Evolving Middlemen Strategies For Simple Supply Chains

Timothy D.B. Gosling

A thesis submitted for the degree of Doctor of Philosophy

Department of Computer Science

University of Essex

2007

To May, Diana, David, Jeremy and David

"Down and safe"- Blake

Abstract

This research is concerned with the evolution of middlemen strategies for simple, but non-trivial, supply chains. The work contributes a language for modelling simple supply chains, a framework for representing middlemen strategies and a platform for evolving these strategies. The work shows that for reasonably complex economic problems, that contain a great deal of uncertainty, the evolution of effective strategies provides a useful technique for investigation.

In the course of this thesis the Simple Supply Chain Model (SSCM) is defined, allowing the modelling of these problems. The SSCM Strategy Framework (SSF) is introduced providing a framework capable of specifying strategies that are able to tackle SSCM problems. The SSCM Market Simulation System (SMSS) strategy evolution platform is then considered. The SMSS supplies a system based on Population Based Incremental Learning with Guided Mutation (PBIL+GM) that is capable of evolving effective SSF-based strategies within SSCM-defined environments. Experimentation using the SMSS demonstrates that effective middlemen strategies may be evolved for a variety of supply chain environments and that these strategies tend to be specialised to their home environments.

Acknowledgments

This research work would not have been possible without the encouragement and guidance of Professor Edward Tsang. A more excellent supervisor and mentor one could not hope for!

I must also express my gratitude towards Dr John Ford and Dr Maria Fasli who provided further advice and support throughout this work.

Very many thanks must also go to my wife, May, family, Diana, David, Jeremy, Choo, Edward, Jenn Ning and Jenn Yu and friends (especially David, Adam and Elias) who provided so much encouragement and support. You all made the hard times easier and the easier times that much more enjoyable. Thank you.

This research was partially funded by the EPSRC and BT through a CASE studentship.

Table of Contents

Chapter 1

Introduction

This research is concerned with the evolution of middlemen strategies for simple, but non-trivial, supply chains.

The work presented here contributes a language for modelling simple supply chains, a framework for representing middlemen strategies and a platform for evolving these strategies. These contributions are elaborated on further in Section 1.4 and listed in Table 1.1.

This chapter provides an introduction to this research and its motivation. Note that within this introduction and throughout the thesis specific supply chains instances are often referred to as markets for the sake of simplicity.

1.1 Motivation

At present various electronic market places, auctions and negotiation systems exist. In the near future full electronic supply chains will be possible and indeed desirable to improve efficiency ([37, 79, 95]).

This situation, however, presents a problem. While humans are good at negotiations and situation analysis, they are less able to handle large volumes of information

and numbers of transactions. What is needed is a computer-based system or strategy for handling these situations. The strategy does not need to be the perfect negotiator, although it must be competent, but it must be able to deal with negotiations more rapidly than a human operator could.

This, however, also presents a problem since complex economic situations, such as supply chains, defy analysis by traditional Game Theory. This includes the relatively simple, but non-trivial, supply chains that this research focuses on.

Game Theory is unable to approach these problems because of the inherent uncertainty in such a dynamic situation. The knowledge and behaviour of participants in the chain is uncertain along with information about likely demand, product availability and product costs. While Game Theory is extremely effective in analysing many situations and identifying potentially effective strategies, with such a degree of uncertainty, the problem becomes intractable. This is elaborated on in Section 2.2. In order to form strategies in these situations it is usual to resort to domain knowledge, however, it is unclear how effective these strategies may be, how they will react in unexpected situations or how to optimise them.

Therefore, there is a need to develop techniques to analyse these problems and generate effective strategies in response to them.

Evolutionary computation (EC) is an effective approach to the generation and optimisation of solutions to a wide range of problems. It has also been applied to the generation of strategies for economic games and been used to optimise negotiation and bargaining systems. In each of these cases, EC is used to tackle comparatively simple one-sided problems that do not approach the complexity and uncertainty inherent in

the supply chains being considered here. Given EC's demonstrated ability to generate strategies for less complex economic problems and its successful track record in solving problems in other domains, its application to the more complex economic problem considered here is reasonable and appropriate.

The behaviour of middlemen strategies and their interaction with other supply chain participants is of primary interest in evolving strategies for these problems. This work focuses on the strategies of these participants since the situation that they face is more complex than that of either pure customers or pure suppliers. To be successful a middleman must service customers by making use of a set of available suppliers and so is presented with a two-sided problem. On one side customer requirements must be fulfilled, but not all customers would prove profitable or their initial requirements unfulfillable. On the other side, obtaining the best price for products is important and may include the selection of the most appropriate supplier. These two problems interact and affect one another with the ability of the middleman to effectively negotiate with suppliers influencing the customers it can profitable deal with and the number of customers and the nature of their requirements influencing which suppliers may be most suitable. This complexity and the need to deal with a dynamic environment make middlemen strategies more interesting than either pure customers or suppliers since the situation faced by both types does not contain the same degree of dynamism.

1.2 Aims And Objectives

The primary aim of this work can be summarised as follows: to evolve effective middleman strategies for the investigation of simple, non-trivial, supply chains.

To achieve this aim, the first objective is to define what problems in particular are to be tackled. Being specific about the problems under consideration is necessary in order to be able to effectively formulate and evaluate strategies through evolution.

In order to make use of evolutionary computation, the next objective is to define a strategy representation. This representation must provide sufficient flexibility to support a wide range of possible behaviours while, at the same time, reducing the possibility of entirely ineffective solutions being generated.

Having devised a strategy representation, the next objective is to develop a platform under which the representation may be used to evolve effective strategies. This includes considering the strategy evaluation mechanism that is critical to the success of the developed platform. This mechanism must both distinguish effectively between the capability of different strategies and not be too computationally expensive. If the mechanism is ineffective there will be insufficient pressure to effectively drive evolution. If the mechanism is too computationally expensive, it will prove impractical to evolve strategies over the long time scales required.

These objectives are summerised below and correlate with the claims laid out in Table 1.1:

1. Develop a way of defining the supply chain situations more precisely so that they can be studied.

2. Develop a way of representing middleman strategies such that they might be evolved.

3. Develop the evolution platform by which middleman strategies maybe evolved.

Having developed a platform for strategy evolution, the aims of experimentation with this platform are as follows. The first aim is to establish that the evolution platform is able to evolve effective strategies and do so under a variety of conditions. The second aim is to investigate how the evolved strategies are affected by the environments in which they form and how they react when placed in a different environment.

