Knowledge-based SOA Framework for Improved Supply Chain Integration and Delivery

By

Paul Moynihan

Submitted to Victoria University in Fulfilment of the Degree of:

Master of Business

in the School of Information Systems

Faculty of Business and Law

November 2013

Abstract

Supply chains are everywhere that people do business, from a single market stall by the side of a road to a multinational corporation. Wherever there are goods and services that need to be used in a scheduled way to create a product that will be sold.

The problem that this thesis addresses is to manage a subset of supply chains; those that the participants of can be connected to electronically via the Internet, and to automatically optimize the flows between multiple supply chains. The methods used are a collection of heuristic rules for routing supply chain flow, a software engine to work through these rules and send messages to web services that encapsulate supply chain entities and a web based interface to view and measure the overall supply chain efficiencies.

The preliminary results show that some supply chains can be managed electronically, and it is possible to squeeze savings in time and money, by managing supply chain flow automatically, and that heuristic rules could work as well as the Simplex method in optimizing multiple supply chains.

Student Declaration

"I, Paul Moynihan declare that the Master by Research thesis entitled "Knowledge-based SOA Framework for Improved Supply Chain Integration and Delivery" is no more than 60,000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work".

Signature: [redacted] Date: 8 November 2013

Acknowledgements

A work such as this doesn’t happen in a vacuum. Many people besides the author made this work possible. First thanks to my principle supervisor Dr. Wei Dai for providing guidance at the right time, and enough financial support to keep my nostrils above the water line, and for standing in my corner when the powers that be were getting restless at the many delays.

Next thanks to my Mum for putting up with her son with his night-owl life style. You gave me a quiet place to work and think, and an Internet connection to gather knowledge. Plus thanks for being a great Mum.

To Catherine Lou who proposed the original problem, that I thought would only take two weeks to solve, but instead took three years. Thanks.

To Professor G. Michael McGrath my secondary supervisor thanks for kindly reading drafts of this work and making insightful comments and helping me to become a better writer.

To Professor Denise Emerson of the University of Florida USA a referee to one of my first papers, even though you don’t know me, thanks for telling me through voluminous criticism how to write a scientific paper. Also for taking the time when you probably thought it should have been binned.

To the administrators at Victoria University thank you for putting up with me for the many delays. I arrogantly thought I was going to finish ahead of time; instead, I finished way over time, how typical of me.

List of Publications

Refereed International Journal and International Conference Papers.

Dai, W., Moynihan, P., Gou, J., Zou, P., Yang, X., Chen, T. and Wan, X. (2007) ‘Services oriented knowledge-based supply chain application’, *Proceedings of the 2007 IEEE International Conference on Services Computing*, IEEE Computer Society, Los Alamitos, CA, USA, pp.660–667, available at http://www.conferences.computer.org/scc/2007/ (3rd place award contest team).

Moynihan, P. and Dai, W. (2010) ‘Knowledge-based service system for supply chain management’, *Proceedings of the 7th IEEE International Conference on Service Systems and Service Management*, June, pp.591–596, IEEE Press, Tokyo, Japan.

Moynihan, P. and Dai, W. (2012) ‘HetNet framework for supply chain management’, *15th International conference on Network-based Information Systems*, 26-28th September 2012, City Campus, La Trobe University.

Moynihan, P. and Dai, W. (2013), ‘Experience in Developing Concurrent Inferences on a De-regulated Supply Chain Network’, in *AINA-2013 27th IEEE International Conference on Advanced Information Networking and Applications Workshops*. In print.

Moynihan, P. and Dai, W. (2013), ‘"A Rule based Service Framework for Supply Chain Management".’, *in International Journal of High Performance Computing and Networking*. In print.

Table of Contents

Table of Figures

1. CHAPTER ONE – INTRODUCTION

1.1 Overview

This chapter is really an overview of the proposed research and covers the following topics. The background to the research, what is being attempted and why there is the need for this research. The contribution to human knowledge this research will make. Also as this research has a strong practical component this chapter elucidates the practical contribution to human methodology in a particular field of endeavour.

This chapter also includes a summary of previous research and a gap in knowledge that this current research attempts to fill. The last half of the chapter is the research design. This has the following subsections. The research methodology which includes top level design diagrams. How the research will be conducted in more detail and how the results will be compared to the alternative approach and how the data will be gathered for the optimization study. A final topic of the research design will be an overview of the implementation of the system. The chapter concludes with a description of the thesis structure.

1.2 Background

A perennial problem with running modern e-business systems is to manage supply chains cost-effectively. To get the maximum efficiency, a higher degree of automation is required. This research investigates steps and processes towards automated supply chain integration and improved coordination of delivery channels in a services oriented e-business. It will show under such an approach potential cost and time savings, as well as an intelligent coordination strategy utilizing existing resources that could be delivered in a real-time fashion.

Additionally, an empirical study using the research prototype to be delivered from this research will be undertaken to show the difference between combinatorial and heuristically guided rules in optimization of supply chains. The prototype will use Service Oriented Architecture (SOA)[1] techniques to initialize, coordinate and monitor supply chains.

Previous research relating to the proposed research target was done from various aspects: empirical study of the benefits of SOA for integration of electronic supply chains Kumar et al. (2007). An electronic hub (e-hub) approach as a coordination mechanism to monitor through the use linear programming models to optimize multiple supply chains was proposed in Fayazbakhsh et al. (2008). A knowledge-based approach using software agents to manage dynamic configuration of multiple supply chains was carried out in Emerson et al. (2004). In answering the proposed research targets, none of the previous work is sufficient as a solution alone.

The main research question to be addressed is an investigation of any differences between combinatorial and heuristically guided rules to improve supply chains coordination. To my knowledge the literature doesn’t have any studies dealing with this issue. Most researchers just assume that standard global optimization techniques will be used. But heuristically guided optimization may perform better at least in some circumstances, and may also be quicker, and may make real time optimization a reality. Additionally, to investigate this question a framework is needed, using SOA, to investigate automated supply chain integration and improve supply chains delivery.

[1] Service Oriented Architecture (SOA) is a software architectural approach where the basic element of design and development is a service. In SOA, applications communicate with each other in such architectures through services.

Typically in the past optimization has been done using mathematical techniques to set up the basic optimization strategy, which then requires human heuristic "tweaking" to achieve optimum efficiency.

The mathematical techniques are limited to relatively small numbers of variables due to the exponential increase in time for a solution; this may not be suitable for a real time response to changes in supply chain events.

The industry benefits of this research are twofold: that heuristic optimization rules could be faster and more useful than standard optimization techniques for real time management of multiple supply chains and SOA technology is really beneficial for remote management of electronic supply chains.

The practical benefits this research could show are that more automation than has hitherto been utilized to maintain and run at peak efficiency multiple supply chains.

Recent developments in e-business and e-commerce applications and faster communication over the internet in general can allow the dynamic re-configuration of supply chains over time to take advantage of better configurations. In such configurations, each business entity in the supply chain can be seen as an agent who makes independent decisions based on information gathered from the next level upstream.

Automated supply chain reconfiguration is beneficial when there are changes in the cost of products or services, resource availability, and customer demands. This assumes that for a given order there are several feasible supply chain configurations that can deliver the order. The number of such feasible configurations increases with the number of stages, products, suppliers, etc. It could be impractical to manually evaluate every feasible configuration in the context of a modern high-speed web-based order processing system (for example amazon.com). The aim of this research is to investigate real time optimisation and dynamic configuration of multiple supply chains. This will lead to more efficiency in production flow and savings in cost, benefiting both lean and agile supply chain strategies by providing the infrastructure to quickly alter the configuration of multiple supply chains leading to quicker set up times for new product lines and avoidance of waste for stable product lines. The research aims to address the following key issue:

- A knowledge-based control of conventional resources (such as existing optimisation techniques) within SOA (Service Oriented Architecture) for improved supply chain configuration.

There are many existing works on optimisation techniques using different approaches, particularly for manufacturing. The proposed research will provide an alternative approach at a meta-level that offers an effective and user friendly application of existing techniques for the problem at hand.

A number of associated features include agility (i.e., adaptive for various supply chain models), and timely response (real-time or just-in-time) to arising business demands.

Agile supply chain management

Both lean and agile supply chain models can benefit from this research. Lean is running a supply chain as optimised as possible, i.e., for a supply chain that doesn’t change often. Agile is running a supply chain that changes frequently without being much optimised. The response can be up to the second or on demand. Some managers like to have some breathing space before they have to make a decision, others like to know what’s happening at all times, so that a quick response can avert a possible disaster.

1.3 Contribution

The academic contribution of this research is twofold: first to answer the question could heuristic optimization rules be faster and more useful than standard optimization techniques for real time management of multiple supply chains, and second is SOA technology really beneficial for remote management of electronic supply chains.

Electronic supply chain management is becoming essential for any modern enterprise business model. I have identified two areas where I believe there is a need for more research. First: an alternative to standard supply chain optimization methods and two: more research into the usefulness of SOA for remote management of supply chains.

With more research in these two areas it may enable real time optimization and configuring of multiple supply chains. This will lead to greater efficiency in production flow and savings in cost, benefiting "lean" supply chain strategies. It will also benefit "agile" supply chain strategies by providing the infrastructure to quickly alter the configuration of multiple supply chains leading to quicker setup time for new product lines.

1.3.1 Practical Contribution

The practical benefits that could come from this research is that more automation than has previous been utilized to maintain and run at peak efficiency multiple supply chains. This will benefit industry initially, through saving companies money and enabling a wider diversification of products, thus enabling greater competitiveness. This will generate greater wealth for society as a whole.

It is anticipated that the proposed research will show that heuristic optimization rules will be quicker than standard optimization techniques in optimizing multiple supply chains but probably not as optimal as the standard techniques. This may enable real time optimization. I also anticipate that SOA will be an enabling technology helping to coordinate information about supply chain entities in real time.

The research aims to address the following key issues:

- A knowledge-based control of conventional resources (such as existing optimization techniques) within SOA for improved supply chain coordination.
- A number of associated features include agility (adaptive for various supply chain models), timely (real-time or just-in-time) response to business demands.
- Flexible engagement with participating business entities.

Flexible engagement with participating business entities.

Some supply chain stake holders may be reluctant to want to participate, out of fear that they may be optimized out of the supply chain flow, due to their unpalatable pricing. The proposed research will argue that buyers pick and choose anyway, and unless you're locked in to a long term contract, people will always look to the cheapest supplier regardless whether it is done with paper or electronically. That said the proposed solution would also be useful to enterprises that have recently acquired new supply chain assets and wish to better integrate and optimize the flow along the new assets and the old assets.

1.3.2 Summary of previous research and gap identified

Electronic management of supply chains has been maturing for the last 20 years. The pervasiveness of the Internet has made supply chain accessibility cheap and easy. It is time now for more automation to be part of the business landscape. The literature reveals various attempts along this route. Multiple supply chains are a fact of life these days, with competition between similar enterprises being decided on supply chain efficiencies. The gap in knowledge the author has identified is: that there has been little research into faster and more flexible supply chain optimization strategies, across multiple supply chains. To support this research there is a need for a model to investigate the cost time benefits of different supply chain optimization strategies. In addition a more thorough use of SOA to provide the glue for tighter integration of multiple heterogeneous supply chains is required.

1.4 Research Design

This proposed research draws on the resources from three different areas: Services Oriented Architecture (Kumar et al. 2007; Lim et al. 2003; Papazoglou et al. 2003), Knowledge Management (Emerson et al. 2004; Fayaz et al. 2005) and Resource Optimization (Bowersox et al. 1999; Mason et al. 2000; Fayaz et al. 2005). The author envisages a potential synergy between these three areas of knowledge that will enable an implemented framework that will demonstrate automated supply chain management and optimization. A conceptual diagram for the proposed research is shown in Figure 1 where the resources are in the ovals and the research challenges and questions are in the rectangles.

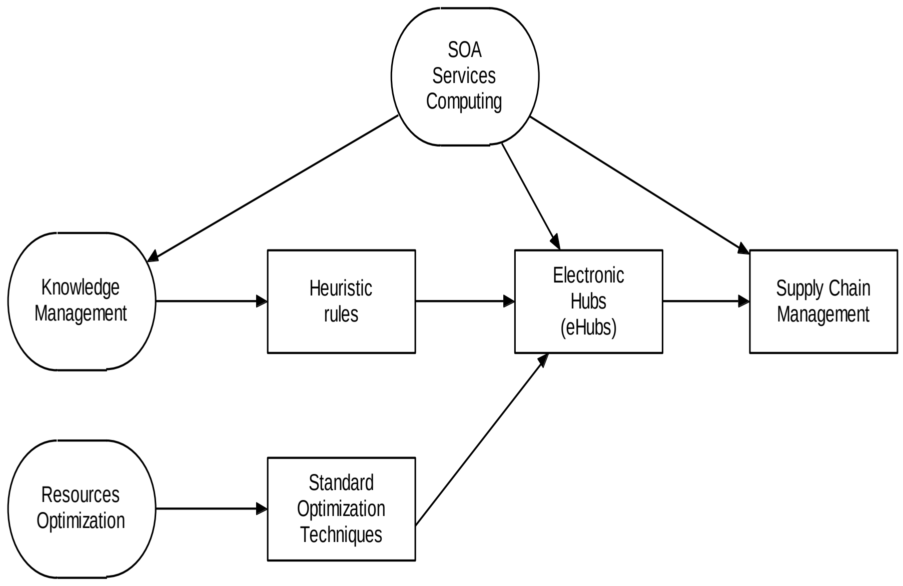


Figure 1 Conceptual Framework

1.4.1 Methodology

The methodology for this research is first to create a framework that will be able to answer two questions: Is it possible to efficiently manage multiple supply chains electronically? Will heuristically guided optimization rules perform better than standard global optimization algorithms in optimizing supply chain flows?

The top level design is shown in figure 2 it shows the research approach and methodology. This diagram says that to create a framework to answer the above questions, certain domains of knowledge are required to plan and create the components of the framework. The knowledge domains are optimization theory, services computing, supply chain and knowledge management. These are used to create the following components of the framework. A rule based reasoner, web services infrastructure, supply chain web service entities and standard resource optimization techniques. The framework once built has two execution pathways, a set of heuristic rules in the rule database and a simplex execution module. These execution pathways generate two sets of data which are collated into a database table. This table comprises the raw data for analysis.

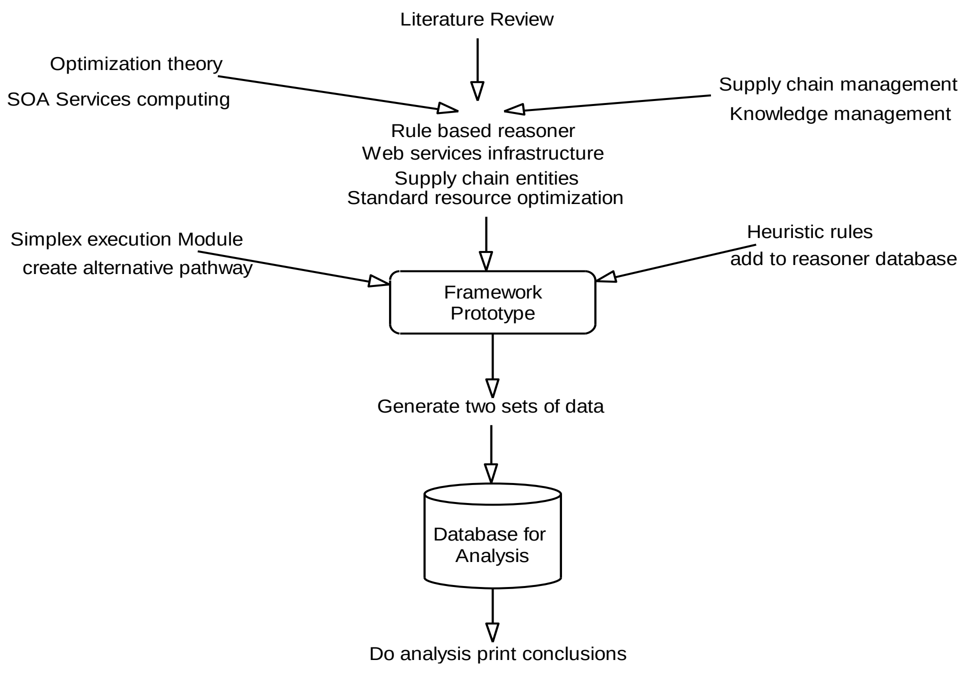


Figure 2 Research Design Overview

To answer the second question a small empirical study will be done using the new framework. Using the new framework we will compare the savings in cost and time of fulfilling some randomly generated retail orders from three different retailers who are members of three separate supply chains under a common trust umbrella. Each supply chain has a retailer, distributor, manufacturer and three suppliers. Real world supply chain data is hard to come by. However, I have managed to locate manufacturer-to-retail supply chain data from both the literature and industry sources that has enabled me to construct 'typical', generic supply chains that approximate real-world situations (Shi 2007, Yaxin 2007). The data sources are such that I should be able to argue that my results will have fairly wide external validity.

The null hypothesis for this study is that there is no difference between the results returned by the heuristic optimization rules and the industry standard Simplex method as regards cost and time savings for fulfilling a number of random orders. The model will be allowed to generate enough orders such that differences if any between the two methods (heuristic versus Simplex) will have a 1% probability of being due to chance alone. I will use some of the tools from the open source statistics package "Dataplot" to help me calculate sample sizes to give a 99% confidence interval and to help me analyze the returned data. I will test data for normality and if confident a normal distribution will apply parametric tests "t test" for similarity between means and "F test" for similarity between standard deviations.

Under each run of the model the supply chain entities from each supply chain will be queried via Web services and their collective response will be optimized using custom hand created optimization rules and also optimized using the industry standard Simplex method. The comparison between these two methods will be returned via a web page.

1.4.2 How Research will be conducted in detail

I will now describe in more detail how the research will be conducted. The Framework as stated above will try to answer two questions: the first question is it possible to manage supply chains electronically? To answer this question I will create a model of some supply chains. The supply chains will be represented by collections of web services. Each web service will be able to be queried as to its current status and be able to return on request a random number within a certain range (price in dollars, or time in days) appropriate to the supply chain artifact that the web service models. Though the web services are hosted locally for convenience, they would operate the same if distributed across the internet, only the URL for each service would be different. There are six services (retailer, distributor, manufacturer and three suppliers) for each of three supply chains, eighteen services in all. The full framework is 100% complete and preliminary results (Moynihan and Dai, 2010) show that it would be possible to monitor and communicate with various supply chain entities and thus manage multiple remote supply chains electronically.

Concerning the second question, will heuristically guided optimization rules perform better than standard global optimization algorithms in optimizing supply chain flows? I have added a heuristically guided optimization module to the Framework and it optimizes supply chain flows and produces time and cost savings for each retail request, for each of the supply chains, compared with non-optimized flows.

I have also added the traditional simplex optimization module to the framework. It also produces time and cost savings for each retail request compared with non-optimized flows. It operates differently from the heuristic rule engine and uses constraint equations to guide its iteration to a globally optimized state.

The Framework integrates two very different techniques - Heuristic and Simplex - and JShop planning techniques will be added later. These can be applied and controlled by the rule based inference engine. My technology makes extensive use of and relies on functionality exposed as services. This allows integration, monitoring and control to be easier.

1.4.3 How the results will be compared to traditional optimization

In comparing the two different systems, a bit of thought is needed, as the two systems operate differently. Though the same cost function is used to evaluate the cost and time of fulfilling the randomly generated retail orders, the heuristic method uses the values of supplier costs, production rate, and distributor stock quantity returned from the supply chain web services to decide which rules will fire from a small database of production rules. A sample heuristic equation is the following:

If a large retail order, low distributor stock, and low production rate (this produces a large time to fulfill order.)

Then choose a supply chain with the smallest retail time, and after completing its current order, to transfer its production stream to the supply chain with the largest retail time.

$$New\ Retail\ Time\ = \frac{PqMax\ \times\ RtMax\ +\ PqMin\ \times\ RtMin}{PqMax\ +\ PqMin}$$

PqMax = Production rate for chain with largest retail time.

RtMax = Retail time for chain with largest retail time.

PqMin = Production rate for chain with smallest retail time.

RtMin = Retail time for chain with smallest retail time.

The action parts of the applicable rules will direct the flows between the different supply chains. As well as directing the supply chain flow the rules will alter the values of the variables in the cost function, which will generate improved costs and times, which are then compared to the non-optimized costs and times.

The simplex method will use the same data but the data will be inserted as coefficients into the decision variables, the constraint equations will model the thresholds for the routing interactions between the different supply chains (similar to the heuristic rules). The constraint equations are then converted into a standard form using additional slack variables (to convert inequalities to equalities) suitable for adding to a two dimensional array called a tableau (see figure 10) certain matrix operations are performed iteratively to the tableau, which creates an adjustment to the variables of the cost function, to create an optimal minimum cost function. See Reveliotis, *(2003)*. The output of the new cost function is compared to the non-optimized cost and times.

1.4.4 How the data will be obtained for the optimization study

It is anticipated that the artificial data will be used through a data generator. After study of a domain of choice, parameters from the data generator could be adjusted or tailored to match the real-world situation. The framework will have extra code added to enable it to run in a loop a certain number of times, each time calling the web services and generating random supply chain data, then calculating the non-optimized then calculating the optimized costs and times (both traditional and heuristic) then comparing them to the non-optimized costs and times and storing the type (both traditional and heuristic) and the difference into an array. This array of data can be analyzed using “Dataplot” or even Excel or “R” to determine if the null hypothesis (that there is no difference between the two optimization methods) can be rejected at the 99% confidence level.

1.4.5 Implementation

Figure 3 shows a more detailed view of the implementation, the application is written in Enterprise level Java. It consists of a Servlet hosted in the Tomcat Servlet container to act as the interface to the Internet.

The Optimization and Control module talks to a rule-based Inference engine and to a web services interface. The web services interface talks to the external web services of the various supply chain entities. The rule engine has rules to control how the optimization and control module should behave based on the data returned from the supply chain entities. Figure 9 shows the inference engine and support classes in more detail.

The Optimization and Control module also sends instructions to the supply chain entities directing them to route supply chain flow to specific individual entities. The heuristically guided rules derive their power from optimizing across the multiple supply chains rather than optimizing within a single chain see Figures 12 and 13.

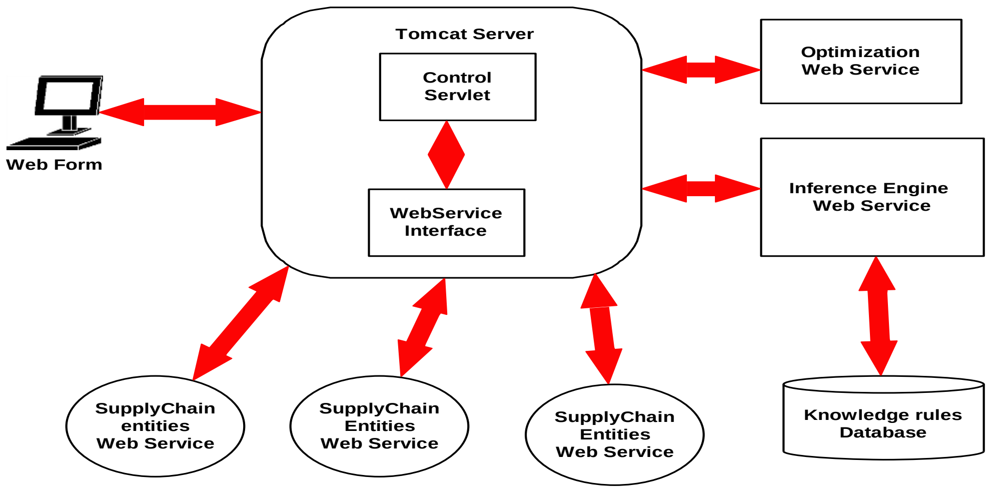

Tomcat Server
Control Servlet
WebService Interface
Web Form
Optimization Web Service
Inference Engine Web Service
SupplyChain entities Web Service
SupplyChain Entities Web Service
SupplyChain Entities Web Service
Knowledge rules Database

Figure 3 Detailed view of the implementation

1.5 Thesis Structure

This thesis consists of six chapters and five appendices. The first chapter gives the background, the contribution to knowledge and the research design. The second chapter is a literature review and discusses the previous work in this field relevant to my research. The third chapter details the system, how it is built and how it works. The fourth chapter is about the evaluation of the system, how it compares with an alternative optimization approach, including data generation and analysis. The fifth chapter is a discussion of and conclusion to the research, including limitations and future research and improvements. There are five appendices: appendix A is a synopsis of my development progress, including highlights and detours. Appendix B is the data from the experiments. Appendix C includes the mathematical derivation of the formulae used in the system. Appendix D is a more detailed discussion of the implementation of parts of the system and the data generation for the experimental results, including screenshots of code snippets. Appendix E is the experimental output from the data analysis package, including parametric tests and confidence intervals.

2. CHAPTER TWO – Literature Review

2.1 Introduction

This chapter is a literature review and discusses the previous work in this field relevant to my research. It has the following topics. Supply chain research which discusses previous attempts to automate supply chain management (supply chain management is mostly manual) and their limitations. The final topic of this chapter is the background to expert systems and production rules (the central decision making and control functionality of my contribution to automating supply chain management).

2.2 Supply Chain Research

A debate concerning supply chain strategy in recent years is the relative merits of Lean and agile philosophies (Christopher et al., 2006). A Lean supply chain strategy focuses on continuous improvement by elimination of all non-value-adding processes and waste across the supply chain. This is appropriate under a steady and stable market and supply process (Mason et al., 2000). The competitiveness of the Lean supply chain is based upon cost leadership and efficiency.

An agile supply chain strategy focuses upon flexibility and a quick response to identify new requirements, and thrives on unpredictable changes, in order to satisfy a customer's needs and try to capture a wider share of the market, and which is appropriate under unstable market conditions or under evolving supply processes (Mason et al., 2000). The competitiveness of the agile supply chain is based on the ability to quickly capture and respond to new opportunities and sell products unmatched by competitors.

Business with agility can rapidly configure itself in response to market opportunities. The agility has to be supported by appropriate processes and structures, and needs the integration of technology, organization and people into a coordinated whole (Zsiflovits and Engelhardt-Nowitzki, 2004). Therefore, human resources and IT infrastructure are needed to support agile systems. The existing work such as those from Hetherington and Ismail (2006), Strohmaier and Rollett (2005), Weizi et al. (2008), and Wu et al. (2006), mainly focused on the descriptive aspects of agile systems. Most of them are lacking the implementation details.

The work presented in this dissertation takes it a step further by using a knowledge-based services-oriented approach in developing an agile system to assist SCM, and ultimately deliver agile on-demand services to participating business entities.

Fayazbakhsh and Razzazi (2008) proposed an electronic hub (e-hub) as a coordination mechanism to monitor (and through the use of linear programming models) to optimize multiple supply chains. They identified that previous study on supply chains were based on simplified models that do not represent real world supply chains with multiple commodities and partners. Their proposal was to model supply chains as a flow network. Ahuja et al. (1993) and Goldberg et al. (1990) provide comprehensive surveys of algorithms for network-flow problems. Their supply chain e-hub was a central coordination mechanism to monitor the current status of all the partners and flows between them. It informs members of the supply chain of optimal decisions about orders after computation. The goal is to minimize operational costs of the whole supply chain while satisfying customer demand. The limitations of the above approach are that its e-hub communication infrastructure is assumed to be a black box and the optimization is to be solved by linear programming techniques (e.g., Karmarkar's algorithm), which for large supply chains could preclude a real time response.

A services-oriented paradigm is the industry trend towards dynamic and agile integrated e-business solutions (Lim and Wen, 2003). Service-oriented architecture (SOA) is a software architectural approach where the basic element of design and development is a service (Papazoglou and Georgakopoulos, 2003). Applications communicate with each other in such architectures through services. Services are self-describing components, which can be recognized by client applications through look up from a registry: universal description, discovery and integration (UDDI). The client application and the service provider communicate via standard protocols (e.g., SOAP, HTTP) and exchange information using standard data formats like XML. The real advantage of SOA lies in its ability to provide seamless integration across business units, customers and partners (Lim and Wen, 2003). By exposing the business services that are available in an organization to external customers, web services and SOA bring additional maintenance costs, since they involve customizing existing legacy interfaces to support the web services application programming interface (API). In Kumar et al. (2007), an empirical study on the benefits of the adoption of SOA was carried out on the performance of the electronic supply chains of a wide cross section of US-based firms. The findings indicated empirical support for a positive relationship between SOA adoption and performance of electronic supply chains. The research results showed that SOA adoption also improved the customer-side effectiveness of electronic supply chains. The real advantage of SOA lies in its ability to provide seamless integration across business units, customers and partners. The limitations for Kumar et al. (2007) are that though they demonstrated supply chain strategy can influence SCI (Supply Chain Integration), they are lacking implementation details. In Lim and Wen (2003), case studies were made where SOA adoption led to potential cost savings and increased business efficiency. The study has found that many large corporations have had successful implementations of web services in their e-commerce and e-supply chain channels (supply chain partners that communicate and do business electronically). However, details are lacking on how SOA may be systematically and effectively applied to e-business, particularly on SCI problems. The limitation of the above work is that the study was limited by the available measures in the data. The data was collected through a survey and their measures are perceived measures rather than actual measures. There were no multiple responses from the same firm, so they were unable to make these perceived measures more robust. Further, they have conducted their analysis at the firm level, while it can be argued that a business unit level or even a process level analysis could give better insights into the organizational impact of SOA adoption. Emerson and Piramuthu (2004) used software agents supported by a knowledge base to manage dynamic configuration of multiple supply chains. Although information plays a major role in effective functioning of supply chains, there is a paucity of studies that deal specifically with the dynamics of supply chains and how data collected in these systems can be used to improve their performance. The research as presented in Emerson and Piramuthu (2004) modeled each business entity in the supply chain as an agent who makes independent decisions based on information gathered from the next level upstream. Using examples, they show performance improvements of the proposed adaptive supply chain configuration framework over static configurations. However, this research did not consider other possible cheaper e-commerce communication infrastructures: rather it assumed all was based on electronic data interchange (EDI). Moreover, global optimization was calculated to be the sum of the individual optimizations performed at each node by the agents managing each pair of nodes using machined learned rules, restricting potential global optimization value-adding, by a central optimization unit.

A preliminary study of integrating a knowledge-based approach to support SCI was conducted through a prototype system delivered by IEEE Services Computing Contest Team (Dai et al., 2007). The work provided much of the insights on integration gaps and potential of improved supply chain delivery under the SOA.

In Fayaz et al. (2005), the researched approach was to augment supply chain simulation by developing a generic ontology that will be used as a centralized, knowledge capturing mechanism. Ontology is a description of the concepts, relationships, set of terms, and languages of a specific domain. Ontology models all the entities and relationships in a domain. It captures the attributes of an entity and inheritance relationships as in object-oriented programming, and it also captures associations such as cardinality as in relational databases (Fayaz et al. 2005). They anticipate that the ontology will enable the user to capture the necessary knowledge to build and generate simulation models. The limitation of this approach was that the ontology they had laboriously developed was too complex to be easily used in generating simulation models.

In Yi-nan and Zhao-fang (2009), an empirical study of the difference between lean and agile supply chain strategies was done with both internal and external integration. Internal integration refers to the extent to which a firm can structure its organizational behaviors, practices, procedures, and strategies into collaborative and manageable processes to establish close relationship between functions and to fulfill its customers' requirements.

While external integration describes the degree to which a firm can establish a close relationship with its key supply chain partners (including customers and suppliers) to structure their inter-organizational behaviors, practices, procedures, and strategies into collaborative and manageable processes to fulfill its customers' requirements (Yi-nan and Zhao-fang, 2009). Their research using regression analyses of the survey data from 604 manufacturing companies in China showed that both lean and agile approaches have significant positive effects on SCI.

In today's competitive environment, it is supply chains rather than companies that compete (Christopher and Towill, 2001). SCM has been a major source of competitive advantage in the USA and, increasingly, in the global economy (Stonebraker and Liao, 2006). The integration of logistics functions, processes and companies, within a firm and across firms, is suggested as a better way to achieve supply chain success, and SCI has been advocated as the key to creating value in SCM. SCI is defined as the extent to which all activities within an organization, and the activities of its suppliers, customers, and other supply chain members, are integrated together (Bowersox et al., 1999). Having an integrated supply chain provides significant competitive advantage including the ability to outperform rivals on both price and delivery (Lee and Bianchi, 1995).

2.3 Background of production rules

Modern production rules are considered these days to be a subset of Event Condition Action rules (ECA) this is a short-cut for referring to the structure of active rules in event driven architecture and active database systems. Event-condition-action (ECA) rules are a natural candidate for the support of reactive functionality on XML repositories (Bailey et. al. 2002).

Such a rule traditionally consisted of three parts:

- The event part specifies the signal that triggers the invocation of the rule.
- The condition part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out.
- The action part consists of updates or invocations on the local data.

This structure was used by the early research in active databases which started to use the term ECA. Current state of the art ECA rule engines use many variations on rule structure (Bailey et. al. 2002). The condition part of ECA which was more emphasized in traditional production rules is:

"If condition A is satisfied then do B. The 'A' portion of the rule is called the antecedent or left hand side (LHS) of the rule. The 'B' portion of the rule is called the consequent or right hand side (RHS) of the rule. If A is true (i.e., all of its conditions are satisfied by data and facts in the fact base) and whatever actions specified in B are accomplished then the rule is said to have been 'fired'."

"The condition 'A' may be a conjunction of conditions A1, A2, ... A(n), which must all be satisfied in order to trigger any actions stipulated by B. Any component of this conjunction may involve a negative."

