produced better results than the GP when mining this process with tasks removed from middle of the process and when tasks were removed from the top.

Process Data 4

Again, like process data 2, removing tasks from the middle of the process made both the structure and the semantics more difficult to mine correctly. The AND sets of the split/join tasks were also incorrectly combined into XOR sets. This possibly points towards the need for revisions to the fitness function to reduce the occurrence of incorrect edge sets.

In the presence of 5% noise with the 'remove from top' task filter this process could be mined correctly on all 5 runs. The semantics for this process could also be resolved correctly on 4 occasions.

The 'remove from middle' task filter set at 5% noise allowed only 3 out of five process structures to be mined correctly. The semantics were slightly incorrect in that two of the AND edge sets were combined into an XOR set on several of the runs.

The GA approach dealt better with the mining of this process with tasks removed from middle of the process and when tasks were removed from the top.

Process Data 6

This was the most complex process to be mined in the presence of noise. The structure and semantics could be mined when tasks were removed from the top of the process, on two out of five runs. Another two runs showed some missing edges. However removing tasks from the middle of the process caused problems with missing edges between tasks and incorrect semantics. The overall structure could still be discerned in most runs when removing middle tasks. The structure of this process does not often occur in real processes, though the lattice construct is still a good test of a process mining approach's ability to mine intricate structures and semantics (this process is, in its semantics, can be more difficult to mine correctly than a larger process, such as process data 5).

The GA approach also produced poor results for the mining of this process with tasks removed from the middle of the process though the results were slightly better than the GP approach when tasks were removed from the top of the process.

6.9. Overall Discussion of Validation

The GP process mining approach is able to resolve a range of process structures from data logs. These structures vary from simple linear structures to the more complex parallel process structures which are difficult to mine. Real process data, provided by BT, has also been mined by this approach. There are a few limitations of the GP approach as the semantics of the more complex processes including the parallel structures are difficult to resolve correctly. The main issues are duplication of edge sets where it is not necessary and incorrect links between process tasks.

The duplication of edge sets is partly due to the stochastic nature of the GP approach and potentially due to the fitness algorithm. This may require further modification to minimise excess edge sets in the semantics of the mined process (this is discussed in the outlook section of this thesis). Occasional mistakes in the split/join semantics also occur in the more complex processes. One or two models over described the data allowing for additional behaviour not recorded in the event log. An example is additional edge nodes in sets, so creating additional XOR splits and joins.

Complex parallel structures, such as process data 7 and 8 can be mined by the GP approach. However, the semantics prove to be more of a challenge for both processes. The semantics of process data 8 are particularly challenging due to the nature of the split/joins within the process, which are XOR in nature.

The GP approach has an ability to mine data in the presence of a low level of noise. The 5% level tolerance has been achievable in the selected processes when removing tasks from the top third of the process. Removing tasks from the middle of the process can cause more errors in both structure and semantics. Most of the errors in mining of

noisy processes are in the semantics and edge set duplication. The edges of noisy models can be trimmed to remove low frequency (potentially incorrect) transitions between tasks. In effect the errors noted in the mining of these processes without the presence of noise were in effect amplified when using the noise filter that removed tasks from the middle of the process. The results obtained for the GP approach point to its success in mining of simple to complex process structures. A basic level of semantics can also be mined using this approach. However, in order to improve the accuracy of the mining results further modifications are required to the fitness and the parsing functions (used to compare GP individuals with the event log data in order to obtain a measure of fitness).

6.10. Summary

The GP approach can mine a range of process structures, without noise and with a low level of noise. The approach also has some success with more complex parallel structures. The XOR/AND semantics can be mined reliably on most of the noise free logs. However, there are a number of drawbacks in that the mining of semantics, for more complex processes, is error prone due to the occasional duplication of edge sets and incorrect interpretation of AND/XOR splits and joins.

The GP approach can cope with 5% noise (removing tasks from the top of the process) on a number of tests completed for this research. Removal of tasks from the middle of the process creates more errors in both structure and semantics. Modifications to the fitness function and the parsing technique used in the GP approach may help in the potential of mining event logs containing higher levels of noise. These enhancements plus a discussion of the findings and conclusions of this research are detailed in the following chapter.

7. Discussion and Conclusions

This chapter will summarise the main findings and observations of the research presented in this thesis. The contributions and limitations of the research will also be highlighted here, along with the conclusions.

7.1. Discussion

This section will discuss the main research findings and observations and provide a summary of the key chapters that comprise this thesis.

7.1.1. Key Observations from the Research

Evolutionary approaches are used in a wide variety of applications, and are now being used as a process mining technique. This thesis presents a new evolutionary approach to process mining using Genetic Programming. Unlike many existing process mining techniques the GP approach utilises a graph based representation and parsing technique. This approach is more flexible and intuitive and also allows for the mining of a wide variety of process constructs, with and without the presence of noise in the process data (process event logs).

The GP approach utilises a number of heuristics to allow for more efficient mining of process (utilising a reduced number of generations), including processes with a high degree of parallel tasks.

The field of process mining is examined in detail including the industry views on this rapidly maturing area. The key findings in the development of the GP approach are presented here along with an outlook for the GP approach and the practice of process mining as a whole.

Literature Review

This research has examined the area of modelling techniques for process mining. A range of techniques exist ranging from simple flow diagrams to more technical approaches such as Integrated Definition Modelling (IDEF), Unified Modelling Language (UML), Petri nets and Event Process Chains (EPC). The literature review conducted for this research demonstrates the vast range of modelling approaches that are available to industry and academia (a topic that will be discussed further in the industry survey sub-section of this chapter).

Process mining, as a technique, has been around since the late nineties (with early practice having been developed directly from data mining techniques). There has been a range of process mining approaches developed since then, many of which take advantage of the increasing availability of event logs (that record execution data for live business processes) from enterprise information systems within medium to large organisations. The vast majority of process mining approaches are custom approaches with many having been developed using techniques and/or experience from data mining practice. In the last few years attempts have been made to harness the power of non deterministic methods in the mining of processes, in order to obtain better mining results when mining complex processes and event logs that contain mining problems such as noise in the data. Initial research has been conducted into such techniques as Neural Networks, Genetic Algorithms (GA), Clustering, Fuzzy Logic and Region Mining and their application to process mining.

Process mining approaches based on GA, Clustering and Region Mining are able to tackle the widest range of process mining problems, though they are not without their limitations (such as time taken to mine results, poor handling of parallel processes and resistance to noise). One particular technique, based on heuristics, has gained appeal with developers of new process mining approaches as a way of providing initial, rough, process models that may be further refined (both the GA and Clustering based approaches utilise heuristics).

Despite the development of a GA based approach the use of evolutionary techniques for process mining still remains a largely unexplored area. This is due to the complexity of translating a complex construct such as a business process to an intermediary representation that an evolutionary technique can then manipulate. However, an evolutionary technique called Genetic Programming (GP) provides a more flexible way of representing individuals. In GP, a population of computer programs is bred; in effect each individual is a program. In fact, several broad types of representation are available with GP linear, tree and graph based representations. Both linear and tree based programs are organised so the code is arranged either in a linear list of instructions or as a tree, whereas tree branches may be selected for representing choice constructs in normal programs. However, it is the graph representation, where programs are represented as process like flowcharts, which holds the most promise for a practice such as process mining.

Industry Survey

In this research, it was necessary to contact industry and seek their views on process mining. In doing so, it was also important to identify their knowledge and awareness of process mining as a practice to ascertain their requirements in this field. The feedback from a set of questionnaires and interviews was revealing in the level of industry knowledge in the area of process mining.

Although many organisations had the potential to conduct process mining (most had access to process data in the form of an event logging system), few actually did. Most process analysis, where it occurred, was manual in nature and utilised the flowchart as the process modelling tool. Few respondents had heard of process mining either as a term or a practice.

However, when introduced to process mining, most respondents were positive towards the technique and saw the potential benefits of being able to mine process diagrams automatically from complex, and sometimes, noisy data. Even those respondents who did not regularly review their processes were positive towards the process mining concept.

The need for industry to gain more knowledge on process modelling techniques and standards was also evident from this survey. Only a few used the Event Process Chains (EPC) form of modelling, and no organisation in the survey was a user of Petri nets, a common technique in literature and subject of much academic research.

Research Gap

It was clear from the literature review that many process mining techniques were developed from existing approaches used in data mining practice. It is also true that while initial research has been carried out in the use of GA techniques the wider field of evolutionary approaches is still to be explored in relation to process mining.

Process modelling techniques in industry mainly concentrate on the use of simple flowcharts. However, in academia research into Petri nets is mature and many process mining approaches apply the theory of Petri nets in their operation. While Petri net theory can offer much to the mining of processes, it is perhaps rather cumbersome to use. In addition there has not been much investigation into alternative non Petri net based representations for process mining practice, in particular the direct use of a graph as a representation.

From literature it was identified that there are many problems that are encountered when mining process models from event logs. There have been attempts to address all of these problems by the development of various mining approaches, though there are a number of complex problems that still require further research.

This research has addressed the areas of evolutionary mining applied to process mining using a Genetic Programming (GP) approach. This approach has used a graph based representation and developed an alternative parsing technique. In addition, the mining problems of noise and complex parallel processes have been addressed by this approach. There has also been an investigation into the development and further use of heuristics in order to aid the mining accuracy and efficiency of the proposed GP approach. This is the first use of GP for process mining. Hence, the proposed GP process mining approach is novel.

Initial GP Approach

The initial development of the Genetic Programming (GP) approach to process mining involved the inception of the graph based representation. This involved the modification and use of JGraphT objects (JGraphT, 2009) to represent individual process graphs within the GP approach. The graph representation was required so that a population of potential process mining solutions could be created for the GP approach to then 'evolve' over a predefined number of generations. It also involved the development of a fitness parsing technique compatible with the graph representation. The parsing technique involved the creation of a new marking scheme, capable of maintaining state of an individual throughout the parsing process (allowing for an audit trail of marking to be preserved in the individual for the duration of that individuals parsing). It was decided to adapt the fitness measure of an existing GA technique. The approaches taken to crossover and mutation used by the existing GA approach were also modified and refined for use in the GP.

This initial GP approach was capable of mining simple process structures correctly and some of the semantics (that is the AND/XOR splits and joins within a process). It also achieved some success with parallel process structures. However, its performance did degrade when encountering more complex structures. The technique, in part, suffered from an inability to represent sing XOR edges within the representation. At this point the initial GP approach required that all node edge sets (that is the task pointers used to indicate which other tasks a task has edges to) should have predetermined semantics (each subset should be marked as AND or XOR forcing the edge nodes within a subset to be in AND or XOR relationships). This limitation need to be removed in order to develop the GP approach further.

Further Development of the GP Approach

Modifications to the original GP approach were required to enable the mining of a wider range of processes. The enhancement concentrated on the redesign of the subsets used in the graph task nodes and involved the removal of edge markings

showing which edges were in AND relationships and XOR relationships (the semantics). This allowed for the representation of single edges that may be placed in XOR or AND relationships at the time of parsing (allowing flexibility in fitness paring for single edge is important when allowing the GP approach to determine the semantics of a given single edge node). A number of parsing rules had to be modified as well to take account of the removal of the edge node subset semantics. A number of alternative approaches to parsing, which utilised the stochastic nature of the GP, were also examined in this research. The modifications also allowed for the full use of the crossover and mutation operators, contributing to the ability of the GP approach to mine a wider range of processes with fewer errors.

A number of heuristics were developed to improve the mining accuracy of the GP technique. In particular additional use was made of the event log to help to determine the correct precedence of tasks in parsing (detailed in chapter 5 section 5.8), so allowing for the potential removal (through mutation) of edges that are out of sequence (earlier tasks relying on input from later tasks). A similar heuristic was developed to aid the mutation operator when selecting tasks to add to an edge node subset. This heuristic also relied on the event log to allow for the selection of valid tasks (tasks that were either earlier or later in the task sequence as derived from the event log; detailed in chapter 5 section 5.7).