These experimental aims are decomposed into the five experimental objectives shown below:

1. Demonstrate that the experimental platform is able to learn middlemen strategies
2. Demonstrate that the learnt strategies are reasonable for the supply chain environment
3. Demonstrate that the experimental platform is able to evolve strategies for a variety of supply chain environments
4. Demonstrate that middlemen strategies tend to converge for a given strategy
5. Demonstrate that middlemen strategies tend to perform best in the environment from which they originate

These experimental objectives are used as the basis of the experimental design discussed in Section 7 and are elaborated upon further there.

1.3 Components Of The Research

With the motivation and objectives of the work discussed above, the research carried out in fulfillment of the these aims is as follows.

Modelling supply chains is accomplished through the Simple Supply Chain Model (SSCM) and investigated in terms of Variant One (SSCM-V1) of this model. The SSCM provides a general method for capturing supply chain situations while SSCM-V1 uses this representation to describe a subset of problems. SSCM-V1 restricting scenarios (S1-3) are used to further specify the problems being considered. This is discussed in Chapter 4.

Middlemen strategies are considered in terms of a strategy framework capable of evolutionary adaptation. This middleman strategy framework is embodied in the SSCM Strategy Framework (SSF) detailed in Chapter 5. Implementation One (SSF-I1) of this framework, in which specific control mechanisms are specified, is used for the evolution of strategies under SSCM-V1 conditions.

For the purposes of investigation the SSCM Market Simulation System (SMSS) provides a platform for the evolution of SSF-I1 based middlemen strategies under SSCM-V1 conditions. The evolutionary component of the SMSS is based on PBIL+GM. The SMSS and its use of PBIL+GM is discussed in Chapter 6.

Experiments to evolve middlemen strategies are conducted using the SMSS and are focused on demonstrating the effectiveness of the SSF and SMSS. In line with the experimental objectives discussed above, this is achieved by first establishing that the SMSS is able to evolve reasonable SSF-I1 based strategies under a set of baseline conditions for the different SSCM-V1 scenarios. Experiments are then conducted to

demonstrate that this is possible for a variety of SSCM-V1 conditions. Finally the SMSS is used to investigate how strategies are affected by environments different to those in which they were originally evolved. The design of these experiments is elaborated upon in Chapter 7, the result being reported in Chapter 8.

The structure of this work is shown graphically in Figure 1.1.

The way in which this work and its component parts relates to work in this and other fields is discussed in Chapter 2. Chapter 3 introduces concepts of direct relevance in a more detailed, context-light, manner.

1.4 Claims

Simple economic problems can be investigated and 'solved' (optimal strategies found) using well-known and studied game-theoretic techniques.

These techniques can not be applied or are impractical to apply in a more complex economic situations where a greater degree of uncertainty exists.

These situations are both common and of increasing importance. There is particular relevance in developing strategies that are appropriate for computers operating in electronic market places.

Since Game Theory can not reasonably be applied to these problems it would be useful to have a mechanism that allows us to investigate and generate reasonable strategies in response to these situations.

The core assertion is that evolutionary computation, appropriately used, allows us to investigate and 'solve' problems of interest.

To this end the work comprises three elements. The first, SSCM, allows us to

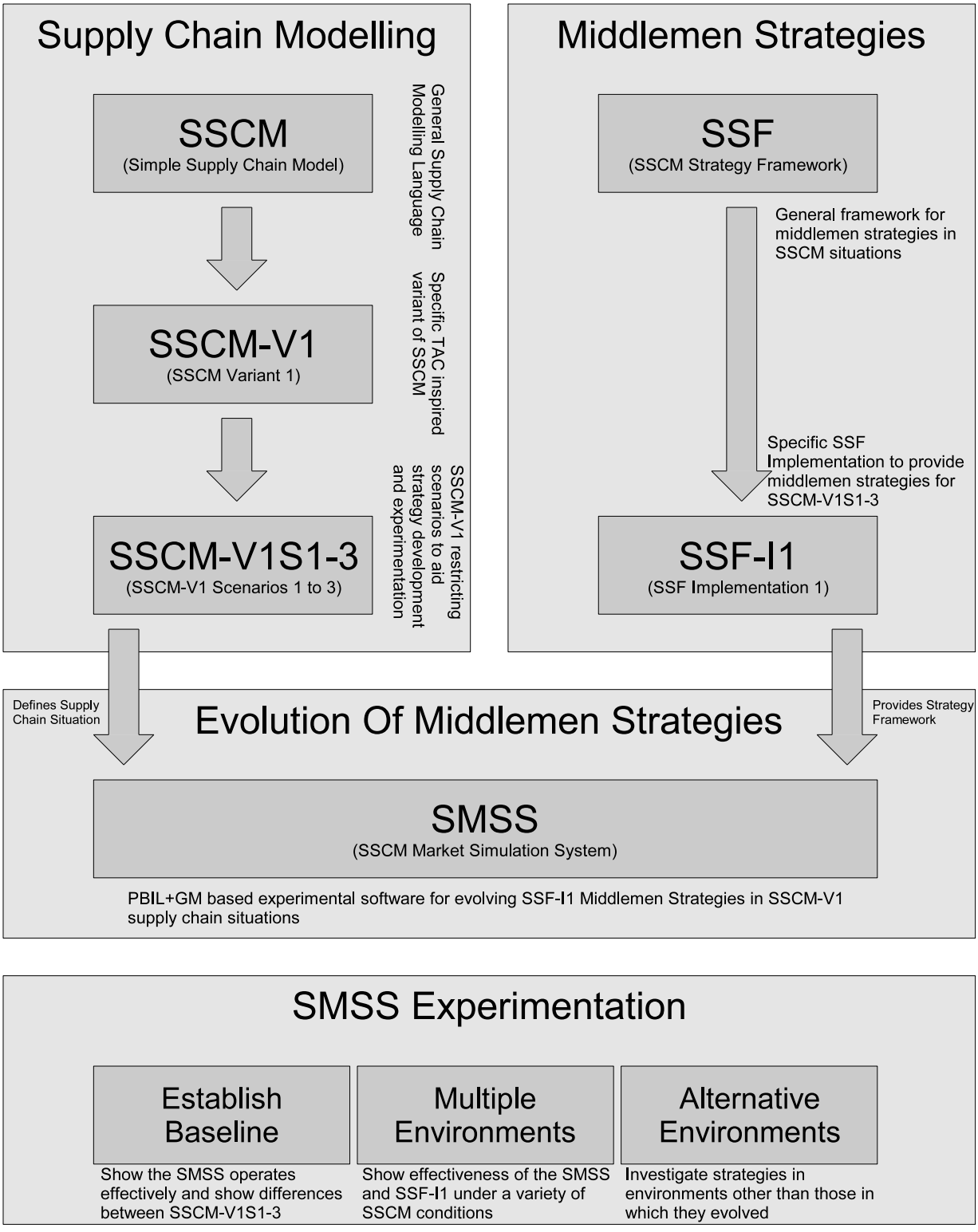

Figure 1.1: Structure Of The Research Work

Table 1.1: The Claims

1	The SSCM provides a way to capture interesting economic problems
2	The SSF provides a feasible framework for defining SSCM middleman strategies
3	The SMSS provides an effective platform for the evolution of strategies within a variety of supply chain environments

capture a set of interesting economic problems that contain a great deal of uncertainty. The second, SSF, provides a way to define middlemen strategies that maybe applied to tackling these problems and which is conducive to evolution. The third, SMSS, provides a way in which to evolve SSF-based strategies through the use of evolutionary computation in the form of PBIL+GM.