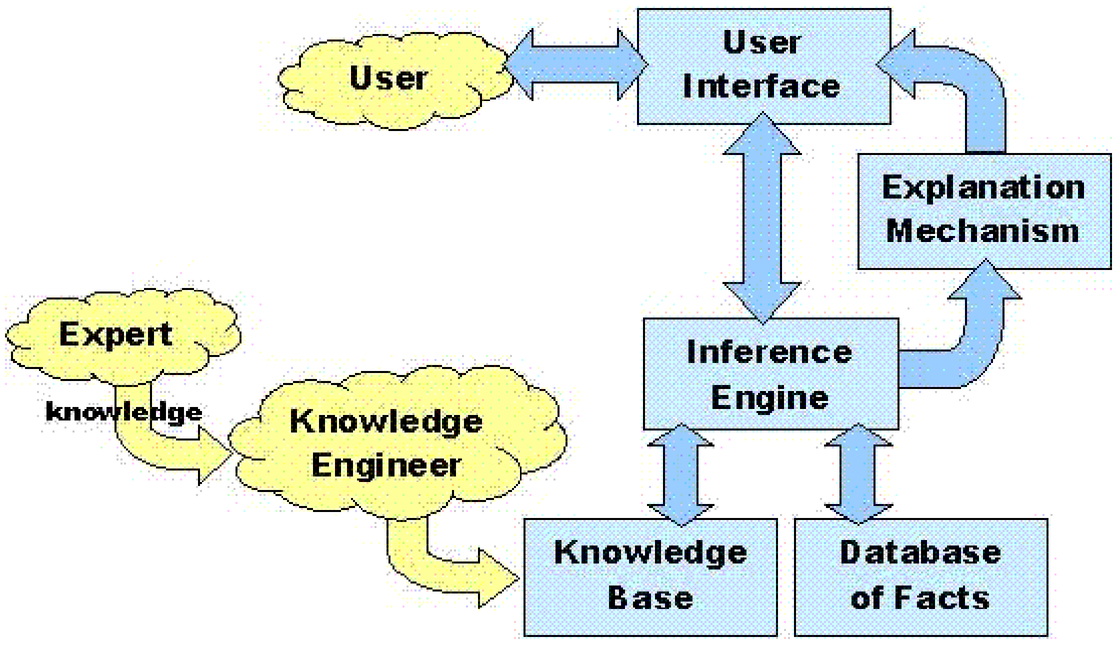


Figure 4 Generic Expert System design

"Likewise 'B' may be a sequence of actions B1, B2 ... B(k), all of which will be taken if the conditional part of the rule is satisfied and the rule is fired." "The relationship between the rule base and the fact base is quite straightforward. If there is a fact in the fact base like 'Var1 = n' and there is a rule in the rule base that states that 'If Var1 = n then B' then this rule is considered for execution (or firing). There may be several rules that are candidates for firing based on the status of the fact base. It is up to the inference mechanism to resolve any conflicts and determine the appropriate firing sequence." "The inference engine (mechanism) is that part of the expert system kernel which supports reasoning about the environment by proper manipulation of its rule and fact bases. It establishes the current state of the environment from its fact base and uses that state information to identify the set of rules whose conditional parts are satisfied by the environment's state. It determines which rules in the rule base are possible candidates for firing based on the circumstance that the conditional part of the rules are satisfied by facts in the fact base. These facts provide an up to date picture of the environment for the expert system.” [Pomykalski et al, (1999), pp.13].

There are two major services, goal driven and event driven these correspond to backward chaining and forward chaining reasoning. Forward chaining or event-driven reasoning is especially important for monitoring functions. Forward chaining works from LHS to RHS of rules. Backward chaining or goal-driven reasoning is especially important for diagnostic activities. Backward chaining works from RHS to the LHS of rules:

The procedure for backward chaining or goal-driven reasoning is:

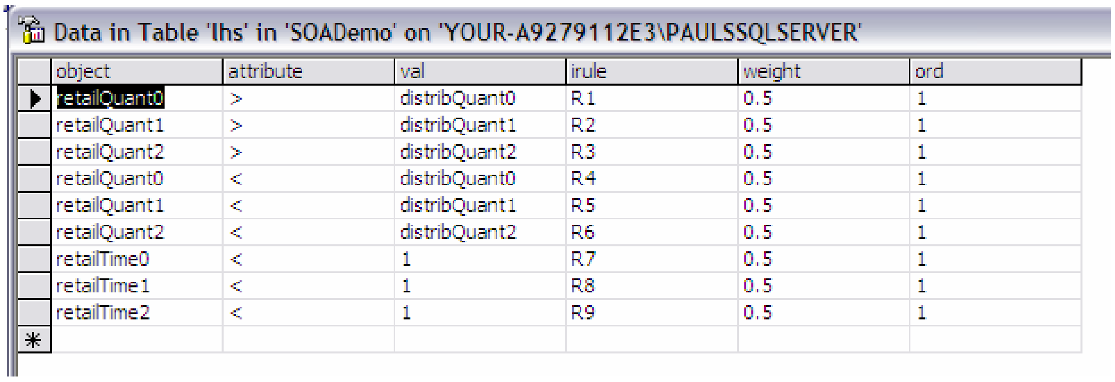
- Select goal to be achieved.
- Identify rules in the rule base whose RHSs reflect the goal.
- Examine the LHS of selected rules.
- Identify the facts and data in the fact base needed to evaluate the LHSs to true.
- Using the identified facts as new goals and going through the previous process continue this backward reasoning process until a goal is proven true.

The procedure for forward chaining or event-driven reasoning is:

- Identify new facts and data in the fact base.
- Identify the rules whose LHSs are satisfied by the selected data and facts.
- If more than one rule is identified resolve conflict and select one rule or sequence of rules according to some priority.
- Fire the rule or sequence of rules.
- The activation of the RHS of the selected rule(s) will result in new facts and data being instantiated in the fact base. These new data and facts can again be used to identify rules whose LHS are satisfied and the forward chaining process can proceed.

To implement this functionality, all the LHSs of the production rules are stored in a database table called ‘lhs’ (see Figure 5). All the RHSs of the production rules are stored in a database table called ‘rhs’ (see Figure 6). This enables efficient searching of LHSs and RHSs. There is a database table called ‘action’ which holds a list of the URL’s of the services that are tied to the action parts of the production rules.

Figure 5 is a screen shot of the database table of the LHS of the production rules.

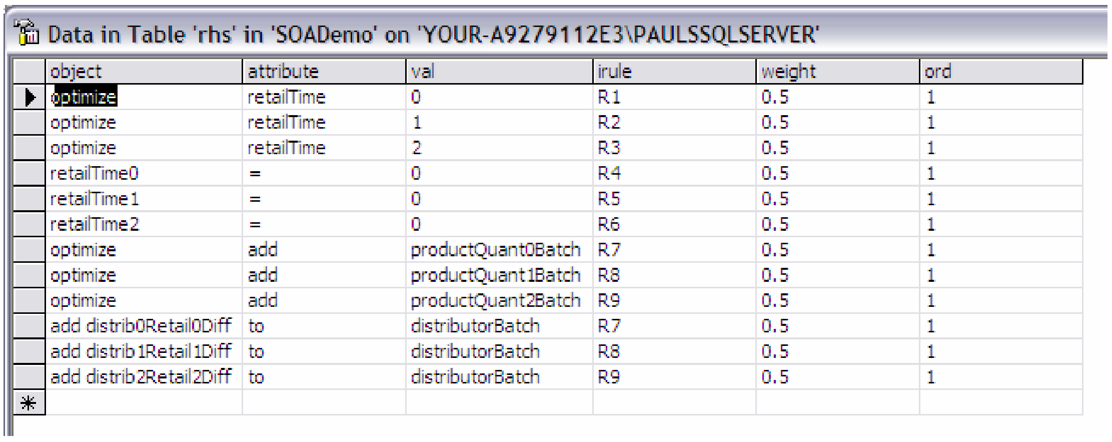


Data in Table 'lhs' in 'SOADemo' on 'YOUR-A9279112E3\PAULSSQLSERVER'

object	attribute	val	irule	weight	ord
retailQuant0	>	distribQuant0	R1	0.5	1
retailQuant1	>	distribQuant1	R2	0.5	1
retailQuant2	>	distribQuant2	R3	0.5	1
retailQuant0	<	distribQuant0	R4	0.5	1
retailQuant1	<	distribQuant1	R5	0.5	1
retailQuant2	<	distribQuant2	R6	0.5	1
retailTime0	<	1	R7	0.5	1
retailTime1	<	1	R8	0.5	1
retailTime2	<	1	R9	0.5	1

Figure 5 LHS data view

Figure 6 is a screen shot of the database table of the RHS of the production rules.



Data in Table 'rhs' in 'SOADemo' on 'YOUR-A9279112E3\PAULSSQLSERVER'

object	attribute	val	irule	weight	ord
optimize	retailTime	0	R1	0.5	1
optimize	retailTime	1	R2	0.5	1
optimize	retailTime	2	R3	0.5	1
retailTime0	=	0	R4	0.5	1
retailTime1	=	0	R5	0.5	1
retailTime2	=	0	R6	0.5	1
optimize	add	productQuant0Batch	R7	0.5	1
optimize	add	productQuant1Batch	R8	0.5	1
optimize	add	productQuant2Batch	R9	0.5	1
add distrib0Retail0Diff	to	distributorBatch	R7	0.5	1
add distrib1Retail1Diff	to	distributorBatch	R8	0.5	1
add distrib2Retail2Diff	to	distributorBatch	R9	0.5	1

Figure 6 RHS data view

It is anticipated that knowledge based approaches have an advantage over traditional approaches in supply chain management and optimization through the following properties:

- Very large expertise in supply chain management is possible to be added to the system databases.
- Continual rule evaluation (forward chain reasoning) emulates highly parallel event driven systems monitoring.
- Goal driven rule evaluation (backward chain reasoning) enables intelligent optimization and feedback control (both forward and backward reasoning).
- Complex monitoring and decision logic kept in one place; easy to move and update.

3. CHAPTER THREE – The System

3.1 Overview

This chapter presents the system solution. The first topic gives the system architecture overview and conceptual model. The next topic describes the system and business services. Then the next topic describes the web based user interface. Then there follows a technical description of the system, including the necessity of the system being thread-safe. This is followed by a description of the alternative traditional optimization approach the Simplex method, including history, maths and walkthrough of an example to illustrate the procedure. Then there follows the hardware configuration and finally the communication protocol for the prototype.

3.2 System Architecture

3.2.1 Overview

The proposed research utilizes a knowledge-based method as an alternative approach to standard global optimization algorithms in optimizing supply chain flows. The methodology for this research is first to create a framework that will be able to answer the following two questions:

- Is it possible to efficiently manage multiple supply chains electronically?
- Will knowledge-based heuristically guided optimization rules perform better than standard global optimization algorithms in optimizing supply chain flows?

To answer the first question, a service system paradigm is adopted to assist dynamic SCM, where the system services and business services are distinguished. To answer the second question, a small empirical study will be done using the new framework. Using the new framework we will compare the savings in cost and time of fulfilling some randomly generated retail orders from a selection of three different retailers who are members of three separate supply chains under a common trust umbrella. Each supply chain has a retailer, distributor, manufacturer and three suppliers.

Under each run of the model the supply chain entities from each supply chain will be queried via web services and their collective response will be optimized using custom hand created optimization rules and also optimized using the industry standard Simplex method. The comparison between these two methods will be returned via a web-based client.

Services in this research are classified into two categories:

- System services
- Business services.

System services have a similar role to the services that are offered from an operating system, which in this case are for managing the framework and network-based resources. Business services are those associated with business processes for performing various business tasks and functions. The solution approach is to develop system services from the proposed framework, coupled with business services to provide complete solutions for business users.

The top-level design is shown in Figure 7 It shows an optimization and control module (e-hub) that communicates with supply chain entities from each of the supply chains via web services.

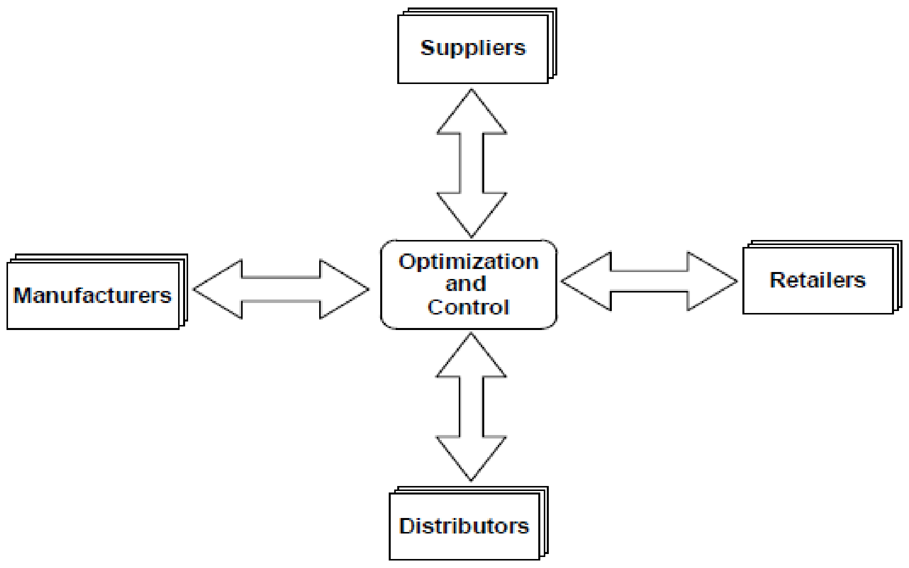


Figure 7 design view of supply chain network control and optimization

The optimisation and control module communicates to a rule-based inference engine through a web services interface. The web services interface interacts with the external web services of the various supply chain entities. The inference engine has rules to control how the optimization and control module should behave based on the data returned from the supply chain entities (see Figure 9).

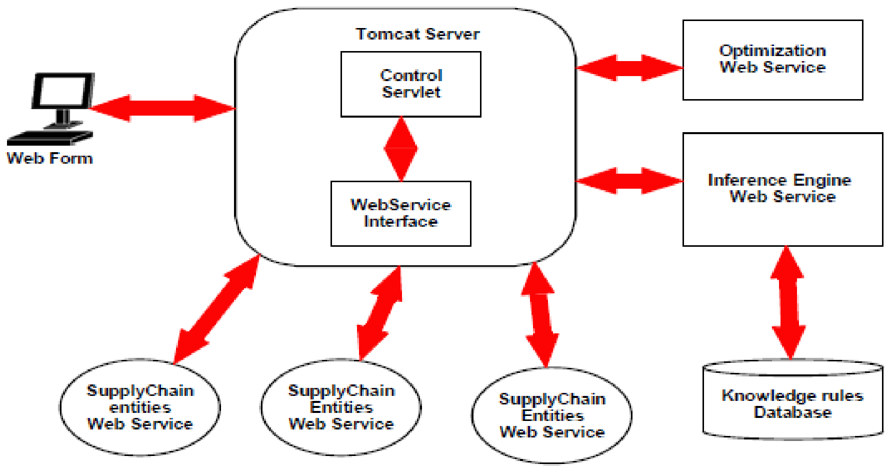

Web Form
Tomcat Server
Control Servlet
WebService Interface
Optimization Web Service
Inference Engine Web Service
SupplyChain entities Web Service
SupplyChain Entities Web Service
SupplyChain Entities Web Service
Knowledge rules Database

Figure 8 design overview of service entities

The optimization web service extracts certain key data items like quantity in units of the various retail orders, quantity in units of stock held in the distributor's warehouses, current production rate in units per day of the various manufacturers and prices of the raw materials for all the suppliers. Certain of these numbers are used to provide an estimate of the time in days to fulfil the retailers order.

$$OrderTime = \frac{DistributorsStockQuantity - retailOrderQuantity}{ManufactProductionRate}$$

There are many variations of this equation used in the prototype, for illustrative and readability purposes only this equation is shown.

After these numbers are obtained the optimization service will invoke the inference engine web service and add these numbers to its working memory (which is implemented as a database table called 'ddt' in SQLServer 2000). If any rules are fired the inference engine will update its working memory, and call any web services that are associated (via the 'action' database table) to the fired rules. The conclusions from the updated working memory are returned via the web service interface to the optimisation web service. This module will parse the returned xml formatted data and call the appropriate optimization function that will calculate an improved order time and price.

3.2.2 Conceptual Model

System services

The system services are implemented as web services, consisting of inference engine and a list of data management and knowledge management services as shown in Figure 9. The inference engine is supported under two main strategies, event driven and goal driven (as shown in Figure 9), which are implemented as services. For example, the URL for externally connecting to the inference engine is:

- URL = http://turquoise.vu.edu.au:8080/axis/services/INDEXService?wsdl
- method to invoke = 'invokeGetGDIResult'
- argument to method = 'optimise'
- login username = 'index'
- password = '**********'.

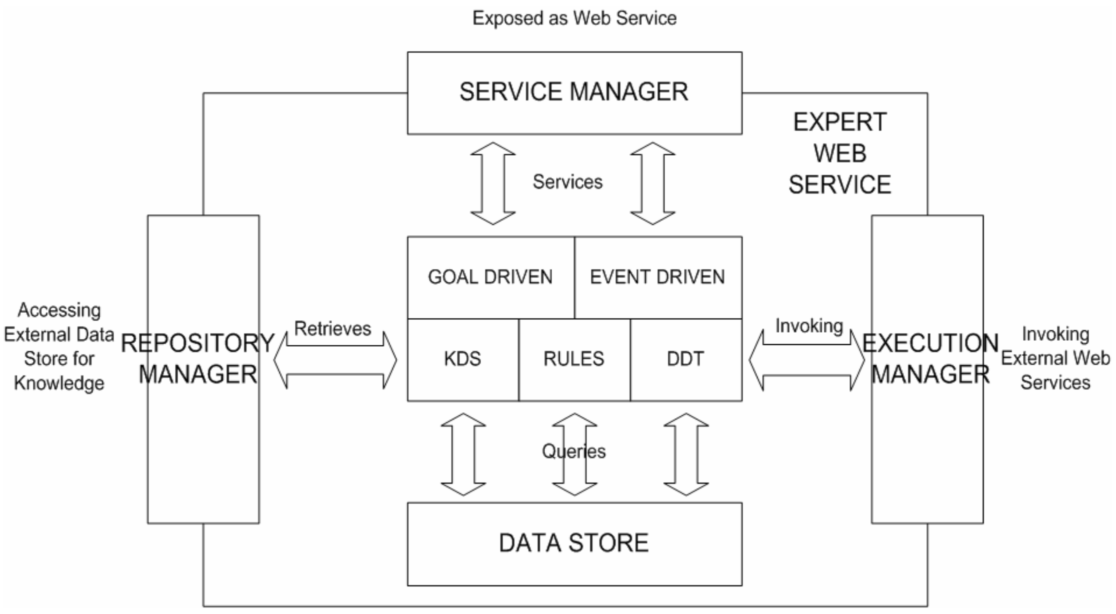


Figure 9 Inference engine system services

Operations of the inference engine

The inference engine operates as a web service and consists of a core set of sub-services with some support services. The core services consist of inference logic (both forward and backward reasoning), and abstractions of rules and working memory. The support services consist of a service execution manager to enable connection to external web services, and a repository manager to communicate with an external database that acts as a repository for the rule base and working memory.

To support our framework, the inference engine has the ability to automatically connect to other computers over a network and execute web services. This enables the system to collect data from other knowledge sources over the WWW and use custom applications to solve complex problems. This ability to connect to and use external resource knowledge greatly enhances the knowledge base and its processing power.

The heuristic method uses the values of supplier costs, production rate, and distributor stock quantity returned from the supply chain web services to decide which rules will fire from a small database of production rules. See Figure 5 for some sample rules. An example rule from Figure 5 say rule 1, reads:

- If retail order quantity from supply chain 0 is greater than current stock in supply chain 0's warehouse then call optimises retail order waiting time for retail order 0 (refer to Figure 6).

The optimisation heuristic for this scenario is to choose the supply chain with the smallest retail order waiting time, and after its order has been fulfilled, to transfer its production stream to the supply chain with the largest retail order waiting time. Or more formally:

- If a large retail order, low distributor warehouse stock, and low manufacturer production rate (this means a large time to fulfil the order).
- Then choose a supply chain with the smallest time to fulfil an order, and after completing its current order, to transfer its production stream to the supply chain with the largest time to fulfil an order.

The math to calculate the new retail order waiting time is:

$$New\ Retail\ Time\ = \frac{PqMax\ \times\ RtMax\ +\ PqMin\ \times\ RtMin}{PqMax\ +\ PqMin}$$

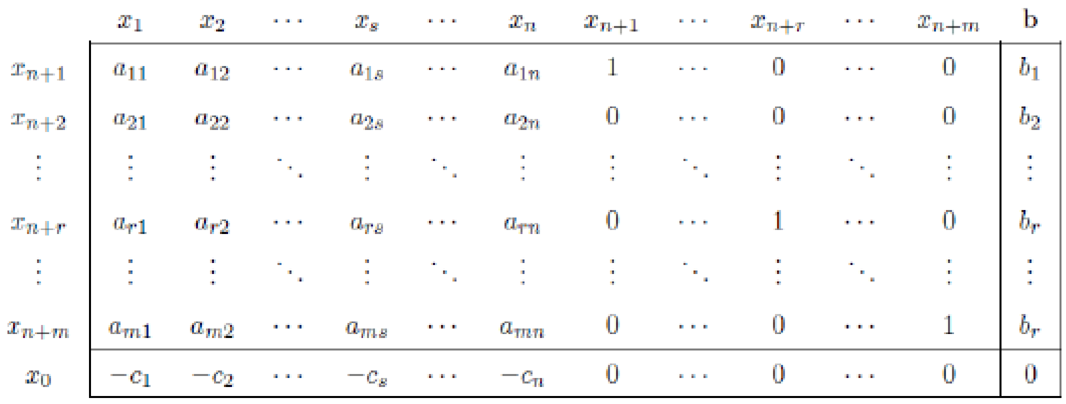
- PqMax production rate for chain with largest retail time
- RtMax retail time for chain with largest retail time
- PqMin production rate for chain with smallest retail time

- RtMin retail time for chain with smallest retail time.

See Appendix C for background to equation.

- Another heuristic that was used in this framework is that retail orders that will be fulfilled in under a day have their supply chain components (production stream and warehouse stock) available for the use of other supply chains. This can be used to increase the production stream and supplement the warehouse stock of other supply chains, therefore reducing the waiting times to deliver orders to all the retailers of other supply chains.
- Another heuristic that was used is to choose the cheapest supplier for each part needed to manufacture an item, even if the supplier traditionally only deals with a particular supply chain.

These are common sense rules to minimise waste in time and cost. They can additionally be used to automatically guide the generation of the constraint equations for the Simplex resource optimisation method. This is possible because SOA makes available the requisite knowledge by monitoring the status of the other supply chains. The action parts of the applicable rules will direct the flows between the different supply chains. As well as directing the supply chain flows the rules will alter the values of the variables in the cost function, which will generate improved costs and times, which are then compared to the non-optimised costs and times.



	x_1	x_2	$\cdots$	x_s	$\cdots$	x_n	x_{n+1}	$\cdots$	x_{n+r}	$\cdots$	x_{n+m}	b
x_{n+1}	a_{11}	a_{12}	$\cdots$	a_{1s}	$\cdots$	a_{1n}	1	$\cdots$	0	$\cdots$	0	b_1
x_{n+2}	a_{21}	a_{22}	$\cdots$	a_{2s}	$\cdots$	a_{2n}	0	$\cdots$	0	$\cdots$	0	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{n+r}	a_{r1}	a_{r2}	$\cdots$	a_{rs}	$\cdots$	a_{rn}	0	$\cdots$	1	$\cdots$	0	b_r
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{n+m}	a_{m1}	a_{m2}	$\cdots$	a_{ms}	$\cdots$	a_{mn}	0	$\cdots$	0	$\cdots$	1	b_r
x_0	$-c_1$	$-c_2$	$\cdots$	$-c_s$	$\cdots$	$-c_n$	0	$\cdots$	0	$\cdots$	0	0

Figure 10 Generalized Simplex tableau

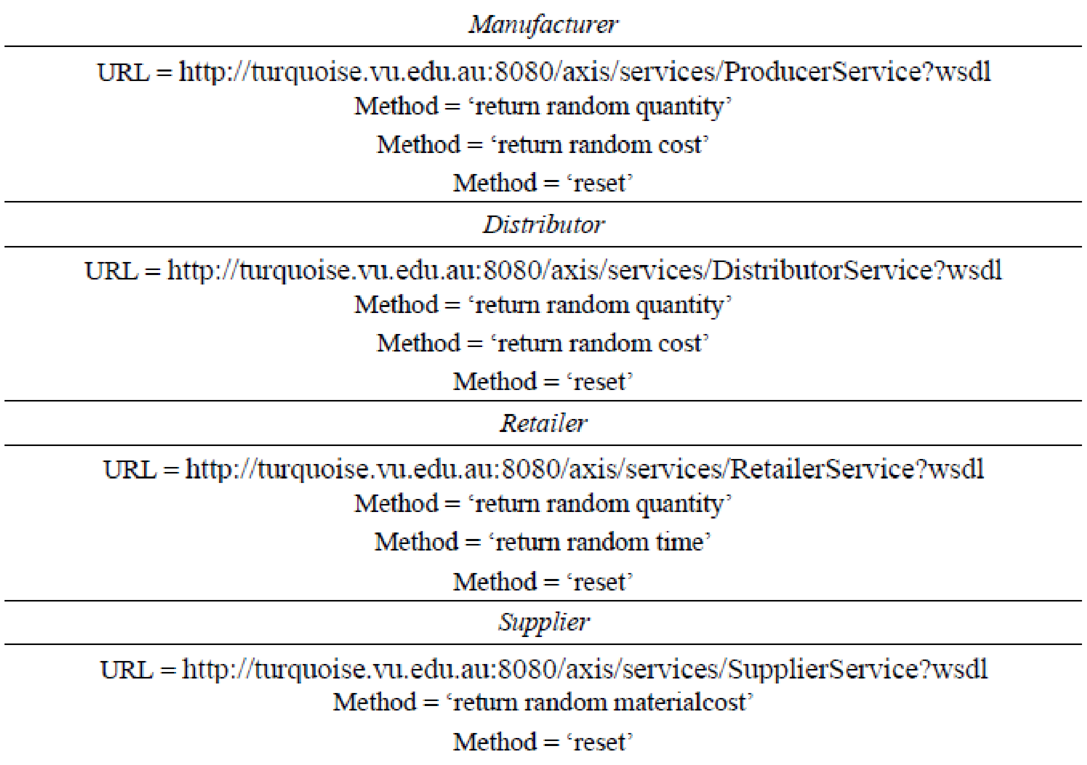
The simplex method will use the same data, but the data will be mapped to the simplex methodology, in this case inserted as coefficients into the decision variables, the constraint equations in the simplex methodology will model the thresholds for the routing constraints between the different supply chains (similar to the heuristic rules). To solve using the simplex methodology the constraint equations are then converted into a standard form using additional slack variables (to convert inequalities to equalities) suitable for adding to a two dimensional array called a tableau (see Figure 10). Certain matrix operations are performed iteratively to the tableau; this adjusts the variables of the cost function, to create an optimal minimum cost function (see Reveliotis, 2003). The output of the new cost function is compared to the non-optimised cost and times. The use of an inference engine for dynamic optimisation and management of multiple supply chains gives great flexibility and easy maintenance of many heuristic rules. The greater processing power enables the use of a more finely grained heuristic modelling. To my knowledge this is unique in the literature.

There are other services that collaborate with the inference engine in the SOA context. For example, a service manager to assist the inference engine to behave as a web service, and expose goal-driven and event-driven inference services. The execution manager invokes external web services for the inference engine when required. These services are invoked when the action parts of the production rules are fired. The invoked services could retrieve data which is added to the fact base table in order to continue the inference process. Consequently, these services could trigger events remotely. The repository manager service manages external database connectivity. This enables multi-knowledge rule bases to be loaded according to the context of the task.

The system services fulfil the roles of service discovery, domain task management, and knowledge maintenance and knowledge execution within SOA where external business services are invoked through services invocation on demand.

Business Services

The supply chains are modeled as collections of web services. Each web service is queried as to its current status, and returns on request a random number within a certain range (price in dollars, or time in days) appropriate to the supply chain artifact that the web service models. Though the web services are hosted locally for convenience, they would operate the same if distributed across the internet, only the URL for each service may be different. There are six services (retailer, distributor, manufacturer and three suppliers) for each of three supply chains, 18 services in all. The full framework is 100% complete and preliminary results (Moynihan and Dai, 2010) show that it would be possible to monitor and communicate with various supply chain entities and thus manage multiple remote supply chains electronically.



Manufacturer

URL = http://turquoise.vu.edu.au:8080/axis/services/ProducerService?wsdl
Method = 'return random quantity'
Method = 'return random cost'
Method = 'reset'

Distributor

URL = http://turquoise.vu.edu.au:8080/axis/services/DistributorService?wsdl
Method = 'return random quantity'
Method = 'return random cost'
Method = 'reset'

Retailer

URL = http://turquoise.vu.edu.au:8080/axis/services/RetailerService?wsdl
Method = 'return random quantity'
Method = 'return random time'
Method = 'reset'

Supplier

URL = http://turquoise.vu.edu.au:8080/axis/services/SupplierService?wsdl
Method = 'return random materialcost'
Method = 'reset'

Figure 11 describes a list of the access details of business services

The manufacturer web service called 'ProducerService' emulates a factory which makes the items that are eventually sold to the customer. When queried it returns two random numbers. The production rate which is 1 to 100 items per day and the cost which is \$1 to \$20 per item. The distributor web service called 'DistributorService' emulates a warehouse which stores the items that are produced by the factory for distribution to retailers who sell to the customer. When queried it returns two random numbers. The current stock level which is 1 to 1,000 items and the storage cost which is \$1 to \$50 per item. The retailer web service called 'RetailerService' emulates a shop which stores a small number of items that are sold to customers. When queried it returns two random numbers. The number of items needed to replenish the stock level which is 1 to 1000 items and the estimated time before stock levels will be zero which is from 1 to 22 days.

The supplier web service called 'SupplierService' emulates a supplier of raw materials to the factory which manufactures the items which are eventually sold to the customer. When queried it returns a random number which is the cost of $1 to $4 per shipment to the factory.

The heuristically guided rules derive their power from optimising across the multiple supply chains rather than optimising within a single chain (see Figure 12 and Figure 13).

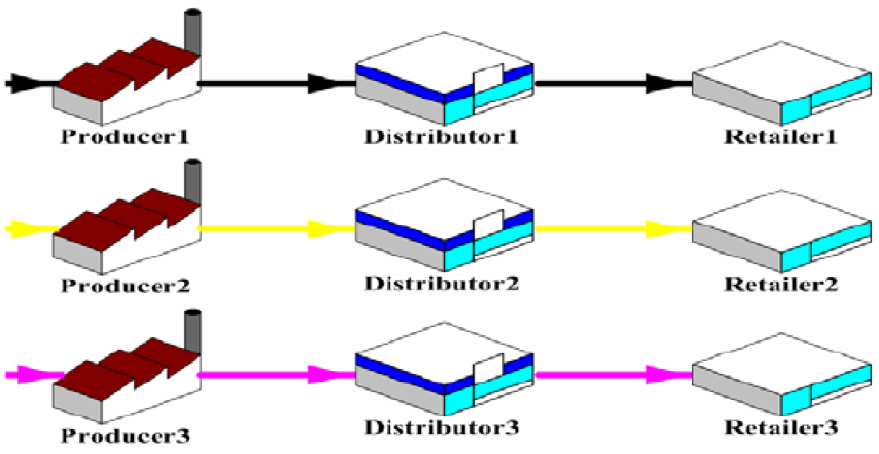


Figure 12 Single isolated supply chains

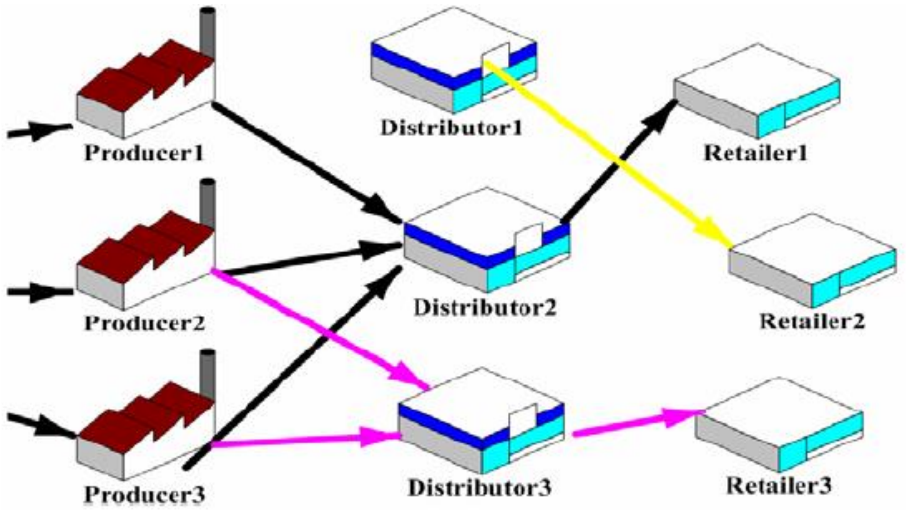


Figure 13 multiple interconnected supply chains

The retailers are part of separate supply chains within larger organizations or have strategic alliances for mutual advantage. See User guide page Figure 16.

The design of the framework distinguishes system services and business services. System services represent the core assets of the framework. Business services are services that interact with external business entities. System services operate in the following categories:

- Knowledge services: directing and coordinating the information flow and task allocation, and problem solving processes.
- Data services: coping with the framework internal and external data retrieval and storage requests.
- Component integrator services: software packaging glue, facilitating systems integration and systems interoperability.
- Communication management services: responsible for all the external communication needs of the framework, such as the facilities for internet communications.
- Business logic modules (BLMs): relational databases, XML data stores.
- Goal driven inferencing (GDI) and event driven inferencing (EDI): services performing goal directed and event driven inferences.

The interactions of system services with business services are described in Figure 14.