The result of the modifications allowed for the mining of more complex process structures and also for the semantics to be resolved correctly for a number of processes. The mining of parallel processes was improved by the development of a heuristic approach that aided the resolution of complex split and join constructs with multiple parallel edges (detailed in chapter 5 section 5.9).

It is likely that the mining of process logs containing noise became possible with the modified GP approach due to the modifications outlined in this section. The removal of the semantic edge markings helped to reduce an inherent level of noise in the process, created by not being able to represent single XOR edges. In addition the heuristics mentioned above (for the mutation operator and operator repair function)

contributed to the GP approaches ability to select viable tasks when modifying individuals. The overall effect was to assist the GP approach, in a non deterministic fashion, in an efficient search for a complete process model from event logs containing a number of incomplete and potentially erroneous traces.

Validation of the GP Approach

Two broad types of event logs were used to evaluate the GP process mining approach. There are a number of event logs and process templates available from literature. These have been artificially created with the aim to demonstrate common process constructs and semantics that occur in real industry based processes. These logs are balanced which means that they contain an equal number of traces for each path that can be taken through the process. The tests have been selected from a number of these artificial data sets as they cover a range of process constructs that the GP approach claims to deal with. In addition two real event logs have been provided for use in the project by the industrial sponsor. The event logs for these real processes are thought to be noise free though they are not guaranteed to be balanced. For a number of process logs a low level of noise was also introduced to test the GP approach's ability to deal with problematic process data.

The results obtained for the artificial event logs showed that the GP approach was capable of mining a range of process structures correctly, and that the approach was also able to resolve the semantics of many medium to complex processes. This was in part due to the heuristic modifications relating to the fitness parsing (detailed in chapter 6 section 6.8), mutation operator and the operator repair function (detailed in chapter 6 section 6.7). Complex parallel process structures and processes with a combination of linear and parallel constructs were successfully mined by the GP approach. Again the use of the parallel process heuristic (detailed in chapter 5 section 5.9) was valuable in the mining of such processes. The mining of complex parallel processes by the GP approach is in stark contrast to the GA approach of Alves de Medeiros (2006) which was not able to mine the two most complex parallel processes, test data sets 3 and 8, correctly on any of the runs (test data set 3 is detailed in chapters 5 and 6 and test data

set 8 is detailed in chapter 7) did prove to be more difficult for the GP approach. Simple semantics involving a limited number of edge node subsets could be mined correctly. However, more complex processes did present the occasional error due to edge node and/or subset duplication. This was most noticeable in the mining of the semantics of test data sets 3 and 8. The GA was more reliable when mining complex semantics, except in processes with a high level of parallel tasks (processes with four or more task edges emanating from a task). The results for the noisy event logs showed that the GP had some resistance to noise when mining simple processes with tasks removed from the top of the process only. It is likely that the modifications to the GP approach outlined in chapter 6 contributed towards this ability. However, like the GA approach, it was not able to mine complex processes perfectly. Again the reliable mining of semantics proved to be more of a challenge for the GP approach than for the GA approach when mining processes with tasks removed from the middle of the traces (that make up the event log of the process).

Potential Applications of the GP Process Mining Approach

From this chapter, it is clear that the GP process mining approach can be further developed in a variety of ways and in a number of directions. It is also true that this approach has wider applicability outside the area of process mining. In the opinion of the author, the GP approach could also be applied in the following areas –

- Scheduling activities.
- Data mining.
- Contextual databases.
- Customer profiling.

The GP approach could be adapted for use as a scheduling application, for example for the delivery of goods to a number of destinations. An initial population of solutions for a scheduling problem can be constructed as graph individuals. The operators could be modified to work on task nodes that represent destinations,

swapping destinations around and creating new links between destinations. The edges in such a system would carry a value representing the distance between destinations. Instead of a marking, system counter objects (Java software objects representing delivery vehicles within the GP approach, capable of marinating state on a particular vehicle) could be passed between the destination nodes. Each counter could have a number representing goods and/or the current loading of the vehicle. The fitness calculation and parsing for such a system would have to take into account the edge values (distances), the number of counters in the graph (vehicles) and perhaps the counter values (stock of goods).

The GP process mining approach lends itself well to the area of data mining. The approach is particularly suited to identifying structural patterns in data, for example the approach could be applied to the monitoring of web site usage, with sections of a site represented with task nodes and edges marked with the amount of visitors clicking between the sections. Popular patterns of browsing activity between the web page sections of a site could be found using a modified version of the GP approach.

Contextual databases, or databases containing objects tagged with metadata, could be mined using this approach. An example of such a database is a document flow system where electronic documents are stored and retrieved using their metadata description (the metadata is usually in the form of an XML string). For example, in terms of a mortgage application a range of documents are required. A modified version of the GP approach could build a flow diagram of the required documents noting when certain documents are missing and their current status (waiting for signatures or completion of extra sections). Based on the contextual metadata for each document different orderings of documents could be presented to users allowing for the paperwork to be prioritised correctly and streamlining the whole mortgage application process (bottlenecks in the application process could also be highlighted using this approach).

The area of customer profiling is a relatively new, and perhaps controversial, area of research. For example, In terms of supermarket shoppers with store loyalty cards, it

is possible to record their purchases over a period of time and build up a profile of their shopping habits. These individual profiles can be built up into an overall picture of shopping habits. Using the GP approach, it may be possible to build up graphs that map the types of products people buy with the layout of a supermarket. Task nodes could represent aisles in the store and edges the flow of shoppers. This way the store layouts could be fine tuned for the customer base that they serve. This would require knowledge of what products the aisles contain and an assumption of the way customers navigate between the aisles to select the products that end in their baskets. However, a new innovation in UK supermarkets is the use of self service checkouts. The customer has a hand held scanner and they are encouraged to scan items as they put them into their basket (so receiving a running cost of their shopping). The information collected from this device could give position and relative time of the effective 'purchase' of an item which may be used as parameters in the GP to give an accurate picture of the way customers actually navigate around a store. Store aisles may then be altered to avoid 'pinch points', where aisles may need to be widened or lengthened to accommodate more shoppers and product combinations may be changed to suit shopping habits that may be particular to a certain store.

These are just four of the potential wider users of the GP process mining approach, showing that a graph representation has particular benefits in the mapping of complex information management problems, and that the GP data mining activity has the potential to address a wide range issues in a more intuitive way.

Feedback gained during the completion of the industry survey, for this thesis (see chapter 4 for the full industry survey) suggests that while industry would benefit from such an approach as process mining, knowledge of process mining practice is still very limited. One potential limitation to the development of process mining approaches, and their acceptance by industry, is the lack of a standardised set of process templates that event logs may be generated from. A few test data event logs have been provided by authors such as Alves de Medeiros (2006) and Herbst (2001), and while the logs provide a good range of processes they do not constitute a definitive collection (Alves de Medeiros, 2006). Such a collection of processes would encompass a wider variety

of constructs and be modelled on patterns that appear in real business processes. However, the production of such a set of process data would require help from industry. The lack of a true graph representation in some process mining approaches may also have a negative impact on process mining results and the development and use of such techniques by industry.

It is clear from the author's investigations that more process mining research needs to be done with the collaboration of industry. This thesis has benefited from an industrial collaboration in the development of the GP process mining approach in terms of access to real process data and feedback on the approach from senior management. When enough industry partners are working with mining researchers, process mining as a practice will start to gain awareness in the business community and software solutions will be adopted by companies wanting to learn more about their processes.

7.1.2. Main Contributions

This research has made a number of contributions to the use of GP as a new process mining technique. In detail, the following contributions are most relevant –

- Provision of a GP process mining representation based on a graph construct (rather than an abstraction of a graph), allowing greater flexibility and the facility for intuitive development to accept future modifications.
- Development of functionality to aid the mining of complex parallel processes without compromising the stochastic search capabilities of the GP approach.
- Provision of a process mining technique with a level of resistance to noisy and unbalanced logs, which has been validated using a range of artificial and real data logs.
- Provision of an accurate and intuitive parsing technique that can maintain state of a process as it is being parsed and is not based on Petri net token parsing.

- Provision of heuristic techniques to aid the accuracy of process mining results and reduce the number of generations required to produce those results.

- Presentation of a survey of industry attitudes to process mining, examining their understanding and use of process mining techniques. Inclusion of a comprehensive review of literature covering key topics such as process modelling through to current process mining practice and the use of soft computing in this field. An analysis of the future direction of process mining detailing the key developments that are most likely to impact the development of new mining approaches and their acceptance in industry.

7.1.3. Limitations of the Research

There are a number of areas that require further research.

- The mining of loop constructs: The current GP technique does not deal with loop constructs in processes, though the suggestions detailed in the outlook section would act as a suitable future research direction.

- The accuracy in the mining of complex semantics: Some complex processes present semantics that are difficult for the GP approach to resolve correctly. The improvements detailed in the outlook section relating to the revision of the fitness function will help to reduce the incidence of multiple edge subsets in tasks which appear when mining complex split/join constructs.

- The range of process types and constructs the GP has been validated against could be wider. This is partly due to the types of event logs that are available. Artificial test data logs are limited in number and deal with the most common process constructs only. Ideally a wider range of real (perhaps event audited) process data could have been sort from a range of companies, though there are confidentiality issues in acquiring data from

industry.

- The resistance to noise of the GP approach could be further improved: This could be solved with the use of clustering with the GP approach, where the main features are identified and refined before recombination into an overall mining result (this modification is also detailed in the outlook section).

- The time taken to mine processes with the GP approach: Again this could be improved with the use of clustering with the GP technique (detailed in the outlook section).

7.1.4. Outlook

This thesis has presented an approach to business process mining based on the technique of Genetic Programming (GP). Evolutionary techniques are of increasing interest in areas such as data mining. The exploration of such techniques for process mining is gaining ground, as researchers look beyond the traditional algorithmic/deterministic approaches (such as provided by van der Aalst et al., (2002)) in this area. This section aims to give a view as to how the GP process mining approach can be further developed, and applied to a wider range of process mining problems.

A number of research directions exist for the further development of the process mining approach –

- Clustering and GP.

- Limitations of the fitness calculation.

- Flexibility of process mining practice.

- Graph visualisation.

- Hyper-heuristic mutation.

- Dealing with loops in GP.
- Use of a decaying rate of mutation.
- Problem feedback crossover.
- User interaction (detailed in Additional Research Directions paragraph)
- Multi-objective optimisation (detailed in Additional Research Directions paragraph)

The Use of Clustering with GP

One of the most important proposed modifications to the proposed GP process mining approach could be the use of clustering with GP. This would allow for improvement to the mining of a range of processes along with a reduction in the amount of time taken to mine even very large event logs. The approach may also allow for the identification and mining of process disparities, a functionality that is beneficial for a number of companies, especially in the world of banking and finance. This concept has been published as Turner and Tiwari (2008).

The use of clustering as a process mining technique has gained ground in recent years with several authors exploring this technique (Alves de Medeiros et al., 2007; Greco et al., 2006). Clustering has the advantage over evolutionary techniques at being much quicker at producing results. It is also capable of allowing for intermediate views of models part way through the mining process, so allowing experts to make manual adjustments if necessary. However models produced by some clustering approaches slightly over describe the event log and produce the so called spaghetti models (allowing for too many links between tasks).

The combination of a clustering technique with the GP approach could be an interesting research direction. It is proposed that the k-means algorithm (Witten and Frank, 2005) could be used as the clustering ‘engine’ and the following stages, as shown in Figure 89, used to mine event logs. In the work of Alves de Medeiros et al.

(2007a) feature selection is performed using an a priori approach (Agrawal and Srikant, 1994) to incrementally create features, based on an initial mined process model, and then check for their occurrence in the event log. In this way features can be identified from the model and the log checked to see if the features are possible. However, instead of using an a priori approach an initial run of the GP approach could be used to incrementally generate features from the models.