As a result of these elements, three core claims are made about the works contributions, see Table 1.1.

Chapter 2

Relationship To Other Work, Comparisons And Issues

This chapter provides an overview of how the work presented here relates to existing research in various fields, with the intention of defining its position in relation to those fields through discussion of its similarities and differences. This discussion helps to highlight some of the issues relating to the SSCM and its use in tackling simple, but non-trivial, supply chains. Chapter 3 provides a more detailed discussion of specific concepts related to this work in a context-light manner, including a more detailed discussion of Game Theory, Evolutionary Computation and the Trading Agent Competition (TAC).

Given the nature of the problem being considered and the approach being taken to tackle it there are a considerable number of research areas that could be and are relevant to this work. Since this thesis is concerned with supply chains, efforts in the area of supply chain management would seem to be of interest, likewise the distributed nature of the problem would suggest work with multi-agent systems, distributed problems solving, distributed optimisation/constraint satisfaction would also be relevant. Since the thesis deals with trading systems, work in bargaining, auctions

and various market simulations would seem to be important. The intended use of evolutionary computation further opens up this area as related and important. To provide an understanding of how the work undertaken here fits into the wider picture the following discussion is organised roughly in accordance with the objectives set down in Chapter 1 rather than by topic area.

Section 2.1 discusses the Simple Supply Chain Model (SSCM), that will be developed in Chapter 4, and how this relates to existing representations and approaches to tackling other economic problems. Section 2.1.1 extends this discussion to consider SSCM-V1; Section 4.3, specifically. Section 2.2 discusses tackling the situations captured by the SSCM and the limitations of Game Theory (GT) for this task. This relates to the SSCM Strategy Framework (SSF) described in Chapter 5. Section 2.3 briefly discusses Evolutionary Computation and its use within the context of the SSCM Market Simulation System (SMSS) introduced in Chapter 6.

2.1 Modelling Economic Problems

The SSCM provides a way to describe simple but non-trivial supply chain situations in their entirety. The SSCM does not attempt to dictate participant behaviour but it does restrict the way in which participants may communicate. The use of the supplied communication mechanism is again left to the individual participants. The problems described by the SSCM are important as they bridge the gap between conventional economics and more complex market simulations while retaining important elements found in real dynamic and uncertain supply chain situations.

The SSCM is related to the original Trading Agent Competition (TAC) by which it was partially inspired. TAC is discussed in relation to SSCM-V1 in Section 2.1.1

and described in greater detail in Section 3.3 of Chapter 3. TAC provides a situation in which middlemen are provided with an allocation of customer requirements which must then be fulfilled by obtaining goods from a number of suppliers and each other via a series of auctions.

The SSCM differs from TAC in a number of important respects. Firstly the SSCM problem is two-sided. Customers in SSCM may be negotiated with to find compromise agreements and do not need to reveal their requirements right away and indeed need not be exclusive to a single middleman. This opens up the possibility of creating more effective middleman strategies to tackle a wider range of situations, it also allows for more direct competition and more complex customer strategies. On the supplier side, individual negotiations are entered in to rather than an auction process used, this effectively reduces the available market information and forces the middlemen to asses which customers they believe can be effectively dealt with. Again this provides a richer environment for the middlemen to operate within. The behaviour of participants is not specified by the SSCM and information about prices and availability of products is not shared; this must be determined by interaction within the system. The lack of market information and restrictions on behaviour create a great deal of uncertainty, uncertainty a middleman would have to face in a real situation. While negotiations between middlemen have not been considered in this work, the SSCM does support this option. Other possibilities are also open. For instance multiple suppliers for a single item, middleman-initiated negotiations with customers and supplier-initiated negotiations with middlemen. These possibilities provide for a wide range of middleman activities and modes of operation. Overall the challenges presented by the SSCM are considerable. As stated before, these challenges

are important to tackle as they relate better to the real uncertainties and dynamic nature of opportunities and problems a participant in a supply chain would face.

During the development of the SSCM it has been intended that customer-middleman interactions essentially form part of a satisfaction problem while middleman-supplier interactions effectively relate to an optimisation problem. Thus, work in distributed constraint satisfaction (DSC) [66] and by association constraint satisfaction (CS) [90] may also be relevant to the model.

Constraint satisfaction models problems in which a number of variables require their values to be specified without violating certain constraints. CS problems (CSPs) are tackled using a central solving system based on many possible techniques. Distributed constraint satisfaction moves the traditional CSP into a distributed domain where different variables are maintained by computationally independent entities. DCS defines problems in a similar way to the SSCM, without restriction on the action of individual participants but some control over what messages are sent and how they are propagated. DCS deals specifically with constraint satisfaction in a cooperative environment where information is shared. SSCM deals with a considerably less formal environment in which cooperation is through mutual advantage and globally good solutions may not be found if it is not to the advantage of individual participants. The communication schemes within SSCM, while adaptable to the given problem domain, are envisaged as being tailored to bargaining and negotiation rather than the relating of constraint information. While there are certainly surface similarities between SSCM and DCS there is considerable divergence in terms of intention and application.

There is a considerable body of work dedicated to various bargaining, negotiation

and auction models [80, 5, 79, 53, 50]. These models in general deal with one-to-one interactions or one-to-many interactions and in the context of SSCM primarily relate to how participants may communicate. The models range from quite simple bargaining situations such as Rubenstein's complete information game [78] to more complex negotiation schemes where multiple issues are under consideration and may be added or removed [15]. These models, while important from a communication point of view, do not capture the larger scope dealt with by the SSCM. While the Rubenstein's bargaining game (and related work [25, 5]), or more relevantly Jennings' auction and negotiation work [3, 2, 85] effectively model one-to-one interactions over a single or multiple issues, they do not capture the larger context within which that interaction is placed. The SSCM is intended to capture that context. Thus, while many of these models deal with the individual interactions in more detail, their conceptual level is different. A further point of relevance and importance is that the SSCM is far removed from complete information games such as Rubenstein's. Participants have very little starting information at all and only the communication protocol to fall back on to make any headway. Rubesnstein's and other bargaining and negotiation models often assume far more information from price distributions to probable participant behaviour, this is something the SSCM does not do. In this sense SSCM has far more in common with other incomplete information games [25, 5]. This has considerable implications for developing strategies to tackle SSCM problems and is discussed further in Section 2.2.