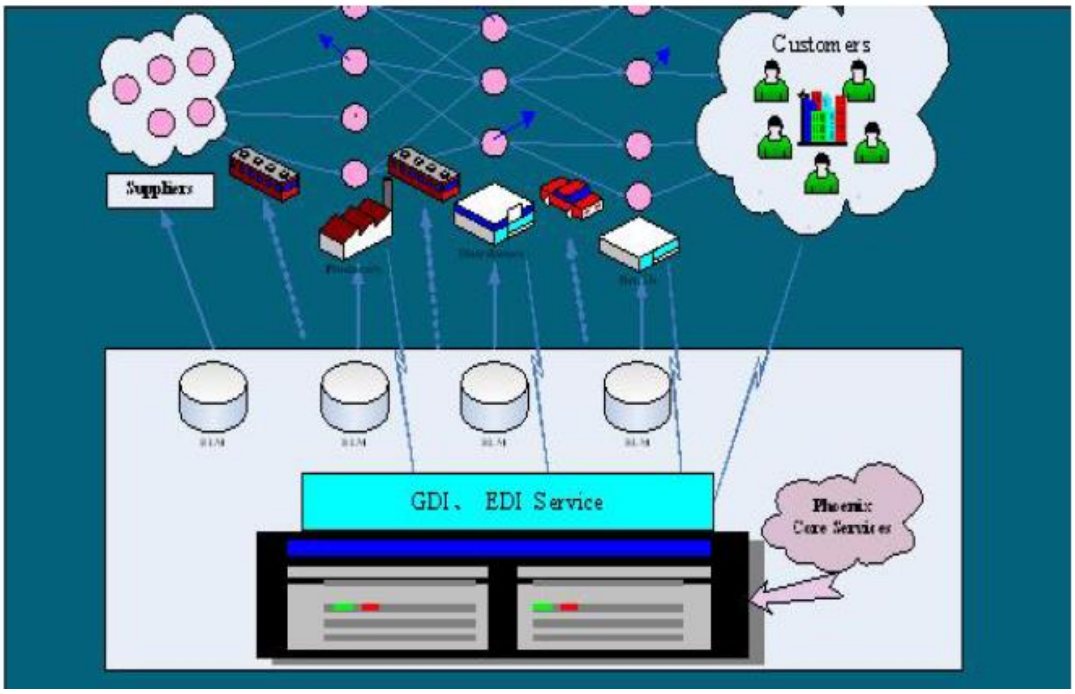


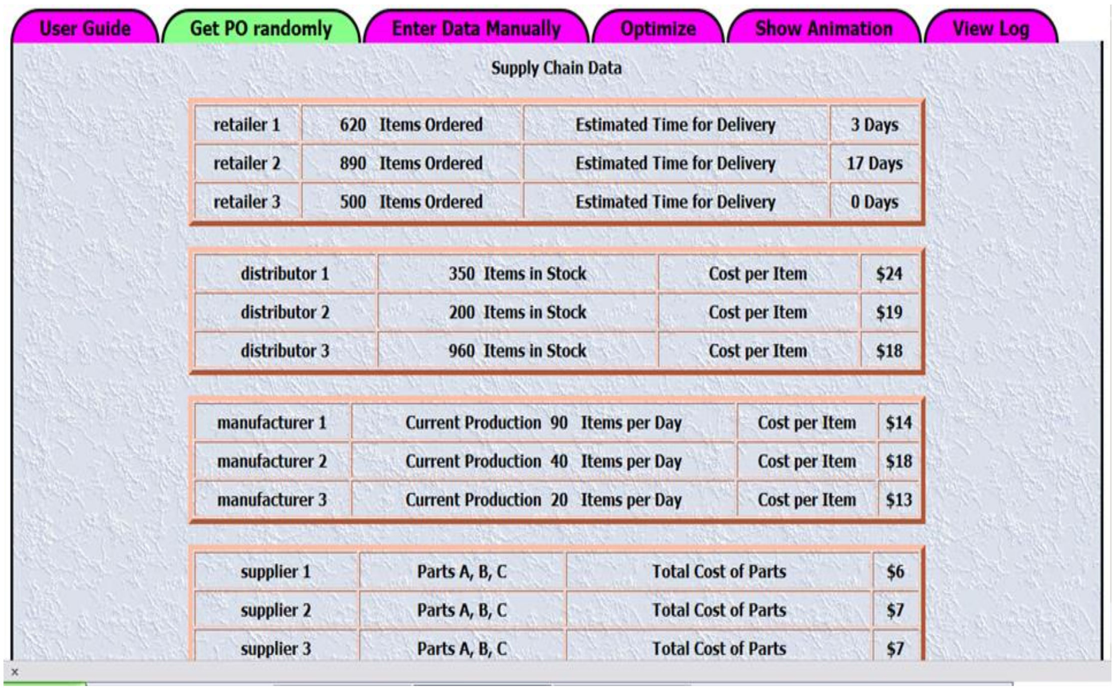
Figure 14 Interactions of system services with business services

3.2.3 User Interface

A prototype of a web-based SCM tool implemented from this research is used. This tool allows the user to see scenarios of various randomly generated supply chain configurations and also see these same configurations after some optimizations have been performed. It is envisaged that in the future, real time data from participating supply chain entities will replace the random data returned by the web services. The online demonstration is available at the URL:

http://turquoise.vu.edu.au:8080/soaDemoNew/UserGuide.html:

See Figure 16, this tabbed page displays some instructions to operate the SCM tool. JavaScript commands control the appearance and functionality of the SCM tool. The web form sends commands via the http protocol to a Servlet hosted in Tomcat called 'MainServlet' this Servlet's job is to first, send reset commands via the web service interface to the web services that model the various supply chain entities. After that task is completed it then sends commands to each of the web services to initialise them with random data within a certain range of values, appropriate to each of the supply chain components, and to return the data to the 'MainServlet'. This is then formatted and inserted into a web page, which is returned to the tabbed page on the users web form. This emulates getting a snap shot of the current state of the users and partners supply chains (see Figure 15).



Supply Chain Data

retailer 1	620 Items Ordered	Estimated Time for Delivery	3 Days
retailer 2	890 Items Ordered	Estimated Time for Delivery	17 Days
retailer 3	500 Items Ordered	Estimated Time for Delivery	0 Days

distributor 1	350 Items in Stock	Cost per Item	$24
distributor 2	200 Items in Stock	Cost per Item	$19
distributor 3	960 Items in Stock	Cost per Item	$18

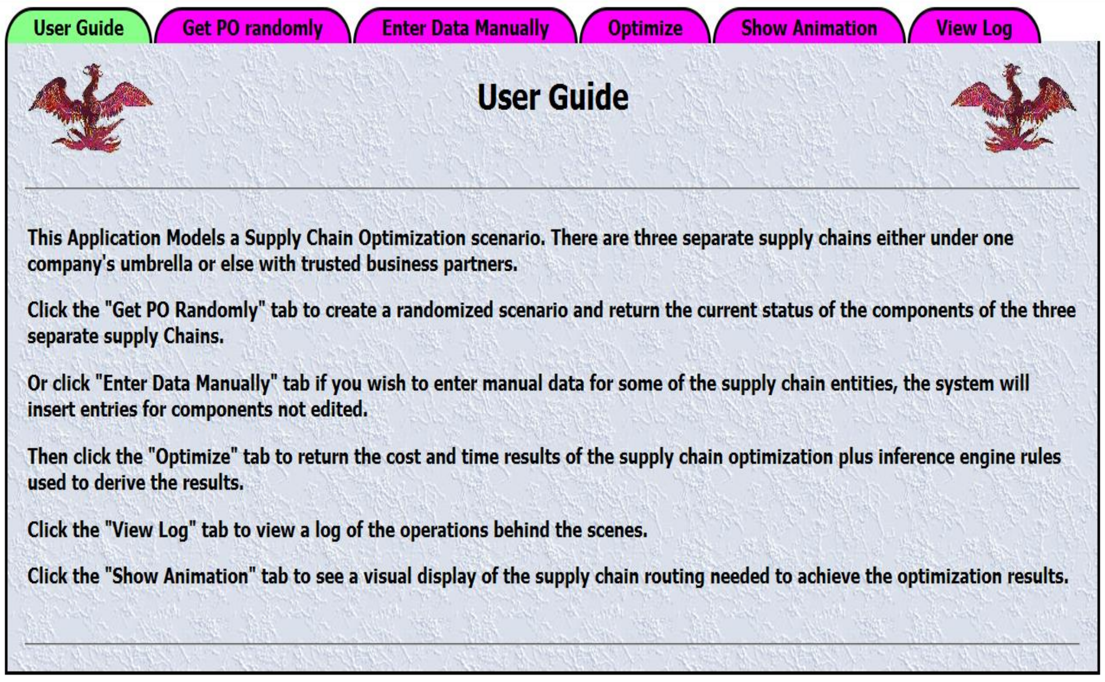
manufacturer 1	Current Production 90 Items per Day	Cost per Item	$14
manufacturer 2	Current Production 40 Items per Day	Cost per Item	$18
manufacturer 3	Current Production 20 Items per Day	Cost per Item	$13

supplier 1	Parts A, B, C	Total Cost of Parts	$6
supplier 2	Parts A, B, C	Total Cost of Parts	$7
supplier 3	Parts A, B, C	Total Cost of Parts	$7

Figure 15 randomly generated scenarios.

Figure 15 shows a scenario, randomly generated, of three supply chains. It shows the current status of the components of three separate supply chains. The user then may wish to see what sort of savings, in terms of time and overall cost that might be achieved, by having the optimization module determine what routing of goods between the various supply chains, that could create the savings. To do this the user will click the 'Optimize' tab. This sends an execution command to the

‘OptimiseServlet’. The OptimiseServlet first sends commands to each of the supply chain entities as web services to send back the previously generated data displayed by the MainServlet. This data is used to populate some java objects that encapsulate the various supply chain states. These hierarchical sets of objects act as repositories for the multiple supply chains state. These objects are then passed to the optimisation class web service. Figure 16 shows the user guide page to prompt the user. Figure 17 shows the costs in time and money of the static supply chain channels based on the generated scenario. Figure 18 shows the dynamically optimised supply chain costs based on the generated scenario.



User Guide

This Application Models a Supply Chain Optimization scenario. There are three separate supply chains either under one company's umbrella or else with trusted business partners.

Click the "Get PO Randomly" tab to create a randomized scenario and return the current status of the components of the three separate supply Chains.

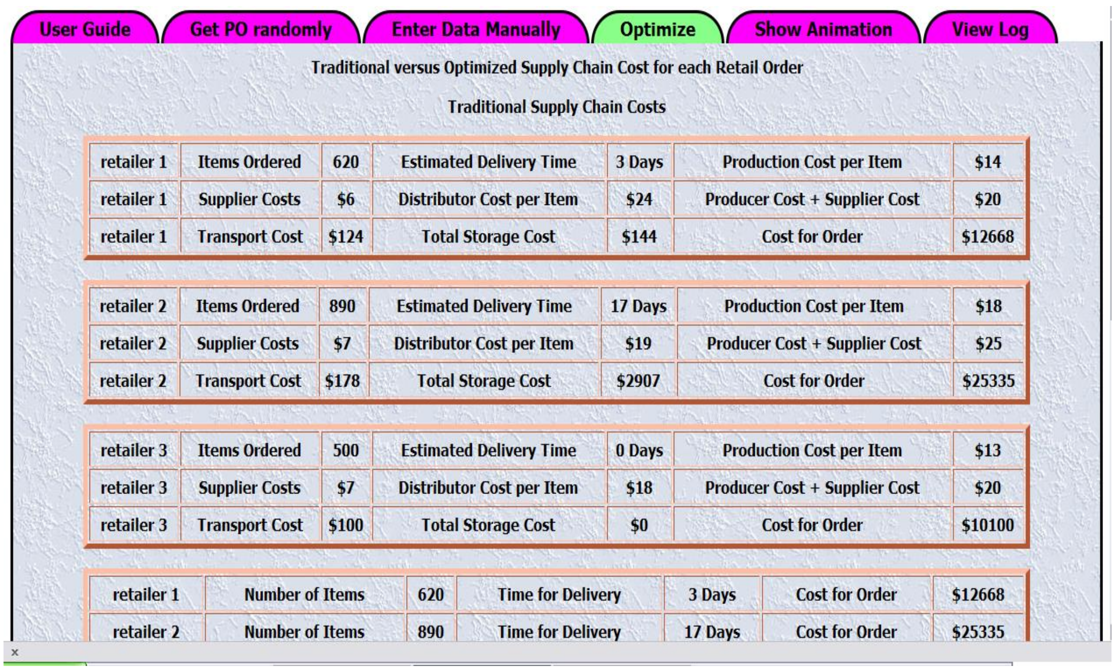
Or click "Enter Data Manually" tab if you wish to enter manual data for some of the supply chain entities, the system will insert entries for components not edited.

Then click the "Optimize" tab to return the cost and time results of the supply chain optimization plus inference engine rules used to derive the results.

Click the "View Log" tab to view a log of the operations behind the scenes.

Click the "Show Animation" tab to see a visual display of the supply chain routing needed to achieve the optimization results.

Figure 16 User Guide



Traditional versus Optimized Supply Chain Cost for each Retail Order

Traditional Supply Chain Costs

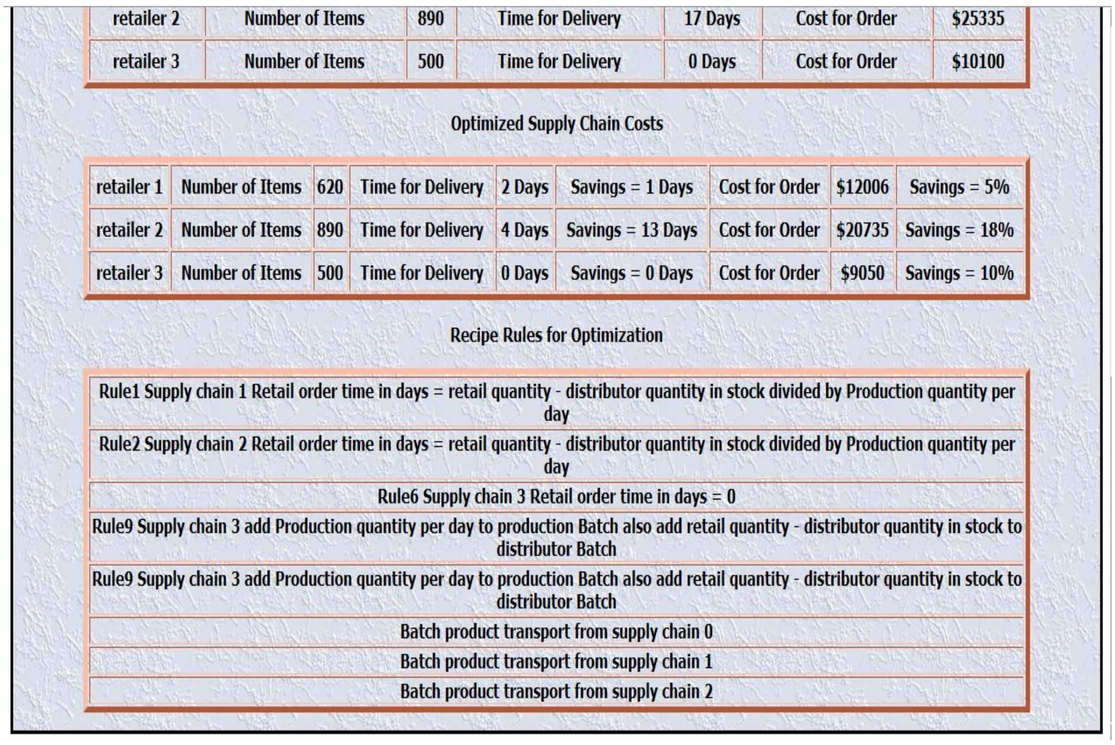
retailer 1	Items Ordered	620	Estimated Delivery Time	3 Days	Production Cost per Item	$14
retailer 1	Supplier Costs	$6	Distributor Cost per Item	$24	Producer Cost + Supplier Cost	$20
retailer 1	Transport Cost	$124	Total Storage Cost	$144	Cost for Order	$12668

retailer 2	Items Ordered	890	Estimated Delivery Time	17 Days	Production Cost per Item	$18
retailer 2	Supplier Costs	$7	Distributor Cost per Item	$19	Producer Cost + Supplier Cost	$25
retailer 2	Transport Cost	$178	Total Storage Cost	$2907	Cost for Order	$25335

retailer 3	Items Ordered	500	Estimated Delivery Time	0 Days	Production Cost per Item	$13
retailer 3	Supplier Costs	$7	Distributor Cost per Item	$18	Producer Cost + Supplier Cost	$20
retailer 3	Transport Cost	$100	Total Storage Cost	$0	Cost for Order	$10100

retailer 1	Number of Items	620	Time for Delivery	3 Days	Cost for Order	$12668
retailer 2	Number of Items	890	Time for Delivery	17 Days	Cost for Order	$25335

Figure 17 Traditional costs derived from scenario



retailer 2	Number of Items	890	Time for Delivery	17 Days	Cost for Order	$25335
retailer 3	Number of Items	500	Time for Delivery	0 Days	Cost for Order	$10100

Optimized Supply Chain Costs

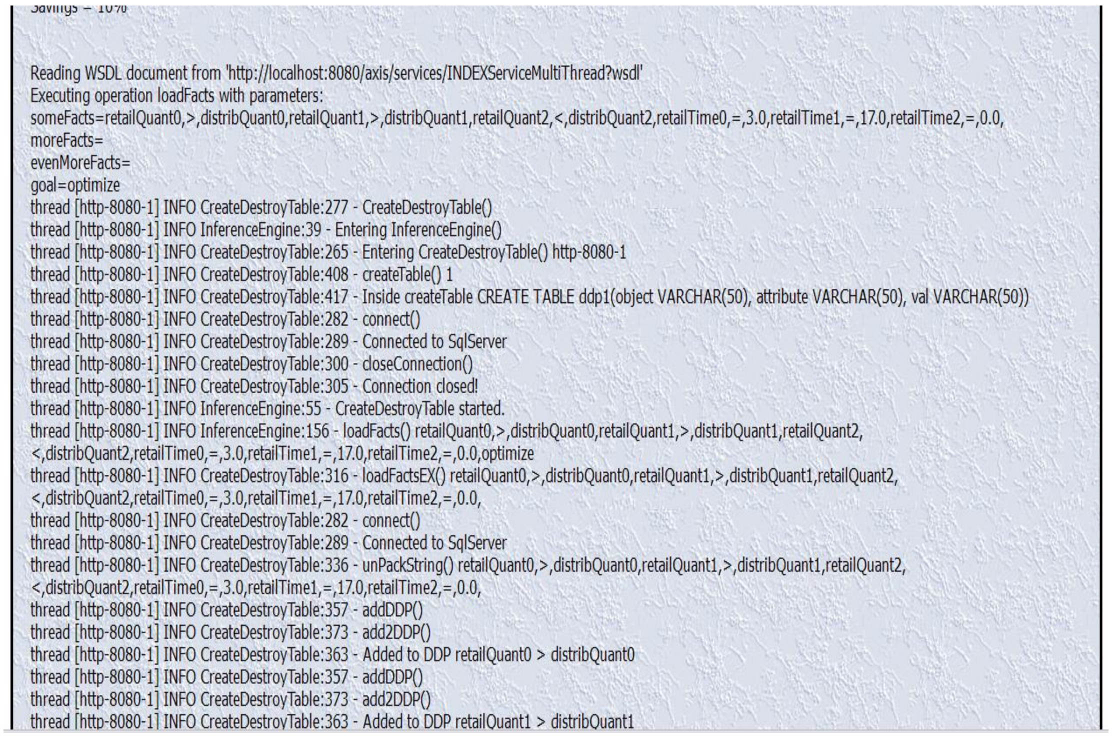
retailer 1	Number of Items	620	Time for Delivery	2 Days	Savings = 1 Days	Cost for Order	$12006	Savings = 5%
retailer 2	Number of Items	890	Time for Delivery	4 Days	Savings = 13 Days	Cost for Order	$20735	Savings = 18%
retailer 3	Number of Items	500	Time for Delivery	0 Days	Savings = 0 Days	Cost for Order	$9050	Savings = 10%

Recipe Rules for Optimization

Rule1 Supply chain 1 Retail order time in days = retail quantity - distributor quantity in stock divided by Production quantity per day
Rule2 Supply chain 2 Retail order time in days = retail quantity - distributor quantity in stock divided by Production quantity per day
Rule6 Supply chain 3 Retail order time in days = 0
Rule9 Supply chain 3 add Production quantity per day to production Batch also add retail quantity - distributor quantity in stock to distributor Batch
Rule9 Supply chain 3 add Production quantity per day to production Batch also add retail quantity - distributor quantity in stock to distributor Batch
Batch product transport from supply chain 0
Batch product transport from supply chain 1
Batch product transport from supply chain 2

Figure 18 Optimized supply chain delivery

The application has a facility to view a log of operations for the interaction between the system web services and the application web services (see Figure 19).



```
Savings = 10%

Reading WSDL document from 'http://localhost:8080/axis/services/INDEXServiceMultiThread?wsdl'
Executing operation loadFacts with parameters:
someFacts=retailQuant0,>,distribQuant0,retailQuant1,>,distribQuant1,retailQuant2,<,distribQuant2,retailTime0,=,3.0,retailTime1,=,17.0,retailTime2,=,0.0,
moreFacts=
evenMoreFacts=
goal=optimize
thread [http-8080-1] INFO CreateDestroyTable:277 - CreateDestroyTable()
thread [http-8080-1] INFO InferenceEngine:39 - Entering InferenceEngine()
thread [http-8080-1] INFO CreateDestroyTable:265 - Entering CreateDestroyTable() http-8080-1
thread [http-8080-1] INFO CreateDestroyTable:408 - createTable() 1
thread [http-8080-1] INFO CreateDestroyTable:417 - Inside createTable CREATE TABLE ddp1(object VARCHAR(50), attribute VARCHAR(50), val VARCHAR(50))
thread [http-8080-1] INFO CreateDestroyTable:282 - connect()
thread [http-8080-1] INFO CreateDestroyTable:289 - Connected to SqlServer
thread [http-8080-1] INFO CreateDestroyTable:300 - closeConnection()
thread [http-8080-1] INFO CreateDestroyTable:305 - Connection closed!
thread [http-8080-1] INFO InferenceEngine:55 - CreateDestroyTable started.
thread [http-8080-1] INFO InferenceEngine:156 - loadFacts() retailQuant0,>,distribQuant0,retailQuant1,>,distribQuant1,retailQuant2,
<,distribQuant2,retailTime0,=,3.0,retailTime1,=,17.0,retailTime2,=,0.0,optimize
thread [http-8080-1] INFO CreateDestroyTable:316 - loadFactsEX() retailQuant0,>,distribQuant0,retailQuant1,>,distribQuant1,retailQuant2,
<,distribQuant2,retailTime0,=,3.0,retailTime1,=,17.0,retailTime2,=,0.0,
thread [http-8080-1] INFO CreateDestroyTable:282 - connect()
thread [http-8080-1] INFO CreateDestroyTable:289 - Connected to SqlServer
thread [http-8080-1] INFO CreateDestroyTable:336 - unPackString() retailQuant0,>,distribQuant0,retailQuant1,>,distribQuant1,retailQuant2,
<,distribQuant2,retailTime0,=,3.0,retailTime1,=,17.0,retailTime2,=,0.0,
thread [http-8080-1] INFO CreateDestroyTable:357 - addDDP()
thread [http-8080-1] INFO CreateDestroyTable:373 - add2DDP()
thread [http-8080-1] INFO CreateDestroyTable:363 - Added to DDP retailQuant0 > distribQuant0
thread [http-8080-1] INFO CreateDestroyTable:357 - addDDP()
thread [http-8080-1] INFO CreateDestroyTable:373 - add2DDP()
thread [http-8080-1] INFO CreateDestroyTable:363 - Added to DDP retailQuant1 > distribQuant1
```

Figure 19 SOA activity log

Figure 20 is a visual display of the supply chain routing needed to achieve the optimization results.

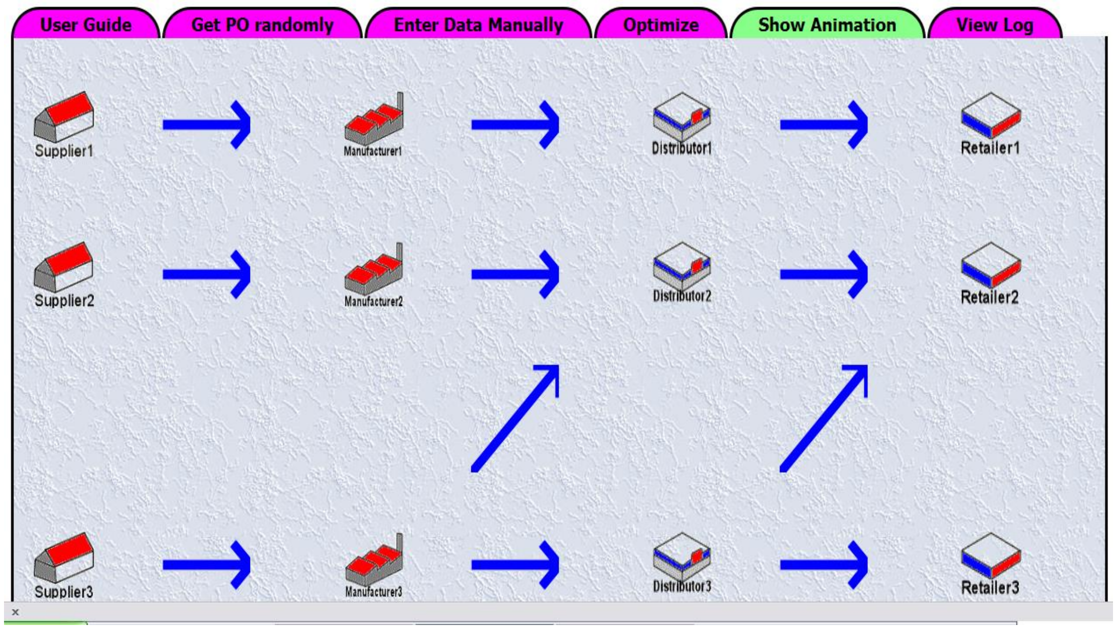
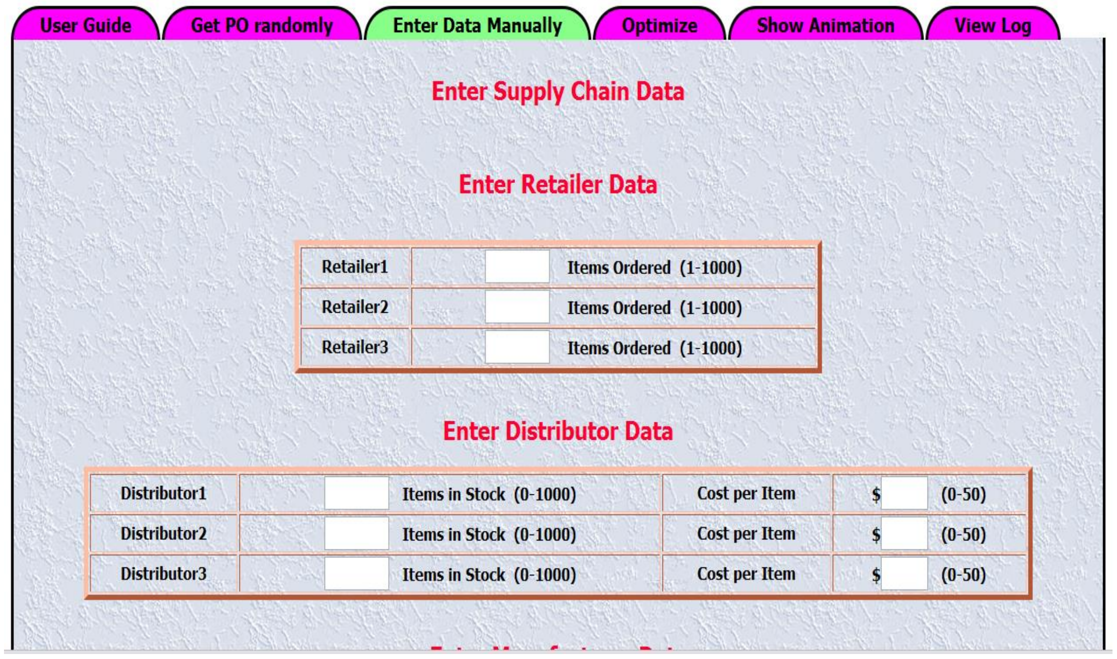


Figure 20 Output of animation tab



Enter Supply Chain Data

Enter Retailer Data

Retailer1		Items Ordered (1-1000)
Retailer2		Items Ordered (1-1000)
Retailer3		Items Ordered (1-1000)

Enter Distributor Data

Distributor1		Items in Stock (0-1000)	Cost per Item	$	(0-50)
Distributor2		Items in Stock (0-1000)	Cost per Item	$	(0-50)
Distributor3		Items in Stock (0-1000)	Cost per Item	$	(0-50)

Figure 21 Option to enter data manually

3.3 Technical Design

3.3.1 Hardware Configuration

The hardware configuration for this prototype is relatively modest, as even cheap computers these days are so powerful. The prototype runs on a modest server called “turquoise” (twin Intel Xeon processors 32bit data path 3.0GHz clock speed 4GB ram) 1MB/sec Internet connection. The system software is: operating system Microsoft Windows Server 2003SP2, Tomcat 6.0.29 Web server, Apache Axis 1.4 SOAP engine and Microsoft SQLServer 2000SP4 relational database.

3.3.1.1 THREAD SAFE INFERENCING

Need for Thread Safe Inferencing

Thread safe inferencing is necessary to allow multiple web clients to run the same set of rules with different input data simultaneously and receive different conclusions based on their input data.

To achieve this goal a series of sub goals has to be met:

- Hopefully be backward compatible with existing client code, in other words the same INDEX (name of inference engine) public API be maintained.
- To support multiple differing conclusions a separate working memory table for each client service request is maintained by each thread.
- To support server robustness a thread pool is used so as to degrade gracefully under a large workload.
- Tomcat web server configuration altered to allow for large numbers of simultaneous web clients.
- INDEX base code made thread safe.

- INDEX retrofitted with a thread creation and maintenance module.

Goals achieved:

- INDEX public API compatibility has been maintained except for a small change.
- Separate working memory table for each client service request is maintained by each thread, and when the http connection thread dies, the table is deleted so the database is not left with artifacts. In SQLServer (INDEX's database) the maximum number of unique tables for each database is 2^{15} or 32,768 tables. So there can be many instances each with their own working memory table.
- A thread pool has been used from "java .concurrent". The pool size has been set at 500 threads by default, but pool size is read at start up from a configuration file. Though this worked, it added an extra layer of complexity and for scalability was removed. Tomcats internal thread pool was leveraged instead.
- Tomcat web server configuration has been altered. A native version of tomcat is used and the server configuration file has been changed to allow for 2000 threads but theoretically could go to 8000 threads.
- INDEX code base has been made thread safe (removed unnecessary code, some non-database methods "synchronized" and reused "Vector" data structure as this is already thread safe. Many internal bugs removed that effected scalability.
- INDEX retrofitted with a thread creation and maintenance module. As the final version of INDEX is only 100K many copies can run concurrently in a typical Server memory space.

The following diagram Figure 22 gives an over view of the concurrency architecture

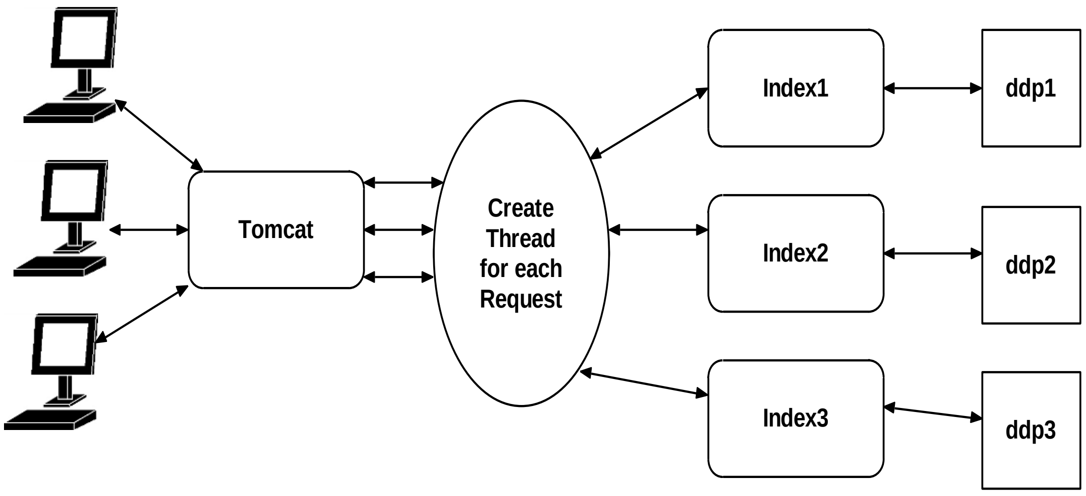


Figure 22 Concurrency Architecture

The multitasking version of INDEX has the following components:

- A thread creation module located in java package "com.index.servicemanager.InferenceEngine" this thread creation module creates a new thread for every web service invocation by external entities.
- The thread creation module also uses a module that creates and manages a database table called "ddp.threadName" which is the working memory for each new instance of INDEX.
- Various synchronized methods located throughout the INDEX system.
- A logging subsystem based on Log4j-1.2.16 with logging statements in every class of INDEX recording current module in focus and current thread name.

Limitations of Thread-based systems

Concurrency is the notion of multiple things happening at the same time. With the proliferation of multicore CPUs and the realization that the number of cores in each processor will only increase, software developers need new ways to take advantage of them. Although modern operating systems are capable of running multiple programs in parallel, most of those programs run in the background and perform tasks that require little continuous processor time. It is the current foreground application that both captures the user's attention and keeps the computer busy. If an application has a lot of work to do but keeps only a fraction of the available cores occupied, those extra processing resources are wasted (Concurrency and Application Design 2013).

Historically, adding concurrency to an application required the creation of one or more additional threads. Writing threaded code can be challenging. Threads are a low-level tool that must be managed manually. Given that the optimal number of threads for an application can change dynamically based on the current system load and the underlying hardware, implementing a correct threading solution becomes difficult. In addition, the synchronization mechanisms typically used with threads add complexity and risk to software designs without any guarantees of improved performance (Concurrency and Application Design 2013).