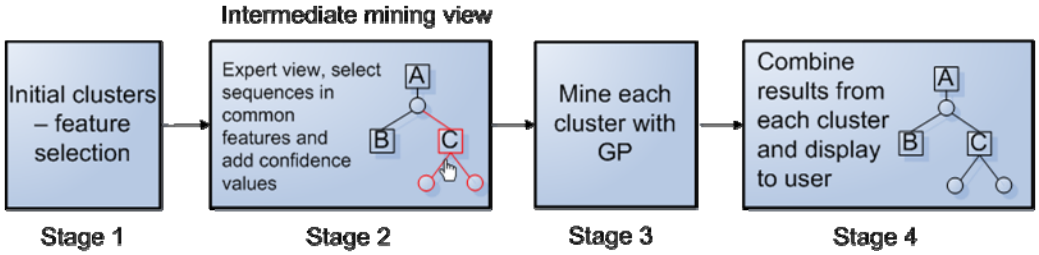


Figure 94: Proposed GP –Clustering Approach

In stage 1 of the approach, outlined in Figure 94, heuristics are used to build an initial set of individuals. The GP approach is then used to select a set of features from models in the initial population. Each feature is then used to form a cluster; event log traces and individuals are apportioned between the different clusters. In stage 2, an intermediate view of each of the fittest models for each cluster is shown so that an expert user may highlight certain parts of the model and assign a confidence level to the highlighted construct if required.

In stage 3 (Figure 94), each cluster is mined with the GP for a certain number of generations. The GP is equipped with a modified fitness function that is based around the ability of each individual to model the selected feature and the event log traces for that cluster. In addition, the GP operators act in a heuristic fashion to refine all of the individuals in each cluster. In stage 4 (Figure 94), the results from each cluster are combined into an overall process mining result.

In the mining of complex processes, there is often an advantage in introducing

expert knowledge in the interpretation of certain sub process patterns. It could be advantageous for an expert, through a visual interface, to select certain sequences in identified common features and ask the algorithm to look for and accept similar sequences in the event log data. In doing this, an expert may have a varying level of confidence in certain decision constructs and task edges (process task links) within a process. In the same way, the sequences that an expert does not hold so much confidence in may be negatively weighted. It is possible to conduct a re-clustering stage after the expert review is complete.

In summary, the advantages of a combined GP clustering approach over a standard GP process mining approach are –

- More refined mining of process features.
- Improvement in the mining of semantics (a problem area for GP and clustering when used independently).
- Accurate representation of low frequency traces.
- Efficient mining of large and complex processes.
- Reduced mining time.

Additional refinements to this approach are possible. In terms of the fitness evaluation of the GP, it is possible to reduce the amount of fitness evaluations undertaken in early generations (Andrews and Prager, 1994). In effect just a sample of individuals need to be fully evaluated for fitness in the early stages, with the rest checked just to see if they retain the process feature that they are being clustered by. The justification for this is that in the early stages individuals are likely to be less fit anyway, only after the completion of a number of generations are fitter individuals likely to emerge. This would reduce the amount of time taken for the approach to execute.

It is possible that the clustering modification would be a suitable vehicle for the development of a process disparity identification approach. Feature selection could be used to try and model any disparities within their own cluster. Instead of then merging

that cluster (or clusters) at the end of process mining, the resulting model could be displayed next to a model for the overall 'typical' process graph. Indeed the expert intervention stage of process mining could be used as a way of quantifying and displaying the early detection of process disparities to users.

Limitations of the Fitness Calculation

In terms of the fitness calculation a number of amendments are suggested for further experimentation in order to address certain limitations experienced with the GP approach. In the fitness function of Alves de Medeiros (2006), which has been adapted for use with the GP approach, the following modification may be appropriate for the issues experienced with some GP results (shown in Figure 95). The original equation currently used in the GP approach is detailed in chapter 4 section 4.4.4.

$$F_{itness}\ (L, In, In[]) = PFcomplete(L, In)$$
$$-\kappa * (PFprecise(L, In, In[])\ - subSetPenalty\ (L, In, In[]))$$

where

$$subSetPenalty(L, In, In[]) = \frac{allEnabledSubsets(L, In)}{\max(allEnabledSubsets(L, In[]))}$$

Figure 95: Modified Fitness Equation for Use in the GP Approach

The modification featured in Figure 95 is proposed to reduce the effect of the GP approach in producing duplicate and unnecessary subsets for individuals. A subSetPenalty function is added to the equation to add a weighting to individuals that express a higher than average amount of subsets (as shown in the third part of the equation in Figure 95). In practice, experimentation will be required to see if the subSetPenalty calculation needs to be turned into a ratio for application to the standard fitness function.

In addition, a similar calculation could be made regarding the number of edge nodes within the average subset and the average input and output set as a whole. In

the case of the GP approach, duplication of edge nodes and subsets tends to lead to over descriptive models (that is models that allow more behaviour, expressed as additional or incorrect task edges, than expressed in the event log).

In terms of parsing, once the aforementioned modifications have been successfully applied to the fitness function it may be possible to remove the additional rule that marked multiple output set subsets in order to reduce the effect of the GP approach in producing additional and unnecessary subsets. In essence, a one to one edge marking could be used as a replacement for this rule (for more information on this additional rule, as used in the GP approach, see chapter 5 and the end of section 5.4).

As an extension to the current research on the mining of business processes it is possible to investigate the area of process disparity. This occurs when several traces within an event log differ substantially from the remainder of the traces, pointing to the possibility of a very different interpretation of the eventual mined process model. Such disparities can potentially be dismissed to noise in the data, though on occasion such traces may point to an unexpected departure from normal process operation. The identification of such disparities may have implications for organisations such as –

- Financial institutions – for the detection of fraud through the identification of suspicious process execution traces.
- Call centres – to check if essential parts of a process are being bypassed.
- Online retailers – to analyse the ordering process a customer must complete in the purchase of goods and services and check for faults in the process.

This practice differs from current process mining approaches that aim to show only the ‘correct’ execution of a process, and ignore occasional traces that do not match the overall mined process.

Flexibility in Process Mining Practice

The use of flexible event logs, for process mining, would be beneficial. Such event logs could be tailored by industry for specific uses, with additional parameters that would be readable by process mining approaches through the use of an XML schema. Research work also needs to be undertaken with industry to construct a more complete set of test data event logs modelled on real process constructs, grouped by industry type and use.

Graph Visualisation

In the GP approach detailed in this thesis, the graphical representations provided are of process structures. The semantics view of a mined process is currently interpreted from the textual output produced by the approach (Petri net diagrams are inferred from the output). A development would be to automatically produce a semantic view of a process via the GP approach (the textual semantic output of the GP approach is shown in Figure 96). In order to do this, a semantic representation would need to be selected. Petri nets have been used in this thesis due to their prevalence in process mining literature. However, the use of an alternative representation format may be appropriate to gain acceptance from the end users of the process mining approach. The industry survey suggested that the knowledge and use of Petri nets within industry were limited. It is also the case that many of the business process management software vendors tend to concentrate on the Event Process Chain (EPC) format when representing process models with their software.

Petri net parsing (van der Aalst et al., 2005a; Alves de Medeiros, et al. 2005) is one way of converting semantic subset detail into a Petri net graph. Such an approach uses the parsing to work out the best way of linking tasks together and representing the AND/XOR splits and joins. However, if EPC is likely to gain wider acceptance with the business community this format should be used to provide semantic representations from processes mined with the GP approach. In converting an existing graph to an EPC the advice of Mendling (2008) on verification, metrics and guidelines for the

creation of correct EPC models, is relevant.

```
Task A

Input Set

{[START : A ]}

Output Set

{[A : C, A : B, A : D]}
```

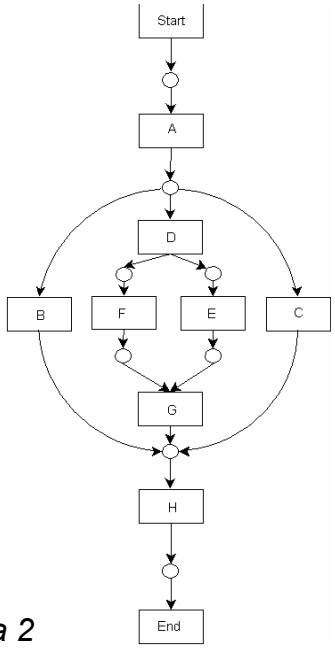


Figure 96: GP Text Output for Task A of Process Data 2

Hyper-heuristic Mutation

Preliminary investigations and development efforts were made, during the course of this thesis, into the application of a new area of evolutionary computing to further develop the GP mutation operator. The field of hyper-heuristics is relatively new. In order to explain hyper-heuristics, it is best to describe meta-heuristic techniques first. Meta-heuristics are techniques such as simulated annealing, genetic algorithms and genetic programming (Ozcan et al., 2008). Such techniques are said to be flexible enough to be applicable to a wide range of problem domains. Hyper-heuristic technique acts on sets of heuristics that are either provided as is or within a meta-heuristic technique, such as GP (Ozcan et al., 2008).

In terms of the GP approach, it is proposed that the hyper-heuristic technique could be used with the mutation operator in order to apply different mutation strategies (in effect heuristics) in an intelligent way.

Three mutation strategies are put forward for the mutation operator: –

- Add one task.
- Remove one task.
- Input/output set problem pairs (beneficial mutation of pairs of tasks)

The ‘add one task’ mutation operator adds a random task to a randomly selected subset. As in the current GP mutation operator, this can occur in the input or output set of any task within an individual. In a similar way, the ‘remove one task’ operator removes a task at random from a randomly selected subset. Both the add and remove task strategies may be heuristic in that problems recorded at the parsing stage can be used to influence the choice of tasks that are acted upon. The ‘input/output set problem pairs’ operator is a different mutation strategy in that it uses information collected about added tokens and tokens left behind to decide if there are connected pairs of problem tasks that may be mutated. A beneficial outcome may be achieved if both tasks are mutated at the same time. The strategy is capable of creating new XOR subsets or new single edge sets. Another difference in this approach to mutation is that more than one mutation strategy can be applied to task for any single mutation. The following rule is used to decide on the type of subset that needs to be created by this mutation strategy.

Added tokens penalise tasks with an XOR split that should be an AND split and tasks with an AND join that should be an XOR join. Similarly tokens left behind penalise tasks with an AND split that should be an XOR split and tasks with an XOR join that should be an AND join The hyper-heuristic mutation technique will at first allow for the random use of the mutation strategies, recording any penalties on a task to be mutated, the strategies applied to the task and the fitness of a mutated task after mutation. This database is then used, after a pre defined set of generations, to influence the choice of mutation strategies used on certain tasks depending on the penalty tokens that they carry. The strategies for one task in an individual could be found by looking at this database of mutation strategies to potentially select the best combination of mutation strategies.

In this way, the hyper-heuristic technique is able to look globally at a range of potential solutions to task-based problems and select the most appropriate combination for a task to be mutated.

Mining Loop Constructs with GP Process Mining Approach

One of the most challenging constructs to resolve correctly in process mining is that of the loop construct. In terms of the GP process mining approach detailed in this thesis, there is no recognition of loop constructs at present. This situation could change with the addition of a number of modifications to the parsing technique employed. In the work of Alves de Medeiros et al. (2004b), the notion of length one and length two loops are put forward (a length one loop involves two tasks, shown as the first illustration from the left in Figure 97; a length two loop involves three tasks, shown as the second illustration from the left in Figure 97) . Weijters and van der Aalst (2003) also put forward the recursive loop where a task may loop back to itself (shown as the third loop in Figure 97). In order to mine such loops (as shown in Figure 97) with the GP approach it is necessary to revise the way that the parsing works. At present, the parsing rejects edges that link back to tasks that are recorded later in the event log task sequence. This creates a problem when trying to identify all three of the aforementioned loop types.