Complex market architectures also exist; one example of these is MAGMA [91] another is MAGNET [21]. MAGMA attempts to provide a complete framework

for agent-based electronic commerce; it deals with issues from individual agent behaviour to banking and goods handling. The MAGMA architecture, while broad in scope and capable of implementing SSCM agents and scenarios, imposes more rigid constraints on agent behaviour than does SSCM; in effect the implementation issues within SSCM and MAGMA overlap but the underlying approach and intention is different. MAGMA could form the basis of an SSCM communication solution. MAGNET is a system focussed on the development of (possibly interlinked) contracts for the fulfilment of agent plans. It encompasses and extends the scope of the MAGMA system, providing a set of agent APIs for use within a regulated market place. Other agent market places exist such as AuctionBot [100] and Kasbah [17], these are effectively focused on how to alleviate users of having to deal with the negotiation process themselves.

Supply Chain Management deals with complex real-world supply chains and the problems faced by organisations trying to optimise them [16, 19]. The SSCM is considerably less complex than the situations dealt with in traditional supply chain management although its results may be applicable in some cases. Further, the SSCM is directed towards a future with greater integration and automation of processes while SCM focuses on providing tools for decision-making and planning.

Distributed Artificial Intelligence (DAI) research has a considerable interest in negotiation between agents and the application of Game Theory to problem solving. The use of negotiation within DAI tends to follow two general approaches each with their own view of the world; the first is that of task assignment with high-level objectives in mind and the second is that of local negotiation between agents where a global view may not be feasible [102]. In DAI systems, however, negotiation is

used between essentially cooperative agents in a manner not too dissimilar to DCS [72]. The cooperative nature of these problems and their primary use with planning systems makes them incompatible with and distinct from the SSCM [19].

2.1.1 Relationships To SSCM-V1

Section 4.3 introduces a more specific SSCM problem, SSCM Variant One (SSCM-V1), based around a travel agent scenario similar to that of the original TAC game. SSCM-V1 restricts the way in which the SSCM is used and makes some assumptions about participant behaviour, this constrains the set of problems that are then considered. SSCM-V1 provides the basis for the development of the SSCM Strategy Framework (SSF), Chapter 5, and investigation of strategy evolution using the SSCM Market Simulation System (SMSS), Chapter 6.

While this SSCM-V1 is similar to the original TAC game there are a number of important differences. Firstly customers are independent entities rather than game server controlled - they can (potentially) negotiate to some limited degree with the middlemen and indeed communicate with multiple middlemen if initially informed about more than one. TAC's insistence that all customer preferences are revealed to middlemen at the beginning of the game does not hold true for SSCM - customers may communicate at any given time with the middlemen they know about and initial communications do not translate into complete information about their preferences. Suppliers in TAC are dealt with via auctions with certain rules applied about how those auctions and the related suppliers behave, SSCM suppliers deal with middlemen in a series of one-to-one negotiations that can include multiple goods as well as price.

Success in TAC is centrally measured and based upon each travel agent's ability to fulfil the supplied customer requirements, SSCM's measure of success (or rather that used within the SMSS) is with regard to the individual middlemen's profitability with the hope that this may prove globally optimal. As with TAC transport times for products are not considered, customers have negotiation cut-off times but delivery details and exceptions are considered outside the model.

Communication between participants is an extremely important issue when modelling a particular problem; the mechanisms and language used has a direct effect on the nature of the agents and their ability to deal with one another. The approach taken to communication has been to make use of a variation on the alternating offers protocol [53]. Any participant may make an offer for goods at a price, the receiving agent may then either accept or reject the offer or make a counterproposal. This scheme while simple allows the research to focus on other areas and allows the agent strategies to be somewhat simpler and computationally less expensive. Other mechanisms could be used, possibly making use of more complex agent communication languages (ACLs) [58]. Argumentation as a mechanism for improving negotiation outcomes by persuading or threatening is currently an active area of research [41, 81, 54] and falls into this 'more complex' category. While argumentation and other systems may be a useful direction to take the work in the future, this is not something that is currently intended.

2.2 Game Theory And Tackling Economic Problems

Having developed a model for a particular problem or problems, the next stage is to develop a strategy or strategies for tackling the model with the aim of informing participant behaviour in these situations.

For many problems, such as Rubinstein's bargaining game mentioned above, Game Theory provides effective techniques for the discovery of optimal strategies. In the case of SSCM problems, Game Theory is of limited help since the situations being dealt with may be sufficiently complex that their analysis would prove intractable. While the SSCM could be used to represent problems of a simple nature, the motivation behind its development has been the capture of more complex situations. To tackle these more complex problems other techniques must therefore be used. A brief introduction to Game Theory and its concepts is provided in Section 3.1.

Game Theory as an approach to tackling SSCM problems is inappropriate due to the wide variety of situations that may be represented and their potential complexity. To accept this it is useful to consider the different elements of a complex SSCM problem that would have to be dealt with in order to form a complete analysis.

Firstly, the product set within the SSCM is dependant upon the particular problem domain being considered. The relationships between products is not captured by the SSCM and therefore its affect on participant behaviour can not be known in advance. To provide a Game Theoretic analysis it would be necessary to know the range of possibilities and how these might affect participant behaviour. It may be possible to construct a Game Theoretic analysis within a particular problem domain, but any

adjustment in the relationship between products would require it to be changed.

Secondly, the SSCM as a whole covers a wide range of possibilities in terms of the participant being considered. Any number of customers, middlemen and suppliers may be present in the system. Customers may have widely divergent requirements and suppliers may be in a position of monopoly or facing strong competition from other suppliers. Either of these participant types may be aware of none, few or many other participants with which they may, or may not, be able to hold a dialogue. Middlemen, meanwhile, have no imposed starting knowledge beyond that of other participants. Any Game Theoretic analysis of this situation would have to take account of each of these possibilities. Again, a specific analysis for a given arrangement of participant distributions and relationships may be possible but would be easily invalidated.

Finally, the SSCM imposes a time-step mechanism, during these time-steps participants may communicate with reference to a further definable component. Unless the specified communication scheme provides a strict ordering mechanism, interaction between participants can essentially occur at any time within the time-step in accordance with the specified scheme. Since participant behaviour is not specified there is no requirement that the communication scheme be used at all, only that it must be adhered to if participants do hold a dialogue. Since communication between participants may occur at any time and these communications may impact what actions should subsequently be taken by the participants, it would be difficult to construct a Game Theoretic model to cope.