For threaded code, locks are one of the traditional ways to synchronize access to resources that are shared between threads. However, the use of locks comes at a cost. Even in the non-contested case, there is always a performance penalty associated with taking a lock. And in the contested case, there is the potential for one or more threads to block for an indeterminate amount of time while waiting for the lock to be released (Concurrency and Application Design 2013).

The biggest problem is that threaded code does not scale very well to arbitrary numbers of cores. The developer cannot create as many threads as there are cores and expect a program to run well. What the developer would need to know is the number of cores that can be used efficiently, which is a challenging thing for an application to compute on its own (Concurrency and Application Design 2013).

Also due to the internal data structures of modern operating systems there is a limit to the maximum number of threads that can be run. For example, for a typical "Linux" server running the latest "pthreads" threading library, the limit is about 100,000 threads. This is why multiple virtual servers are run in a federation of servers to handle large numbers of users for cloud computing applications.

Although threads have been around for many years and continue to have their uses, they do not solve the general problem of executing multiple tasks in a scalable way. With threads, the burden of creating a scalable solution rests squarely on the shoulders the developer. The application developer has to decide how many threads to create and adjust that number dynamically as system conditions change. Another problem is that the application assumes most of the costs associated with creating and maintaining any threads it uses (Concurrency and Application Design 2013).

Instead of relying on threads, some modern operating systems like OS X and iOS (Apple's desktop and mobile operating systems) take an asynchronous design approach to solving the concurrency problem. Asynchronous functions have been present in operating systems for many years and are often used to initiate tasks that might take a long time, such as reading data from the disk. When called, an asynchronous function does some work behind the scenes to start a task running but returns before that task might actually be complete. Typically, this work involves acquiring a background thread, starting the desired task on that thread, and then sending a notification to the caller (usually through a call-back function) when the task is done. In the past, if an asynchronous function did not exist for what you want to do, the developer would have to write their own asynchronous function and create their own threads. But now, OS X and iOS provide technologies to allow the developer to perform any task asynchronously without having to manage the threads themselves (Concurrency and Application Design 2013).

There are various scales of concurrency:

- The term thread is used to refer to a separate path of execution for code.
- The term process is used to refer to a running executable, which can encompass multiple threads.
- The term task is used to refer to the abstract concept of work that needs to be performed, which can encompass multiple processes.

In my system the scale of concurrency is the process level. Within the application Inference Engine there was too much that had to be performed serially (changing the order of execution of the subsystems changed the result) for threads to be helpful. Other than each process should be able to share resources (web server and database) the only threading needed was to create a separate process for each user request. Most of the thread management was done by the web server and the database.

Steps to Implementation

The thread creation and maintenance module was finished. It was harder than first thought. Concurrent systems are hard to debug and test. It wasn't until testing that the logic flaws in the module became apparent.

The backwards compatibility with the INDEX API was broke slightly. This was because previously facts had been loaded into the working memory table (ddp) directly by applications, using SQL calls to SQLServer, instead of going through INDEX. What this meant was that the facts added to working memory go into the database table ddp instead of database table ddp(threadNumber).

The code to add facts to working memory was moved into INDEX so that INDEX will be able to manage working memory on a per thread basis. This means that to add facts to working memory an application will call a method in INDEX and pass the facts to that method rather than calling SQLServer directly. This is only a small change to the public API.

The following are some screenshots of the new INDEX being used with the “SOADemo” application.

The next two screenshots Figure 23 and Figure 24 shows the log of interactions behind the scenes, it shows the new logging framework for INDEX. It has been set up to log current thread, log-level, method, line number and comment. It also shows a new working memory table being created. See half way down the following screenshot at “Thread-92”.

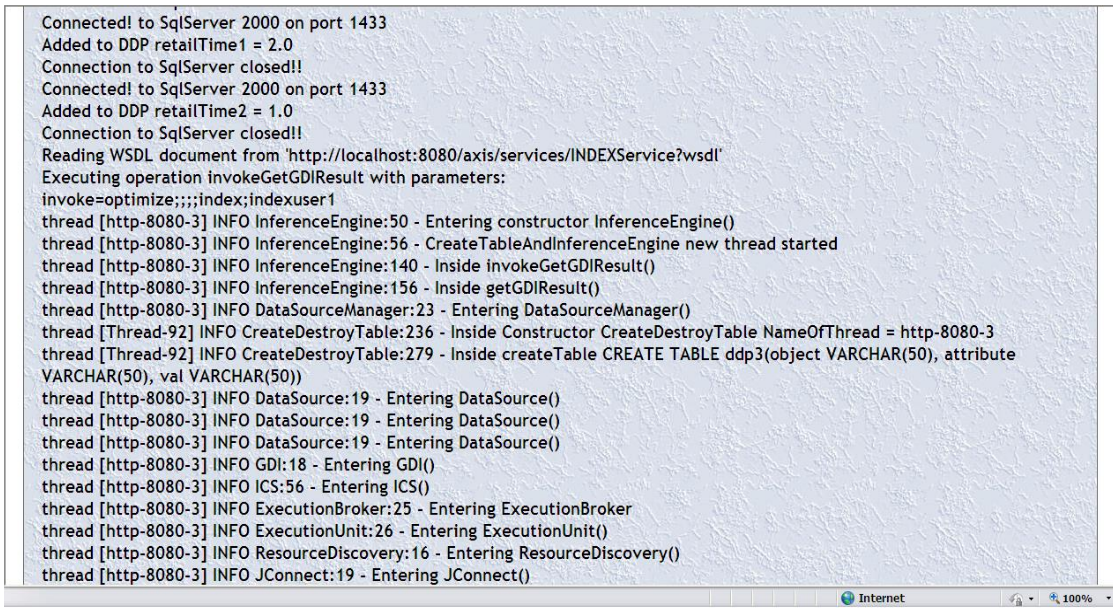

```
Connected! to SqlServer 2000 on port 1433
Added to DDP retailTime1 = 2.0
Connection to SqlServer closed!!
Connected! to SqlServer 2000 on port 1433
Added to DDP retailTime2 = 1.0
Connection to SqlServer closed!!
Reading WSDL document from 'http://localhost:8080/axis/services/INDEXService?wsdl'
Executing operation invokeGetGDIResult with parameters:
invoke=optimize;;;;index;indexuser1
thread [http-8080-3] INFO InferenceEngine:50 - Entering constructor InferenceEngine()
thread [http-8080-3] INFO InferenceEngine:56 - CreateTableAndInferenceEngine new thread started
thread [http-8080-3] INFO InferenceEngine:140 - Inside invokeGetGDIResult()
thread [http-8080-3] INFO InferenceEngine:156 - Inside getGDIResult()
thread [http-8080-3] INFO DataSourceManager:23 - Entering DataSourceManager()
thread [Thread-92] INFO CreateDestroyTable:236 - Inside Constructor CreateDestroyTable NameOfThread = http-8080-3
thread [Thread-92] INFO CreateDestroyTable:279 - Inside createTable CREATE TABLE ddp3(object VARCHAR(50), attribute
VARCHAR(50), val VARCHAR(50))
thread [http-8080-3] INFO DataSource:19 - Entering DataSource()
thread [http-8080-3] INFO DataSource:19 - Entering DataSource()
thread [http-8080-3] INFO DataSource:19 - Entering DataSource()
thread [http-8080-3] INFO GDI:18 - Entering GDI()
thread [http-8080-3] INFO ICS:56 - Entering ICS()
thread [http-8080-3] INFO ExecutionBroker:25 - Entering ExecutionBroker
thread [http-8080-3] INFO ExecutionUnit:26 - Entering ExecutionUnit()
thread [http-8080-3] INFO ResourceDiscovery:16 - Entering ResourceDiscovery()
thread [http-8080-3] INFO JConnect:19 - Entering JConnect()
```

Figure 23 Log of interactions behind the scenes

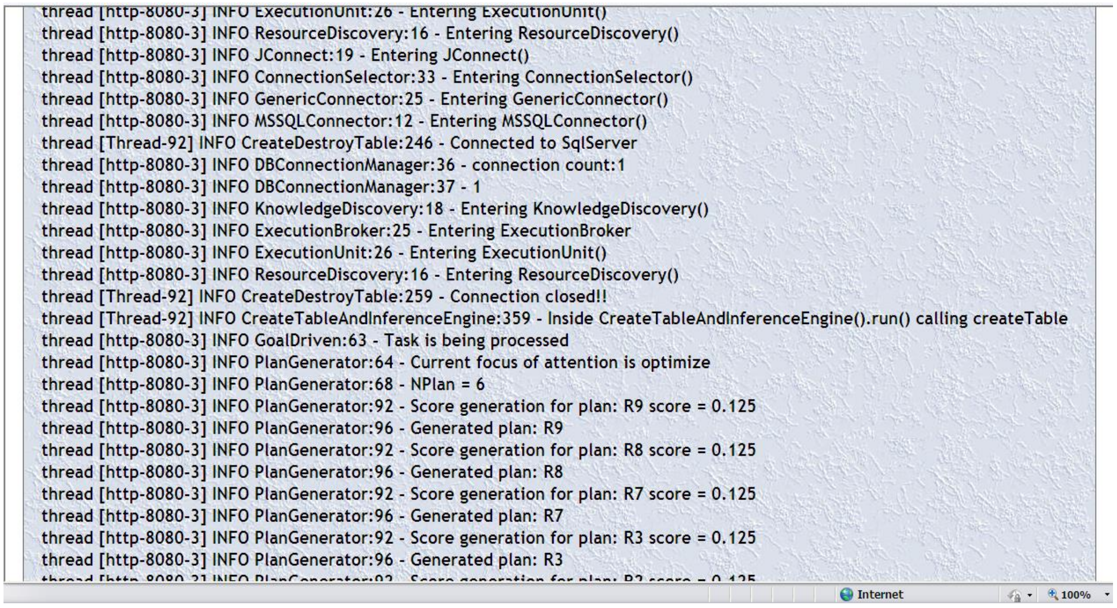
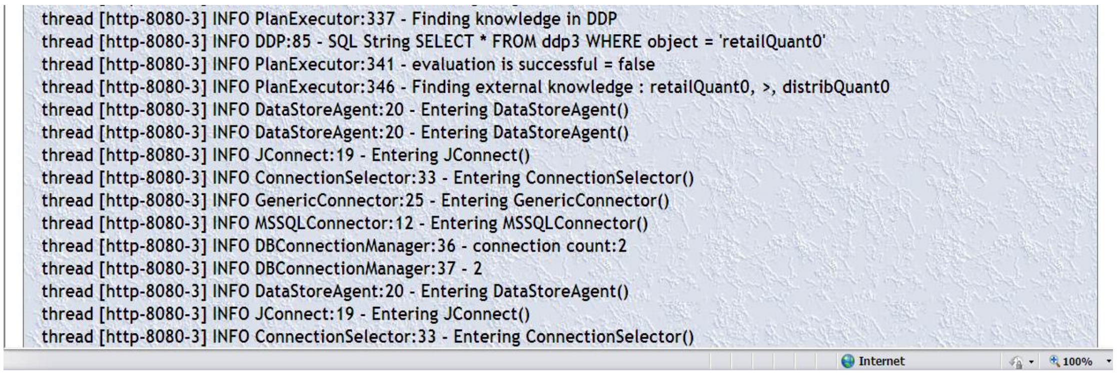

```
thread [http-8080-3] INFO ExecutionUnit:26 - Entering ExecutionUnit()
thread [http-8080-3] INFO ResourceDiscovery:16 - Entering ResourceDiscovery()
thread [http-8080-3] INFO JConnect:19 - Entering JConnect()
thread [http-8080-3] INFO ConnectionSelector:33 - Entering ConnectionSelector()
thread [http-8080-3] INFO GenericConnector:25 - Entering GenericConnector()
thread [http-8080-3] INFO MSSQLConnector:12 - Entering MSSQLConnector()
thread [Thread-92] INFO CreateDestroyTable:246 - Connected to SqlServer
thread [http-8080-3] INFO DBConnectionManager:36 - connection count:1
thread [http-8080-3] INFO DBConnectionManager:37 - 1
thread [http-8080-3] INFO KnowledgeDiscovery:18 - Entering KnowledgeDiscovery()
thread [http-8080-3] INFO ExecutionBroker:25 - Entering ExecutionBroker
thread [http-8080-3] INFO ExecutionUnit:26 - Entering ExecutionUnit()
thread [http-8080-3] INFO ResourceDiscovery:16 - Entering ResourceDiscovery()
thread [Thread-92] INFO CreateDestroyTable:259 - Connection closed!!
thread [Thread-92] INFO CreateTableAndInferenceEngine:359 - Inside CreateTableAndInferenceEngine().run() calling createTable
thread [http-8080-3] INFO GoalDriven:63 - Task is being processed
thread [http-8080-3] INFO PlanGenerator:64 - Current focus of attention is optimize
thread [http-8080-3] INFO PlanGenerator:68 - NPlan = 6
thread [http-8080-3] INFO PlanGenerator:92 - Score generation for plan: R9 score = 0.125
thread [http-8080-3] INFO PlanGenerator:96 - Generated plan: R9
thread [http-8080-3] INFO PlanGenerator:92 - Score generation for plan: R8 score = 0.125
thread [http-8080-3] INFO PlanGenerator:96 - Generated plan: R8
thread [http-8080-3] INFO PlanGenerator:92 - Score generation for plan: R7 score = 0.125
thread [http-8080-3] INFO PlanGenerator:96 - Generated plan: R7
thread [http-8080-3] INFO PlanGenerator:92 - Score generation for plan: R3 score = 0.125
thread [http-8080-3] INFO PlanGenerator:96 - Generated plan: R3
thread [http-8080-3] INFO PlanGenerator:92 - Score generation for plan: R2 score = 0.125
```

Figure 24 Continuation of log of interactions

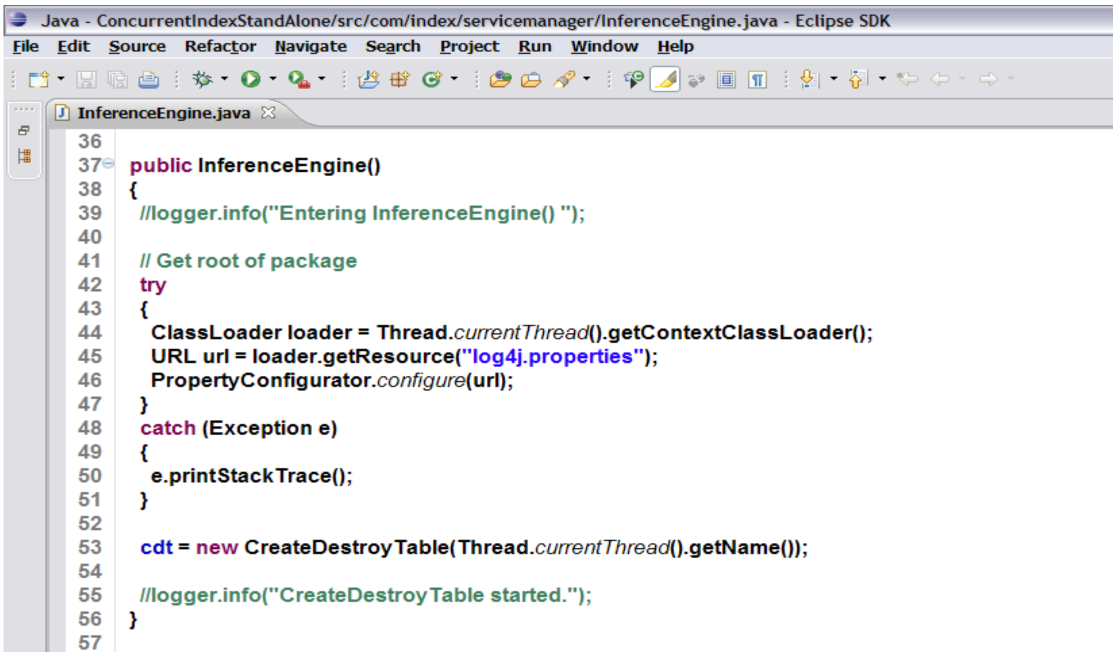
The next screenshot Figure 25 shows working memory table "ddp3" after being created for "Tomcat" thread "http-8080-3" being accessed by a core service "DDP.java" line 85, looking for fact "retailQuant0". See the first two lines.



```
thread [http-8080-3] INFO PlanExecutor:337 - Finding knowledge in DDP
thread [http-8080-3] INFO DDP:85 - SQL String SELECT * FROM ddp3 WHERE object = 'retailQuant0'
thread [http-8080-3] INFO PlanExecutor:341 - evaluation is successful = false
thread [http-8080-3] INFO PlanExecutor:346 - Finding external knowledge : retailQuant0, >, distribQuant0
thread [http-8080-3] INFO DataStoreAgent:20 - Entering DataStoreAgent()
thread [http-8080-3] INFO DataStoreAgent:20 - Entering DataStoreAgent()
thread [http-8080-3] INFO JConnect:19 - Entering JConnect()
thread [http-8080-3] INFO ConnectionSelector:33 - Entering ConnectionSelector()
thread [http-8080-3] INFO GenericConnector:25 - Entering GenericConnector()
thread [http-8080-3] INFO MSSQLConnector:12 - Entering MSSQLConnector()
thread [http-8080-3] INFO DBConnectionManager:36 - connection count:2
thread [http-8080-3] INFO DBConnectionManager:37 - 2
thread [http-8080-3] INFO DataStoreAgent:20 - Entering DataStoreAgent()
thread [http-8080-3] INFO JConnect:19 - Entering JConnect()
thread [http-8080-3] INFO ConnectionSelector:33 - Entering ConnectionSelector()
```

Figure 25 New working memory table "ddp3" being accessed.

The following screenshot Figure 26 shows some source code for the constructor for the new inference engine.

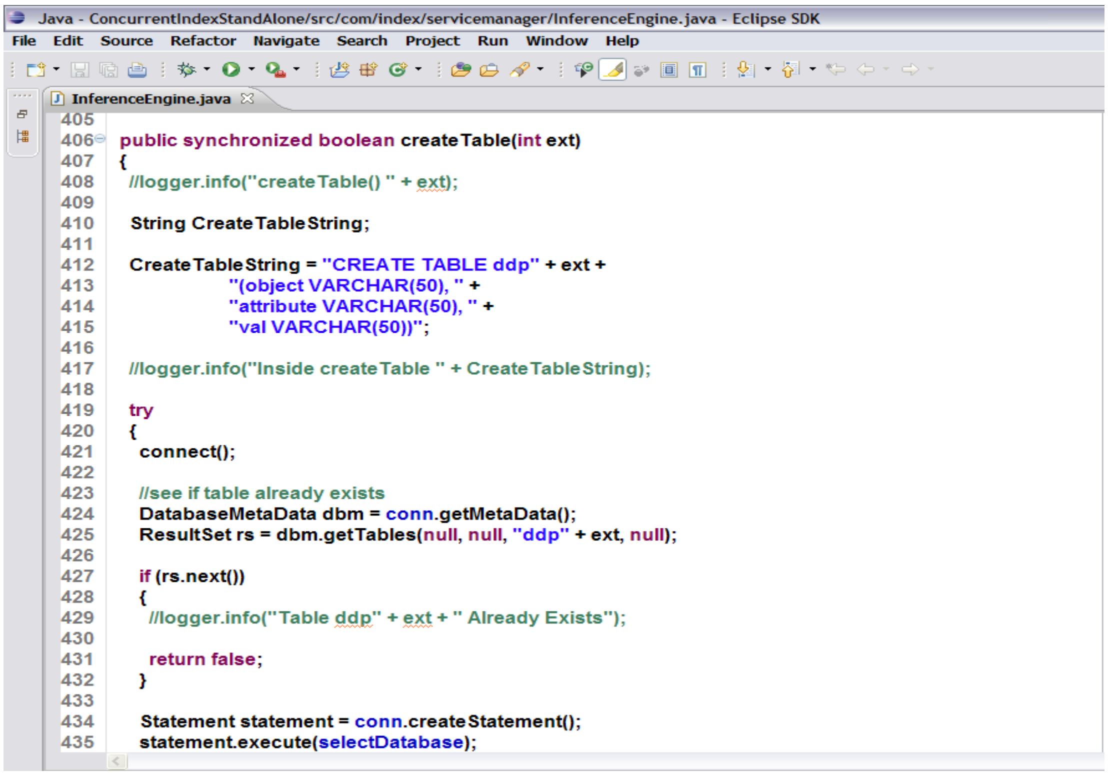


Java - ConcurrentIndexStandAlone/src/com/index/servicemanager/InferenceEngine.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
InferenceEngine.java

```

public InferenceEngine()
{
 //logger.info("Entering InferenceEngine() ");

 // Get root of package
 try
 {
  ClassLoader loader = Thread.currentThread().getContextClassLoader();
  URL url = loader.getResource("log4j.properties");
  PropertyConfigurator.configure(url);
 }
 catch (Exception e)
 {
  e.printStackTrace();
 }

 cdt = new CreateDestroyTable(Thread.currentThread().getName());

 //logger.info("CreateDestroyTable started.");
}

```

Figure 26 Source code for constructor of new Inference Engine.

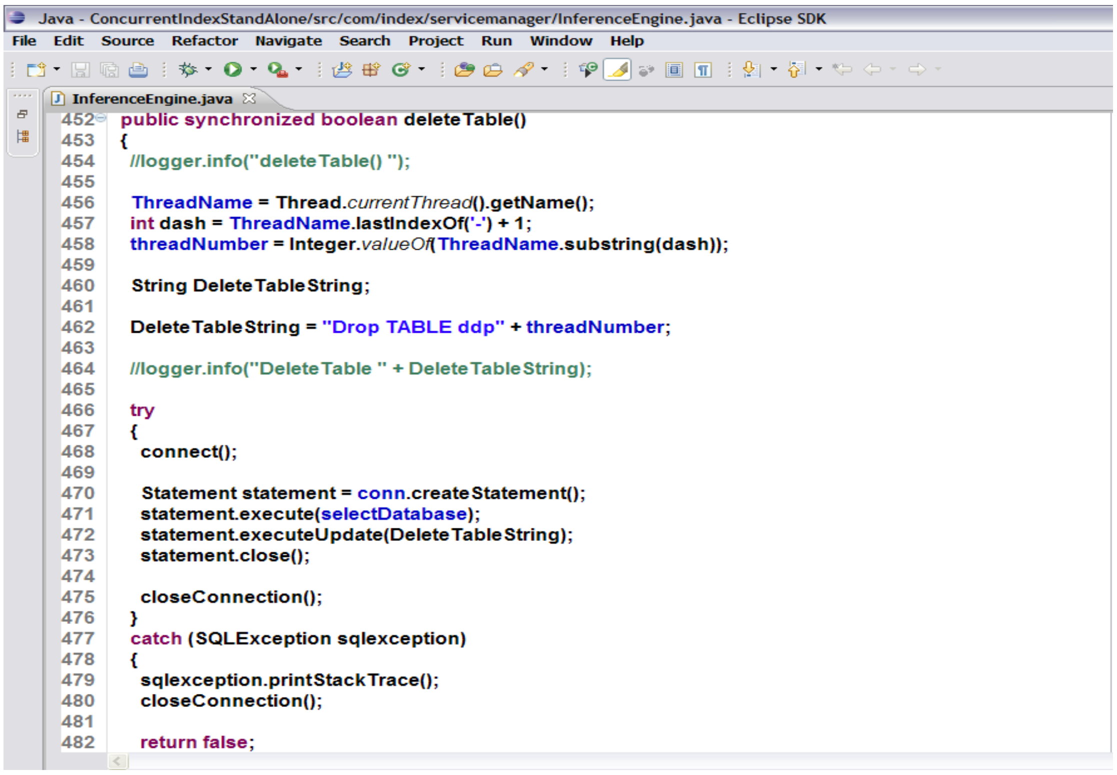
The following screenshot Figure 27 shows some source code for each connection thread creating a working memory table.


Java - ConcurrentIndexStandAlone/src/com/index/servicemanager/InferenceEngine.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
InferenceEngine.java

```java

public synchronized boolean createTable(int ext)
{
 //logger.info("createTable() " + ext);

  String CreateTableString;

  CreateTableString = "CREATE TABLE ddp" + ext +
                "(object VARCHAR(50), " +
                "attribute VARCHAR(50), " +
                "val VARCHAR(50))";

  //logger.info("Inside createTable " + CreateTableString);

  try
  {
   connect();

   //see if table already exists
   DatabaseMetaData dbm = conn.getMetaData();
   ResultSet rs = dbm.getTables(null, null, "ddp" + ext, null);

   if (rs.next())
   {
    //logger.info("Table ddp" + ext + " Already Exists");

    return false;
   }

   Statement statement = conn.createStatement();
   statement.execute(selectDatabase);
```

Figure 27 Source code for connection thread create working memory table.

The following screenshot Figure 28 shows some source code for each connection thread to destroy the working memory table after thread is destroyed.


Java - ConcurrentIndexStandAlone/src/com/index/servicemanager/InferenceEngine.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
InferenceEngine.java

```java
public synchronized boolean deleteTable()
{
 //logger.info("deleteTable() ");

 ThreadName = Thread.currentThread().getName();
 int dash = ThreadName.lastIndexOf('-') + 1;
 threadNumber = Integer.valueOf(ThreadName.substring(dash));

 String DeleteTableString;

 DeleteTableString = "Drop TABLE ddp" + threadNumber;

 //logger.info("DeleteTable " + DeleteTableString);

 try
 {
  connect();

  Statement statement = conn.createStatement();
  statement.execute(selectDatabase);
  statement.executeUpdate(DeleteTableString);
  statement.close();

  closeConnection();
 }
 catch (SQLException sqlexception)
 {
  sqlexception.printStackTrace();
  closeConnection();

  return false;
```

Figure 28 Source code for connection thread destroy working memory table.

See `APPENDIX A` for a more in depth discussion drawn heavily from my development diary. It shows the complex issues that can arise from creating innovative software.

SIMPLEX METHOD

History of Simplex Optimization

Introduction

Optimization as a technique answers the questions what is the best result that can be achieved; what do I have to adjust to attain the highest profits; or what do I do to achieve the lowest cost, or waste. These problems involve making the most efficient use of your resources. Optimization problems are often classified as linear or nonlinear, depending on whether the relationships in the problem are linear with respect to the variables.

Operations Research

Operations research is a discipline that deals with the application of analytical methods to help make better decisions (Operations Research 2012). The terms management science and decision science are synonyms. Using techniques from modeling, statistical analysis, and optimization, operations research attempts to create optimal or near-optimal solutions to complex decision-making problems. As a formal discipline, operations research originated in the efforts of military planners during World War II. After the war it began to be applied to similar problems in industry (Operations Research 2012).

Linear Programming

The most fundamental and pervasive type of optimization problem is a linear program (LP) of the form:

$$minimize\ (or\ maximize)\ c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$such\ that$

$$A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n\ (\leq, =, \geq)\ b_1$$

$$A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n\ (\leq, =, \geq)\ b_2$$

$$\vdots \qquad\qquad \ldots \qquad\qquad \vdots$$

$$A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n\ (\leq, =, \geq)\ b_m$$

$$L_1 \leq x_1 \geq U_1$$

$$L_2 \leq x_2 \geq U_2$$

$$L_n \leq x_n \geq U_n$$

$$where\ A_{ij}, c_j, b_i, L_j, U_j\ are\ known\ real\ numbers;$$

and $(x_1,\ x_2,\ \cdots,\ x_n)$ are the decision variables (unknowns) for which optimal values are sought. The expression being optimized is called the objective function and $(c_1, c_2, \cdots, c_n)$ are the objective coefficients.

The relationships whose senses are expressed with $(\leq, =, \geq)$ are the constraints; $(A_{i1}, A_{i2}, \cdots, A_{in})$ are the coefficients; and b_i is the right-hand side value for the i^{th} constraint. L_j *and* U_j represent lower and upper bounds for the j^{th} decision variable and can be finite or infinite.

The problem of solving a system of linear inequalities dates back at least as far as Fourier, after whom the method of Fourier-Motzkin elimination is named. The earliest linear programming was first developed in Russia by Leonid Kantorovich in 1939 (Linear Programming 2012). It was used during the Second World War for planning to reduce costs to the army. The three founding figures of the subject are Leonid Kantorovich, who developed the earliest linear programming problems, George Dantzig, who published the simplex method in 1947, and John von Neumann, who developed the theory of the duality in the same year. The method was kept secret until 1947 when George B. Dantzig published the simplex method and John von Neumann developed the theory of duality (Given any linear program called the primal, there is another related linear program called the dual. The dual is the negative transpose of the primal program [Vanderbei 2007].)

$$Primal\ Problem\ in\ standard\ form:$$

$$maximize\ \ \sum_{j=1}^{n} c_j\, x_j$$

$$subject\ to\ \ \sum_{j=1}^{n} a_{ij}\, x_j \leq b_i \quad i = (1, \ldots, m)$$

$$x_j \geq 0 \quad j = (1, \ldots, n)$$

Dual is negative transpose of primal.

$$Dual\ Problem\ in\ \text{standard}\ form:$$

$$-maximize \quad \sum_{i=1}^{m} -b_i\, y_i$$

$$subject\ to \quad \sum_{i=1}^{m} -a_{ij}\, y_i \leq -c_j \qquad j = (1, \dots, n)$$

$$y_i \geq 0 \qquad i = (1, \dots, m)$$

and applied it in the field of game theory (Linear Programming 2012).

The linear-programming problem was first shown to be solvable in polynomial time by Leonid Khachiyan in 1979, but a larger theoretical and practical breakthrough in the field came in 1984 when Narendra Karmarkar introduced a new interior-point method for solving linear-programming problems (Linear Programming 2012).

To give an example of the usefulness of linear programming, say you want to find the best assignment of 70 people to 70 jobs (Dantzig's original example). The number of possible configurations exceeds the number of particles in the known universe. Yet, it takes only a relatively short time to find the optimum solution by posing the problem as a linear program and applying the Simplex algorithm. This is because the algorithm drastically reduces the number of possible solutions that must be checked (Linear Programming 2012).

Linear programming is an important part of the field of optimization for several reasons. Many practical problems in operations research can be expressed as linear programming problems. Network flow problems and multicommodity flow problems are special cases of linear programming. A number of algorithms for other types of optimization problems work by solving LP problems as sub-problems. Linear programming is very heavily used in microeconomics and company management, such as planning, production, transportation, and other issues. Most companies would like to maximize profits or minimize costs with limited resources (Linear Programming 2012).

Algorithm

In mathematical optimization, Dantzig's simplex algorithm (or simplex method) is a popular algorithm for linear programming. The journal “Computing in Science and Engineering” listed it as one of the top 10 algorithms of the twentieth century.

The simplex method starts at the origin and follows a path along the edges of a polytope (a multi-dimensional graphic model of the solution space) to a vertex of the polytope where the maximum occurs. There are some limitations of the simplex method that can occur in supply chain management. One is that it is not suitable where some (MIP or Mixed Integer Programming) or all (ILP or Integer Linear Programming) of the constraints are integer variables. These have to be solved with another optimization method (“branch and bound” as used in the open source linear programming solver “LPSolve) Some Integer programing problems theoretically are not solvable in “Polynomial time”.

There are many descriptions in the literature of the simplex algorithm. One of the simplest descriptions I have read is from a linear algebra book called “Linear Algebra with Applications” by W. Keith Nicholson, 3rd Edition PWS Publishing Boston. The procedure is:

- From the problem derive a model that you wish a solution for.
- Convert to standard form.
- Apply the first part of the simplex method to determine if the model is feasible.

- If it is feasible use the simplex method to find which of the basic feasible points is the maximum.

I will now describe the algorithm in more detail. Assume that we are trying to maximize the objective function and there are n variables and m constraints. Let p equal the objective function.

- Prepare the initial Tableau. From the model introduce slack variables and convert any inequalities to equations, including the objective function. Prepare the initial tableau (an augmented (non-square) matrix of coefficients and constants). The augmented matrix has (m+1) rows. The coefficients of p are made the last row.
- Test for optimality. For a given tableau the corresponding basic feasible solution is optimal, if there is no entry in the last row (except the last entry in that row) that is negative. If this is the case then stop, the maximum value of the objective function is the last entry in the last row of the tableau. Otherwise continue.
- Choose the pivot column. This is any column (except the last column) whose last row entry (this is the row containing the coefficients of the objective function) is the most negative. If there is a tie, choose either one.
- Test for unbounded objective function. This occurs if no entry in the pivot column is positive. If this is the case then stop, the objective function has no maximum. Otherwise continue.
- Choose the pivot entry in the pivot row. For each positive entry in the pivot column divide that entry into the last entry in that entries row. Choose the entry with the lowest ratio from the divisions. If two ratios are equal then choose either. This may lead to an infinite loop (called cycling), generally this

is rare, as floating point round off error usually prevents it, there are algorithms that can deal with this case.

- Make the pivot column basic. Use elementary row operations to make the pivot entry 1 and every other entry in the pivot column (including the last) zero.
- Loop back to the step testing for optimality.

I will illustrate the algorithm with a small example.

$$Maximize\ \ p = 2x_1 + 3x_2 - x_3 \ \ \text{subject to:}$$

$$x_1 + 2x_2 + 2x_3 \leq 6$$

$$3x_1 - x_2 + x_3 \leq 9$$

$$2x_1 + 3x_2 + 5x_3 \leq 20$$

$$x_i \geq 0 \ \ for\ i = 1, 2, 3$$

The first step in the procedure is to convert the constraints from inequalities to equalities. This is achieved by introducing new variables x_4, x_5 and x_6 (called slack variables), one for each constraint. Also write the objective function as a fourth equation. The new problem is to:

$$maximize\ p = 2x_1 + 3x_2 - x_3 + 0x_4 + 0x_5 + 0x_6 \ \ subject\ to:$$

$$\begin{aligned}
x_1 + 2x_2 + 2x_3 + x_4 \qquad\qquad &= 6 \\
3x_1 - x_2 + x_3 \qquad + x_5 \qquad &= 9 \\
2x_1 + 3x_2 + 5x_3 \qquad\qquad + x_6 &= 20 \\
-2x_1 - 3x_2 + x_3 \qquad\qquad + p &= 0
\end{aligned}$$

$$x_i \geq 0 \ \ for\ i = 1, 2, 3, 4, 5, 6$$

The objective function p is considered as yet another variable, the augmented matrix or initial matrix for this system of equations is:

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & p & \\ \end{array}$$

$$\begin{bmatrix} 1 & 2 & 2 & 1 & 0 & 0 & 0 & 6 \\ 3 & -1 & 1 & 0 & 1 & 0 & 0 & 9 \\ 2 & 3 & 5 & 0 & 0 & 1 & 0 & 20 \\ -2 & -3 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

In the last row there are two negative entries so this tableau is not optimal. So to continue with the simplex algorithm we must determine the pivot column, this is the column associated with the most negative entry in the last row which in this case is the second column, the one with minus three in the last row. The next step is to test for an unbounded objective function; in this case there are two entries that are positive in the pivot column so the objective function is bounded and has a maximum value yet to be determined.