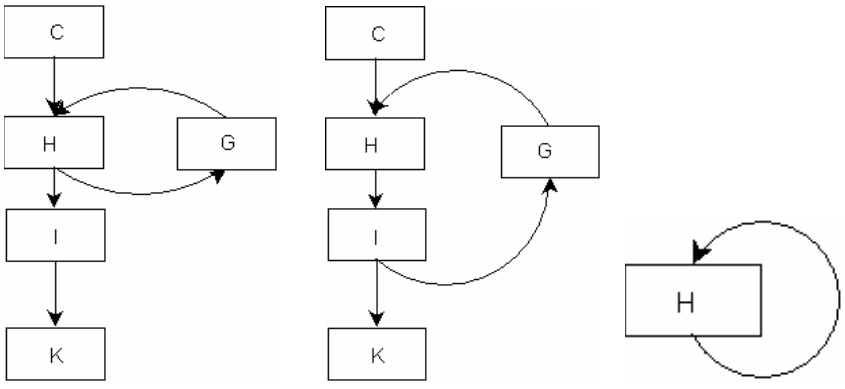


Figure 97: A Recursive Loop, Length One Loop and Length Two Loop (from Left to Right)

It is also the case that the nature of the task marking may also create possible confusion with the introduction of loop constructs. As a solution to this problem, it is proposed that further use is made of individuals from a population. In chapter 6, individuals from the initial population were used in order to aid the identification of tasks that were in parallel. A similar heuristic is proposed for the identification of the loop constructs outlined here.

It is proposed that a number of individuals from each generation could be assessed for loop constructs and the relevant tasks (involved in forming the loop) identified. Loops can be detected by checking the input set of each task for links to tasks that are later in the task sequence as recorded in the event log task sequence (a search can be performed through the event log for a number of traces that contain both tasks and a trace selected at random to act as the heuristic guide). Once a task has been found that is later in the task sequence, a series of backtracking steps can be taken to evaluate the extent of the loop and highlight the number of tasks involved. Alternatively, a more formal approach can be taken using an algorithm designed to find strongly connected components such as Tarjan's algorithm (Nuutila and Soisalon-Soininen, 1994) and its modified successors. Strongly connected components within a directed graph are components where the constituent vertices are connected in both directions (in effect forming a loop) (Tarjan, 1971). Tarjans algorithm provides a way of backtracking through a graph to identify strongly connected components. With such a formal approach, tasks that have edges back to later tasks in their input sets can be investigated to try and find strongly connected components in the GP individuals.

Once the loops have been located in a number of individuals, the task sets that comprise the loops can be compared to find a definitive set of loops. This information is important for the parsing stage as normal parsing activity can be suspended when a task is thought to be part of a loop construct (i.e. no penalties are added until it is clear that no loop exists). A special marking can then be used to indicate a cycle path through a graph. The cycle information can also be used to adjust the heuristics that depend on finding earlier and later tasks (for the mutation operator). When a task is

thought to be part of a cycle, the cycle information could be used or a random task selection made instead. A similar adjustment can also be made for the parallel task component. The issue of infinite loops and loop termination criteria, present in many computer science problems, does not arise in the parsing of cyclical process structures due to the nature of the event log traces. Each trace contains a finite number of tasks and the number of task iterations and loop termination is apparent from reading the trace.

Use of a Decaying Rate of Mutation

In early preparatory work for this thesis experiments were carried out with the GA approach of Alves de Medeiros (2006). In this work, the rate of mutation was altered based on a decaying rate and a number of process data sets were mined in order to ascertain the effect on the quality of mined models. This approach may be applicable to the GP approach, and could help with the accuracy of mined models. For more details on this set of experimentation, please look at Appendix A.

Problem Feedback Crossover

Again in early preparatory work for this thesis, an additional set of experiments were carried out with the GA approach of Alves de Medeiros (2006) involving the use of a problem feedback loop. The concept behind this research was to find out if problem tasks identified in an individual at parsing time, when fed back to the crossover operator as potential crossover points, could have a beneficial effect on the quality of mined process models. Like the decaying rate of mutation this approach may be applicable to the GP approach, and could help with the accuracy of mined models. For more details on this set of experimentation, please look at Appendix B.

Additional Research Directions

Further research could be undertaken into user interaction with process mining tools. This research could influence the design of Graphical User Interfaces (GUIs) for such applications and guide the level interactivity required and the look to the software.

Design lessons may also be learnt from existing Business Process Management tools. It is possible that future process mining applications will be required to perform multiobjective mining, in that there will be interest not just in the ordering of the tasks but in various parameters that may be associated with tasks such as cost, time and ownership. The use of multiobjective optimisation in process mining may required for such applications where a balance must be found between, potentially, competing objectives.

7.2. Conclusions

This thesis has detailed the development of a new business process mining approach that uses Genetic Programming (GP). The following points demonstrate how the objectives for this project, set out in chapter 1, have been met.

- Research has been carried out in the form of a literature review into the state of the art in business process mining. As part of the review, methods of modelling processes have been examined. A comprehensive assessment of the current techniques used to mine processes has been given in text and tabular form, including a comparison table matching techniques to process mining problems. The area of soft computing techniques and their relevance to process mining has also been explored. As a result of this review, it has been possible to identify a gap in the current research concerning the use of evolutionary techniques and in particular the use of Genetic Programming as a new way of mining event logs for process models. In addition it is clear that any new approach to process mining could benefit from an improved representation format based on an actual graph.

- A picture of industry understanding and views on process mining has also been established in a survey undertaken for this project. While knowledge of process mining is limited in industry, the concept of the automatic reconstruction of process models from data was welcomed and seen as a

useful tool. Process modelling practice was limited on the whole, with flowcharts being the main technique used. Most respondents did not review their processes on a regular basis. This survey has highlighted the need to communicate to and work with industry on process mining issues, an aim this research has met.

- A new business process mining approach has been developed using Genetic Programming (GP). The GP approach uses a graph representation and is able to mine a range of process structures (such as split and joins, linear and parallel constructs). The approach utilises a new marking scheme which can maintain the state of individual graphs as they are being parsed.

- The proposed GP business process mining approach has also been modified to mine complex parallel processes, allowing for the correct mining of semantics and the mining of event logs containing noise. Heuristics have been introduced to improve the mutation operator and the operator repair functions (called after the crossover or mutation has occurred to repair broken graphs). Heuristics have also been used to improve the fitness parsing allowing removal of edges that are known to be out of sequence. The removal of semantics from the edge node subsets of the tasks in the GP representation, in combination with the heuristics (outlined in this thesis) has led to an improvement in the mining results achieved in the original GP process mining approach. Heuristics in particular have allowed for the mining of event logs with noise due to the use of the event log when selecting tasks to become new edge nodes (with the mutation operator).

- The proposed approach has been validated using 8 event logs containing the descriptions of processes ranging from simple to complex structures (and semantics). To complement the test data event logs, two additional and real event logs (the data has been extracted from live running processes) have been provided by the industrial partner. The results show that the

proposed GP approach can mine structures, including complex parallel structures, reliably and render the semantics correctly for a range of event logs. The mining of correct process structures from event logs containing noise, with the GP approach, has also been demonstrated.

This research provides a solid foundation for further development of the GP approach as a business tool relevant to both the current and future business process mining needs of industry.

8. References

Adams, N. M., Hand, D. J. and Till, R. J. (2001). Mining for classes and patterns in behavioural data, *Journal of the operational research society,* vol. 52, no. 9, pp. 1017-1024.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules, in Bocca, J. B., Jarke, M. and Zaniolo, C. (eds.) *VLDB,* Morgan Kaufmann, San Francisco, pp. 487-499.

Agrawal, R., Gunopulos, D. and Leymann, F. (1998). Mining process models from workflow logs, in Schek, H. J. (ed.), *Proceedings of the 6th international conference on extending database technology: Advances in database technology,* Springer Verlag, Heidelberg, pp 469-490.

Aguilar-Saven, R. S. (2004). Business process modelling: Review and framework, *International journal of production economics,* vol. 90, no. 2, pp. 129-149.

Alves de Medeiros, A. K. (2006). *Genetic process mining*, PhD. Thesis, Eindhoven Technical University, Eindhoven, The Netherlands.

Alves de Medeiros, A. K., Guzzo, A., Greco, G., van der Aalst, W. M. P., Weijters, A. J. M. M., van Dongen, B. F. and Sacca, D. (2007a). Process mining based on Clustering: A quest for precision. In: *Business process management workshops 2007, Brisbane, Australia, 25th-28 September 2007*.

Alves de Medeiros, A. K., Pedrinaci, C., van der Aalst, W. M. P., Domingue, J., Song, M., Rozinat, A., Norton, B. and Cabral, L. (2007b). An outlook on semantic business process mining and monitoring, in Meersman, R., Tari, Z. and Herrero, P. (eds.), *OTM 2007 Workshops,* Springer Verlag, Heidelberg, pp. 1244-1255.

Alves de Medeiros, A. K., van der Aalst, W. M. P. and Weijters, A. J. M. M. (2003). Workflow mining: current status and future directions, in Meersman, R. E. A. (ed.), *CoopIS/DOA/ODBASE 2003,* Springer Verlag, Heidelberg, pp. 389-406.

Alves de Medeiros, A. K., van Dongen, B. F., van der Aalst, W. M. P. and Weijters, A. J. M. M. (2004c). Process mining for ubiquitous mobile systems: An overview and a concrete algorithm, in Baresi, L., Dustdar, S., Gall, H., et al (eds.), *Ubiquitous Mobile Information and Collaboration Systems (UMICS 2004),* Springer Verlag, Heidelberg, pp. 151-165.

Alves de Medeiros, A. K., van Dongen, B. F., van der Aalst, W. M. P. and Weijters, A. J. M. M. (2004b). *Process mining: Extending the a-algorithm to mine short loops,* BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven.

Alves de Medeiros, A. K., Weijters, A. J. M. M. and van der Aalst, W. M. P. (2007c). Genetic process mining: An experimental evaluation, *Data mining and knowledge discovery,* vol. 14, no. 2, pp. 245-304.

Alves de Medeiros, A. K., Weijters, A.J.M.M. and van der Aalst, W. M. P. (2005). Genetic process mining: A basic approach and its challenges. In: *Business Process Management Workshops (BPM),* Nancy, France, September 5, 2005.

Alves de Medeiros, A. K., Weijters, A. J. M. M. and van der Aalst, W. M. P. (2004a). *Using genetic algorithms to mine process models: Representation, operators and results,* BETA Working paper series, WP 124.

Amaravadi, C. S. and Lee, I. (2005). The dimensions of process knowledge, *Knowledge and process management,* vol. 12, no. 1, pp. 65-76.

Andersson, B., Bider, I., Johannesson, P. and Perjons, E. (2005). Towards a formal definition of goal-oriented business process patterns, *Business process management journal,* vol. 11, no. 6, pp. 650-662.

Andrews, M. and Prager, R. (1994). Genetic programming for the acquisition of double auction market strategies, in Kinnear, K. E. J. (ed.) *Advances in genetic programming,* The MIT Press, Cambridge, pp. 355-368.

Ashlock, D., Smucker, M. and Walker, J. (1999). Graph based genetic algorithms, *Congress on Evolutionary Computation (CEC99), vol. 2,* 6-9 July 1999, IEEE, Washington, DC, USA, pp. 1362.

Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice, Oxford, University Press, Oxford.

Banzhaf, W., Nordin, P., Keller, R. E. and Francone, F. D. (1998). *Genetic programming: An introduction,* Morgan Kaufmann, San Francisco.

Biermann, A. and Feldmann, J. (1972). On the synthesis of finite state machines from samples of their behaviour. IEEE transactions in computing, Vol. 21, No. 6, pp. 592-597.