Generally, for a specific SSCM problem within a known domain where assumptions can be made about product and participant relationships and for which a strict communication mechanism exists that imposes some order on interactions, it may

well be possible to construct a Game Theoretic analysis to determine how the participants should interact. However, if the conditions are changed a new analysis would need to be undertaken. Within a particular domain for a restricted set of conditions it may be possible to find more general solutions but, as the diversity of situations increases this would rapidly become intractable.

If instead an adaptable participant behavioral framework is created and instances of that framework can be exposed to a range of likely problems with the view to evaluating and adjusting its operation such that it works more effectively, an approximate solution can be found to any set of problems providing they can be captured and described by the SSCM. This is the aim of using Evolutionary Computation within this work as opposed to traditional Game Theoretic analysis of very specific problem instances. Since the problems faced in tackling the SSCM are common to any complex situation with many participants, this general approach is now common within Computational Economic [99, 44].

To this end Chapter 5 discusses the SSCM Strategy Framework (SSF) that has been developed to guide middlemen participants in tackling SSCM-defined problems. The SSF takes an approach similar to a reactive system, as opposed to a planning system, and focuses on fulfilling groups of customer requirements. Part of the responsibilities of this framework are the communication and negotiation with other supply chain participants. The negotiation mechanisms in particular may be varied within the framework and the consideration of an optimal approach is needed.

With regards to bargaining and negotiation mechanisms, and as has been previously mentioned, there is a considerable body of existing work within this field. While

complete information games have had perfect strategies devised ([78]) and various incomplete games have too ([25, 5]), this is largely unhelpful in this context for the reasons discussed above. Partly this is due to the potential (perceived) irrationality of participants (the SSCM does not assume rationality), partially due to the potential multi-issue nature of negotiations, but mostly due to the degree of incomplete information the participants face and the dynamics of the situation. The participant involved in one negotiation is highly likely to be involved in others, many of which will be related; the ability of the participant to know even its own item valuation is limited and indeed it may be better to stop negotiations on short notice if conditions change. For middlemen its customers and suppliers, both of whom it may be in negotiations with simultaneously, primarily determine these conditions.

To provide the SSF with a flexible negotiation mechanism, the approach taken here is based on Matos's work ([67]) due to its high degree of flexibility and limited ability to track other player's negotiation strategies (these mechanisms have also been used in the auction domain [1]). This is critical in giving the participants a range of approaches to use in tackling different negotiators. A further reason for using this negotiation mechanism was its prior use with evolutionary computation, something intended here too.

2.3 Evolutionary Computation For Strategy Generation

The SSCM Market Simulation System (SMSS), Chapter 6, is designed to evaluate SSCM scenario strategies and allow for their optimisation using Evolutionary Computation. The basic system concept is built around a feedback loop. SSCM participants

exist within a market and are able to communicate with the SSCM-applied restrictions, timing and configuration being applied externally. The middleman within this market are highly parameterised, through the SSF, and are the main focus of interest. The customer and supplier while parameterisable are so to a lesser extent and, in effect, make up the environment within which the middlemen operate. The different market participants are configured, allowed to operate in the market and then evaluated. The evaluation of middlemen within the market provides the information that acts as the basis for the evolution of improved middlemen strategies.

Evolutionary Computation is a powerful optimisation tool that has been applied to many domains including negotiation and auction research [67, 1]. The evolutionary component of the SMSS is based around Population Based Incremental Learning (PBIL) [8, 9]. PBIL was selected as it has been applied successfully to various problems ([45, 87, 84, 38]) and allows learning to be undertaken using a small population of middlemen. More recently, work has been carried out reproducing Rubinstein's research in to evolving strategies for Iterated Prisoners Dilemma and comparing a traditional GA approach to that of PBIL ([29]). In this work a variation on PBIL, PBIL with Guided Mutation (PBIL+GM), is used and achieves good results when compared against a Genetic Algorithm (GA).

Since the use of large populations are impractical due to limited computational resources, PBIL provides an effective solution. PBIL combines evolutionary notions with those of reinforcement learning, the traditional population essentially being replaced by a probability distribution. Within the simulation system middlemen agents may be configured from the PBIL distribution; evaluation of middlemen performance over time may then be used to update the distribution and used to configure a new

set of middlemen for testing. The effectiveness of one middleman is measured relative to the rest of the population. Other learning approaches may have been possible (including basic GA's) but PBIL's ability to work with small (sample) population sizes over repeated rounds was an important, and deciding, consideration.

A more general introduction to Evolutionary Computation and PBIL in particular is provided in Section 3.2.

2.4 Conclusions

The problems captured by the Simple Supply Chain Model cover a wide range of situations and as such attempts to tackle them touch on a variety of fields concerned with distributed problem solving, optimisation and planning.

The specific travel agent like SSCM problem considered during the course of this thesis, SSCM-V1, shares some similarities with the original TAC game but provides a far richer environment. This includes a two-sided problem for the middlemen in which negotiations with both customers and suppliers may be considered and far less information being available on pricing and product availability.

The development of strategies to tackle the SSCM relates to a broad swathe of work that attempts to develop strategies for problems with less uncertainty. It also relates to work attempting to tackle similar problems through the creation of more complex agent organisations.

Evolutionary computation provides a powerful tool for the development of strategies. Population Based Incremental Learning (PBIL) is particularly suited to the problems faced here due to its ability to operate effectively using small sample populations.

Chapter 3

Literature Survey

This chapter provides a context-light review of research work closely associated with this thesis.

Section 3.1 provides a brief introduction to Game Theory as both a modelling tool and as an approach to strategy discovery. Section 3.2 introduces Evolutionary Computation, the approach taken within this work for strategy discovery. Section 3.3 discusses the Trading Agent Competition, that partially inspired this work. Section 3.4 provides an overview of the bandwidth trading problem. Finally, Section 3.5 lists prior publications by the author that relate to this work.

3.1 Game Theory

Work in Game Theory [73, 71, 24] may be viewed either as an attempt to explain the behaviour of participants in some setting (descriptive) or as way of helping to inform participants in a given situation about how they ought to act (prescriptive). In each case a model of the situation must first be built before an analysis can reveal or inform behaviour [76, 55]. The model of a problem includes what actions each participant is able to make, what order actions are taken in, what participants know

about each other's actions and what pay-off each participant will receive as a result of the series of actions.