The next step is to choose the pivot entry in the pivot column. For each positive entry in the pivot column, in this case 2 in row 1 and 3 in row 3, divide into the last entry of each entries row in this case that is 6 / 2 = 3 and 20 / 3 = 6.7 and choose the lowest ratio, in this case 3 which corresponds to 2 in row 1, this is the pivot entry.

The next step is to make the pivot column basic. Use elementary row operations to make the pivot entry 1 and every other entry in the pivot column (including the last) zero. In this case the result is the following tableau:

$$\begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & p & & \\ \end{array}$$

$$\begin{bmatrix} \frac{1}{2} & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 3 \\ \frac{7}{2} & 0 & 2 & \frac{1}{2} & 1 & 0 & 0 & 12 \\ \frac{1}{2} & 0 & 2 & -\frac{3}{2} & 0 & 1 & 0 & 11 \\ -\frac{1}{2} & 0 & 4 & \frac{3}{2} & 0 & 0 & 1 & 9 \end{bmatrix}$$

We test for optimality, in this case there is still a negative entry in the last row (-1/2), so the tableau is still not optimal, we iterate once more. We choose another pivot column; in this case it is column 1 as it contains the only negative entry. The next step is to choose a pivot entry. There are three positive entries to test so first row 3 / ½ = 6 second row 12 / 7/2 = 24/7 third row 11 / ½ = 22. The lowest ratio is 24/7 which corresponds to column 1 row 2 which is 7/2 this is the new pivot entry. The next step is to use elementary row operations to make the pivot entry 1 and every other entry in the pivot column (including the last) zero. In this case the result is the following tableau:

$$\begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & p \end{array}$$

$$\begin{bmatrix} 0 & 1 & \frac{5}{7} & \frac{3}{7} & -\frac{1}{7} & 0 & 0 & \frac{9}{7} \\ 1 & 0 & \frac{4}{7} & \frac{1}{7} & \frac{2}{7} & 0 & 0 & \frac{24}{7} \\ 0 & 0 & \frac{12}{7} & \frac{-11}{7} & -\frac{1}{7} & 1 & 0 & \frac{65}{7} \\ 0 & 0 & \frac{30}{7} & \frac{11}{7} & \frac{1}{7} & 0 & 1 & \frac{75}{7} \end{bmatrix}$$

We test for optimality again, as there are no more negative entries in the last row this tableau is now optimal.

The objective function maximum is $p = \frac{75}{7}$ when the decision variables are:

$$x_1 = \frac{24}{7},\ x_2 = \frac{9}{7},\ x_3 = 0.$$

3.3.2 Communication Interface

The communication interface to my prototype is a TCP/IP back bone supporting HTTP packets which wraps SOAP xml packets. In other words the Internet with web service software enabled for each computer for each supply chain entity.

4. CHAPTER FOUR - System Evaluation

4.1 Overview

This chapter describes the evaluation of the system with respect to the comparison of the two methods for optimization of supply chain flows. To do this a testing framework had to be created to generate the experimental results, and the results had to be analyzed. The Testing Framework topic covers the creation of the testing framework which includes sub topics of discussing the alternative optimization method which includes applying the simplex method to supply chains by modelling them as network flow transportation models.

Another topic is the generation and collection of the data which includes the collection procedure. The data analysis section includes a background to data analysis focusing on exploratory data analysis. The following section called analysis of results takes the background on exploratory data analysis and applies it to the experimental results and the final section called outcomes describes the result of the data analysis.

4.2 Testing Framework

4.2.1 Alternative Optimization Method

The alternative optimization method chosen to compare with the heuristic method to optimize multiple supply chains was a network flow transportation model. A network flow transhipment model was tried but was infeasible, because manufacturer flow rate was always metered by the day while the distributor to retailer arc of the network flow was always a lot larger and intermittent and the distributor transhipment nodes net throughput (flow in minus flow out) was never zero (the transhipment model theory states that transhipment nodes have to have zero net throughput for the model to be applicable). The model chosen was two network flow transportation models back to back see Figure 31. The distributors to retail arcs were optimized first see Figure 30 this gave the quantities that were left in the distributor's warehouses. Then the manufacturers to distributor arcs then were optimized using the new distributor warehouse quantities. See Figure 29. Network flow transportation models can be solved with linear programming techniques. For the automated system the open source linear programming solver "LPSolve" was used as the solver.

The objective function and constraints for the distributor to retailer nodes are:

Objective function

min: -210 C1 +110 C2 +210 C3 +210 C4 +530 C5 +630 C6 +370 C7 +690 C8 +790 C9;

Constraints

+C1 +C2 +C3 <= 360;

+C4 +C5 +C6 <= 780;

+C7 +C8 +C9 <= 940;

+C1 +C4 +C7 = 570;

+C2 +C5 +C8 = 250;

+C3 +C6 +C9 = 150;

The routings from the solution of the distributor to retailer transportation model is:

Objective function: = 195500

Constraint Variable [0] = 0	Distributor1 to Retailer1
Constraint Variable [1] = 210	Distributor1 to Retailer2
Constraint Variable [2] = 150	Distributor1 to Retailer3
Constraint Variable [3] = 570	Distributor2 to Retailer1

Constraint Variable [4] = 40	Distributor2 to Retailer2
Constraint Variable [5] = 0	Distributor2 to Retailer3
Constraint Variable [6] = 0	Distributor3 to Retailer1
Constraint Variable [7] = 0	Distributor3 to Retailer2
Constraint Variable [8] = 0	Distributor3 to Retailer3

Dummy variables

Constraint Variable [9] = 0	Distributor4 to Retailer1
Constraint Variable [10] = 0	Distributor4 to Retailer2
Constraint Variable [11] = 0	Distributor4 to Retailer3

The objective function and constraints for the manufacturer to distributor nodes are:

Objective function

min: -180 C1 +30 C2 +30 C3 -200 C4 +10 C5 +10 C6 -190 C7 +20 C8 +20 C9;

Constraints

+C1 +C2 +C3 <= 30;

+C4 +C5 +C6 <= 10;

+C7 +C8 +C9 <= 20;

+C10 +C11 +C12 <= 150;

+C1 +C4 +C7 +C10 = 210;

+C2 +C5 +C8 +C11 = 0;

+C3 +C6 +C9 +C12 = 0;

The routings from the solution of the manufacturer to distributor transportation model are:

Optimized Objective value = -11200

Optimized Constraints Routings [0] = 30 Manufacturer1 to Distributor1

Optimized Constraints Routings [1] = 0 Manufacturer1 to Distributor2

Optimized Constraints Routings [2] = 0 Manufacturer1 to Distributor3

Optimized Constraints Routings [3] = 10 Manufacturer2 to Distributor1

Optimized Constraints Routings [4] = 0 Manufacturer2 to Distributor2

Optimized Constraints Routings [5] = 0 Manufacturer2 to Distributor3

Optimized Constraints Routings [6] = 20 Manufacturer3 to Distributor1

Optimized Constraints Routings [7] = 0 Manufacturer3 to Distributor2

Optimized Constraints Routings [8] = 0 Manufacturer3 to Distributor3

Dummy variables

Optimized Constraints Routings [9] = 150 Manufacturer4 to Distributor1

Optimized Constraints Routings [10] = 0 Manufacturer4 to Distributor2

Optimized Constraints Routings [11] = 0 Manufacturer4 to Distributor3

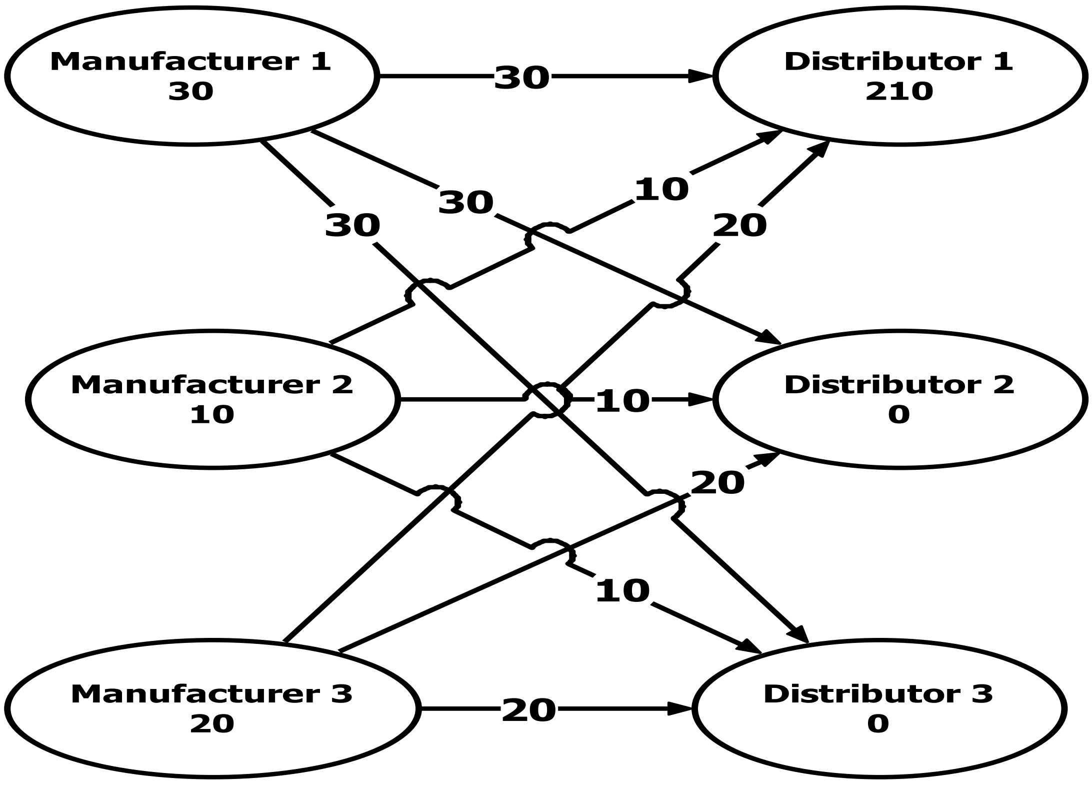


Figure 29 Network flow transportation model of Manufacturer to Distributor

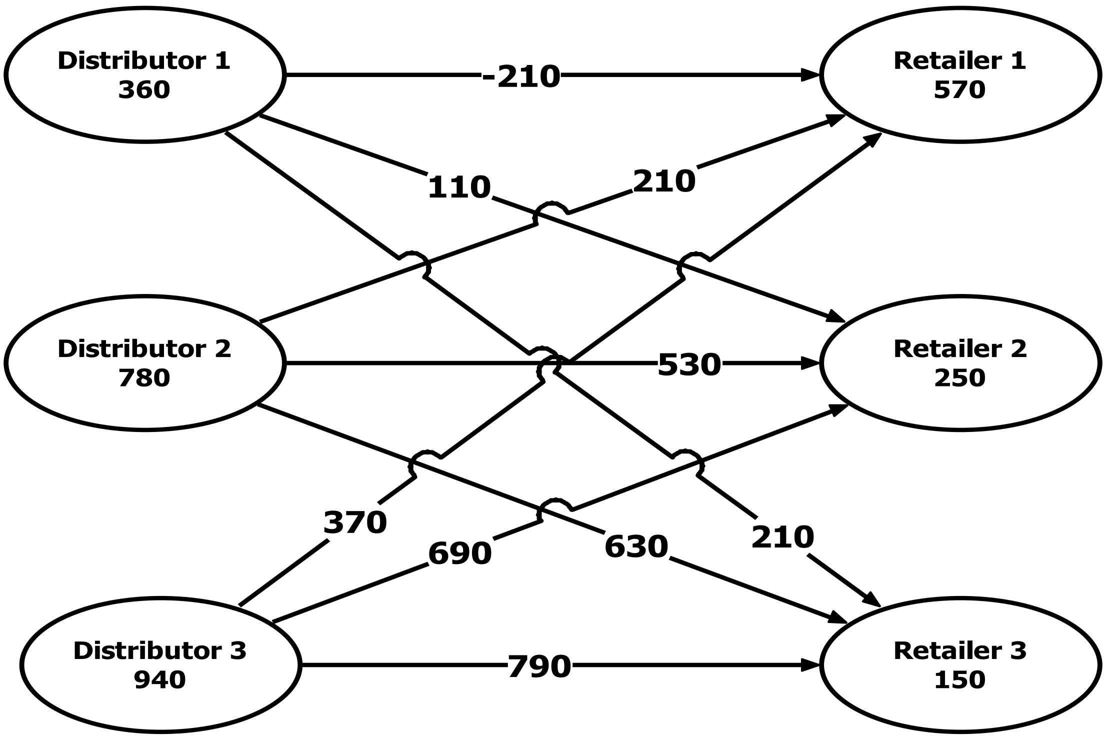


Figure 30 Network flow transportation model of Distributor to Retailer

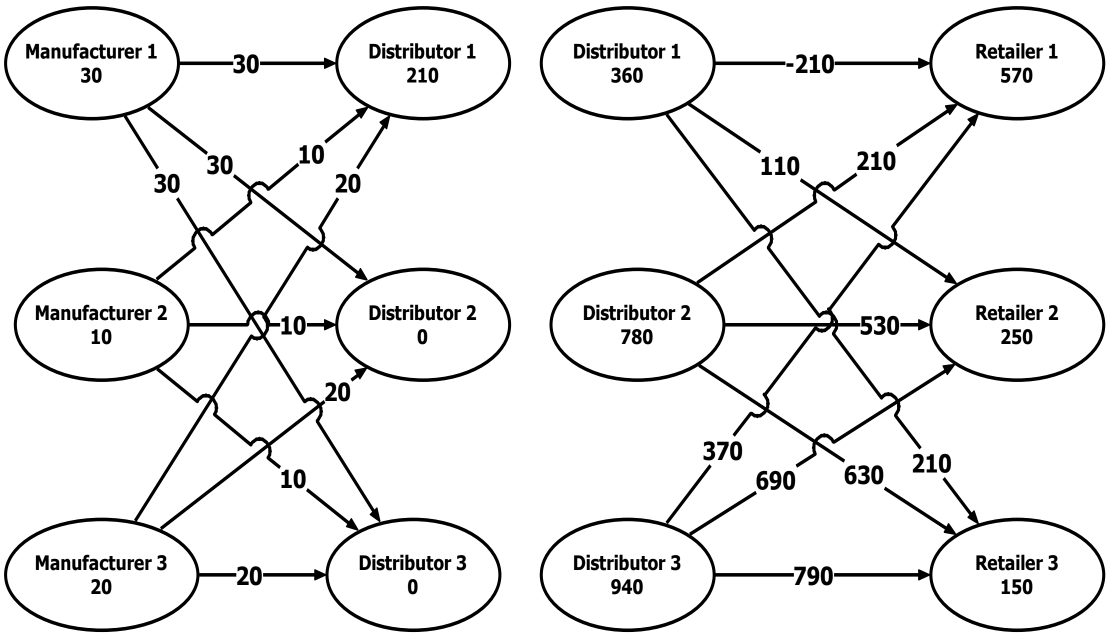


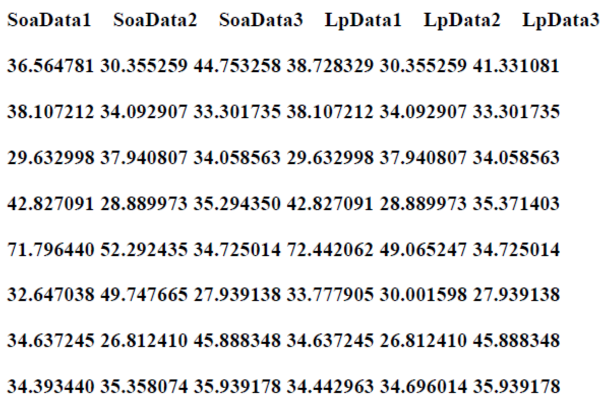
Figure 31 Network flow optimization model

4.3 Collection of Data

4.3.1 Collection Procedure

Data generation begins by clearing the data file and calling the data generation method 100 times to create 100 data points for each supply chain. Each call will generate a random scenario for the supply chain entities and pass it to the heuristic supply chain optimization module, which optimizes it, and calculates a percentage improvement in cost to fulfill a retail order, then adds it to a data structure. At the same time a copy of the same random scenario is sent to the linear programming solver, which optimizes it in its own way and calculates a percentage improvement in cost to fulfill the same retail order, and then adds it to a separate data structure. Both data structures are returned to a module that formats and writes them to an external file. See APPENDIX D.

Examples of the file format see Figure 32 the numbers are percentage improvement in cost to fulfill an order compared to non-optimized costs. See APPENDIX B for all the data.



```
SoaData1  SoaData2  SoaData3  LpData1  LpData2  LpData3
36.564781 30.355259 44.753258 38.728329 30.355259 41.331081
38.107212 34.092907 33.301735 38.107212 34.092907 33.301735
29.632998 37.940807 34.058563 29.632998 37.940807 34.058563
42.827091 28.889973 35.294350 42.827091 28.889973 35.371403
71.796440 52.292435 34.725014 72.442062 49.065247 34.725014
32.647038 49.747665 27.939138 33.777905 30.001598 27.939138
34.637245 26.812410 45.888348 34.637245 26.812410 45.888348
34.393440 35.358074 35.939178 34.442963 34.696014 35.939178
```

Figure 32 Example of Data file format

The data was analyzed with a free data analysis package from National Institute of Standards and Technology (NIST) called "Dataplot" it is written in FORTRAN and has been constantly improved over the last 30 years. They also have a data analysis manual see reference (NIST e-Handbook of Statistical Methods 2012).

4.4 Technique of Analysis

4.4.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach for data analysis that employs a variety of techniques (mostly graphical) to:

- Maximize insight into a data set;
- Uncover underlying structure;
- Extract important variables;
- Detect outliers and anomalies;
- Test underlying assumptions;

- Develop parsimonious models; and
- Determine optimal factor settings.

The EDA approach is an attitude about how a data analysis should be carried out. EDA is an approach to data analysis that postpones the usual assumptions about what kind of model the data follows with the more direct approach of allowing the data itself to reveal its underlying structure and model (NIST e-Handbook of Statistical Methods 2012 chapter 1).

The seminal work in EDA is Exploratory Data Analysis, Tukey, (1977). Over the years it has benefitted from other publications such as Data Analysis and Regression, Mosteller and Tukey (1977), Interactive Data Analysis, Hoaglin (1977), The ABC's of EDA, Velleman and Hoaglin (1981). Most EDA techniques are graphical in nature with a few quantitative techniques. The reason for the reliance on graphics is that the main role of EDA is to open-mindedly explore, and graphics draws upon the natural pattern-recognition capabilities that we all possess. (NIST e-Handbook of Statistical Methods 2012 chapter 1).

The particular graphical techniques employed in EDA are often quite simple, consisting of various techniques of:

- Plotting the raw data (such as data traces, histograms, bihistograms, probability plots, lag plots, block plots, and Youden plots.
- Plotting simple statistics such as mean plots, standard deviation plots, box plots, and main effects plots of the raw data.
- Positioning such plots so as to maximize our natural pattern-recognition abilities, such as using multiple plots per page.

Every measurement process, however complicated, has certain underlying assumptions. This section deals with what those assumptions are.

Underlying Assumptions

There are four assumptions that underlie all measurement processes; namely, that the data from the process behaves like:

- Random drawings;
- From a fixed distribution;
- With the distribution having a fixed location;
- The distribution having a fixed variation.

The "fixed location" differs for different problem types. The simplest problem type is univariate; that is, a single variable. For the univariate problem, the general model:

Response = deterministic component + random component, becomes

Response = constant + error (NIST e-Handbook of Statistical Methods 2012 chapter 1).

For this case, the "fixed location" is simply the unknown constant. The process is considered to be operating under constant conditions that produce a single column of data with the properties that:

- Data are uncorrelated with each other;
- Random component has a fixed distribution;
- Deterministic component consists of only a constant;
- Random component has a fixed variation.

The importance of the univariate model is that it can be extended to where the deterministic component is not just a constant, but a function of many variables, and the objective is to characterize and model the function. The point is that if the engineer succeeds in choosing a good model, then the differences (residuals) between the raw response data and the predicted values from the fitted model should themselves behave like a univariate process. The residuals from this univariate process fit will behave like:

- Random drawings;
- From a fixed distribution;
- With fixed location;
- Fixed variation.

Thus if the residuals from the fitted model behaves like the ideal, then the testing of the underlying assumptions becomes a tool for the validation and quality of fit of the chosen model. If the four underlying assumptions hold, then we have achieved probabilistic predictability--Moreover, if the four assumptions are valid, then the process is amenable to the generation of valid scientific and engineering conclusions (NIST e-Handbook of Statistical Methods 2012 chapter 1).

Techniques for Testing Assumptions

Because the validity of the final scientific/engineering conclusions is linked to the validity of the underlying univariate assumptions, it follows that there is a necessity that each one of the above four assumptions be routinely tested.

The following EDA techniques are simple, efficient, and powerful for the routine testing of underlying assumptions:

- Run sequence plot (Y_i versus i)
- Lag plot (Y_i versus Y_{i-k})
- Histogram (counts versus subgroups of Y)
- Normal probability plot (ordered Y versus theoretical ordered Y)

The four EDA plots can be juxtaposed for a quick look at the characteristics of the data. The example plots below are ordered as follows:

- Run sequence plot - upper left
- Lag plot - upper right
- Histogram - lower left
- Normal probability plot - lower right

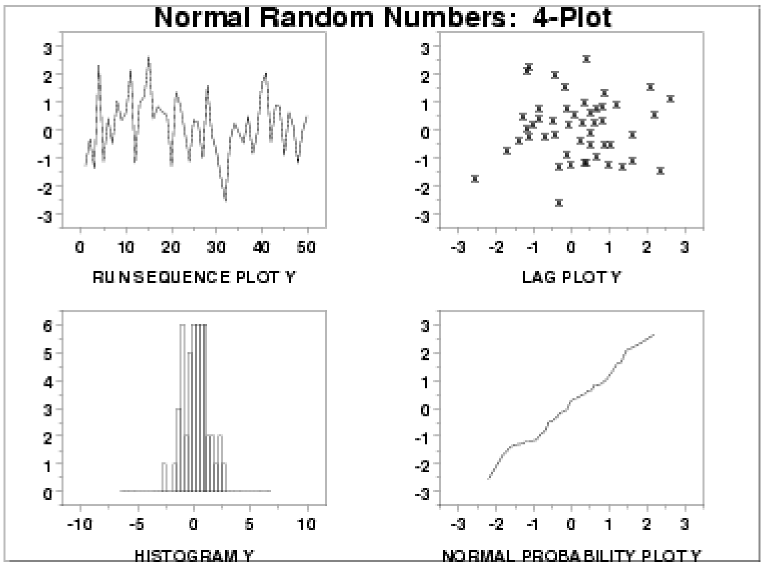


Figure 33 Example of a random process.

This 4-plot reveals a process that has fixed location, fixed variation, is random, apparently has a fixed approximately normal distribution, and has no outliers. If one or more of the four underlying assumptions do not hold, then it will show up in the various plots as demonstrated in the following example.

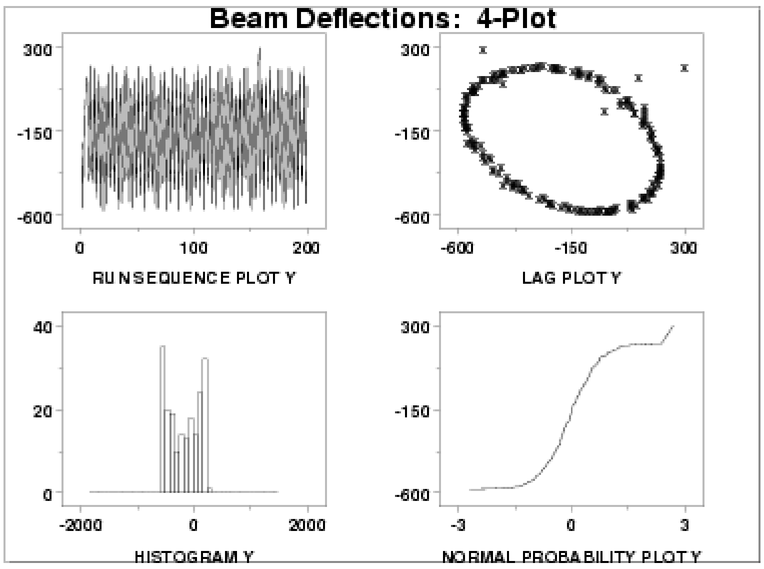


Figure 34 Example of a non-random process.

This 4-plot reveals a process that has fixed location, fixed variation, is non-random (oscillatory), has a non-normal, U-shaped distribution, and has several outliers. The four EDA plots discussed on the previous page are used to test the underlying assumptions:

- Fixed Location: If the fixed location assumption holds, then the run sequence plot will be flat and non-drifting.
- Fixed Variation: If the fixed variation assumption holds, then the vertical spread in the run sequence plot will be the approximately the same over the entire horizontal axis.
- Randomness: If the randomness assumption holds, then the lag plot will be structure less and random.
- Fixed Distribution: If the fixed distribution assumption holds, in particular if the fixed normal distribution holds, then the histogram will be bell-shaped, and the normal probability plot will be linear.

If all four of the assumptions hold, then the process is said by definition to be "in statistical control" (NIST e-Handbook of Statistical Methods 2012 chapter 1).

Consequences of Non-Randomness

The randomness assumption is the most critical but the least tested. If the randomness assumption does not hold, then:

- All of the usual statistical tests are invalid.
- The calculated uncertainties for commonly used statistics become meaningless.
- The calculated minimal sample size required for a pre-specified tolerance becomes meaningless.
- The simple model: y = constant + error becomes invalid.
- The parameter estimates become non-supportable.

One common type of non-randomness is autocorrelation. Autocorrelation is the correlation between Y_t and Y_{t-k}, where k is an integer that defines the lag for the autocorrelation. Autocorrelation is a time dependent non-randomness. This means that the value of the current point is highly dependent on the previous point if k = 1 (or k points ago if k is not 1). Autocorrelation is detected via an autocorrelation plot or a lag plot.

If the data are not random due to autocorrelation, then:

- Adjacent data values may be related.
- There may not be n independent snapshots of the phenomenon under study.
- There may be undetected "junk"-outliers.
- There may be undetected "information-rich"-outliers.

Consequences of Non-Fixed Location Parameter

The usual estimate of location is the mean

$$\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

For N measurements $(Y_1, Y_2, \dots, Y_N)$

If the run sequence plot does not support the assumption of fixed location, then

- The location may be drifting.
- The single location estimate may be meaningless (if the process is drifting).
- The choice of location estimator (e.g., the sample mean) may not be optimal.
- The usual formula for the uncertainty of the mean:

  May be invalid and the numerical value may be too small.
- The location estimate may be poor.
- The location estimate may be biased.

Consequences of Non-Fixed Variation Parameter

The usual estimate of variation is the standard deviation

$$s(\bar{Y}) = \frac{1}{\sqrt{(N-1)}}\sqrt{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$$

For N measurements $(Y_1, Y_2, \dots, Y_N)$

If the run sequence plot does not support the assumption of fixed variation, then

- The variation may be drifting. The single variation estimate may be meaningless (if the process variation is drifting).
- The variation estimate may be poor.
- The variation estimate may be biased.

Consequences Related to Distributional Assumptions

Scientists and engineers routinely use the mean (average) to estimate the "middle" of a distribution. It is not so well known that the variability and the noisiness of the mean as a location estimator are intrinsically linked with the underlying distribution. For certain distributions, the mean is a poor choice. For any given distribution, there exists an optimal choice-- that is, the estimator with minimum variability/noisiness. This optimal choice may be, for example, the median, the midrange, the mid mean, the mean, or something else. The implication is to "estimate" the distribution first, and then--based on the distribution--choose the optimal estimator. The resulting parameter estimators will have less variability if this approach is followed (NIST e-Handbook of Statistical Methods 2012 chapter 1).

Other consequences that flow from problems with distributional assumptions are:

Distribution

- The distribution may be changing.
- The single distribution estimate may be meaningless (if the process distribution is changing).
- The distribution may be markedly non-normal.
- The distribution may be unknown.
- The true probability distribution for the error may remain unknown.

Model

- The model may be changing.
- The single model estimate may be meaningless.
- The default model Y = constant + error may be invalid.
- If the default model is insufficient, information about a better model may remain undetected.

- A poor deterministic model may be fit.
- Information about an improved model may go undetected.

Process

- The process may be out-of-control.
- The process may be unpredictable.
- The process may be un-modelable.

(NIST e-Handbook of Statistical Methods 2012 chapter 1)

4.5 Analysis of Results

Based on the above discussion my analysis went like this: Dataplot was the data analysis package that I used to analyse and graph my data. Dataplot is a free data analysis package from the NIST statistical engineering division. It is written in FORTRAN and has been continually improved over the last 30 years, and comes with its own easy to understand scripting language. After reading the data file into Dataplot a 4-plot of each column of data (each column is a variable which is a supply chain optimized by either the heuristic method or linear programming method) was done to see if the variables percentage of improvement could be adequately modelled as an independent random sample drawn from a normal distribution. If the assumptions hold, then use parametric tests for each pair of variables (each supply chain optimized by heuristic and linear programming methods) to test if there is any difference between the two methods. The parametric tests to be used in this case are Student's t-test for equality between the means and the Fisher F-test for equality of variance. The null hypothesis is that the means and standard deviations are equal which would imply that each member of the pairs of variables came from the same normal distribution and that there is no difference between the optimization methods at a certain level of confidence. The alternative hypothesis is that there is a difference between the means and standard deviations which would imply that the members of each pair of variables came from different normal distributions and that there is a difference between the optimization methods for each supply chain at a certain level of confidence.

4.5.1 Outcomes

Figure 35 shows that for supply chain 1 using the heuristic optimization method that the 100 data points came from a fixed normal distribution. This is because based on a previous discussion the top left graph (run sequence plot) is flat and non-drifting which means the mean is constant and the vertical spread in the run sequence plot is approximately the same over the entire horizontal axis which means the variance is constant. Also the top right hand graph (lag plot) is structure less and random which means that the data are independent random samples (no autocorrelation). Also the bottom left hand graph (histogram plot) is bell shaped and the bottom right hand (normal probability plot) is linear which means that the data come from a normal distribution with a fixed mean and standard deviation.

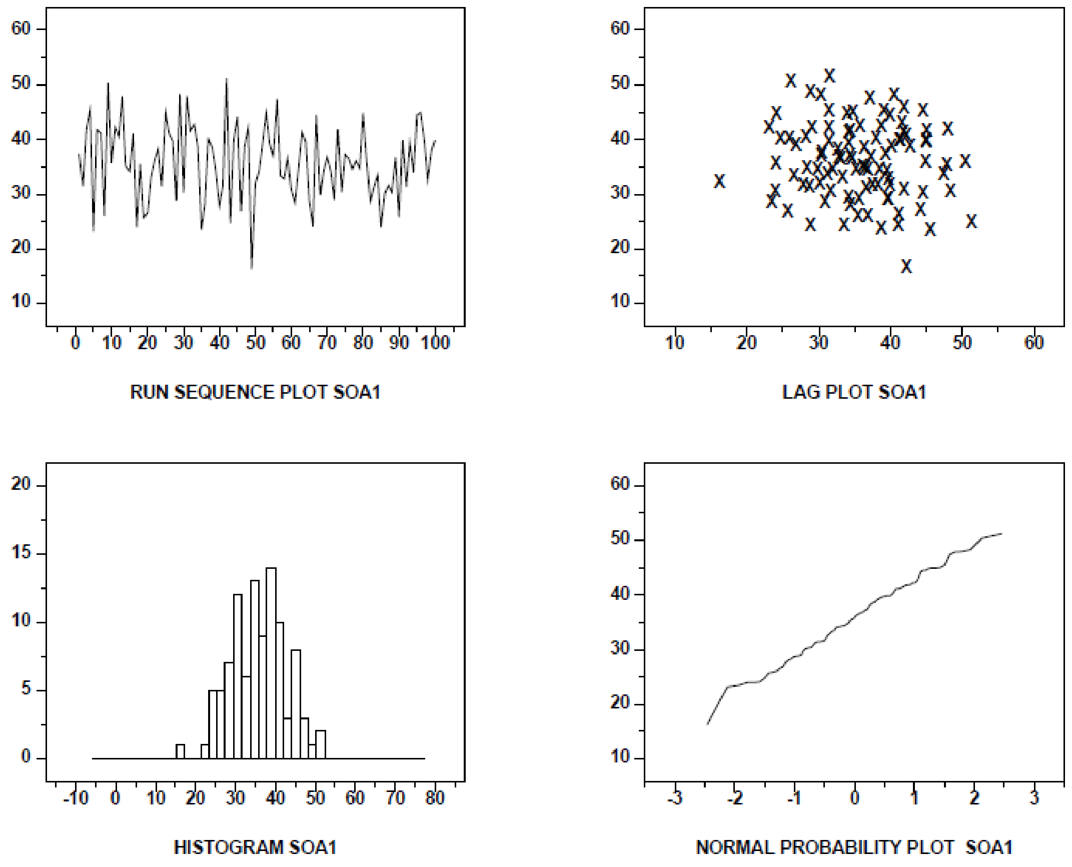


Figure 35 Supply chain 1 Heuristic 4-plot

Figure 36 shows that for supply chain 2 using the heuristic optimization method that the 100 data points came from a fixed normal distribution though with some more variability. This is because based on a previous discussion the top left graph (run sequence plot) is flat and non-drifting which means the mean is constant and the vertical spread in the run sequence plot is approximately the same over the entire horizontal axis which means the variance is constant. Also the top right hand graph (lag plot) is structure less and random which means that the data are independent random samples (no autocorrelation). Also the bottom left hand graph (histogram plot) is bell shaped and the bottom right hand (normal probability plot) is linear which means that the data come from a normal distribution with a fixed mean and standard deviation.