Brace, I. (2004). *Questionnaire design: How to plan, structure and write survey material for effective market research*, Kogan Page , London.

Chen, K. C. W. and Yun, D. Y. Y. (2003). Discovering process models from execution history by graph matching, in Liu, J., Cheung, Y. and Yin, H. (eds.) *Intelligent Data Engineering and Automated Learning (IDEAL 2003),* Springer Verlag, Heidelberg, pp. 887-892.

Coad, P. and Yourdon, E. (1991). *Object oriented analysis,* Prentice Hall, Englewood Cliffs NJ.

Cook, J. E., Du, Z., Liu, C. and Wolf, A. L. (2004). Discovering models of behaviour for concurrent workflows, *Computers in industry,* vol. 53, no. 3, pp. 297-319.

Cook, J. E. and Wolf, A. L. (1998a). Discovering models of software processes from event-based data, *ACM transactions on software engineering and methodology,* vol. 7, no. 3, pp. 215-249.

Cook, J. E. and Wolf, A. L. (1998b). Event-based detection of concurrency. In: *6th International symposium on the foundations of software engineering,* Lake Buena Vista, Florida, ACM, New York, pp. 35-45.

Cox, E. (2005). *Fuzzy modelling and genetic algorithms for data mining and exploration,* Elsevier, San Francisco.

Daida, J.M., Hilss, A.M., Ward, D.J., Long, S.L., (2005). Visualising tree structures in genetic programming, *Genetic Programming and Evolvable Machines,* 6: pp.79-110.

Das, S. and Mozer, M. (1994). A unified gradient -descent/clustering architecture for finite state machine induction. In: *Proceedings of the 1993 conference*: *Morgan kaufmann advances in neural information processing systems*, vol. 6. Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 19-26.

Davenport, T. H. (1993). *Process innovation: Re-engineering work through information technology,* Harvard Business School, Boston.

Deb, K. (1995). *Optimisation for engineering design: Algorithms and examples,* Prentice Hall, New Delhi, India.

Dehnert, J. and Zimmermann, A. (2005). On the suitability of correctness criteria for business process models. In: van der Aalst, W. M. P., Benatallah, B., Caasati, F., et al (eds.), *Business Process Management (BPM 2005),* Springer Verlag, Heidelberg, pp. 386-391.

Dietz, J. L. G. and Habling, N. (2004). The notion of business process revisited. In: Meersman, R. and Tari, Z. (eds.), *On the move to meaningful internet systems 2004: CoopIS, DOA and ODBASE 2004,* Springer Verlag, Heidelberg, pp. 85-100.

Dong, G. and Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. In: *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining,* ACM, pp. 43-52.

Dumas, M., van der Aalst, W. M. P. and ter Hofstede, A. H. M. (eds.) (2005). *Process aware information systems,* Wiley, Hoboken, New Jersey.

Dustdar, S., Hoffmann, T. and van der Aalst, W. (2004). *Mining of ad-hoc business processes with teamlog*, Technical University of Vienna, Vienna.

Ferreira, D., Zacarias, M., Malheiros, M. and Ferreira, P. (2007). Approaching process mining with sequence clustering: Experiments and findings. In: Alonso, G., Dadam, P. and Rosemann, M. (eds.), *Business Process Management (BPM 2007),* Springer Verlag, Heidelberg, pp. 360-374.

Gaaloul, W. and Godart, C. (2005). Mining workflow recovery from event based logs. In: van der Aalst, W. M. P., Benatallah, B., Casati, F., et al (eds.), *Business Process Management (BPM 2005),* Springer Verlag, Heidelberg, pp. 169-185.

Garvin, G. A. (1998). The process of organisation and management, *Sloan management review,* vol. 39, no. 4, pp. 33-50.

Giaglis, G. M., Paul, R. J. and Doukidis, G. I. (1996). Simulation for intra-organisational business process modelling, in Charnes, J. M., Morrice, D. T., Brunner, D. T., et al (eds.), *Proceedings of the 1996 winter simulation conference,* ACM Press, New York, pp. 1297-1304.

Golani, M. and Pinter, S. (2003). Generating a process model from a process audit log. In: van der Aalst, W. M. P., ter Hofstede, A. H. M. and Weske, M. (eds.), *Business Process Management (BPM 2003),* Springer Verlag, Heidelberg, pp. 1020.

Goldberg, D. E., (1989). *Genetic algorithms: In search, optimization and machine learning*, Addison-Wesley, Wokingham, England.

Greco, G., Guzzo, A., Pontieri, L. and Sacca, D. (2006). Discovering expressive process models by clustering log traces, *IEEE Transactions on knowledge and data engineering,* vol. 18, no. 8, pp. 1010-1027.

Greco, G., Guzzo, A., Manco, G. and Sacca, D. (2005). Mining and reasoning on workflows, *IEEE transactions on knowledge and data engineering,* vol. 17, no. 4, pp. 519-534.

Greco, G., Guzzo, A., Pontieri, L. and Sacca, D. (2004). Mining expressive process models by clustering workflow traces. In: Dai, H., Srikant, R. and Zhang, C. (eds.), *Pacific-Asia conference on knowledge discovery and data mining 2004,* Springer Verlag, Heidelberg, pp. 52.

Gunther, C. W., Rozinat, A., van der Aalst, W. M. P. and van Uden, K. (2008). *Monitoring deployed application usage with process mining,* BPM 08-11, BPM Center Org.

Gunther, C. W. and van der Aalst, W. M. P. (2007). Fuzzy mining: Adaptive process simplification based on multi perspective metrics. In: Alonso, G., Dadam, P. and Rosemann, M. (eds.), *Business Process Management (BPM 2007),* Springer Verlag, Heidelberg, pp. 328-343.

Hammori, M., Herbst, J. and Kleiner, N. (2004). Interactive workflow mining. In: Desel, B. P. and Weske, M. (eds.), *Business Process Management (BPM 2004),* Springer Verlag, Heidelberg, pp. 211.

Harmon, P. (2003). *Business process change: A manager's guide to improving, redesigning, and automating processes,* Morgan Kaufmann, San Francisco.

Havey, M. (2005). *Essential Business Process Modelling,* O'Reilly, Sebastapol, CA.

Herbst, J. (2001). *Ein induktiver Ansatz zur Akquisition und Adaption von Workow-Modellen*, PhD Thesis, Universitat Ulm, Ulm, Germany.

Herbst, J. and Karagiannis, D., (2004). *Workflow mining with InWoLvE*, Computers in industry, vol. 53, no. 3, pp. 245-264.

Herbst, J. and Karagiannis, D. (1998). Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. In: *9th International Workshop on Database and Expert Systems Applications*, Vienna, Austria, 25-28 August 1998.

Hirasawa, K., Okubo, M., Katagiri, H., Hu, J. and Murata, J. (2001). Comparison between Genetic Network Programming (GNP) and Genetic Programming (GP) In: *Proceedings of the 2001 Congress on Evolutionary Computation (CEC), vol 2.* 27-30 May 2001, Seoul, South Korea, IEEE, pp. 1276.

Hornby, G. S. (2006). Shortcomings with using edge encodings to represent graph structures, *Genetic Programming and Evolvable Machines,* vol. 7, no. 3, pp. 231-252.

Hwang, S.Y., Wei, C.P. and Yang, W.S., (2004). *Discovery of temporal patterns from process instances*, Computers in industry, vol 53, no. 3, pp. 345-364.

Hwang, S. Y. and Yang, W. S. (2002). On the discovery of process models from their instances, *Decision support systems,* vol. 34, no. 1, pp. 41-57.

Iba, H., Sato, T., and de Garis, H. (1995). Numerical genetic programming for system identification. In Rosca, J.P., (ed.), Proceedings of the Workshop on Genetic Programming: From Theory to Real World applications, Tahoe City, CA, pp. 64-75.

Jablonski, S. (2000). Workflow management between formal theory and pragmatic approaches. In: van der Aalst, W. M. P., Desel, J. and Oberweis, A. (eds.) *Business process management - Models, techniques, and empirical studies,* Springer Verlag, Heidelberg, pp. 35-83.

Jensen, K. (1987). *Petri nets: Central models and their properties*, Springer Verlag Heidelberg.

JGraph (2009). *Citing Internet resources (WWW document).* http://www.jgraph.com/, (accessed 31st March 2009).

JGraphT (2009). *Citing Internet resources (WWW document).*http://www.jgrapht.org/, (accessed 31st March 2009).

Juhas, G., Lorenz, R. and Desel, J. (2005). Can I execute my scenario in your net, in Ciardo, G. and Darondeau, P. (eds.), *Applications and Theory of Petri Nets (ICATPN 2005),* Springer Verlag, Heidelberg, pp. 289.

Kim, C. H., Weston, R. H., Hodgson, A. and Lee, K. H. (2003). The complementary use of IDEF and UML modelling approaches, *Computers in industry,* vol. 50, no. 1, pp. 35-56.

Koza, J. R. (1994). *Genetic programming II: Automatic discovery of reusable programs,* The MIT Press, Cambridge.

Koza, J. R. (1992). *Genetic programming,* The MIT Press, Cambridge.

Kueng, P. and Kawalek, P. (1997). Goal-based business process models: Creation and evaluation, *Business process management,* vol. 3, no. 1, pp. 17-38.

Kumar, H., Joshi, A.H., Banka, K.K., and Rockett, P. I. (2008). Evolution of hyper-heuristics for the Biobjective 0/1 knapsack problem by multiobjective genetic programming, In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO),* 12–16 July 2008, Atlanta, USA, pp. 1227-1234.

Lamma, E., Mello, P., Riguzzi, F. and Storari, S. (2007). Applying inductive logic programming to process mining. In: Blockeel, H., Ramon, J., Shavlik, J., et al. (eds.) *Inductive Logic Programming,* Springer Verlag, Heidelberg.

Lee, L. L. (2005). Balancing business process with business practice for organisational advantage, *Journal of knowledge management,* vol. 9, no. 1, pp. 29-41.

Lewis, T., Pérez-Quiñones, M. A. and Rosson, M. B. (2004). A comprehensive analysis of object oriented design: Towards a measure of assessing design ability. In: *34th ASEE/IEEE frontiers in education conference,* IEEE, vol. 3, pp. S3H/16-S3H/21.

Li, C., Reichert, M. and Wombacher, A. (2008). Discovering reference process models by mining process variants. In: *Proceedings of 6th International Conference on Web Services (ICWS 08),* IEEE Computer Society Press, pp. 45-53.

Li, Y. and Feng, Y. Q. (2007). An automatic business process modelling method based on markov transition matrix in BPM, *Proceedings of the 2006 International Conference on Management Science and Engineering ICMSE,* pp. 46.

Lindsay, A., Downs, D. and Lunn, K. (2003). Business processes - Attempts to find a definition, *Information and software technology,* vol. 45, no. 15, pp. 1015-1019.

Majeed, H. and Ryan, C. (2006). Using context aware crossover to improve the performance of GP. In: *The Genetic and Evolutionary Computation Conference (GECCO) 2006,* July 8–12, 2006, Seattle, Washington, USA.

Majeed, H. and Ryan, C. (2007a). Context aware mutation: A modular, context aware mutation operator for genetic programming. In: *The Genetic and Evolutionary Computation Conference (GECCO) 2007, July 07-11, 2007, London, UK, pp. 1651-1658.*

Majeed, H. and Ryan, C. (2007b). On the constructiveness of context-aware crossover. In: *The Genetic and Evolutionary Computation Conference (GECCO) 2007, July 07-11, 2007, London, UK, pp. 1659-1666.*

Mannila, H. and Rusakov, D. (2001). Decomposition of event sequences into independent components. In: *First SIAM international conference on data mining*, SIAM, pp. 1-17.

Maruster, L., van der Aalst, W. M. P., Weijters, T., van den Bosch, A. and Daelemans, W. (2002a). Automated discovery of workflow logs from hospital data. In: *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*.