Traditionally, Game Theoretic models are analysed with consideration of different concepts of the relationships and equilibrium between participant strategies. Two important concepts in this regard are the Nash Equilibrium (NE) and Sub-Perfect Game Equilibrium (SPE). NE is the set of possible results reached by a player using its best possible action in response to actions of its opponents. SPE is the result of a game if each player acts such that a NE strategy is played at each sub-game, avoiding the worst possible outcomes at each stage of the game.

While these concepts are useful in analysing Game Theoretic problems they can often be insufficient on their own. In situations where populations of strategies may interact over time the survival and domination of a given strategy in the population is explained by declaring it an Evolutionary Stable Strategy (ESS), that is, a strategy that dominates the population and can not suffer from invasion by other (mutant) strategies. This concept has importance for trying to explain work such as that by Axelrod in to the Iterated Prisoners Dilemma (IPD) [6] where a Genetic Algorithm, see Section 3.2.1, was used to evolve strategies. This work, however, arrives at a successful (but not necessarily optimal, [11, 12, 62, 52]) strategy, not through the analysis of the IPD problem, but through the automatic generation of strategies via evolution. ESS in this context can only be used to try and explain the result after the fact rather than produce an effective strategy on its own. This trend continues today and is now often used within Agent-Based Computational Economics (ACE) [44] where complex problems may be modelled and observed through the use of interacting software agents.

3.2 Evolutionary Computation

Evolutionary Computation (EC) [7] is a branch of machine learning inspired by the process of evolution found in nature. Work in EC focuses on the discovery (or optimisation) of solutions to a modelled problem using a particular representation. Populations of solutions, or representations of that population, are improved through an iterative process of solution-generation and evaluation. The evaluation of solutions being used to inform the next generation process.

Evolutionary Computation has spawned a considerable body of research concerning many possible approaches and their application to a wide range of problems. These approaches are primarily defined by their particular solution representation mechanisms and the manner in which evaluation information is used to inform new solution-generation.

Two well established EC approaches are Genetic Algorithms (GAs) [69, 27], that uses a fixed length string representation, and Genetic Programming (GP) [10, 59], that uses a branching, variable-size representation. More recently, Estimation of Distribution Algorithms (EDAs) [61, 63], that take a statistical approach to the evolution process, have attracted considerable attention. Population Based Incremental Learning (PBIL) [8, 9, 84], as used in this work, is one example of an EDA.

3.2.1 Genetic Algorithms

Genetic Algorithms act upon a population of solution strings. Individual strings comprise a series of alleles (a chromosome) and each allele represents some parameter in a possible solution. Thus, the values of the alleles in a string correspond to the values of the parameters in a solution. An example of GA representation is shown in

Section 3.2.3. GAs aim to iteratively improve a randomly initialised starting population in order to ultimately find good solutions. In each iteration the population is evaluated and this evaluation used to select parent strings from which to form new strings in a new population. Cross-over and mutation operators are commonly used, respectively, to mix existing strings to form new solutions and add new variations to the population. Cross-over achieves this by taking some head proportion of one solution and combining it with the tail of another and vice versa to create two novel solutions. Mutation either adjusts or replaces an existing allele value, and is often applied to new strings after cross-over. Having selected sufficient parents to generate enough new strings for a new population the process may then be repeated.

Genetic Algorithms are well established and have been applied to a wide variety of problems. For instance GAs can be applied to the optimisation of an aircrafts design when that design is considered as a list of specifying parameters [13]. Other applications have included such things as molecular modelling [14], decision support systems for the aesthetic design of bridges [26], process planning for machine shops [101] and improving the performance of protein folding simulations [92]. GAs are also commonly used to evolve the design of artificial neural networks [47, 68]. In relation to this work, strategy generation for the well known Iterated Prisoner's Dilemma (IPD) problem has been tackled by EC in general, [64], and GAs in particular, [6, 29], adding to the considerable debate about what an optimal solution is in this Game Theory defined situation.

3.2.2 Genetic Programming

Genetic Programming also makes use of a population of solutions but in contrast to GAs individuals are comprised of a set of terminal and non-terminal symbols that may be composed in some way to form a solution. GP is powerful in that it allows the representation of more complex problem solutions at the expense of more complex representation and recombination operators. The basic algorithm operates in the same way as GA, evaluating the population of solutions and using this information to create a new population. Great care must be paid to the operators used for the generation of new solutions although variations on cross-over and mutation are often applied.

Genetic Programming is again well established and has proven effective for solution-generation in many domains. For example, GP has been used to discover new, more effective, alternatives to proportional-integral-derivative (PID) controllers that are a commonly used feedback loop component of industrial process control systems [51]. Like PID controllers these new Keane-Koza-Streeter (KKS) controllers, are able to adjust a processes controlling parameters on the basis of a received error signal from that process. The development of KKS may as a result have direct real-world impact in industry.

The RoboCup competition pits computer controller players against one another in both pure simulation and various robotic arenas in an approximation of the game of soccer [49, 48]. Within this environment GP has been used to evolve high-level soccer playing behaviour for agents engaged in the simulation environment and achieved reasonable success against human created teams [65].

With more direct relationship to this work GP has also been used to evolve near-optimal solutions [43, 30] for the Rubinstein's bargaining game [77] comparable to the Game Theory derived solution. Within this game two players split a unit pie between them, alternately making offers and accepting or rejecting those offers. A rejected offer leading to the rejector making the next offer. Both players suffer a discount of their final reward that increases the longer it takes to reach agreement, so providing an incentive to agree earlier.

3.2.3 Estimation Of Distribution Algorithms And Population Based Incremental Learning

Estimation of Distribution Algorithms take a different, more statistical approach to evolutionary processes. While the specifics vary from algorithm to algorithm the basic idea is to build a probability model of the population of solutions and use this to inform specific solution-generation either for a complete new population or for individual solutions for evaluation. Within this work Population Based Incremental Learning (PBIL), an EDA, is used as the evolutionary component of the SMSS, Chapter 6, and is the basis for further EDA discussion below. Other EDA algorithms include Compact Genetic Algorithms (CGAs) [35], Univariate and Bivariate Marginal Distribution Algorithms (UMDAs and BMDAs) [74, 75], and Factorized Distribution Algorithms (FDAs) [70].

PBIL effectively makes use of the same kind of solution representation as a GA, however, instead of maintaining a population of solutions, PBIL maintains a probability distribution that effectively represents that population. For each allele in the solution representation PBIL maintains the frequency or probability of each value for that allele existing in an instantiated solution string. This is shown in Figure 3.1.

PBIL is then able to act directly on this probability distribution rather than maintaining a real population of solutions.