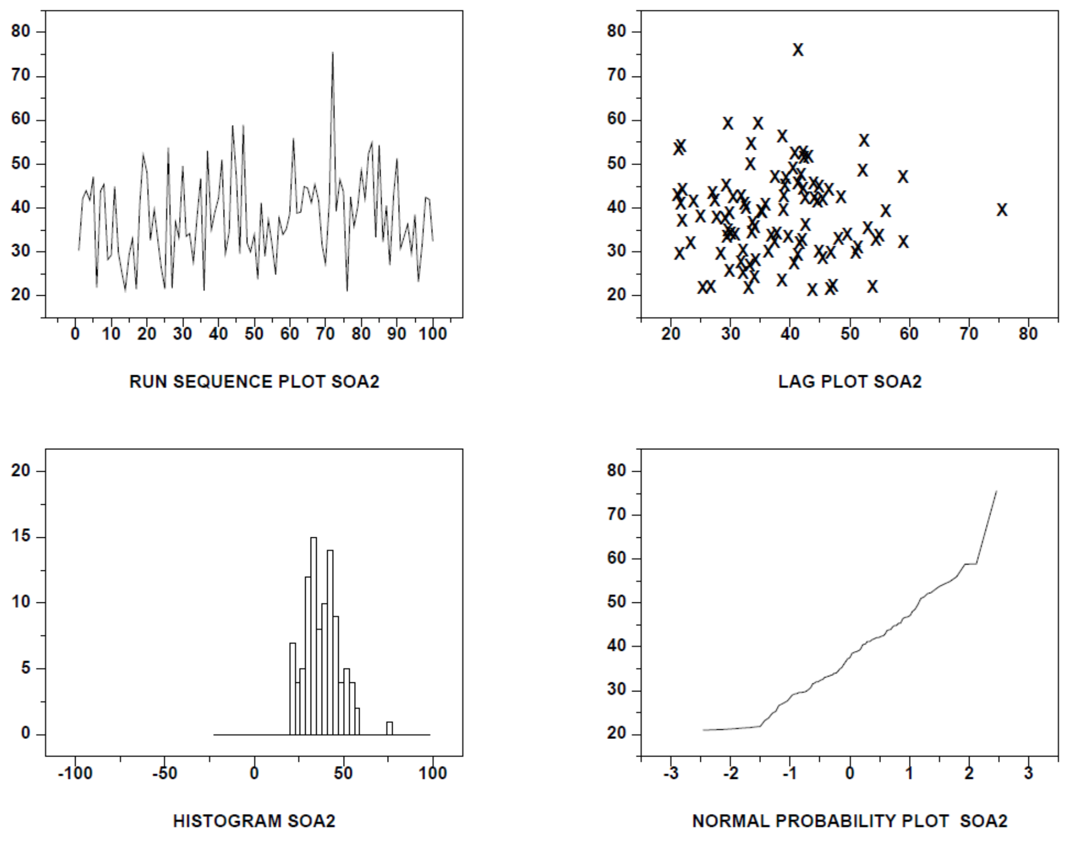


Figure 36 Supply chain 2 Heuristic 4-plot

Figure 37 shows that for supply chain 3 using the heuristic optimization method that the 100 data points came from a fixed normal distribution. This is because based on previous discussion the top left graph (run sequence plot) is flat and non-drifting which means the mean is constant and the vertical spread in the run sequence plot is approximately the same over the entire horizontal axis which means the variance is constant. Also the top right hand graph (lag plot) is structure less and random which means that the data are independent random samples (no autocorrelation). Also the bottom left hand graph (histogram plot) is bell shaped and the bottom right hand (normal probability plot) is linear which means that the data come from a normal distribution with a fixed mean and standard deviation.

The data from supply chain 2 has more variability than supply chains 1 and 3 this difference repeats in the following three sets of graphs for the linear programming optimization.

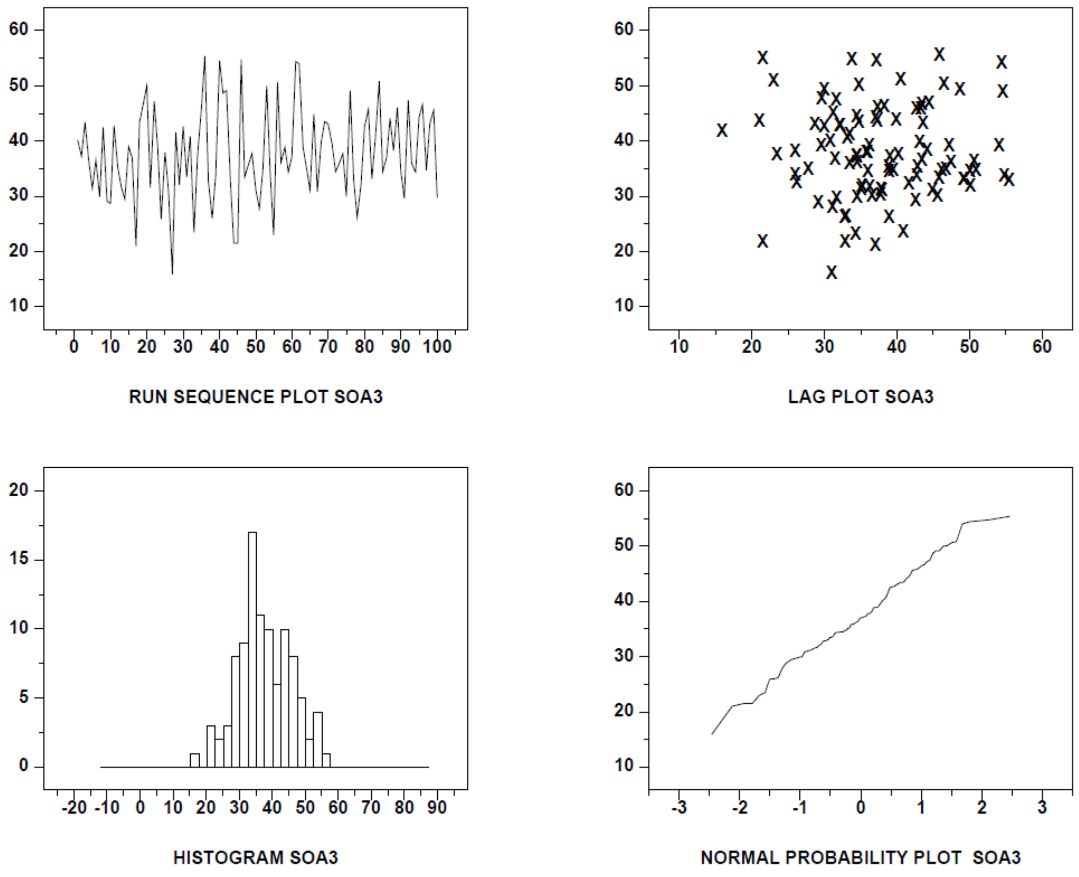


Figure 37 Supply chain 3 Heuristic 4-plot

Figure 38 shows that for supply chain 1 using the Linear programming optimization method that the 100 data points came from a fixed normal distribution. This is because based on previous discussion the top left graph (run sequence plot) is flat and non-drifting which means the mean is constant and the vertical spread in the run sequence plot is approximately the same over the entire horizontal axis which means the variance is constant. Also the top right hand graph (lag plot) is structure less and random which means that the data are independent random samples (no autocorrelation). Also the bottom left hand graph (histogram plot) is bell shaped and the bottom right hand (normal probability plot) is linear which means that the data come from a normal distribution with a fixed mean and standard deviation.

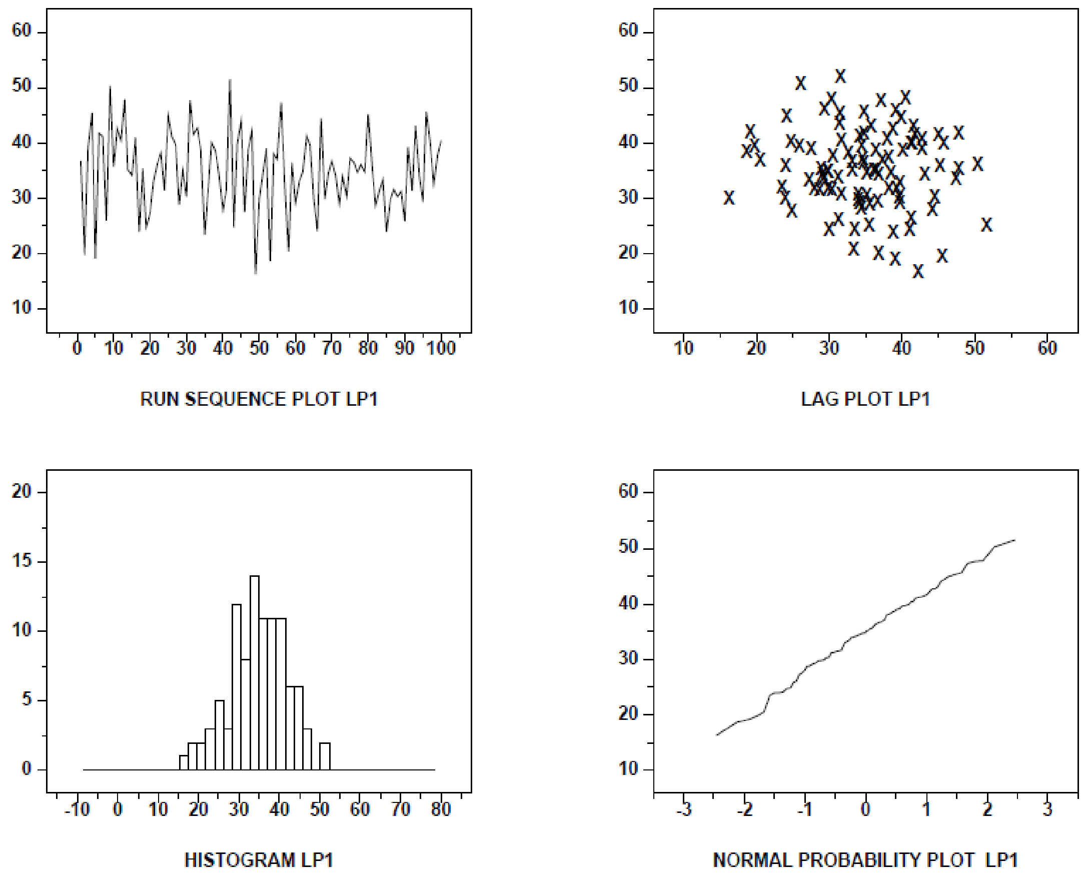


Figure 38 Supply chain 1 Linear Programming 4-plot

Figure 39 shows that for supply chain 2 using the Linear programming optimization method that the 100 data points came from a fixed normal distribution, though there is more variation and some outlier data points. This is from a possible bug in my code that in about 5% of cases the linear programming method results in an increase of order costs instead of a decrease. In my code for those cases, the values are forced to zero, notice those values in the top left graph (run plot) spiking downwards to zero and the drop off at the beginning of the bottom right graph (normal probability plot).

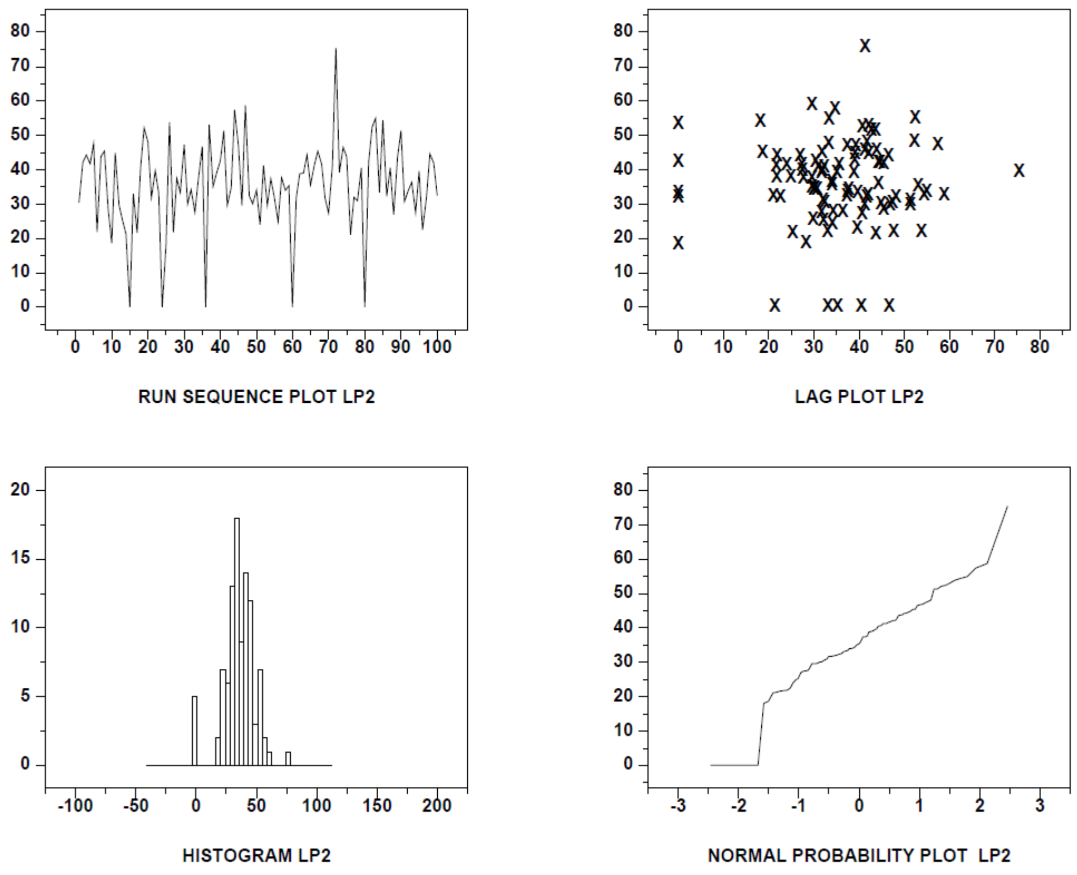


Figure 39 supply chain 2 Linear Programming 4-plot

Figure 40 shows that for supply chain 3 using the Linear programming optimization method that the 100 data points came from a fixed normal distribution. This is because based on previous discussion the top left graph (run sequence plot) is flat and non-drifting which means the mean is constant and the vertical spread in the run sequence plot is approximately the same over the entire horizontal axis which means the variance is constant. Also the top right hand graph (lag plot) is structure less and random which means that the data are independent random samples (no autocorrelation). Also the bottom left hand graph (histogram plot) is bell shaped and the bottom right hand (normal probability plot) is linear which means that the data come from a normal distribution with a fixed mean and standard deviation.

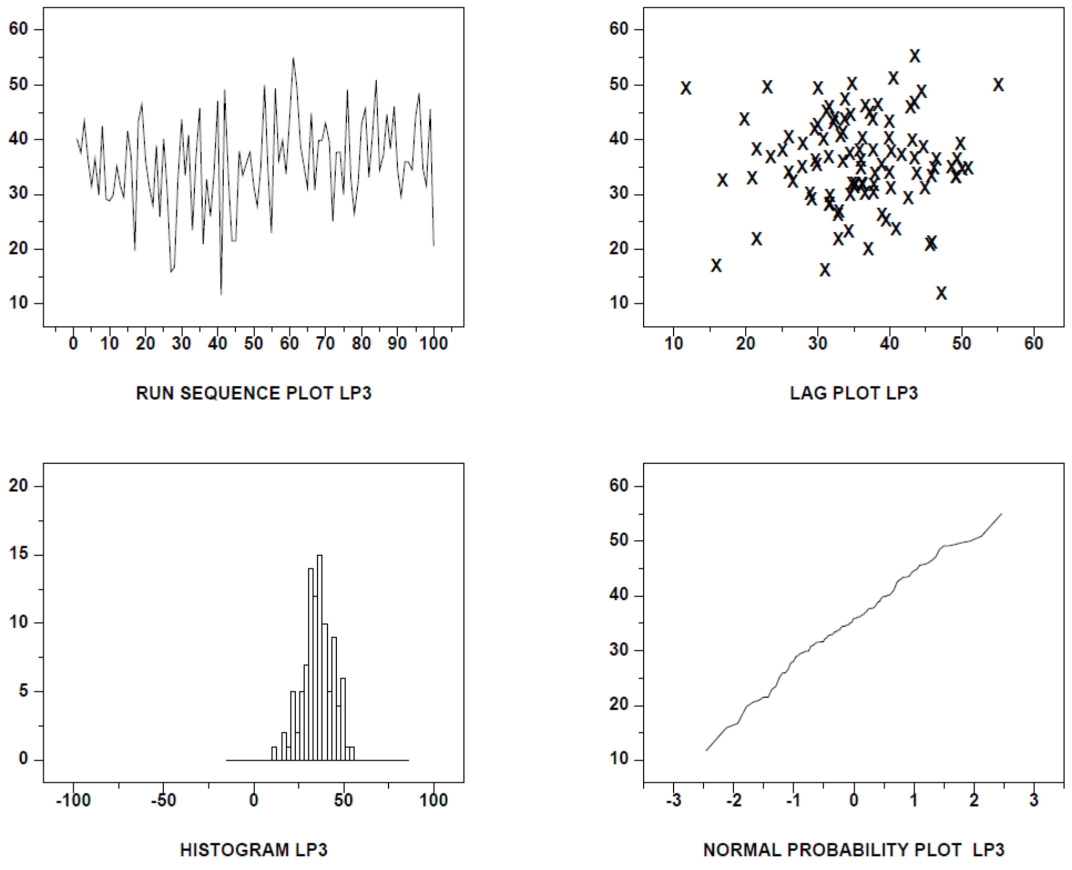


Figure 40 Supply chain 3 Linear Programming 4-plot

Another exploratory data analysis technique that Dataplot has for comparing two samples for equality of location and variation is called a Bi-Histogram this technique combines two histogram plots like in a mirror to see if they are similar in means and variability. The Bi-Histograms for each supply chain, plotting histograms of heuristic and linear programming improvement samples are shown see Figure 41 – Figure 43 they suggest that there is little difference in the means and standard deviations between the two optimization methods for each supply chain. This suggests they came from similar populations. As the samples appear to come from normal populations, parametric tests for mean and standard deviation would be appropriate to quantify the similarity between the two optimization methods. From these parametric tests (see Appendix E) at the 99% confidence level the null hypothesis that there is no appreciable difference between the two optimization methods is correct.

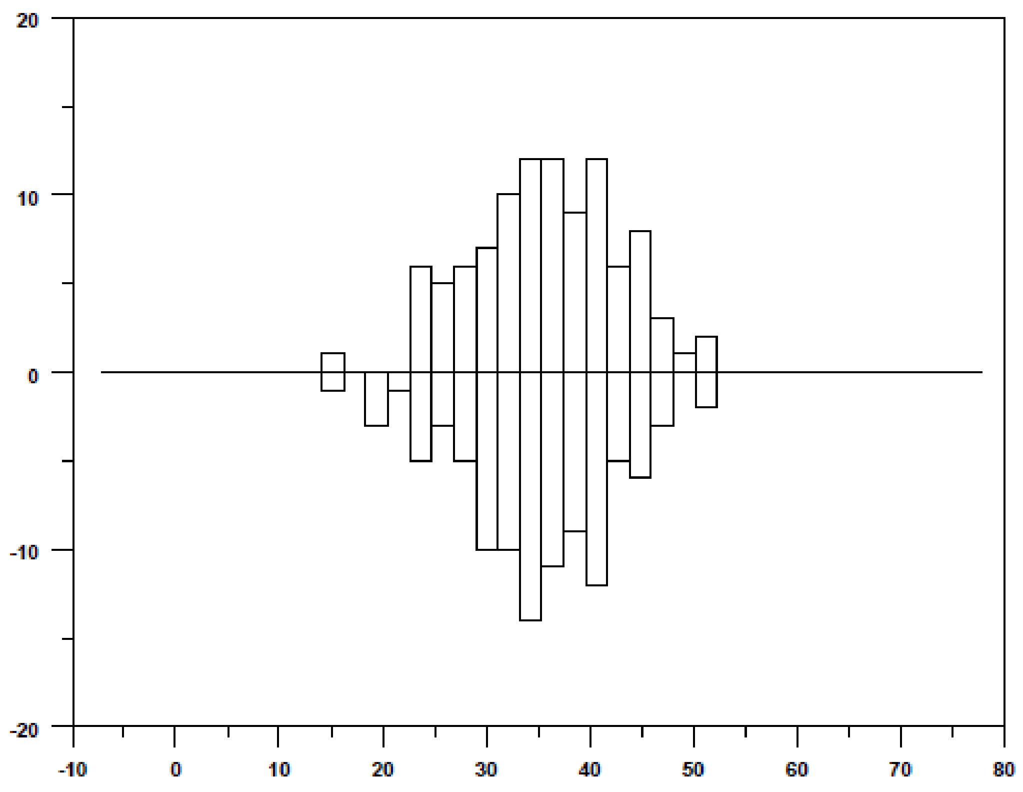


Figure 41 Supply chain 1 Bi-Histogram

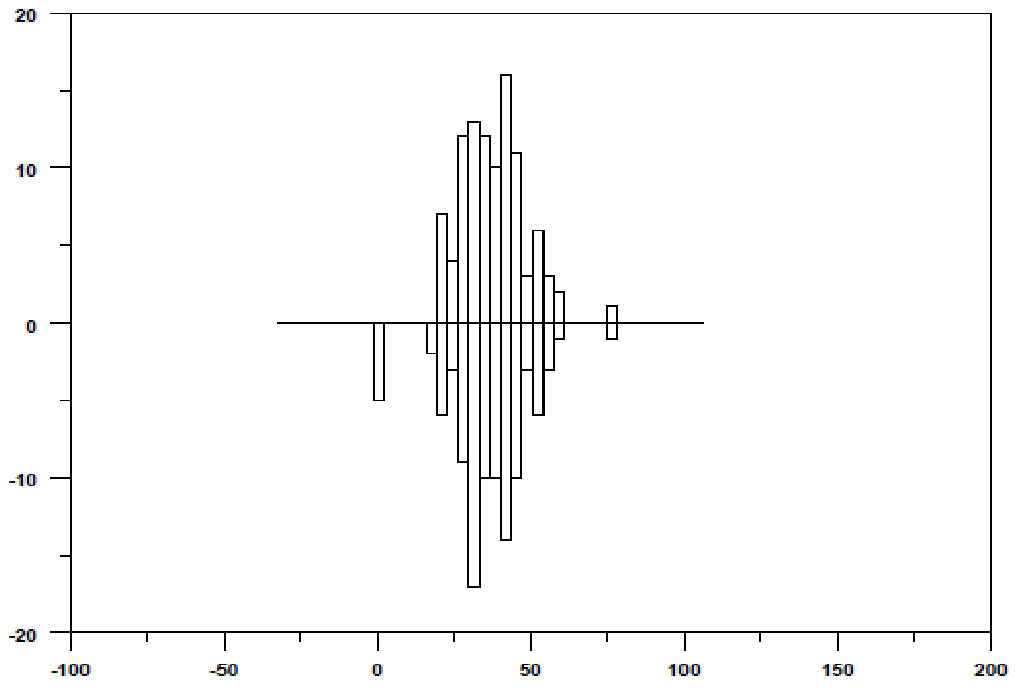


Figure 42 Supply chain 2 Bi-Histogram

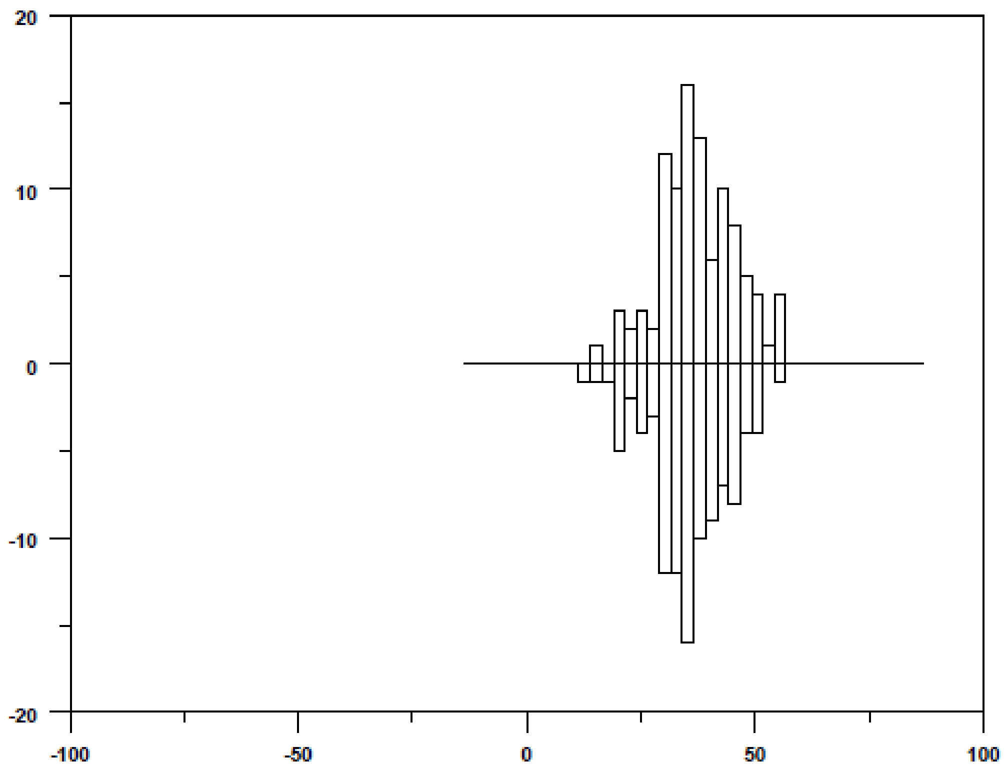


Figure 43 Supply chain 3 Bi-Histogram

See Appendix E for Dataplot output of the parametric tests.

The above results show that there is not much difference between the two optimization techniques. This is due to the fact that both are equally good at routing goods to the supply chain that needs the most help in fulfilling orders but the heuristic method was much easier to implement and manage changes in complexity as well as potentially being more efficient.

5. CHAPTER FIVE – Discussion and Conclusion

5.1 Overview

This chapter is a discussion of the research and conclusions reached; it also includes a discussion of the limitations of this research and direction for improvements with future research.

5.2 Recapitulation and Discussion

Electronic management of supply chains has been maturing for the last 20 years. The pervasiveness of the internet has made supply chain accessibility cost-effective and far-reaching into various business communities. There is great opportunity for more automation within the business landscape. The literature reveals various attempts along this route. This research has identified that there is need for more research into faster and more flexible supply chain optimisation strategies across multiple supply chains. This research adopts an SOA model to provide the glue for integrating loosely-coupled business services across multiple heterogeneous supply chains to answer the first research question raised. Concerning the second research question, a heuristically guided optimisation module has been developed and added to the framework. The framework integrates two very different techniques, i.e., heuristic and simplex techniques, and is open for additional planning techniques. All these resources can be applied and controlled by the rule-based inference engine. The research makes extensive use of and relies on existing resources exposed as services. This allows integration, monitoring and control to be far easier within SOA. Preliminary results showed that optimised supply chains delivered much better time and cost savings for each retail request, compared with traditional static supply chains. Also that of the two techniques used for optimization of supply chains there was no significant difference between them in savings in compared with traditional static supply chains.

5.3 Study Limitations

The limitations of this research are that the data was simulation data not real world data. This was unfortunate, but was unavoidable, as the sources for real world data I approached were not willing to share their supply chain data; not even anonymously. After consulting a supply chain academic, who stated that companies would be unwilling to share data, as it gave away too much knowledge of their business and in essence, could be considered their intellectual property?

5.4 Directions for Future Research

Some further research that is planned, is to see if routing rules can be acquired automatically by machine learning techniques. Also additional research that is planned is to see if multi-criteria optimization methods might be helpful in optimization of supply chains. We also hope to have a better visual overview of supply chain interactions, of which an example is shown in Figure 20. An additional research future research improvement is keeping a log of each user's actions so that in case of failure, safe recovery mechanisms could be called into play. This would be essential for a real world system. Some extra supply chain contextual information could be logged as well which would be useful for auditing purposes as well as to provide for more complex reasoning rules. It would also be good to have access to the infrastructure from tablets and mobile devices. The web pages would be smaller and simpler to interact with, but be essentially unchanged. Any device capable of running a browser could use web services to interact with my system. An extra security layer would be needed to prevent unauthorized access.

REFERENCES

Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. (1993) *Network Flows: Theory, Algorithms, and Applications,* Prentice Hall.

Bailey, J., Papamarkos, G., Poulovassilis, A., Wood, P.T., (2002) *An Event-Condition-Action Language for XML* in Proc. WWW '2002, pages 486-495, Hawaii, 2002.

Bowersox, D.J., Closs, D.J. and Stank, T.P. (1999) *21st Century Logistics: Making Supply Chain Integration a Reality*, Council of Logistics Management, Michigan State University.

Chopra, S. and Meindl, P. (2004) Supply *Chain Management: Strategy, Planning, and Operation,* Prentice-Hall Publishing.

Christopher, M., Peck, H. and Towill, D. (2006) 'A taxonomy for selecting global supply chain strategies', *The International Journal of Logistics Management*, Vol. 17, No. 2, pp.277–287.

Christopher, M. and Towill, D. (2001) 'An integrated model for the design of agile supply chains', *International Journal of Physical Distribution & Logistics Management*, Vol. 31, pp.235–246.

Concurrency and Application Design (2013) available at https://developer.apple.com/library/ios/DOCUMENTATION/General/Conceptual/ConcurrencyProgrammingGuide/ConcurrencyandApplicationDesign/ConcurrencyandApplicationDesign.html, (accessed 1st November 2013).

Dai, W., Moynihan, P., Gou, J., Zou, P., Yang, X., Chen, T. and Wan, X. (2007) 'Services oriented knowledge-based supply chain application', *Proceedings of the 2007 IEEE International Conference on Services Computing*, IEEE Computer Society, Los Alamitos, CA, USA, pp.660–667, available at http://www.conferences.computer.org/scc/2007/ (3rd place award contest team).

Emerson, D. and Piramuthu, S. (2004) 'Agent-based framework for dynamic supply chain configuration', *Proceedings of the 37th Hawaii International Conference on System Sciences,* pp.1–9.

NIST/SEMATECH e-Handbook of Statistical Methods (2012) available at http://www.itl.nist.gov/div898/handbook/, (accessed 24th January 2012).

Fayaz, M., Rabello, L. and Mollaghasemi, M. (2005) ‘Ontologies for supply chain simulation modeling’, *Proceedings of the 2005 Winter Simulation Conference*.

Fayazbakhsh, K. and Razzazi, M. (2008) ‘Coordination of a multi-commodity supply chain with multiple members using flow networks’, *IEEE Second International Conference on the Digital Society,* pp.25–30.

Goldberg, A.V., Tardos, E., and Tarjan, R.E. (1990) *Network Flow Algorithms, Paths, Flows, and VLSI-Layout,* Springer-Verlag, pp.101–164.

Hetherington, M. and Ismail, H.S. (2006) ‘Qualitative examination of how agility and agile manufacturing fit with traditional strategy and the Triz framework’.

Horvath, L. (2001) ‘Collaboration: the key to value creation in supply chain management’, *Supply Chain Management: An International Journal*, Vol. 6, No. 5, pp.205–207.

Kumar, S., Dakshinamoorthy, V. and Krishnan, M.S. (2007) ‘Does SOA improve the supply chain? An empirical analysis of the impact of SOA adoption on electronic supply chain performance’, *Proceedings of the 40th Hawaii International Conference on System Sciences*, pp.1–10.

Lee, H.L. and Bianchi, C. (1995) ‘The evolution of supply chain management models and practices at Hewlett Packard’, *Interface,* Vol. 25, No. 5, pp.42–63.

Lim, B. and Wen, J. (2003) ‘Web services: an analysis of the technology, its benefits, and implementation difficulties’, *Information Systems Management.*

Linear Programming (2012) available at
http://en.wikipedia.org/wiki/Linear_programming.htm (accessed 7th January 2012).

Mason, D., Naylor, B. and Towill, D.R. (2000) ‘Engineering the legible supply chain’, *International Journal of Agile Managements Systems*, Vol. 2, No. 1, pp.54–61.

Moynihan, P. and Dai, W. (2010) ‘Knowledge-based service system for supply chain management’, *Proceedings of the 7th IEEE International Conference on Service Systems and Service Management*, June, pp.591–596, IEEE Press, Tokyo, Japan.

Operations Research (2012) available at
http://en.wikipedia.org/wiki/Operations_research.htm (accessed 7th January 2012).

Papazoglou, M.P. and Georgakopoulos, D. (2003) ‘Service-oriented computing’, *Communications of the ACM*, Vol. 46, No. 10, pp.24–25.

Pomykalski, J.J., Truszkowski, W.F. and Brown, D.E. (1999) ‘Expert systems’, in Webster, J. (Ed.): *Wiley Encyclopedia for Electrical and Electronics Engineering*.

Reveliotis, S. (2003) *An Introduction to Linear Programming and the Simplex Algorithm*, available at http://www.isye.gatech.edu/~spyros/index.html> (accessed on 22 February 2010).

Stonebraker, P.W. and Liao, F. (2006) 'Supply chain integration: exploring product and environmental contingencies', *Supply Chain Management: An International Journal*, Vol. 11, No. 1, pp.34–43.