Maruster, L., Weijters, A. J. M. M., van der Aalst, W. M. P. and van den Bosch, A. (2002b). Process mining: Discovering direct successors in process logs. In: *Proceedings of the 5th International Conference on Discovery Science (Discovery Science 2002),* Springer Verlag, Berlin, pp. 364-373.

Melao, N. and Pidd, M. (2000). A conceptual framework for understanding business processes and business process modelling, *Information systems journal,* vol. 10, no. 2, pp. 105-129.

Mendling, J. (2008). *Metrics for process models: Empirical foundations of verification, error prediction, and guidelines for corrections,* Springer, Heidelberg.

Meyer, R. J., Menzel, C. P., Painter, M. K., deWitte, P. S., Blinn, T. and Perakath, B. (1995). *Information Integration for Concurrent Engineering (IICE) IDEF3 process description capture,* KBSI-IICE-90-STR-01-0592-02, KBSI, College Station.

Miller, J.F., (1999). An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach, In: *The Genetic and Evolutionary Computation Conference (GECCO)*, 1999, Orlando, Florida, USA, 14-17 July, pp. 1135-1142.

Miller, J. F. and Banzhaf, W. (2003). Evolving the program for a cell: From French flags to Boolean circuits, in Kumar, S. and Bentley, P. (eds.) *On Growth, Form and Computers,* Academic Press, New York, pp. 278-302.

Murata, T. (1989). Petri nets: Properties, analysis and applications, *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541 - 580.

Negnevitsky, M. (2005). *Artificial intelligence,* 2nd ed, Addison Wesley, Harlow.

Neiger, D. and Churilov, L. (2004). Goal-oriented business process modelling with EPCs and value-focused thinking, in Desel, J., Pernici, B. and Weske, M. (eds.), *Business Process Management (BPM) 2004,* Springer Verlag, Heidelberg, pp. 98-115.

Nikovski, D. and Baba, A. (2007). *Workflow trees for representation and mining of implicitly concurrent business processes,* TR2007-072, Mitsubishi Electric Research Laboratories, Cambridge.

Nonaka, I., Toyama, R. and Konno, N. (2000). SECI, Ba and leadership: A unified model of dynamic knowledge creation, *Long range planning,* vol. 33, no. 1, pp. 5-34.

Nuutila, E. and Soisalon - Soininen, E. (1994). On finding strongly connected components in a directed graph, *Information processing letters,* vol. 49, no. 1, pp. 9-14.

Ozcan, E., Burak, B. and Erkan Korkmaz, E. (2008). A comprehensive analysis of hyper-heuristics, *Intelligent data analysis,* vol. 12, pp. 3-23.

Pei, M., Goodman, E., Punch, W. and Ding, Y. (1995). Further research on feature selection and classification using genetic algorithms for classification and feature extraction, *Proceedings of the annual meeting of the classification society of North America,* Denver, Colorado.

Peleg, M., Soffer, P. and Ghattas, J. (2007). Mining process execution and outcomes. In: *Business process management workshops 2007*, Brisbane, Australia, September 24th-31st.

Phalp, K. and Shepperd, M. (2000). Qualitative analysis of static models of processes, *The journal of systems and software,* vol. 52, no. 2-3, pp. 105-112.

Quatrani, T. (2003). *Introduction to the unified modelling language*. Rational Rose, http://www.cs.nmt.edu/~cs328/reading/intro_rdn.pdf, (accessed 31st March 2009).

Regev, G., Alexander, I. F. and Wegmann, A. (2005). Modelling the regulative role of business processes with use and misuse cases, *Business process management journal,* vol. 11, no. 6, pp. 695-708.

Reijers, H. A. (2003). *Design and control of workflow processes,* Springer Verlag, Berlin.

Reijers, H. A. and Vanderfeesten, T. P. (2004). Cohesion and coupling metrics for work-flow process design. In: Desel, J., Pernici, B. and Weske, M. (eds.), *Business Process Management (BPM) 2004,* Springer Verlag, Heidelberg, pp. 290-305.

Ren, S., Zhang, X. D. and Zhang, X. P. (1997). A new generation of decision support systems for advanced manufacturing enterprises, *Journal of intelligent manufacturing,* vol. 8, no. 5, pp. 335-343.

Robson, C. (2002). *Real World Research: A resource for social scientists and practitioner researchers*, Blackwell, Oxford.

Rohloff, M. (1996). Reference model and object oriented approach for business process design and work-flow management. In: *Proceedings of the 1996 information systems conference of New Zealand,* New Zealand, Palmerston North, October 30th-31st.

Rozinat, A. and van der Aalst, W. M. P. (2006). Decision mining in ProM. In: Dustdar, S., Fiadeiro, J. L. and Sheth, A. (eds.), *Business Process Management (BPM 2006),* Springer Verlag, Heidelberg, pp. 420-425.

Rumbaugh, J., Jacobson, I. and Booch, G. (2005). *The unified modelling language reference manual,* Addison Wesley, London.

Sadiq, S. and Orlowska, M. E. (2000). On capturing exceptions in work-flow process models. In: Proceedings of the 4th international conference on business information systems, Springer Verlag, Heidelberg, pp. 3-19.

Scheer, A. W., Abolhassan, F., Jost, W. and Kirchmer, M. (2004). *Business process automation ARIS in practice,* Springer Verlag, Heidelberg.

Schimm, G. (2004). Mining exact models of concurrent workflows, *Computers in industry,* vol. 53, no. 3, pp. 265-281.

Schimm, G. (2003). Mining most specific workflow models from event-based data. In: van der Aalst, W. M. P., ter Hofstede, A. H. M. and Weske, M. (eds.), *Business Process Management (BPM) 2003,* Springer Verlag, Heidelberg, pp. 1021.

Sheth, A. (1997). From contemporary workflow process automation to adaptive and dynamic work activity coordination and collaboration. In: *NSF workshop on workflow and process automation,* IEEE, pp. 24.

Shirakawa, S., Ogino, S. and Nagao, T. (2007). Graph structured program evolution, *9th Annual Genetic and Evolutionary Computation Conference, GECCO 2007,* Jul

7-11, London, United Kingdom, Association for Computing Machinery (ACM), New York, NY, pp. 1686.

Smith, S. F. (1980). *A learning system based on genetic adaptive algorithms, PhD. Thesis,* University of Pittsburgh.

Tarjan, R. (1971). Depth-first search and linear graph algorithms, In: *Annual Symposium on Switching and Automata Theory (SWAT 1971),* 13-15 Oct., pp. 114-121.

Teller, A. (1996). Evolving programmers: The co-evolution of intelligent recombination operators. In: Angeline, P. J. and Kinnear, J., K.E. (eds.) *Advances in genetic programming,* The MIT Press, Cambridge, pp. 45-68.

Tsai, A., Wang, J., Tepfenhart, W. and Rosea, D. (2006). EPC workflow model to WIFA model conversion. In: *Systems, man and cybernetics, 2006, SMC '06', IEEE International Conference*, vol.4, 8–11 Oct., Taipei, pp.2758–2763.

Thurner, V. (1997). Business process modelling in software development. In: Tolvanen, J. P. and Winter, A. (eds.), *CAiSE'97 doctoral consortium, fachbericht informatik 14/97,* Universität Koblenz-Landau, Koblenz.

Tiwari, A., Turner, C. J. and Majeed, B. (2008). A review of business process mining: State of the art and future trends, *Business Process Management Journal (BPMJ),* vol. 14, no. 1, pp. 5-22.

Turner, C. J. and Tiwari, A. (2009). An exploration of genetic process mining, in Avineri, E., Köppen, M., Dahal, K., et al (eds.) *Applications of soft computing updating the state of the art,* Springer, Heidelberg, pp. 199-208.

Turner, C. J. and Tiwari, A. (2008). Process mining: A soft computing approach. In: *Multi-strand conference: Creating wealth through research and innovation (CMC 2008),* 6-7 May, Cranfield, UK, Cranfield University.

Turner, C. J. and Tiwari, A. (2007a). An experimental evaluation of genetic process mining. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO),* 7th – 11th July, London, pp. 2268.

Turner, C. J. and Tiwari, A. (2007b). An experimental evaluation of feedback loops in a business process mining genetic algorithm. In: *IEEE Congress on Evolutionary Computation (CEC),* 25th – 28th September 2007, Singapore, pp. 2679-2686.

Turner, C. J., Tiwari, A. and Mehnen, J. (2008). A genetic programming approach to business process mining. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO),* 12–16 July 2008, Atlanta, USA, pp. 1307-1314.

Turner, C. J., Vergidis, K. and Tiwari, A. (2007). Surveying business processes: The industry perspective. In: *Proceedings of the 5th International Conference on Manufacturing Research (ICMR-2007),* 11th – 13th September, Leicester, UK, Leicester University, Leicester, UK, pp. 313-317.

van der Aalst, W. M. P., Rubin, V., van Dongen, B. F., Kindler, E. and Gunther, C. W. (2006). *Process mining: A two step approach using transition systems and regions,* BPM-06-30, BPM Center, Eindhoven.

van der Aalst, W. M. P., Rubin, V., Verbeek, H. M. W., van Dongen, B. F., Kindler, E. and Gunther, C. W. (To be published 2009). Process mining: A two-step approach to balance between underfitting and overfitting, *Software and systems modelling*.

van der Aalst, W. M. P. (2005). Business alignment: using process mining as a tool for delta analysis and conformance testing, *Requirements engineering journal,* vol. 10, no. 3, pp. 198-211.

van der Aalst, W. M. P. (1999). Generic workflow models: How to handle dynamic change and capture management information. In: *Proceedings of 1999 IFCIS International Conference on Cooperative Information Systems, 1999. CoopIS '99,* IEEE, pp. 115.

van der Aalst, W. M. P. and Alves de Medeiros, A. K. (2005). Process mining and security: Detecting process executions and checking process conformance, *Electronic notes in theoretical computer science,* vol. 121, pp. 3-21.

van der Aalst, W. M. P., Alves de Medeiros, A. K. and Weijters, A. J. M. M. (2005a). Genetic process mining, In: Ciardo, G., Darondeau, P., (eds.) *Applications and theory of petri nets,* Springer Verlag, Heidelberg, pp. 48-69.

van der Aalst, W. M. P., de Beer, H. T. and van Dongen, B. F. (2005b). Process mining and verification of properties: An approach based on temporal logic. In: Meersman, R. and Tari, Z. (eds.) *On the move to meaningful internet systems 2005,* Springer Verlag, Heidelberg, pp. 130-149.

van der Aalst, W. M. P., Hofstede, A. H. M. and Weske, M. (2003b). Business process management: A survey. In: van der Aalst, W. M. P., ter Hofstede, A. H. M. and Weske, M. (eds.), *Business Process Management (BPM 2003),* vol. 2678, Springer Verlag, Heidelberg, pp. 1019.

van der Aalst, W. M. P. and Song, M. (2004). Mining social networks: Uncovering interaction patterns in business processes. In: Desel, J., Pernici, B. and Weske, M. (eds.), *Business Process Management (BPM 2004),* Springer Verlag, Berlin, pp. 244-260.

van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. J. M. M. (2003a). Workflow mining: A survey of issues and approaches, *Data & knowledge engineering,* vol. 47, no. 2, pp. 237-267.

van der Aalst, W. M. P. and van Hee, K. (eds.) (2002). *Workflow management models, methods, and systems,* MIT Press, London.

van der Aalst, W. M. P. and Weijters, A. J. M. M. (2004). Process mining: A research agenda, *Computers in industry,* vol. 53, no. 3, pp. 231-244.

van der Aalst, W. M. P., Weijters, A. J. M. M. and Maruster, L. (2004). Workflow mining: Discovering process models from event logs, *IEEE Transactions on knowledge and data engineering,* vol. 16, no. 9, pp. 1128-1142.