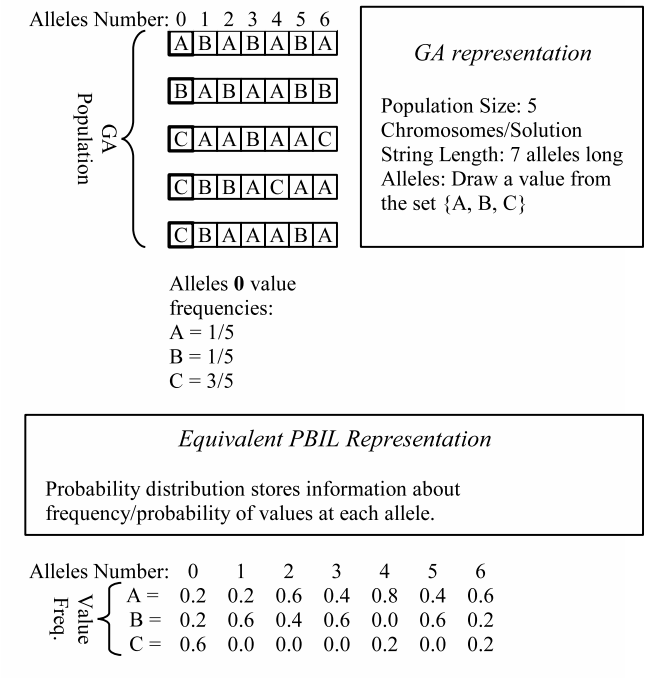

Figure 3.1: Comparison of PBIL and GA representation

While a GA improves its population of solutions over successive iterations, PBIL attempts to improve its probability distribution such that the probability of good values for alleles is increased and the probability of ineffective values is decreased. Improvement of the probability distribution is achieved by instantiating solutions probabilistically from the distribution, evaluating them for their fitness and using this information to adjust, or reinforce, the probability distribution. For example, if a tournament of five solutions is generated from the probability distribution the best of these may be used to reinforce the probability distribution by increasing the probability of values for its alleles by some amount. The reverse is also possible,

taking the worst member of the tournament and reducing the probability of its allele values recurring.

The instantiation of values for solution parameters from the allele value probabilities provides a good mix of values for the parameters that should improve over time. This process is often complemented by a small probability of an arbitrary value being selected at some alleles for a generated solution, in much the same way as mutation is applied in the generation of new GA solution strings after cross-over.

The adjustment of the probability distribution towards (or away) from selecting a given value is achieved via a reinforcement learning rule and mediated by a learning rate, these are similar to those employed for the update of artificial neural network [93] weights.

The specific mechanism used in this work for the generation of solutions from the probability distribution is discussed in Section 6.3. How the probability distribution is updated as a result of evaluations is discussed in Section 6.5.

PBIL has been effectively applied to a range of problems. For instance it has previously been applied to the optimisation of parameters for search algorithms competing in a variety of problems [45], helping to find the most effective algorithm with the most effective parameter set for a given situation.

PBIL has also been applied to tactical driving [87]. In this context PBIL has been used to optimise the controlling parameters of a set of software agents. Each agent of the set is individually responsible for some aspect of driving behaviour, the set acting together to manoeuver the vehicle in a safe and effective manner.

PBIL has also been used to help classify cirrhotic patients receiving transjugular

intrahepatic portosystemic shunt (TIPS) treatment for portal hypertension [38]. Patients receiving this treatment suffer a risk of death within six months and PBIL was used as a component of a system to determine the risk factors involved.

Relating more directly to this work, PBIL has been compared to a traditional GA in the Iterated Prisoner's Dilemma problem [29, 30]. In this work the effectiveness of evolving strategies for IPD was compared between a simple GA and PBIL. PBIL was found to have comparable performance to the GA but superior performance when population sizes were smaller.

3.3 The Trading Agent Competition

In 2000 the first Trading Agent Competition (TAC) was run to provide a focus for research efforts in to autonomous trading agents [88]. TAC was based around a travel agent scenario in which software agents took the place of travel agents attempting to arrange trips between TACTown and Boston (where the competition was being held). Agents in this scenario deal with three types of goods, flights, accommodation and entertainments, and attempt to fulfill a fixed, known, set of customer requirements. Agents bid in auctions (that end at unknown times) to obtain flights and accommodation from suppliers and are also able to trade a predetermined allocation of entertainments amongst one another using a similar mechanism. The customer requirements comprise a known set of preferences for particular goods with associated rewards for fulfillment. A game controller distributes customer requirements, handles the operation of suppliers (in a publicly known way) and maintains the auctions. The final allocation of acquired goods to customers determines an agent's score in

the game. This allocation process may be completed automatically by the game controller using a greedy algorithm on behalf of an agent if it is not forthcoming with its own allocation in time. Penalties arise for unfulfilled requirements and unallocated goods

A summary of the strategies produced for the first TAC competition, TAC-00, is provided by Stone [86]. One example of a TAC-00 strategy is RoxyBot [34]. RoxyBot breaks down the problems of TAC into allocation, assigning goods to clients at the end of the game and completion, and completion, determining the amount of goods to buy given client preferences, existing holdings and the market price. RoxyBot uses a novel 'priceline' data structure with beam search and a greedy heuristic to aid in the second of these problems.

Following the success of TAC-00 the competition has been ongoing, including TAC-01 ([97, 98])/02/03/04/05 and 06. A statistical analysis of TAC-01 [60] suggested that TAC was unable to distinguish one agent as being statistically better than all the others, but was able to determine groups of agents that acted more effectively than others. This analysis suggested various methods of improving the situation. One successful agent that participated in TAC-01 was ATTac-2001 [82]. This agent made use of a new boosting based algorithm for conditional density estimation. This was used in order to learn the likely price of products within a game given the current known market conditions.

TAC-02's best strategies were the agents WhiteBear [94] and SouthamptonTAC [36]. An analysis of TAC-02, [33] provides an overview of all the TAC-02 strategies and notes that as the competition has matured the diversity of techniques applied and their motivation has increased.

In 2003 the trading agent competition expanded to include a new supply chain management based problem TAC-SCM [4, 96]. TAC-SCM is based around a factory producing personal computers (PCs). This problem combines tasks including responding to customer request for quotes (RFQs), issuing RFQs for components from suppliers (with which to build PCs), scheduling of a plant to produce PCs from the basic components and deciding which customers to delivery which PCs to and when. PCs of various configurations may be built with dependencies between some components. An inventory fee is charged for the storage of components and assembled PCs in order to provide a disincentive for stockpiling. Following the introduction of TAC-SCM, the original TAC game was continued under the name TAC-Classic and continued to be run in parallel with TAC-SCM.