Strohmaier, M. and Rollett, H. (2005) 'Future research challenges in business agility – time, control and information systems', *Proceedings of the 2005 Seventh IEEE International Conference on E-Commerce Technology Workshops* (CECW'05), pp.1–7.

Vanderbei, R. (2007) 'Linear Programming' chapter 5, *Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544,* pp.4-5.

Weizi, L., Huiying, G., Zhijun, Y. and Kecheng, L. (2008) 'On dynamic agility of inter-organizational processes', *IEEE Xplore,* pp.1–5.

Wu, X., Fang, G. and Wu, Z. (2006) 'The dynamic IT capabilities and firm agility: a resource-based perspective.'

Yi-nan, Q. and Zhao-fang, C. (2009) 'The impact of supply chain strategies on supply chain integration', *2009 International Conference on Management Science & Engineering*, 14–16 September, pp.534–540, Moscow, Russia.

Zsiflovits, H.E. and Engelhardt-Nowitzki, C. (2004) 'An analysis of frameworks for measuring supply chain agility', *IEEE Xplore*, pp.87–95.

APPENDIX A

The following discussion is drawn heavily from my development diary (the unashamed use of the first person part of speech). It shows the complexity that can arise from creating innovative software.

To test the finished system I thought I would use a website stress testing tool called "The Grinder3", it has a learning curve as the tests have to be written in the Python language, which I used to be familiar with, but haven't used for a while. It is a very powerful tool. But in the end, it took too long to set up to get a suit of website stress tests to run and so; I wrote my own humble test harness tool in java.

I worked on the load facts module on the client side. Testing had shown that the client side module needs to be isolated to a single class to prevent multiple differing web service calls creating unnecessary threads and connections to the database. The server side code was done and it is now a part of the INDEX web service. At the time it still had some problems. I fixed two main bugs that were causing INDEX to throw two SQL exceptions. One was that I needed to test that the new ddp(threadnumber) table that was created upon the receiving of a new http connection, was not already existing in the database (I have to query the database first before creating the table).

And the second one was that the method to clear the table was not passed the right thread number so couldn't select the right table name. These are now fixed and there are no more exceptions but there was still erratic behaviour, all the facts were not added to the right table.

Another thread is interrupting, causing the final facts to be added to another threads working memory table. Some of this is due to the client application adding facts from an external class, which causes another web service call, which creates another http connection, which causes another instance of INDEX to be created with its own working memory table. The threads need to be more isolated from each other. I may use thread-local variables to provide this isolation.

To get the load facts module to load facts within the same thread as the http connection (this was to ensure that the right working memory table was loaded with facts, as a new table was created for each call to INDEX for inferencing services.) I needed to load them all within the same web service call. To do this the dynamic invoker had to be able to pass an array of the facts to the web service. I had modified the dynamic invoker (a program to call a generic web service) to be able to pass an array of strings to the web service. At first I thought this had worked (no errors), but until I had modified INDEX and rebuilt the service I didn’t know for sure. Unfortunately after I had done this the dynamic invoker started throwing errors up.

Axis 1.4 (a java API for adding web service functionality to applications) will allow simple arrays of primitive types to be passed across the protocol but not arrays of arrays (I later found this to be wrong). I realized then that I was attempting to pass an array of string arrays. (An array of facts, each fact has an object, attribute and value strings.) If each fact were packed into a string with a delimiter character separating each object attribute and value string, then there would be only a simple array of strings to be passed to the web service. At the time and after searching on the web I formed the conclusion that I would probably need to migrate to axis2.

Some other ideas that I considered then were having a static singleton class manage http sessions with transaction ID’s but that extra housekeeping code in the end wasn’t necessary.

Getting the web service infrastructure to pass bulk data was the first short term goal. I obtained the source code and java docs for axis-1.4, jaxrpc, and wsdl4j. I set these up so that eclipse (a java development tool) would be able to access them so I could set breakpoints inside the axis libraries and have pop-up windows for the documentation. I then wrote a web service to test passing an array of strings to and from a web service. The web service test code worked when called from a local main method.

I used my normal procedure to build and deploy it as an axis mediated web service inside tomcat. It built normally but when I tried to deploy it, the axis admin client raised a strange axis error that I had never seen before. I checked the procedure and the code and everything was ok, the procedure and libraries worked flawlessly if you followed the procedure rigidly. After a few hours of checking, and trying other code, I concluded that my axis installation had just gone bad, and I would have to replace it entirely. I restored a clean version from my archives, and the web service deployed perfectly. But when I ran the application, there were errors from the dynamic invoker from the client side this time (I didn't know this at the time, I thought it might have been a fault in the server side). I kept adding libraries to the client side until the errors subsided; eight libraries (approx. 2 MB) are the minimum needed to run the dynamic invoker to be able to call a web service.

After getting all this working the web service still wouldn't pass a complete array only the name back and forth from the web service to the dynamic invoker. I read all of the axis documentation and set some hidden settings (server-conf.wsdd is the file to configure axis capability) it still wouldn't work, the documentation said that axis can pass arrays of various type and java beans and all kinds of mime attachments. Then I tried all the samples, most of them didn't work, they were composed of conglomerations from various other projects, they were there to give you ideas not necessarily working code. I tried pieces of code from the various samples but still only a single item from the array would be sent or received.

I also tried a couple of other web service technologies Axis2 and JaxWS (apache cxf is the implementation). Cxf crashed my eclipse installation, it might be a good system but I will need a better computer. Axis2 didn't have a dynamic invoker; I would have to create one from scratch, again a good system but a lot more time to get it to do the simple thing that the project needed.

I decided that I would write my own and use a packed string to pass facts across the web service barrier. This I know will work, as the dynamic invoker can pass single strings and numbers to and from a web service.

At last I now know why the dynamic invoker would not pass arrays even though axis does support them. First a little history, the original dynamic invoker was written by "Davanum Srinivas" (who worked for Yahoo) in 2001 as a command line tool to test a web service installation. It uses JAXRPC (Java XML extensions for Remote Procedure Calls) and axis calls to collect data from the command line and pass it to the web service from any environment, it is completely stub less (you don't need receiving code on the web service side for it to work). As complex data is not usually passed from the command line Davanum used a simple serializer (a program to manage data transfer across a protocol) from axis to manage the data types. A simple serializer will only pass simple strings and numbers. Axis also comes with other serializers, an array serializer for instance which is the default, but is not used by the current dynamic invoker.

I did not alter the dynamic invoker in the short term (murphy's law had leaned heavily on me and I didn't want to waste any more time) I used packed strings to pass the facts across, in java (for my application) a single packed string will be able to carry about a 1000 facts and I set up the INDEX service to be able to pass five packed strings back and forth in one go, this will be sufficient in the short term, I will alter the dynamic invoker if in the future I need more functionality.

The load facts module took longer than I thought. I first built a simple version using string arrays; this would pack an array of strings into a single string separated by commas (CSV's). I also wrote a method that would unpack a packed string into an array of strings. These all worked perfectly. I then wrote a simple web service that I could send a packed string to, and it would unpack the string internally. This worked as a java application. I then packaged and deployed it as a web service in Tomcat. This worked perfectly, verifying that it was possible to send multiple facts and or rules to a web service in one call.

I then tried to retrofit the "SOADemo" and INDEX to see if this approach would work for them. I first tried with the "SOADemo". Instead of using a String array for the client side to aggregate the facts, I used a String Buffer object (similar to a String Builder object but thread safe) to build up the packed string as I went. I created a "loadFacts" method to handle the client side fact aggregation.

This method was to be called multiple times throughout the "SOADemo" whenever a fact triplet was to be sent to INDEX. This method would accept a fact triplet (object, attribute and value) plus a Boolean (true or false) variable called "sendFacts", this variable if set to false was to tell the "loadFacts" method to both aggregate the fact triplet and return for more facts to aggregate, or if set to true to aggregate the fact triplet and send the fact aggregation to INDEX. This was to allow the reuse of the "loadFacts" method and tell the method when enough facts had been collected to send to the INDEX web service for inferencing.

This client side method worked perfectly as a java application. To test it as a web browser application I had to set up three programs to run concurrently, an Eclipse debugging application, a debug version of Tomcat and a web browser to initiate the "SOADemo". A stub was inserted into INDEX to receive the packed string.

There were quite a few problems to get this to work properly. Some of which was that the debug version of Tomcat is a separate application to the Tomcat usually used as a server. This debug version uses "jpda" technology to enable it to act as a client server so that Eclipse can send and receive debugging information on port 8000. The start-up script for the debug version of Tomcat has to have some magic commands added to it. This debug version of tomcat was so that you can debug servlets running inside of the Tomcat container.

The problem of using the debug version of tomcat was in moving a copy of the "SOADemo" and INDEX web service to a separate version of Tomcat and not editing some of the configuration files for the demo and INDEX. They needed some path strings changed to reflect the location of the new version. Anyway after tracking down all the errors and fixing them, I was able to verify that the right packed strings were getting correctly passed to INDEX.

I then started on the Server side with INDEX. I wrote an extended method inside INDEX that unpacked the strings sent over from the client side, and added this data to a thread created working memory table called "ddp (threadnumber)". This worked perfectly. The problem now, was that the facts (though all added to the database table on the same thread of execution) was still a different thread than the client call, to start inferencing on the added data.

The adding of facts and the command to start inferencing needs to be on the same "http" connection thread, so that the facts go into the right working memory table, because the temporary working memory table name is created for each "http" connection thread number. Because of this requirement I decided that the load facts

“INDEX “method should immediately call the inferencing subsystem after loading the facts into ddp. This was logically what always happened anyway, but a separate web service call to initiate inferencing was always made. This testing showed the extra web service call actually initiated another “http” connection, which didn’t matter in the single threaded version of INDEX, but is an important difference in a multi-threaded version. Also how do you easily track which data or commands go with which http connection over time, and have this scalable?

I made the load facts module in INDEX call the inferencing subsystem. This worked; a single “http” connection thread handled the loading of facts and inferencing and returned the right results to the user that initiated the service.

The multi-threaded version of INDEX now worked, but it only worked under the Eclipse debugger, this is because the debugger makes the INDEX worker thread wait until SQLServer has returned from its database update. Outside the debugger the thread doesn’t wait long enough and the data is not added to the right working memory table which causes “no conclusion” to be returned. So far making the thread sleep for 100ms was not enough time (I later found if I fixed the code it wouldn’t need any wait). It was exciting the project was nearly finished.

I got INDEX working outside the debugger. A thread sleep of 300ms got INDEX and “SQLServer” communicating together allowing the facts to be added to the temporary working memory which then allowed INDEX to come to the right conclusion and the results to be returned to the webpage properly. On moving INDEX to the production version of Tomcat I found that I had to increase the thread sleep time to 500ms.

There were some significant bugs still left in the SOADemo notably in INDEX. The demo did run better, the bugs were typically, for a multithreaded application, intermittent; and they would significantly compromise scalability.

Now I will talk about the last part of INDEX development and testing. I used a program called "FindBugs" it is a program similar to C's Lint in that it reads source code and identifies stylistic inconsistencies and most importantly potential errors.

It identified about 50 errors on the first run which I fixed. A snippet of the output follows:

```
com.index.inference.core.DDP.create(String, String, String) may fail to close
Statement    DDP.java    /ConcurrentIndex/src/com/index/inference/core    line
243    FindBugs Problem (Normal Priority)
com.index.inference.core.DDP.updateOA(String, String, String) may fail to close
Statement    DDP.java    /ConcurrentIndex/src/com/index/inference/core    line
184    FindBugs Problem (Normal Priority)
```

The main progress in resolving the faults occurred when I created a better testing framework. This was one that by passed the internet and let me test INDEX threading as a standalone application. This allowed me to see clearly what was going on and fix errors and inefficiencies.

Because of this better testing, I rewrote some of the Multi-threaded parts of INDEX. This created a lot more stability and fixed the intermittent bugs, and I didn't need to add any more wait states to enable thread synchronization.

There was one last bug still to be fixed. For a reason I can't recall I removed the logging framework from INDEX, most likely to reduce the logging output and found another bug (the last major one I hope) that was hiding behind the logging framework thread (the logging framework runs in its own thread). This had been masking a thread issue with the database. It only occurred intermittently when the thread timing was just right. The first thread that closed the database connection caused the other threads to raise an exception about the database connection being closed.

In a typical run when this bug occurs, thread “main-1” finishes first and closes the database connection and returns the correct conclusion, and when thread “main-2” tries to access the database and close it the SQLServer driver raises an exception, as it can’t be closed twice, this is a bug in the JDBC SQLServer driver. I quote from an Internet blog:

“Apparently with the MSSQL driver (and also the Data Direct Driver) you cannot close a connection twice. If you do so, something will get screwed in the driver. It won’t show up all the time, just when you push the driver. So check your code and make sure you don't close a connection twice. Data Direct is responsible for making the MS driver, so perhaps someone should tell these guys about the problem. “

This causes thread “main-2” to return “no conclusion”

After investigating this situation I was able to ascertain that two classes are responsible for the database connection thread race condition. They are KDS.java and Rule.java. They connect and close the connection with the database, differently from the rest of INDEX.

The solution is to make KDS and Rule conform to the rest of INDEX (quick to do) or the make the other classes conform to KDS and Rule (slower to do). I hope to try the easy way soon. Update! I still haven’t fixed it, INDEX keeps running flawlessly. I will have to fix it sometime.

A snippet from the log of a successful run follows:

```
Thread: main-1 running
retailQuant[0] = 520.0
retailQuant[1] = 430.0
retailQuant[2] = 100.0
distribQuant[0] = 690.0
distribQuant[1] = 940.0
distribQuant[2] = 160.0
manufactQuant[0] = 10.0
manufactQuant[1] = 90.0
manufactQuant[2] = 40.0
```

```
Some Facts are =
retailQuant0,<,distribQuant0,retailQuant1,<,distribQuant1,retailQuant2,<,distribQuan
t2,retailTime0,=,0.0,retailTime1,=,0.0,retailTime2,=,0.0,

Thread: main-2 running
retailQuant[0] = 130.0
retailQuant[1] = 370.0
retailQuant[2] = 890.0
distribQuant[0] = 620.0
distribQuant[1] = 700.0
distribQuant[2] = 670.0
manufactQuant[0] = 90.0
manufactQuant[1] = 50.0
manufactQuant[2] = 100.0

Some Facts are =
retailQuant0,<,distribQuant0,retailQuant1,<,distribQuant1,retailQuant2,<,distribQuan
t2,retailTime0,=,0.0,retailTime1,=,0.0,retailTime2,=,0.0,retailQuant0,<,distribQuant0,
retailQuant1,<,distribQuant1,retailQuant2,>,distribQuant2,retailTime0,=,0.0,retailTi
me1,=,0.0,retailTime2,=,2.0,

thread [main-1]  INFO InferenceEngine:39 - Entering InferenceEngine()
thread [main-1]  INFO CreateDestroyTable:265 - Entering CreateDestroyTable()
main-1

thread [main-2]  INFO InferenceEngine:39 - Entering InferenceEngine()
thread [main-2]  INFO CreateDestroyTable:265 - Entering CreateDestroyTable()
main-2

thread [main-1]  INFO DataSourceManager:23 - Entering DataSourceManager()
thread [main-1]  INFO DataSource:19 - Entering DataSource()
thread [main-1]  INFO GDI:18 - Entering GDI()
thread [main-1]  INFO GoalDriven:28 - Entering GoalDriven()

thread [main-2]  INFO GDI:18 - Entering GDI()
thread [main-2]  INFO GoalDriven:28 - Entering GoalDriven()
thread [main-2]  INFO ICS:56 - Entering ICS()

thread [main-1]  INFO ICS:56 - Entering ICS()
thread [main-1]  INFO KDS:28 - Entering KDS()
. . .
thread [main-1]  INFO CreateDestroyTable:464 - DeleteTable Drop TABLE ddp1
. . .
thread [main-2]  INFO Fact:19 - Entering Fact()
thread [main-2]  INFO CreateDestroyTable:464 - DeleteTable Drop TABLE ddp2

Inference results =
<?xml version="1.0" encoding="utf-8" ?>
<conclusion>
	<result number="1">
```

```
                <ObjectName>optimize</ObjectName>
                <ObjectAttribute>retailTime</ObjectAttribute>
                <ObjectValue>2</ObjectValue>
        </result>
        . . .
        <result number="9">
                <ObjectName>optimize</ObjectName>
                <ObjectAttribute>add</ObjectAttribute>
                <ObjectValue>productQuant2Batch</ObjectValue>
        </result>
        <result number="10">
                <ObjectName>add distrib2Retail2Diff</ObjectName>
                <ObjectAttribute>to</ObjectAttribute>
                <ObjectValue>distributorBatch</ObjectValue>
        </result>
</conclusion>

Inference results =
<?xml version="1.0" encoding="utf-8" ?>
<conclusion>
        <result number="1">
                <ObjectName>retailTime0</ObjectName>
                <ObjectAttribute>=</ObjectAttribute>
                <ObjectValue>0</ObjectValue>
        </result>
        <result number="2">
                <ObjectName>optimize</ObjectName>
                <ObjectAttribute>add</ObjectAttribute>
                <ObjectValue>productQuant0Batch</ObjectValue>
        </result>
        .  .  .
        <result number="8">
                <ObjectName>optimize</ObjectName>
                <ObjectAttribute>add</ObjectAttribute>
                <ObjectValue>productQuant2Batch</ObjectValue>
        </result>
        <result number="9">
                <ObjectName>add distrib2Retail2Diff</ObjectName>
                <ObjectAttribute>to</ObjectAttribute>
                <ObjectValue>distributorBatch</ObjectValue>
        </result>
</conclusion>
```

The log snippet shows that the inference engine is running concurrently with two different sets of facts and returning two different conclusion xml files. One file from thread “main-1” has ten results the other from “main-2” has nine.

APPENDIX B

Data from experiments

SoaData1	SoaData2	SoaData3	LpData1	LpData2	LpData3
37.445877	30.387577	40.187366	36.798466	30.387577	40.187366
31.478830	42.146862	37.284042	19.747478	42.146862	37.609524
41.822319	44.022949	43.446621	39.092403	44.303745	43.450703
45.502251	41.765820	36.271744	45.502251	41.765820	36.271744
23.121887	47.154095	31.452332	19.162468	47.643986	31.505575
41.762100	21.925781	36.574764	41.762100	21.925781	36.574764
41.119404	43.867088	29.956314	41.243534	43.867088	29.956314
26.084146	45.400459	42.511681	26.084146	45.400459	42.511681
50.338131	28.294250	29.181549	50.338131	28.294250	29.181549
35.733803	29.287676	28.712944	35.733803	18.636715	28.902822
42.237091	44.883530	42.816181	42.723499	44.883530	29.874987
40.482914	29.786386	35.178989	40.482914	29.786386	35.178989
47.826069	25.315979	31.662405	47.826069	25.315979	31.662405
35.155399	21.397947	29.529404	35.155399	21.397947	29.529404
34.190350	29.394468	39.013344	34.190350	0.000000	41.629219
41.035347	33.049236	36.973633	41.035347	33.049236	36.973633
23.981462	21.578676	21.017944	23.981462	21.686468	19.802719
35.467323	40.749229	43.461914	35.467323	40.753548	43.461914
25.711170	52.110165	46.495102	24.837156	52.192753	46.495102
26.580370	48.142971	50.096596	27.274776	48.142971	36.134060
33.108913	32.613529	31.632917	33.108913	31.839254	31.653080
36.285183	39.691704	47.188648	36.373512	39.733593	27.959463
38.231617	33.175610	38.936687	38.231617	33.175610	38.936687
31.432983	26.652046	25.962576	31.470415	0.000000	25.962576
44.966431	21.674229	37.964592	44.966431	18.147520	40.151188
41.194607	53.809608	30.997095	41.194607	53.809608	30.997095
39.714970	21.838804	15.918496	39.714970	21.838804	15.918496
28.828367	36.820415	41.629436	28.828367	37.515465	16.769573
48.294563	33.315670	32.092342	35.126713	33.315670	32.358799
30.306334	49.604771	42.690998	30.306334	47.346325	43.745911
47.935848	33.541904	33.470661	47.744919	30.138584	33.470661
41.572495	34.216347	40.890835	41.572495	34.216347	40.890835
42.717327	27.669548	23.488764	42.717327	27.669548	23.488764
38.637024	37.416618	37.290737	38.768188	37.416618	36.632320
23.459551	46.725433	45.818233	23.459551	46.725433	45.818233
28.143150	21.198072	55.406879	31.650574	0.000000	20.942051
40.066578	53.044743	32.792130	40.066578	53.044743	32.792130
38.438984	35.090508	25.968002	38.438984	35.090508	25.968002
34.317337	38.931236	33.805405	34.317337	38.984116	33.805405
27.827913	42.323181	54.592453	27.927000	42.644730	47.175575
31.530207	51.014797	48.670521	31.530207	51.389019	11.684208
51.189148	29.584789	49.196194	51.580654	29.584789	49.196194

24.711519	34.649628	32.865803	24.711519	34.649628	32.865803
39.874805	58.892639	21.506458	39.874805	57.439495	21.506458
44.157879	46.842739	21.503315	44.157879	46.842739	21.503315
26.834238	29.581400	54.761421	27.508131	29.581400	37.942219
38.684444	58.975391	33.498493	38.684444	58.771896	33.498493
42.199867	32.089130	35.788639	42.199867	32.381897	35.788639
16.235897	30.034075	37.772327	16.235897	30.034075	37.772327
31.964680	34.016342	31.086821	29.761696	34.094631	31.542221
34.190662	23.791967	27.807692	34.459843	23.992048	27.812437
39.014210	41.248936	34.749050	39.014210	41.248936	34.749050
44.923767	29.074965	50.026215	18.618029	29.588465	50.026215
39.252670	37.351395	34.258118	38.139309	37.358196	34.258118
37.090458	32.097168	22.965963	37.090458	32.097168	22.965963
47.388638	24.913689	50.640583	47.388638	24.913689	49.368958
33.331173	37.756027	36.187180	33.331173	37.756027	36.187180
32.771908	34.030022	38.903557	20.502810	34.030022	39.950302
36.599098	35.269096	34.418392	36.599098	35.307045	33.743473
30.837542	38.669689	37.165077	29.189596	0.000000	43.503685
28.359264	56.009281	54.428272	33.164005	31.733278	55.093418
34.397217	38.860435	54.037834	34.746365	38.860435	49.750919
41.359474	39.081032	38.853088	41.359474	39.081032	38.853088
39.569889	44.881733	35.071159	39.625515	44.303421	35.071159
28.870424	44.579815	31.173021	29.943531	35.609585	31.173021
24.125790	41.259373	44.902885	24.125790	41.259373	44.902885
44.490772	45.402431	30.840382	44.490772	45.402431	30.840382
29.867676	41.622211	39.890354	29.867676	41.622211	39.890354
34.331810	31.675602	43.601673	34.546814	31.675602	39.945076
36.807846	27.354225	43.112305	36.807846	27.354225	43.112305
34.155266	41.343586	39.517467	34.155266	41.343586	39.517467
29.038916	75.606819	34.490707	29.038916	75.414520	25.117392
41.834141	39.293262	35.907337	33.961220	39.293262	37.694729
30.446678	46.505630	37.737961	30.446678	46.505630	37.737961
37.247162	43.705307	29.981106	37.247162	43.705307	30.011595
36.473984	21.069162	49.173157	36.473984	21.069162	49.173157
34.593655	42.566910	32.948341	34.709000	32.020748	32.948341
36.083580	35.872738	26.209795	36.083580	30.984301	26.568142
34.724567	40.470905	32.229446	34.774860	40.470905	32.196903
44.822392	48.587185	42.632851	45.188770	0.000000	42.877003
35.571751	42.180237	45.773800	35.571751	42.180237	45.773800
28.698391	52.369221	33.137074	28.698391	52.369221	33.137074
31.219032	54.953312	40.504368	31.219032	54.953312	40.504368
33.478046	33.414536	50.885288	33.478046	33.414536	50.913002
23.960686	54.352303	34.449825	23.960686	54.352303	34.449825
30.127153	32.307259	37.025406	29.785507	32.307259	37.192440
31.616390	40.640060	44.114517	31.616390	40.640060	44.675392
30.352238	26.985983	38.242050	30.352238	26.960960	38.347107
36.806316	43.072483	46.131725	31.258556	43.639179	46.131725
25.878668	51.379986	34.469631	25.878668	51.379986	34.469631
39.877708	30.765829	29.613733	39.223724	30.765829	29.624447
31.413034	33.644001	47.432949	31.413034	33.959858	35.974304

```
39.476830 36.338009 35.979744 43.145683 36.338009 35.979744
33.975143 29.745092 34.419724 33.975143 27.452793 34.547440
44.523067 38.563381 44.382015 29.345318 39.590729 44.382015
44.940281 23.232840 46.666210 45.669342 22.553053 48.557198
39.693180 31.747107 34.758862 39.693180 31.747107 34.758862
32.611973 42.494766 43.258156 32.611973 44.651356 31.570852
37.950684 42.011677 45.576920 37.950684 42.011677 45.576920
39.898361 32.494362 29.773643 40.456894 32.494362 20.638634
```

APPENDIX C

The original given model for cost per unit to manufacture, transport and store was:

$$tc = 1000[P + (a + b + c)] + \frac{950}{500}SL + \sum_{t=1}^{19} d\,t$$

- P = manufacturing costs per unit
- a = supply cost per unit from supplier a
- b = supply cost per unit from supplier b
- c = supply cost per unit from supplier c
- SL = $100 transport cost per 500 units
- tc = total cost per unit

$$tc = P + a + b + c + \frac{950}{500}SL + \sum_{t=1}^{n} d\,t$$

$\sum_{t=1}^{n} d\,t$ = storage costs $d per unit for t days models a trickle feed from manufacturer with constantly increasing storage space needed over n days.

Simplify for calculation purposes:

$$\sum_{t=1}^{n} d\,t = d + 2d + 3d + 4d + \cdots + nd$$

$$= d \times (arithmetic\ progression)$$

$$= d \times (1 + 2 + 3 + \cdots n)$$

Multiply generic sum of arithmetic series by d:

$$\sum_{t=1}^{n} d\,t = \frac{1}{2}d(n+1)^2 - \frac{1}{2}d(n+1)$$

$$\sum_{t=1}^{n} d\,t = \frac{1}{2}d\,n^2 - \frac{1}{2}d\,n$$

Collect terms and simplify:

$$\sum_{t=1}^{n} d\,t = \frac{1}{2} d\,n(n+1)$$

Then final simplified model for calculation purposes of the total cost per unit to manufacture, transport and store for n days is:

$$tc = P + a + b + c + \frac{19}{10} SL + \frac{1}{2} d\,n(n+1)$$

For the scenario of a supply chain with the longest delivery time (large retail order, low distributor stock, and low manufacturing production throughput, this produces a large time to fulfill order). The heuristic rule for this scenario is to choose the supply chain with the smallest delivery time, and after its current order has been fulfilled, to transfer its production stream to the supply chain with the longest delivery time. The math to calculate the new improved delivery time is:

For each isolated supply chain the normal delivery time depends on what is available from the warehouse and how fast the manufacturer can make up the difference.

$Let\ rq = retail\ order\ in\ number\ of\ units$

$Let\ dq = number\ of\ units\ available\ from\ distributors\ warehouse$

$Let\ mq = manufacturers\ production\ throughput\ in\ units\ per\ day$

The un-optimized delivery time in days for single isolated supply chain is:

$$oldTime = \frac{rq - dq}{mq}$$

For shared supply chains if the supply chain with fastest throughput can supplement the supply chain with the least throughput then:

$Let\ PqMax =$ production throughput in units per day with longest delivery time.

$Let\ PqMin\ =$ production throughput in units per day with least delivery time.

$Let\ RtMax =$ Time in days for supply chain with longest time to deliver.

$Let\ RtMin =$ Time in days for supply chain with smallest time to deliver.

Then the optimized delivery time for the supply chain with the least throughput is:

$$newTime = \frac{(RtMax - RtMin)\ PqMax}{PqMax + PqMin} + RtMin$$

Expand and then simplify:

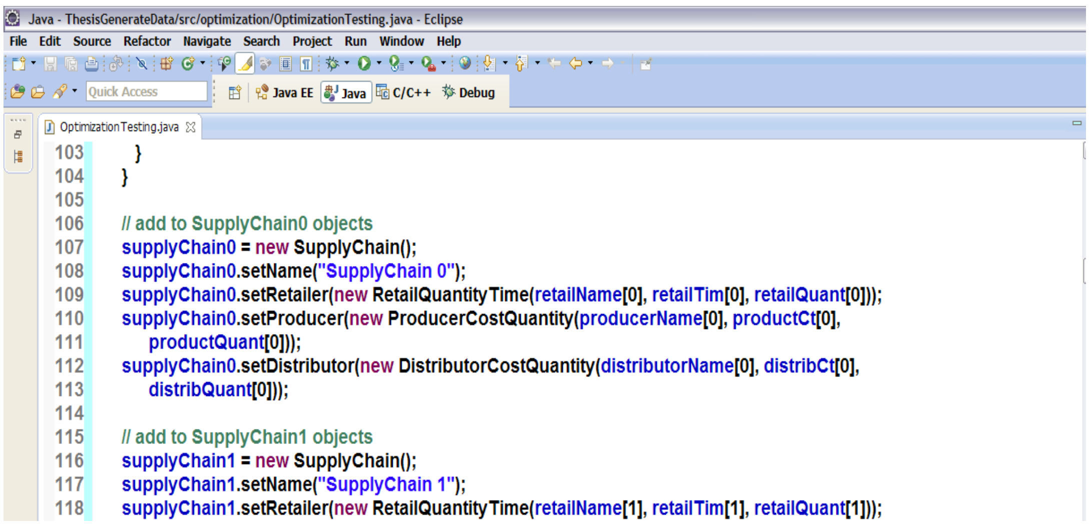
$$newTime = \frac{PqMax\ RtMax}{PqMax + PqMin} - \frac{PqMax\ RtMin}{PqMax + PqMin} + \frac{PqMax\ RtMin + RtMin\ PqMin}{PqMax + PqMin}$$

$$newTime = \frac{PqMax\ RtMax + RtMin\ PqMin}{PqMax + PqMin}$$

APPENDIX D

Optimization procedures

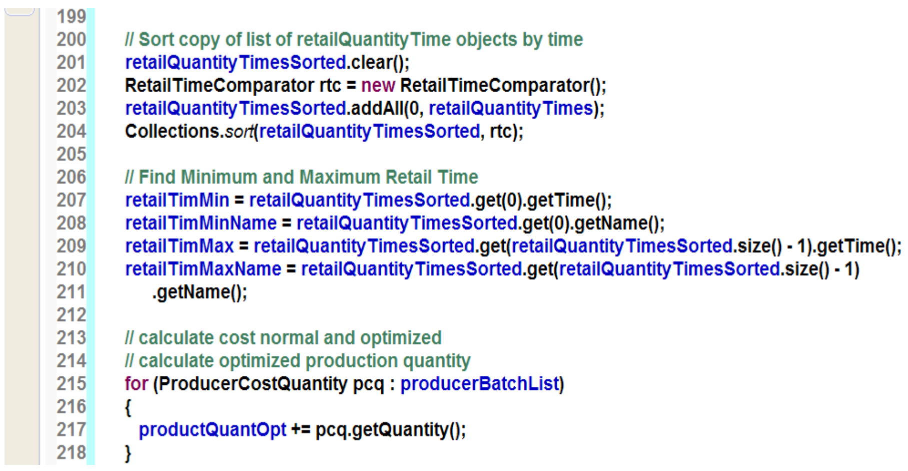
The first procedure is to create and populate the data structures that will contain and encapsulate the hierarchical data objects; list of supply chains, individual supply chains and the individual supply chain artifacts (retailers, distributors, manufacturers and suppliers).