van der Aalst, W. M. P., Weijters, A. J. M. M. and Maruster, L. (2002). *Workflow mining: Which processes can be rediscovered?* Beta Working Paper WP74, Eindhoven University of Technology, Eindhoven.

van Dongen, B. F., Alves de Medeiros, A. K., Verbeek, H. M. W., Weijters, A. J. M. M. and van der Aalst, W. M. P. (2005b). The ProM Framework: A new era in process mining tool support. In: Ciardo, G. and Darondeau, P. (eds.), *26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005),* Springer Verlag, Heidelberg, pp. 444-454.

van Dongen, B. F. and Jansen-Vullers, M. H. (2005). *EPC verification in the ARIS for MySAP reference model database,* BETA Working Paper Series, WP 142, Department of Technology Management, Eindhoven University of Technology, Eindhoven.

van Dongen, B. F. and van der Aalst, W. M. P. (2005b). A meta model for process mining data, *Proceedings of the CAiSE'05 workshops*, Porto, Portugal, 13th-14th June, 2005. pp. 309-320.

van Dongen, B. F. and van der Aalst, W. M. P. (2004a). EMIT: A process mining tool. In: Cortadelle, J. and Reisig, W. (eds.), *25th International Conference on Applications and Theory of Petri Nets (ICATPN 2004),* Springer Verlag, Heidelberg. pp. 454-463.

van Dongen, B. F. and van der Aalst, W. M. P. (2004b). Multi-phase process mining: Building instance graphs, in Atzeni, P., Chu, W., Lu, H., et al (eds.), *Conceptual modelling - ER 2004,* Springer Verlag, Heidelberg, pp. 362-376.

van Dongen, B. F., van der Aalst, W. M. P. and Verbeek, H. M. W. (2005a). Verification of EPCs: Using reduction rules and petri nets, in Pastor, O. and e Cunha, J. F. (eds.), *17th International Conference on Advanced Information Systems Engineering (CAiSE 2005),* Springer Verlag, Heidelberg, pp. 372.

van Dongen, B. F. and van der Aalst, W. M. P. (2005a). Multi-Phase process mining: Aggregating instance graphs into EPCs and petri nets. In: *2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management (PNCWB) at the ICATPN 2005, pp. 35-58.*

van Hee, K., Sidorova, N., and Voorhoeve, M. (2003). Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In van der Aalst, W.M.P. and Best, E. (Eds.), *Application and Theory of Petri Nets 2003*, Springer-Verlag, Berlin, pp. 335-354.

Vergidis, K., Turner, C. J. and Tiwari, A. (2008). Business process perspectives: Theoretical developments vs. real-world practice, *International journal of production economics,* vol. 114, no. 1, pp. 91-104.

Wainwright, C. E. R. and Lee, P. G. (2000). Manufacturing planning within holistic strategic operations, *Journal of materials processing technology,* vol. 107, pp. 372-383.

Wainwright, C. E. R. and Ridgway, K. (1995). The application of GRAI as a framework for manufacturing strategy process. In: *Fourth international conference on factory 2000 - Advanced factory automation,* IEE, London, pp. 294.

Wang, J., Fan, Z., Terpenny, J. P. and Goodman, E. D. (2005). Knowledge interaction with genetic programming. In: Mechatronic systems design using bond graphs, *IEEE Transactions on systems, man and cybernetics part c: Applications and reviews,* vol. 35, no. 2, pp. 172-182.

Wang, E. A., Richter, A. and Cheng, B. H. C. (1997). Formalizing and integrating the dynamic model within OMT. In: *Proceedings of the 19th international conference on Software engineering,* ACM Press, Boston, pp. 45-55.

Wang, Y, Leung, H.K.N. (2001). A benchmark - based adaptable software process model, *Proceedings of 27th Euromicro conference., 4th Sept. - 6th Sept., Warsaw. pp. 216-224.*

Wang, J., Rosca, D., Tepfenhart, W. and Milewski, A. (2005). An intuitive formal approach to dynamic workflow modelling and analysis. In: van der Aalst, W. M. P., Benatallah, B., Caasati, F., et al (eds.), *Business Process Management (BPM 2005),* Springer Verlag, Heidelberg. pp. 137-152.

Weijters, A. J. M. M. and van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using little thumb, *Integral computer-aided engineering,* vol. 10, no. 2, pp. 151-162.

Weijters, A. J. M. M. and van der Aalst, W. M. P. (2001). Process mining: Discovering workflow models from event based data, in *Proceedings of the 13th*

Belgium-Netherlands conference on artificial intelligence, Amsterdam, The Netherlands, pp. 283-290.

Witten, I. H. and Frank, E. (2005). *Data mining: practical machine learning tools and techniques,* Elsevier, Amsterdam.

Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision processes, *IEEE transactions on systems, man and cybernetics*, vol. SMC-3, no. 1, pp. 28-44.

Zelm, M., Vernadat, F. B. and Kosanke, K. (1995). The CIMOSA business modelling process, *Computers in industry,* vol. 27, no. 2, pp. 123-142.

Zhang, S. H., Gu, N., Lian, J. X. and Li, S. H. (2003). Workflow process mining based on machine learning, *Proceedings of the second international conference on machine learning and cybernetics,* IEEE, vol. 4, pp. 2319- 2323.

[Page left intentionally blank]

9. Appendix A: Decaying Rate of Mutation Experiments

In early preparatory work for this thesis experiments were carried out with the GA approach of Alves de Medeiros (2006). In this work the rate of mutation was altered according to a decaying rate and a number of process data sets were mined in order to ascertain the effect on the quality of mined models. This work has been published as (Turner and Tiwari, 2009). Authors on evolutionary techniques, such as Cox (2005), point to the use of a decaying rate of mutation in genetic algorithms. It is claimed that such a function may help the genetic algorithm to converge on a solution sooner by gradually reducing genetic diversity, from a relatively high base, over the generations.

$$Pm = Pm \times min\left(\frac{l}{g^c}, 1\right) \qquad (1)$$

In equation 1 above the non-uniform decay algorithm of Cox (2005) is shown. The notation *Pm* represents the probability that an individual will mutate. The notation *l* represents the switchover limit, the mutation rate stays at its initial value until $g^c = l$ (*l* is set to 1 in this paper). The notation g^c is a generation counter. The algorithm for mutation is written as pseudo code in Figure 1.

```
MutationR = MutationR * min(1.0 / GenerationCount)
If MutationR < newRandomD Then
    If  MutationR > new(RandomD) then
        Case DeleteActivity if RandomD > 0.2
        Case  InsertActivity > 0.3
        Case RearrangeElements > 0.5
    End select
End if
Return Mutation choice
```

Figure 1: Implementation of the mutation function

In Figure 1 above the notation MutationR refers to the mutation rate calculation. The notation new(RandomD) indicates a function that returns RandomD as a new random number between 0.0 and 1.0. The case statement below MutationR shows the three types of mutation that can be carried out.

The choice of the values for the mutation experiment were influenced by Negnevitsky (Negnevitsky, 2005), who suggests the mutation probability for genetic algorithms is usually between 0.001 and 0.01. Cox (2005) also suggests the following formula for determining a suitable rate of mutation, shown below in equation 2.

$$m_r = \max\left(0.01, \frac{1}{N} \right) \qquad (2)$$

In equation 5 above m_r represents the probability of mutation or mutation rate. The notation N represents the population size. In the work of Alves de Medeiros (2006) the population size was 100, giving a mutation rate of 0.01 by the formula. Table 1 sets out the range of values that were used in the testing of the decaying rate of mutation.

Table 1: Decaying rate of mutation experiment values

	Mutation rate starting value		**Mutation rate starting value**
Test 1	0.1	**Test 8**	0.25
Test 2	0.15	**Test 9**	0.27
Test 3	0.18	**Test 10**	0.28
Test 4	0.2	**Test 11**	0.3
Test 5	0.21	**Test 12**	0.35
Test 6	0.22	**Test 13**	0.4
Test 7	0.24	**Test 14**	0.5

Table 2 details the process data sets that were used in this experimentation. The use of the decaying rate for the mutation operator was most noticeable in the results for process data 1. The use of the decay operator with mutation rate set initially between the values of .2 and .3 benefited the mining of the process model.

Table 2: Process data sets for use with decaying rate of mutation experiments

	Sequences	**Parallelism**	**Loops**
Process Data 1	Low	Low	Low
Process Data 2	Low	Low	Medium
Process Data 3	Medium	Medium	Low
Process Data 4	High	Low	Low
Process Data 5	Low	High	Low

The settings of 0.2, 0.21, 0.25, 0.27 and 0.28 allowed for the correct mining of the process model. This was an improvement over the results when utilising the standard non decaying rate of 0.2 used by (Alves de Medeiros, 2006), where the structure for process data 1 could not be resolved correctly (the correct structure for process data 1, as mined using the decaying rate, is shown in Figure 2). The use of the decaying rate was less noticeable for the remaining four process data sets.

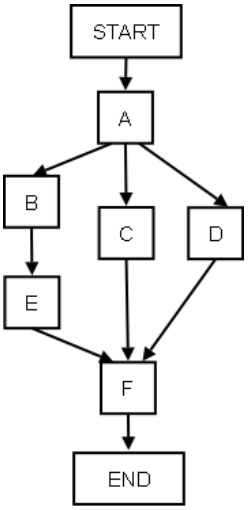


Figure 2: Correct structure for process data 1

Although the results obtained from this experimentation were exploratory in nature, it is possible that they may have a beneficial effect in the mining of processes with the GP approach. The use of the decaying rate was beneficial in the case of one process and did not negatively impact on the mining of the other processes. Again in the experimentation there was a concentration on mining structures rather than semantics, so it would be interesting to ascertain the potential benefit of the decaying rate in resolving the AND/XOR nature of process split and join points.

10. Appendix B: Problem Feedback Crossover

Again in early preparatory work for this thesis an additional set of experiments were carried out with the GA approach of Alves de Medeiros (2006) involving the use of a problem feedback loop. This work has been published as Turner and Tiwari (2007b).

The concept behind this research was to find out if problem tasks identified in an individual at parsing time, when fed back to the crossover operator as potential crossover points, could have a beneficial effect on the quality of mined process models. In this experimentation when a token is added against a task, due to a parsing problem (such as a missing edge etc.), that task is recorded in a log along with a count of the amount of tokens added against it.

A roulette wheel selection process is used to select a task that appears in both individuals to act as a crossover point. The pseudo code for the roulette wheel selection function is shown in Figure 1.

Figure 1: Pseudo code for the roulette wheel selection process

Select two individuals for crossover

rouletteWheel(Individual 1)

Identify which tasks in the individual have also been recorded as parsing problems – as Array2

populateWheel(Array2)

rouletteWheel.selectTask(Random)

Return task

Crossover individuals at task returned

The same five process data sets that were used for the decaying rate of mutation experimentation (detailed in the previous section of this chapter) were also used for the tests here. Two types of test carried out are of interest here –

- Roulette wheel selection with a problem feedback loop
- No feedback for first 500 generations then phased introduction of roulette wheel problem feedback

The roulette wheel problem feedback loop test used the wheel to select a crossover point based on the problems encountered in parsing process traces from the event log. The roulette wheel selection process is so called as each candidate for selection is given a proportionate amount of the circumference of the wheel depending on its frequency. The proportion of the wheel attributed to a candidate dictates its chances of being selected when the wheel is spun (Deb, 1995).

The phased introduction of the roulette wheel selection problem feedback loop means that for the first 500 generations the standard settings are used then the roulette wheel method of selection, with problem feedback, is gradually phased in for the next 100 generations to become the only method of deriving the crossover point. The concept behind the phased introduction of feedback loops is that the first 500 generations benefit from random crossover, so giving a greater variety of potential process models. The second 500 generations benefit from a gradual introduction of the feedback loops so process models at this point are crossed over on the basis of problem tasks. This approach is likely to benefit the mining of more complex process models, containing both high and low frequency task transitions.