Java - ThesisGenerateData/src/optimization/OptimizationTesting.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
C/C++
Debug
OptimizationTesting.java

```
        }
    }

    // add to SupplyChain0 objects
    supplyChain0 = new SupplyChain();
    supplyChain0.setName("SupplyChain 0");
    supplyChain0.setRetailer(new RetailQuantityTime(retailName[0], retailTim[0], retailQuant[0]));
    supplyChain0.setProducer(new ProducerCostQuantity(producerName[0], productCt[0],
        productQuant[0]));
    supplyChain0.setDistributor(new DistributorCostQuantity(distributorName[0], distribCt[0],
        distribQuant[0]));

    // add to SupplyChain1 objects
    supplyChain1 = new SupplyChain();
    supplyChain1.setName("SupplyChain 1");
    supplyChain1.setRetailer(new RetailQuantityTime(retailName[1], retailTim[1], retailQuant[1]));
```

Figure 44 Data structures

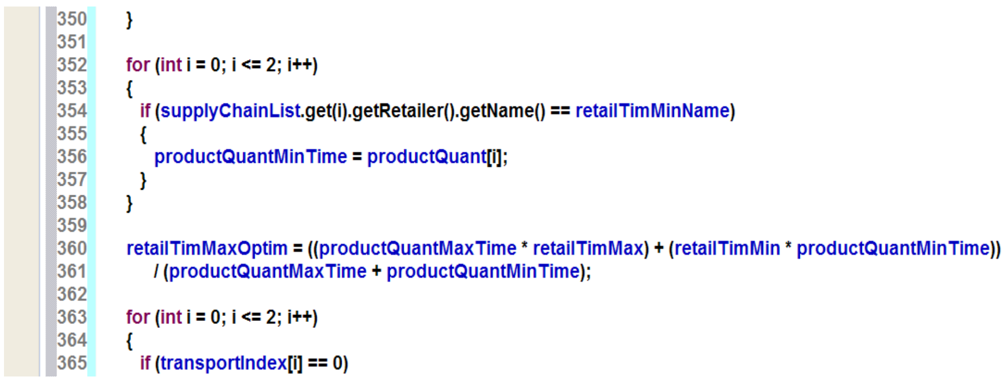
Data structures are sorted to find largest and smallest times to fulfill retail orders.



```java

// Sort copy of list of retailQuantityTime objects by time
retailQuantityTimesSorted.clear();
RetailTimeComparator rtc = new RetailTimeComparator();
retailQuantityTimesSorted.addAll(0, retailQuantityTimes);
Collections.sort(retailQuantityTimesSorted, rtc);

// Find Minimum and Maximum Retail Time
retailTimMin = retailQuantityTimesSorted.get(0).getTime();
retailTimMinName = retailQuantityTimesSorted.get(0).getName();
retailTimMax = retailQuantityTimesSorted.get(retailQuantityTimesSorted.size() - 1).getTime();
retailTimMaxName = retailQuantityTimesSorted.get(retailQuantityTimesSorted.size() - 1)
    .getName();

// calculate cost normal and optimized
// calculate optimized production quantity
for (ProducerCostQuantity pcq : producerBatchList)
{
  productQuantOpt += pcq.getQuantity();
}
```

Figure 45 Sort retail time objects by time

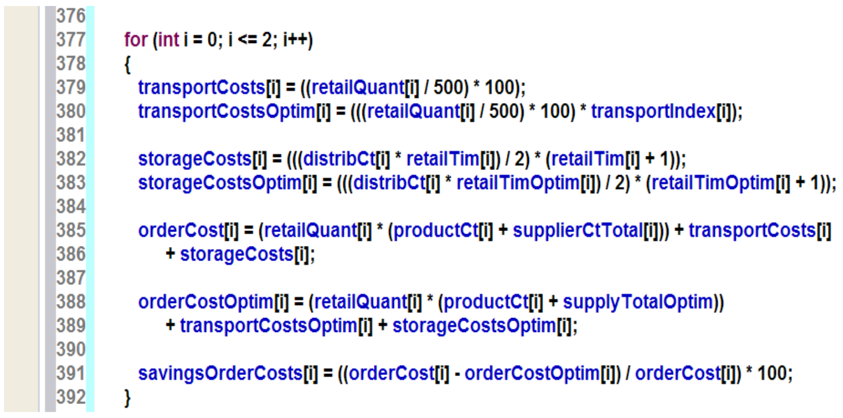
Calculate optimized retail order times by applying equation see line 360



```java
350    }
351
352    for (int i = 0; i <= 2; i++)
353    {
354      if (supplyChainList.get(i).getRetailer().getName() == retailTimMinName)
355      {
356        productQuantMinTime = productQuant[i];
357      }
358    }
359
360    retailTimMaxOptim = ((productQuantMaxTime * retailTimMax) + (retailTimMin * productQuantMinTime))
361        / (productQuantMaxTime + productQuantMinTime);
362
363    for (int i = 0; i <= 2; i++)
364    {
365      if (transportIndex[i] == 0)
```

Figure 46 Optimized retail order times by equation

Calculate savings in cost components (optimized order, transport and storage costs)

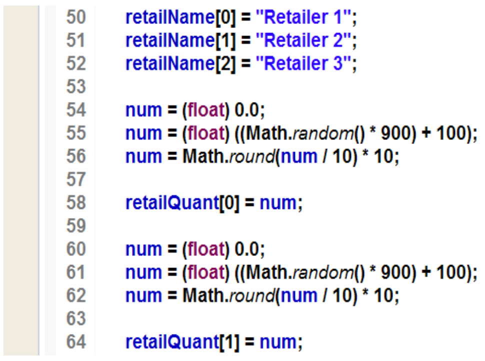


```java

for (int i = 0; i <= 2; i++)
{
  transportCosts[i] = ((retailQuant[i] / 500) * 100);
  transportCostsOptim[i] = (((retailQuant[i] / 500) * 100) * transportIndex[i]);

  storageCosts[i] = (((distribCt[i] * retailTim[i]) / 2) * (retailTim[i] + 1));
  storageCostsOptim[i] = (((distribCt[i] * retailTimOptim[i]) / 2) * (retailTimOptim[i] + 1));

  orderCost[i] = (retailQuant[i] * (productCt[i] + supplierCtTotal[i])) + transportCosts[i]
      + storageCosts[i];

  orderCostOptim[i] = (retailQuant[i] * (productCt[i] + supplyTotalOptim))
      + transportCostsOptim[i] + storageCostsOptim[i];

  savingsOrderCosts[i] = ((orderCost[i] - orderCostOptim[i]) / orderCost[i]) * 100;
}
```

Figure 47 Optimized order, transport and storage costs

Data Generation procedures

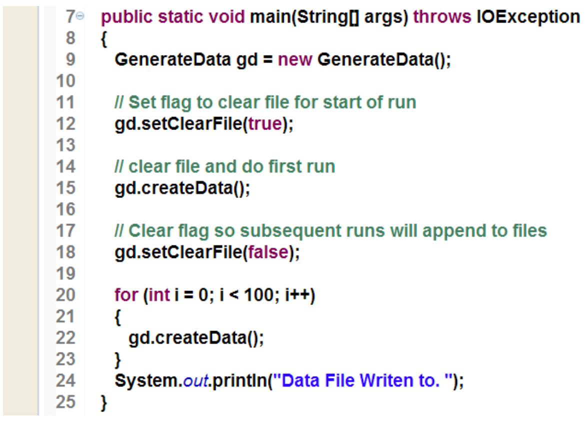
The data generation procedures are first, the individual supply chain artifacts are filled with randomized data (within certain ranges).



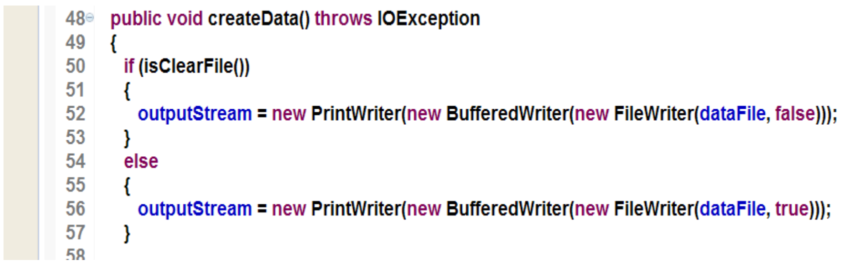
```
retailName[0] = "Retailer 1";
retailName[1] = "Retailer 2";
retailName[2] = "Retailer 3";

num = (float) 0.0;
num = (float) ((Math.random() * 900) + 100);
num = Math.round(num / 10) * 10;

retailQuant[0] = num;

num = (float) 0.0;
num = (float) ((Math.random() * 900) + 100);
num = Math.round(num / 10) * 10;

retailQuant[1] = num;
```

Figure 48 Fill supply chain artifacts with randomized data

The randomized supply chain artifacts are ready to be sent to two optimization modules. One the heuristic based approach that uses the optimization model from my web based research prototype, but the internet components have been removed to aid efficiency (the heuristic based optimization module will be called at least 100 times in each run). The second is a linear programming Simplex based module; this will also be called at least 100 times during a run. Next a data file is initialized and told to generate 100 data sets for each Heuristic and Simplex modules and a flag is set to tell the system that it is writing to the file for the first time so that the file will be cleared of old data or else it will be appended to the file during each run.

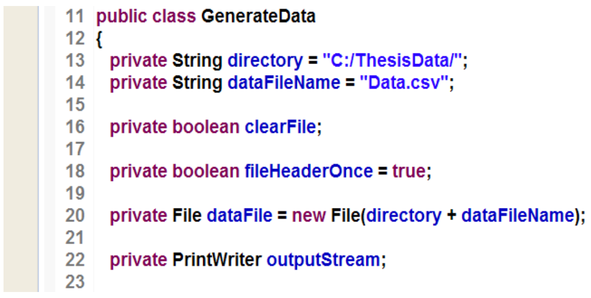


```
public static void main(String[] args) throws IOException
{
  GenerateData gd = new GenerateData();

  // Set flag to clear file for start of run
  gd.setClearFile(true);

  // clear file and do first run
  gd.createData();

  // Clear flag so subsequent runs will append to files
  gd.setClearFile(false);

  for (int i = 0; i < 100; i++)
  {
    gd.createData();
  }
  System.out.println("Data File Writen to. ");
}
```



```
public void createData() throws IOException
{
  if (isClearFile())
  {
    outputStream = new PrintWriter(new BufferedWriter(new FileWriter(dataFile, false)));
  }
  else
  {
    outputStream = new PrintWriter(new BufferedWriter(new FileWriter(dataFile, true)));
  }
```

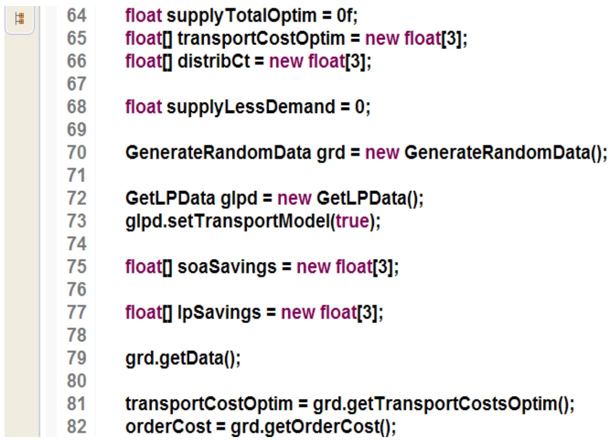
Figure 49 Generate data and write to file



```
public class GenerateData
{
  private String directory = "C:/ThesisData/";
  private String dataFileName = "Data.csv";

  private boolean clearFile;

  private boolean fileHeaderOnce = true;

  private File dataFile = new File(directory + dataFileName);

  private PrintWriter outputStream;

```

Figure 50 Create data file name

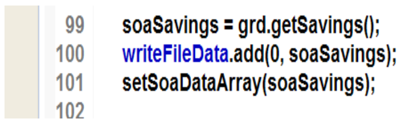
Next some variables are created to handle the dataflow and control signals between the two optimization modules and the calling module.



```
float supplyTotalOptim = 0f;
float[] transportCostOptim = new float[3];
float[] distribCt = new float[3];

float supplyLessDemand = 0;

GenerateRandomData grd = new GenerateRandomData();

GetLPData glpd = new GetLPData();
glpd.setTransportModel(true);

float[] soaSavings = new float[3];

float[] lpSavings = new float[3];

grd.getData();

transportCostOptim = grd.getTransportCostsOptim();
orderCost = grd.getOrderCost();
```

Figure 51 Variables for data flow and control

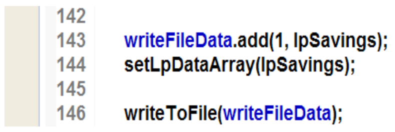
The randomized data is sent to the heuristic optimization module first and the savings are returned and added to a common data structure.



```
soaSavings = grd.getSavings();
writeFileData.add(0, soaSavings);
setSoaDataArray(soaSavings);
```

Figure 52 Data from Heuristic optimization model ready for writing to file

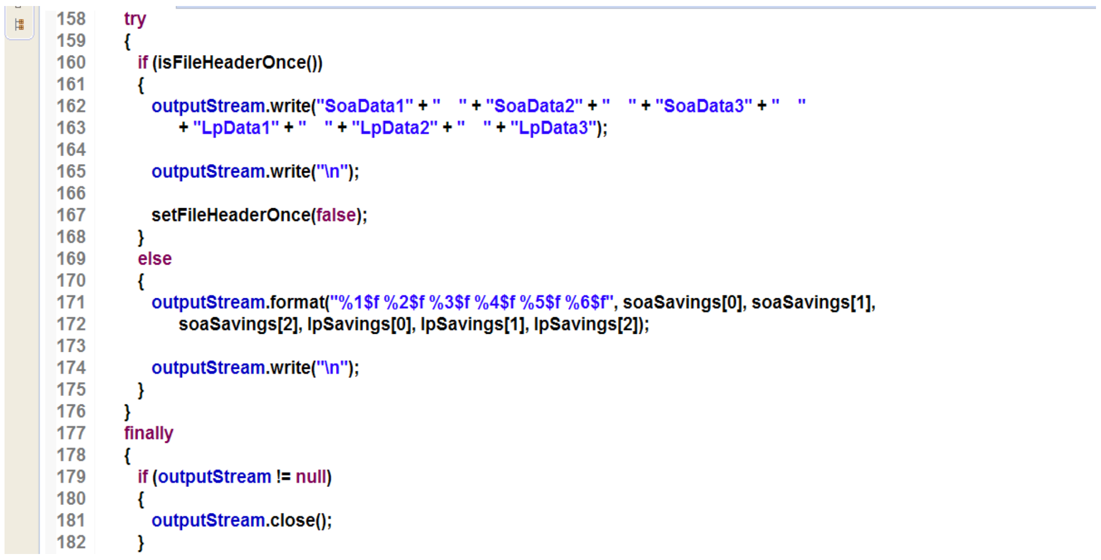
Then the same randomized supply chain data is sent to the Simplex module and the savings are returned and added to the same common data structure as the heuristic module was added to and then the common data structure is written to the data file.



```

writeFileData.add(1, lpSavings);
setLpDataArray(lpSavings);

writeToFile(writeFileData);
```

Figure 53 Data from Simplex module written to file

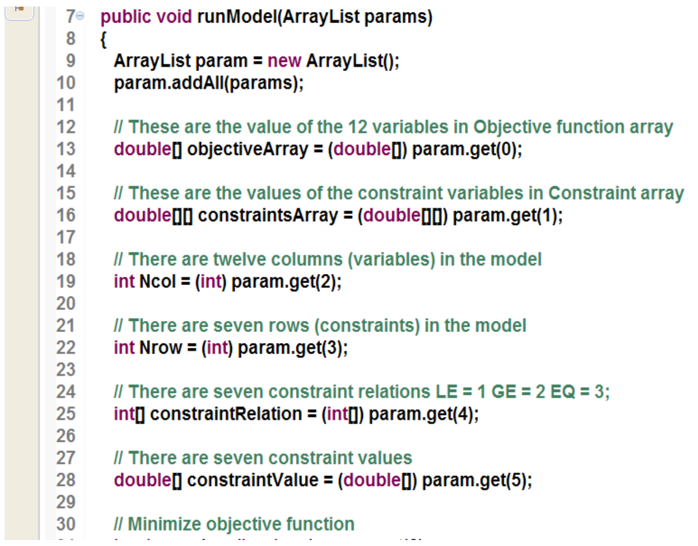
The method “writeToFile()” writes column headings then formats the data and writes a line of data to the file.



```
try
{
  if (isFileHeaderOnce())
  {
    outputStream.write("SoaData1" + "    " + "SoaData2" + "    " + "SoaData3" + "    "
        + "LpData1" + "    " + "LpData2" + "    " + "LpData3");

    outputStream.write("\n");

    setFileHeaderOnce(false);
  }
  else
  {
    outputStream.format("%1$f %2$f %3$f %4$f %5$f %6$f", soaSavings[0], soaSavings[1],
        soaSavings[2], lpSavings[0], lpSavings[1], lpSavings[2]);

    outputStream.write("\n");
  }
}
finally
{
  if (outputStream != null)
  {
    outputStream.close();
  }
```

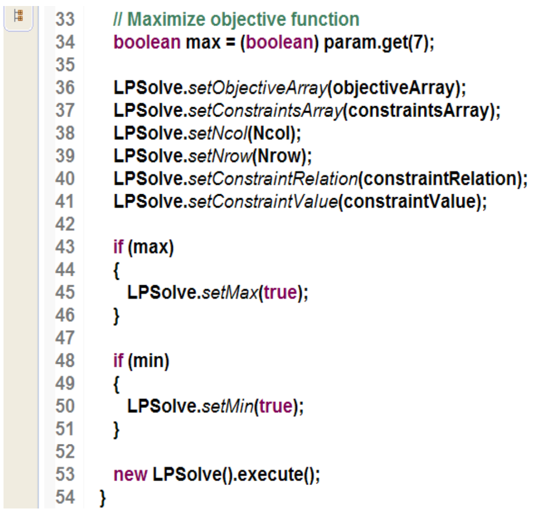
Figure 54 Method writeToFile()

See APPENDIX B for example of file format.



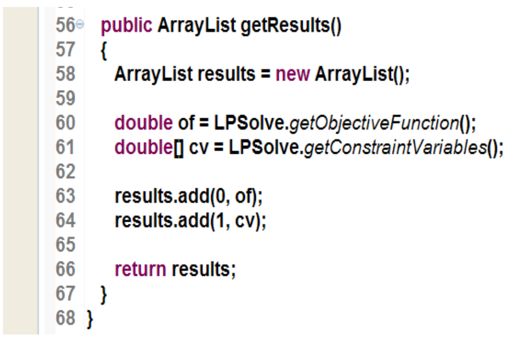
```
public void runModel(ArrayList params)
{
  ArrayList param = new ArrayList();
  param.addAll(params);

  // These are the value of the 12 variables in Objective function array
  double[] objectiveArray = (double[]) param.get(0);

  // These are the values of the constraint variables in Constraint array
  double[][] constraintsArray = (double[][]) param.get(1);

  // There are twelve columns (variables) in the model
  int Ncol = (int) param.get(2);

  // There are seven rows (constraints) in the model
  int Nrow = (int) param.get(3);

  // There are seven constraint relations LE = 1 GE = 2 EQ = 3;
  int[] constraintRelation = (int[]) param.get(4);

  // There are seven constraint values
  double[] constraintValue = (double[]) param.get(5);

  // Minimize objective function
```

Figure 55 Simplex Solver driver initialize variables



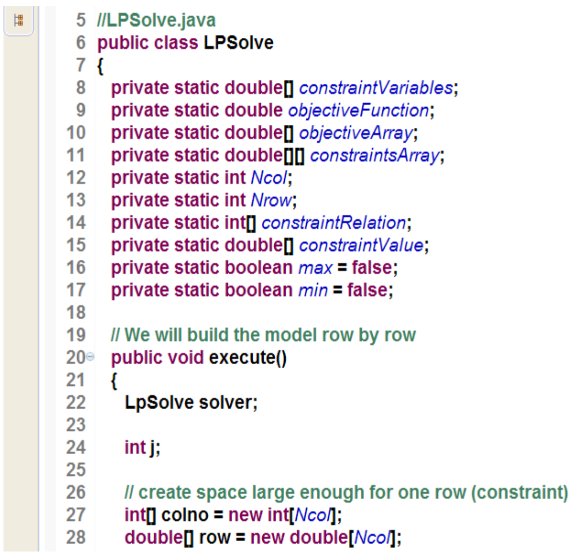
```
        // Maximize objective function
        boolean max = (boolean) param.get(7);

        LPSolve.setObjectiveArray(objectiveArray);
        LPSolve.setConstraintsArray(constraintsArray);
        LPSolve.setNcol(Ncol);
        LPSolve.setNrow(Nrow);
        LPSolve.setConstraintRelation(constraintRelation);
        LPSolve.setConstraintValue(constraintValue);

        if (max)
        {
          LPSolve.setMax(true);
        }

        if (min)
        {
          LPSolve.setMin(true);
        }

        new LPSolve().execute();
    }
```

Figure 56 Simplex Solver driver add data from model to solver



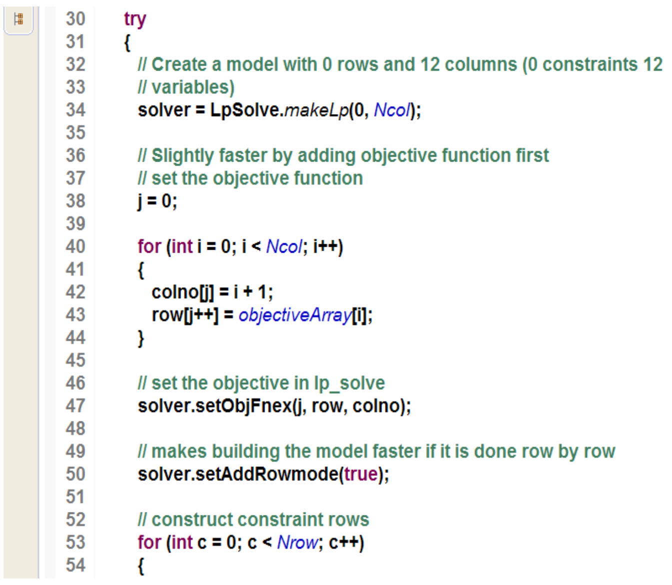
```
    public ArrayList getResults()
    {
      ArrayList results = new ArrayList();

      double of = LPSolve.getObjectiveFunction();
      double[] cv = LPSolve.getConstraintVariables();

      results.add(0, of);
      results.add(1, cv);

      return results;
    }
  }
```

Figure 57 Simplex Solver driver run model and return results



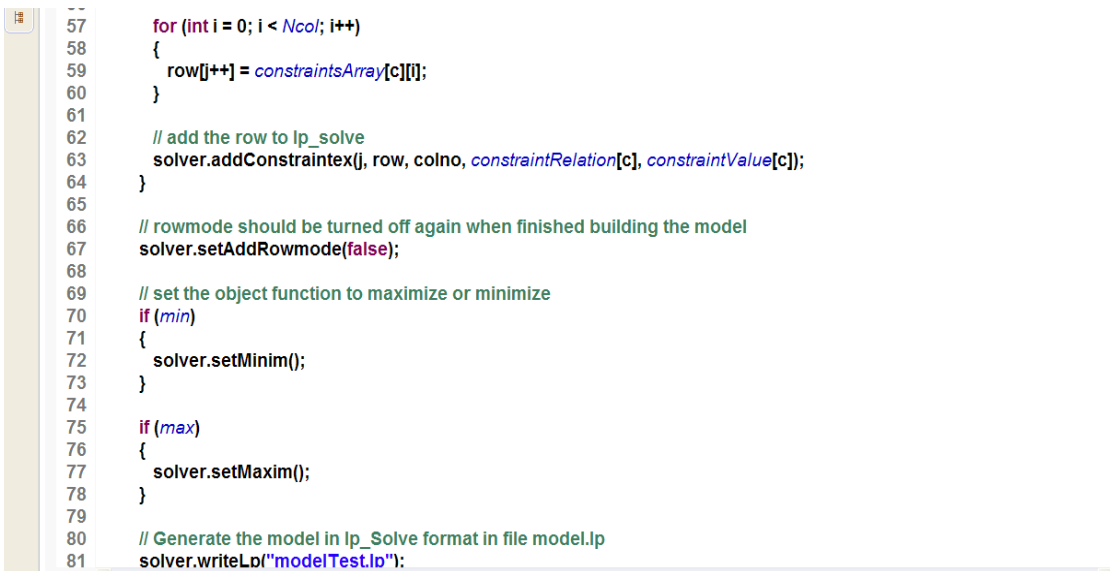
```java
//LPSolve.java
public class LPSolve
{
  private static double[] constraintVariables;
  private static double objectiveFunction;
  private static double[] objectiveArray;
  private static double[][] constraintsArray;
  private static int Ncol;
  private static int Nrow;
  private static int[] constraintRelation;
  private static double[] constraintValue;
  private static boolean max = false;
  private static boolean min = false;

  // We will build the model row by row
  public void execute()
  {
    LpSolve solver;

    int j;

    // create space large enough for one row (constraint)
    int[] colno = new int[Ncol];
    double[] row = new double[Ncol];
```

Figure 58 Simplex Solver itself



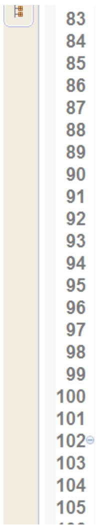
```java
    try
    {
      // Create a model with 0 rows and 12 columns (0 constraints 12
      // variables)
      solver = LpSolve.makeLp(0, Ncol);

      // Slightly faster by adding objective function first
      // set the objective function
      j = 0;

      for (int i = 0; i < Ncol; i++)
      {
        colno[j] = i + 1;
        row[j++] = objectiveArray[i];
      }

      // set the objective in lp_solve
      solver.setObjFnex(j, row, colno);

      // makes building the model faster if it is done row by row
      solver.setAddRowmode(true);

      // construct constraint rows
      for (int c = 0; c < Nrow; c++)
      {
```

Figure 59 Simplex Solver build model



```
        for (int i = 0; i < Ncol; i++)
        {
          row[j++] = constraintsArray[c][i];
        }

        // add the row to lp_solve
        solver.addConstraintex(j, row, colno, constraintRelation[c], constraintValue[c]);
      }

      // rowmode should be turned off again when finished building the model
      solver.setAddRowmode(false);

      // set the object function to maximize or minimize
      if (min)
      {
        solver.setMinim();
      }

      if (max)
      {
        solver.setMaxim();
      }

      // Generate the model in lp_Solve format in file model.lp
      solver.writeLp("modelTest.lp");
```

Figure 60 Simplex Solver finishes building model



```
      // We only want to see important messages on screen while solving
      solver.setVerbose(LpSolve.IMPORTANT);

      // Calculate a solution
      solver.solve();

      // solution is calculated, now lets get some results
      objectiveFunction = solver.getObjective();
      constraintVariables = solver.getPtrVariables();

      // clean up all memory used by lp_solve
      solver.deleteLp();
    }
    catch (LpSolveException e)
    {
      e.printStackTrace();
    }
  }

  public static double[] getConstraintVariables()
  {
    return constraintVariables;
  }
```

Figure 61 Simplex Solver solve model and return results to driver

APPENDIX E

The following Dataplot output are the parametric tests for each supply chain LP1 is linear programming optimization for supply chain 1 and SOA1 is heuristic optimization method for supply chain 1.

Two Sample t-Tests for Equal Means

First Response Variable: LP1

Second Response Variable: SOA1

H0: Population Means Are Equal

Ha: Population Means Are Not Equal

Sample One Summary Statistics:

Number of Observations:	100
Sample Mean:	34.91064
Sample Standard Deviation:	7.245900
Sample Standard Deviation of the Mean:	0.7245900

Sample Two Summary Statistics:

Number of Observations:	100
Sample Mean:	35.79009
Sample Standard Deviation:	6.899811
Sample Standard Deviation of the Mean:	0.6899811

Two-Tailed Test (Assume Equal Variances)

H0: u1 = u2; Ha: u1 <> u2

			Null
Significance	Test	Critical	Hypothesis

```
          Level      Statistic    Value (+/-)         Conclusion
-----------------------------------------------------------------
          50.0% -0.8789650       0.6757308           REJECT
          80.0% -0.8789650       1.285842            ACCEPT
          90.0% -0.8789650       1.652586            ACCEPT
          95.0% -0.8789650       1.972017            ACCEPT
          99.0% -0.8789650       2.600887            ACCEPT
          99.9% -0.8789650       3.340340            ACCEPT
```

Two-Tailed Test (Assume Unequal Variances)

H0: u1 = u2; Ha: u1 <> u2

```
-------------------------------------------------------------------
                                                    Null
   Significance       Test          Critical        Hypothesis
          Level      Statistic    Value (+/-)       Conclusion
-------------------------------------------------------------------
          50.0% -0.8789650       0.6757339           REJECT
          80.0% -0.8789650       1.285852            ACCEPT
          90.0% -0.8789650       1.652604            ACCEPT
          95.0% -0.8789650       1.972046            ACCEPT
          99.0% -0.8789650       2.600948            ACCEPT
          99.9% -0.8789650       3.340461            ACCEPT
```

Two Sample F-Tests for Equal Standard Deviations

First Response Variable: LP1

Second Response Variable: SOA1

H0: Sigma1 = Sigma2

Ha: Sigma1 not equal Sigma2

Sample One Summary Statistics:

```
Number of Observations:           100
Sample Mean:                        34.91064
Sample Standard Deviation:            7.245900
```

Sample Two Summary Statistics:

```
Number of Observations:           100
Sample Mean:                        35.79009
Sample Standard Deviation:            6.899811
```

Conclusions (Upper 1-Tailed Test)

```
H0: sigma1 = sigma2; Ha: sigma1 <> sigma2
---------------------------------------------------------------
                                              Null
  Significance     Test         Critical      Hypothesis
       Level     Statistic   Region (>=)     Conclusion
---------------------------------------------------------------
       50.0%   1.102835      0.9999990         REJECT
       75.0%   1.102835      1.145651          ACCEPT
       90.0%   1.102835      1.295129          ACCEPT
       95.0%   1.102835      1.394061          ACCEPT
       97.5%   1.102835      1.486234          ACCEPT
       99.0%   1.102835      1.601498          ACCEPT
       99.9%   1.102835      1.873411          ACCEPT
```

Two Sample t-Tests for Equal Means

```
First Response Variable:  LP2

Second Response Variable: SOA2

H0: Population Means Are Equal

Ha: Population Means Are Not Equal
```

Sample One Summary Statistics:

```
Number of Observations:              100

Sample Mean:                           35.66766

Sample Standard Deviation:             12.80811

Sample Standard Deviation of the Mean:    1.280811
```

Sample Two Summary Statistics:

```
Number of Observations:              100

Sample Mean:                           37.98339

Sample Standard Deviation:             10.04330

Sample Standard Deviation of the Mean:    1.004330
```

Two-Tailed Test (Assume Equal Variances)

```
H0: u1 = u2; Ha: u1 <> u2

----------------------------------------------------------------

                                              Null

  Significance     Test       Critical      Hypothesis

        Level    Statistic   Value (+/-)    Conclusion

----------------------------------------------------------------

        50.0% -1.422772     0.6757308        REJECT

        80.0% -1.422772     1.285842         REJECT

        90.0% -1.422772     1.652586         ACCEPT

        95.0% -1.422772     1.972017         ACCEPT

        99.0% -1.422772     2.600887         ACCEPT
        99.9% -1.422772      3.340340            ACCEPT
```

Two-Tailed Test (Assume Unequal Variances)

```
H0: u1 = u2; Ha: u1 <> u2
---------------------------------------------------------------
                                                  Null
  Significance      Test          Critical        Hypothesis
        Level     Statistic   Value (+/-)        Conclusion
---------------------------------------------------------------
        50.0% -1.422772      0.6758016           REJECT
        80.0% -1.422772      1.286087            REJECT
        90.0% -1.422772      1.653028            ACCEPT
        95.0% -1.422772      1.972708            ACCEPT
        99.0% -1.422772      2.602326            ACCEPT
        99.9% -1.422772      3.343215            ACCEPT
```

Two Sample F-Tests for Equal Standard Deviations

```
First Response Variable:  LP2
Second Response Variable: SOA2
H0: Sigma1 = Sigma2
Ha: Sigma1 not equal Sigma2
```

Sample One Summary Statistics:

```
Number of Observations:               100
Sample Mean:                           35.66766
Sample Standard Deviation:             12.80811
```

Sample Two Summary Statistics:

Number of Observations: 100

Sample Mean: 37.98339

Sample Standard Deviation: 10.04330

Conclusions (Upper 1-Tailed Test)

H0: sigma1 = sigma2; sigma1 <> sigma2

H0: sigma1 = sigma2; Ha: sigma1 <> sigma2

```
----------------------------------------------------------------
                                                 Null
  Significance     Test        Critical       Hypothesis
      Level     Statistic   Region (>=)      Conclusion
----------------------------------------------------------------
      50.0%   1.626362      0.9999990         REJECT
      75.0%   1.626362      1.145651          REJECT
      90.0%   1.626362      1.295129          REJECT
      95.0%   1.626362      1.394061          REJECT
      97.5%   1.626362      1.486234          REJECT
      99.0%   1.626362      1.601498          REJECT
      99.9%   1.626362      1.873411          ACCEPT
```

Two Sample t-Tests for Equal Means

First Response Variable: LP3

Second Response Variable: SOA3

H0: Population Means Are Equal

Ha: Population Means Are Not Equal

Sample One Summary Statistics:

Number of Observations: 100

Sample Mean: 35.67877

Sample Standard Deviation: 8.415528

Sample Standard Deviation of the Mean: 0.8415528

Sample Two Summary Statistics:

Number of Observations: 100

Sample Mean: 37.67469

Sample Standard Deviation: 8.288676

Sample Standard Deviation of the Mean: 0.8288676

Two-Tailed Test (Assume Equal Variances)

H0: u1 = u2; Ha: u1 <> u2

```
---------------------------------------------------------------
                                              Null
  Significance   Test      Critical        Hypothesis
       Level   Statistic  Value (+/-)      Conclusion
---------------------------------------------------------------
       50.0% -1.689746     0.6757308         REJECT
       80.0% -1.689746     1.285842          REJECT
       90.0% -1.689746     1.652586          REJECT
       95.0% -1.689746     1.972017          ACCEPT
       99.0% -1.689746     2.600887          ACCEPT
       99.9% -1.689746     3.340340          ACCEPT
```

Two-Tailed Test (Assume Unequal Variances)

H0: u1 = u2; Ha: u1 <> u2

```
-----------------------------------------------------------------
                                              Null
  Significance   Test      Critical        Hypothesis
       Level   Statistic  Value (+/-)      Conclusion
```

```
---------------------------------------------------------------
       50.0% -1.689746     0.6757311       REJECT
       80.0% -1.689746     1.285843        REJECT
       90.0% -1.689746     1.652588        REJECT
       95.0% -1.689746     1.972020        ACCEPT
       99.0% -1.689746     2.600893        ACCEPT
       99.9% -1.689746     3.340352        ACCEPT
```

Two Sample F-Tests for Equal Standard Deviations

```
First Response Variable:  LP3
Second Response Variable: SOA3
H0: Sigma1 = Sigma2
Ha: Sigma1 not equal Sigma2
```

Sample One Summary Statistics:

```
Number of Observations:            100
Sample Mean:                        35.67877
Sample Standard Deviation:           8.415528
```

Sample Two Summary Statistics:

```
Number of Observations:            100
Sample Mean:                        37.67469
Sample Standard Deviation:           8.288676
```

Conclusions (Upper 1-Tailed Test)

```
H0: sigma1 = sigma2; Ha: sigma1 <> sigma2
---------------------------------------------------------------
                                                  Null
  Significance    Test        Critical         Hypothesis
       Level     Statistic   Region (>=)       Conclusion
```

```
-----------------------------------------------------------------
        50.0%   1.030843      0.9999990        REJECT
        75.0%   1.030843      1.145651         ACCEPT
        90.0%   1.030843      1.295129         ACCEPT
        95.0%   1.030843      1.394061         ACCEPT
        97.5%   1.030843      1.486234         ACCEPT
        99.0%   1.030843      1.601498         ACCEPT
        99.9%   1.030843      1.873411         ACCEPT
```