For the phased introduction of roulette wheel selection technique, the following equation 1 was used for the phasing –

$$Rp = Rp \times (rate \times RgenIndex(genIndex - 500)) \qquad (1)$$

The structure of process data 1 could not be mined correctly using the standard crossover, with no problem feedback. However the use of the roulette feedback loop did allow for the correct mining of the structure for this process. In terms of the phased

introduction of roulette feedback loop this performed slightly better than standard settings though a number of errors remained in the semantics mined process model.

The results for the other four data sets showed that no particular improvement in process mining accuracy could be achieved with either version of the roulette wheel selection operator over the standard settings.

It must be noted with this experimentation that working out which task has had a token added against it is difficult with the Petri net based parsing system employed by Alves de Medeiros (2006). In effect parsing errors are mixed with a small percentage of potential errors, which may not be errors at all. In addition it is not always possible to work out the context of the problem using this parsing technique, for example which tasks are surrounding the problem task, is there a sub graph of tasks that are incorrect etc.

These results are potentially interesting and point towards the use of such feedback loops in the GP approach. The GP technique benefits from a more flexible representation, allowing for the identification of problem tasks and sub graphs containing problem graphs with a higher level of accuracy than possible with the GA approach. The advantage of being able to identify sub graphs is that the edges between problem tasks and other task may be examined and a mutation/crossover strategy arrived at to potentially correct the problem.

The concept of identifying sub graphs is also interesting in that this could be the basis for a new form of crossover that swaps actual sections of graphs (instead of acting on just the task subsets). This would mean that a new set of operator repair functions would need to be designed in order to account for sections of graphs changing not just the edges to and from one task. This concept can be combined with a recently developed technique for use with GP called context aware crossover (Majeed and Ryan, 2006). Context aware crossover allows for the intelligent the placement of sub-trees within an individual. In effect a number of viable locations within an individual are examined and the best selected before a swap takes place. Although this

technique has been developed for GP applications with tree representations it could be adapted for a graph representation as used by the GP approach of this thesis.

11. Appendix C: BT Event Logs Data Questionnaire

The aim of this questionnaire is to gain feedback on the results obtained by the GP process mining approach on test data sets 1 and 2 (provided by BT). The questions all refer to the structure of the two processes mined from data set 1 and 2.

For data set 1

Q1: Was the general order (sequence) of tasks correct?

Yes □ No □

If no, which tasks were in the incorrect order?

Q2: Were there any process constructs missing (e.g. missing parallel constructs, splits or joins)?

Yes □ No □

Q3: Are there any tasks missing from the process?

Yes □ No □

If no, which tasks are missing, where should they be placed?

Q4: Did the mined result differ from your current understanding of the live process?

Yes □ No □

If yes, how did it differ?

Q5: Did the result provide a constructive insight into your process?

Yes □ No □

If Yes or NO, in what way did it provide (or not provide) a constructive insight?

Q6: Does the mining result obtained provide a useful contribution to process management practice within your organisation?

For data set 2

Q1: Was the general order (sequence) of tasks correct?

Yes □ No □

If no, which tasks were in the incorrect order?

Q2: Were there any process constructs missing (e.g. missing parallel constructs, splits or joins)?

Yes □ No □

If no, which constructs were missing?

Q3: Are there any tasks missing from the process?

Yes □ No □

If no, which tasks are missing, where should they be placed?

Q4: Did the mined result differ from your current understanding of the live process?

Yes □ No □

If yes, how did it differ?

Q5: Did the result provide a constructive insight into your process?

Yes □ No □

If Yes or NO, in what way did it provide (or not provide) a constructive insight?

Q6: Does the mining result obtained provide a useful contribution to process management practice within your organisation?

12. Appendix D: Industry Survey Interview Questionnaire

Intelligent Decision Support & Optimisation for Process Mining and Conformance

Sponsored by: EPSRC (Grant No: EP/C54899X/1), British Telecom

Chris Turner

School of Applied Sciences,

Cranfield University, Cranfield,

Bedfordshire, MK43 0AL, UK.

Questionnaire for Interviews

Introduction

The aim of this project is to develop a Soft Computing based framework for capturing the business processes in an automated manner, optimising the process design interactively, and identifying the extent of disparity with the optimal in the continuous improvement process. The application scope of this generic framework is in the computer assisted business processes, especially in the Service and IT industries. Examples include order processing and fault handling.

Purpose of the Questionnaire

The purpose of this questionnaire is to identify the procedures and / or practices in industry for capturing the business processes, optimising the process design, and identifying the extent of disparity of an individual's/team's process with the recommended standard process. It will also capture the industrial requirements for an intelligent tool capable of supporting the above requirements.

Definition of a Business Process

The term Business Process has been defined as 'any set of activities performed by a business that is initiated by an event, transforms information, materials, or business commitments, and produces an output'.

Definition of Business Process Mining and Soft Computing Techniques

Business process mining allows for data stores of past process executions to be mined to produce graphical representation of processes. Soft computing is a consortium of methodologies (such as fuzzy logic, neural networks and evolutionary computation) that exploits the tolerance for imprecision, uncertainty, approximate reasoning and partial truth in order to mimic the remarkable human capability of making decisions in real-life ambiguous environments. Soft computing therefore helps in dealing with subjective/qualitative information and knowledge, and also helps in monitoring people's performance. It has become popular in developing systems that encapsulate human expertise.

Section A: Business Process Management & Modelling

This section of the questionnaire tries to understand how your company perceives business process. It also deals with business process management and modelling issues.

Q1. How does your company perceive/understand the concept of business process?

Q2. Who holds a holistic view/knowledge about each business process?

Q3. What is the most common approach when it comes to modelling/representing a business process?

Q4. Is there a particular methodology used (e.g. IDEF)?

Q5. Which would you name as the most common business process patterns/features (e.g. parallel flow, feedback loops, and decision points)?

Q6. Do you use business process management software within your organisation (such as SAP, ARIS or Tibco)?

Q7. Are there regular reviews of business processes within your organisation?

Q8. Do you use a particular methodology for the review?

Q9. What are the main steps of the review?

Section B: Business Process Mining

This section of the questionnaire aims to find out what process automation (and mining) activities your company undertakes.

Q10. What is the split between automated and manual processes within the organisation? Is the data from past business process executions retained?

Q11. Is the execution of business processes monitored?

Q12. Is data of past business executions retained?

Q13. Are process mining techniques employed in the organisation to extract actual processes from business process execution data? Which software tools are used for this task?

Q14. Which software tools are used to perform data mining tasks and why?

Q15. How do you identify 'bottleneck' problems in existing processes?

Section C: Business Process Analysis and Optimisation

(This section refers to work completed by the second researcher on the wider Intelli- process project (Intelligent Decision Support for Process Re-design and Conformance, EPSRC project EP/C54899X/1) of which this thesis is part)

Q16-Q22……

Section D: Industry Requirements for a Software Tool

This section of the questionnaire records your company's requirements for a process mining, optimisation and conformance tool

Q23. If a software tool was to be implemented, how would you describe its main functionalities to assist business process analysis and optimisation?

Q24. Do you require the ability to reconstruct actual process executions from stored process data?

Q25. Do you need an index or framework to quantify disparities between ideal template processes and actual business processes?

13. Appendix E: Industry Survey Internet Questionnaire

Questionnaire

Intelligent Decision Support for Process Re-design and Conformance (Intelli-Process Project)

Introduction: The purpose of this questionnaire is to identify the practice in industry for capturing the business processes, optimising the process design, and identifying the extent of disparity of an individual's/team's process with the recommended standard process. It will also capture the industrial requirements for an intelligent tool capable of supporting the above requirements. This work is part of the Intelli-Process Project, undertaken at Cranfield University. The aim of this project is to develop a Soft Computing based framework for the computer-assisted business processes, especially in the Service and IT industries.

Instructions: This questionnaire is divided into **six sections**. Please complete it in sequence and complete each section before progressing to the next one. When you have completed the questionnaire, please submit it via **email or fax** (contact details are provided at the bottom of this page).

Disclaimer: The answers provided here are confidential and only used for the

purposes of this research. A short report will be distributed to all those who took part in the survey.

Contact details:

Chris Turner

Manufacturing Department,

School of Applied Sciences,

Cranfield University, Cranfield,

Bedfordshire, MK43 0AL, UK.

SECTION A – Business process perception & modelling

Q1. Please select the phrase that reflects most appropriately the current practice within your company:

- ☐ The various departments work independently.
- ☐ Although there is departmental segmentation, there is an informal co-operation for certain processes (e.g. cross-functional teams)
- ☐ The company is organised around business processes
- ☐ Other, please specify in the box

Q2. Which of the definitions below is closest to your understanding of the term 'business processes':

- ☐ A dynamic ordering of work activities across time and place with a beginning, an end, and clearly identified inputs and outputs
- ☐ A set of logically related tasks performed by specific actors to achieve a defined business outcome.
- ☐ A construct with complex socio-technical interrelations
- ☐ Other, please specify in the box below

Q3. For the business processes that currently exist in your organisation:

- No one has explicit knowledge about the complex process flow
- There is a specific process owner who is responsible for each particular business process
- The process knowledge is shared among the main actors of the process
- Other, please specify in the box below:

Q4. Are there regular reviews of business processes within your organisation?

Yes No

If yes please provide details on the methodology used and the main steps:

Q5. What is the most common modelling/representation approach of a business processes used in your company?

□ Simple flowcharts with no predefined notation

Common practice □ Frequently Used □ Sometimes □ Rarely □ Never □

□ IDEF0 or IDEF3

Common practice □ Frequently Used □ Sometimes □ Rarely □ Never □

□ Petri-nets

Common practice □ Frequently Used □ Sometimes □ Rarely □ Never □

□ Other software assisted representation method

Common practice □ Frequently Used □ Sometimes □ Rarely □ Never □

□ Documentation only (text based description)

Common practice □ Frequently Used □ Sometimes □ Rarely □ Never □

□ Other, please specify in the box below:

Common practice □ Frequently Used □ Sometimes □ Rarely □ Never □

Q6. Please tick the main business process patterns that exist within your business processes: (more than one choice is allowed here)

□ Sequential flow (one task is executed after another)

□ Parallel flow (two or more tasks are executed in parallel)

□ Decision point (a point in the process where a decision about the flow is taken)

□ Feedback loop (a point in the process where the flow is directed on a previous activity)

□ Other process pattern (please specify in the box below)

Q7 . Do you use business process management software?

Yes □ No □

If yes, which one do you use?

□ ARIS

□ SAP

□ Tibco software

□ Other, please specify in the box below

SECTION B – Business process mining

Q8. What is the split between automated and manual processes within your organisation?

Q9. Is the execution of business processes monitored?

Yes □ No □

Any additional comments:

Q10. Is data of past business process executions retained?

Yes □ No □

Any additional comments:

Q11. Are process mining techniques employed in your organisation to extract actual processes from business process execution data?

Yes □ No □

If yes, what techniques are used?

Q12. Which software tools are used to perform data mining tasks and why?

Q13. How do you identify 'bottleneck' problems in existing processes?

SECTION C – Business process analysis & optimisation:

(This section refers to work completed by the second researcher on the wider Intelli- process project (Intelligent Decision Support for Process Re-design and Conformance, EPSRC project EP/C54899X/1) of which this thesis is part)

Q14-19…….

SECTION D – Industry requirements for a software tool:

Q20. If a software tool was to be implemented, how would you describe its main functionalities to assist business process analysis and optimisation?

Q21. Do you require the ability to mine actual process executions from data stores?

Yes □ No □

Q22. Do you need an index or framework to quantify disparities between ideal template processes and actual business processes?

Yes □ No □

Please use the space below to add any comments you might have, or further explain any business process related issues that were not covered by the questions above:

Thank you very much for your time