For TAC-05, TAC-Classic was won by the Mertacor agent [89], using an approach based on individual bidding strategies for each product type combined with linear programming. TAC-SCM was won by the TacTex-05 agent [23] using a strategy combining various approaches including Bayesian modelling of likely customer demand and adaptation of play to known opponents based on passed experience. An analysis of entrants to both TAC-05 competitions is provided in [99].

3.4 Bandwidth Trading

Trading network bandwidth as a commodity is, despite early setbacks, seen as a potentially effective way to make more efficient use of large, modern, heterogeneous telecommunications networks, [46, 83, 18, 39, 42]. The motivation for using such a scheme is an improvement in network utilisation leading to increased revenues for both the network operators and service providers.

The effective trading of bandwidth is contingent upon the adoption of standard service agreements by all participating buyers and sellers. These agreements are needed to smooth over the differences between specific network architectures and operation as well as providing benchmarks for quality of service and contingencies for when commitments are not met.

Along with the adoption of standard agreements, appropriate technology is also required such that the agreements can be acted upon and monitored in a timely and effective manner. More recent network automation mechanisms would help to enable this [46, 40, 39].

3.5 Previous Published Works Relating To This Thesis

Prior publications have provided an overview of the SSCM and SSCM-V1, [28, 31, 30], the SSF, [32, 30] and SMSS, [32, 30], in the context of evolving middlemen strategies.

The evolutionary component of the SMSS, PBIL+GM, has previously been used to study the Iterated Prisoners Dilemma game in comparison with a GA, [29].

Chapter 4

The Simple Supply Chain Model (SSCM)

The principle aim of this work is to study interesting economic problems that cannot be approached by traditional techniques. Specifically we are interested in the interaction between participants in a supply chain and what strategies may be employed by those participants. The first step towards achieving this is the ability to specify these problems in an unambiguous way. The Simple Supply Chain Model (SSCM) provides a way to capture the elements of a supply chain in order that it might be studied. The SSCM defines a supply chain in terms of its participants, the customers, suppliers and middlemen that operate in the market, the length of the market in time and the way in which participants may communicate. While the SSCM defines what these participants know at the outset it does *not* specify how they should act.

Section 4.1 of this chapter introduces the SSCM and describes how it specifies the different supply chain elements. Section 4.2 goes on to demonstrate how the SSCM can be used to model two familiar problems, a travel agent scenario and bandwidth trading. Section 4.3 describes SSCM Variant One (SSCM-V1), a specific version of

the captured travel agent scenario that is used as the basis of experimentation in this work. Section 4.4 discusses the SSCM-V1 Scenarios, a series of restrictions placed upon the SSCM-V1 for the purposes of further investigation. Since the SSCM captures the complete start state of a supply chain, discussing specific instances or classes of problems can prove problematic. Section 4.5 introduces methods for describing a supply chain more abstractly for the purposes of easier discussion and consideration. The chapter concludes with a summary in Section 4.6.

4.1 The SSCM

The SSCM is designed to specify supply chains in terms of their participants starting knowledge and the way in which these participants may interact through communication. Three types of participants are represented customers, suppliers and middlemen, each of which has a different set of starting knowledge and different objectives within the supply chains. Customers seek to fulfill some set of requirements, suppliers wish to sell their products at a good price and middlemen aim to bring these two parties together in order to leverage a profit.

The supply chains specified by the SSCM, while simple in themselves, are complex from an economic point of view. From the perspective of a middleman in the chain the problem presented is two-sided and contains considerable uncertainty. On the customer-middleman side a satisfiability problem exists. Can the middleman find a product set that will satisfy customers and still make a profit? On the middleman-supplier side an optimisation problem exists. Can the middleman obtain the required products and do so at the best possible price? The problem faced by middlemen in an SSCM environment is elaborated upon further in Section 4.3.1.

Participants are defined in terms of their knowledge at the beginning of the market. For customers this is a set of acceptable alternative requirements along with the budget available to fulfill these and the set of middlemen known. For suppliers the knowledge consists of the products they are able to supply, the amount available, their basic value and the middlemen they may be able to sell to. Middlemen are defined simply in terms of the customers and suppliers they are aware of.

In addition to this basic information the model has a total time length restriction and all participants have a limit imposed on the number of outbound communications they are able to make per time unit. Further, the way in which participants may interact is constrained by specifying the communication scheme used as this will likely have a bearing on participant behaviour and effectiveness.

4.1.1 The SSCM - Top Level Description

A top level SSCM description includes the products to be traded, P, the set of supplier, S, middlemen, M, and customer, C, information, the duration of the supply chain, T_{total} and T_{active}, and the communication mechanism being used by participants, Com. See Definition 4.1 and Table 4.1.

$$SSCM = \{P, S, M, C, T_{total}, T_{active}, Com\} \tag{4.1}$$

The total duration of a market is the time in which all participants have to communicate in. The active period specifies the length of time at the end of the market during which customer requirements occur. That is to say, customers only require products within the active period.

The active period may be set to less than that of the total duration in order

Table 4.1: SSCM Top Level

P	The set of Products
S	The set of Suppliers
M	The set of Middlemen
C	The set of Customers
T_{total}	The total duration of the market
T_{active}	The duration of the markets active period
Com	The communication scheme in use

that middlemen and suppliers have a chance to fulfill the customers with the earliest requirements. If the active period is identical to the total duration, middlemen and suppliers may have no opportunity to fulfill a customers requirement. This inability to fulfill a customer's requirement would not result from strategic inability but simply through lack of available time. The inclusion of a possibly shorter active period is intended to mitigate this.

4.1.2 The SSCM - Traded Products

The set of products, P, traded in the captured supply chain is specified as follows in Definition 4.2 and Table 4.2.

$$P = \{P_1, .., P_{pn}\} \tag{4.2}$$

Table 4.2: P, Products

P_x	A numbered product 1 to pn
pn	The total number of products in the model

4.1.3 The SSCM - Suppliers

The specification of a supply chain supplier, S_x, consists of the information about which products it is able to supply, what other participants it knows of and what its

communication restrictions are. This is shown in Definition 4.4 and Table 4.3.

$$\begin{aligned} S &= \{S_1, .., S_{sn}\} \\ S_x &= \{SP_x, SKMM_x, SKC_x, SKS_x, SComOut_x\} \end{aligned} \tag{4.3}$$

Table 4.3: S, Suppliers

S_x	A numbered supplier 1 to sn
sn	The total number of suppliers in the model
For each supplier x	
SP_x	Information related to the set of products supplied by the supplier
$SKMM_x$	The set of known middlemen
SKC_x	The set of known customers
SKS_x	The set of known suppliers
$SComOut_x$	The number of outbound communication allowed in one time-step

Each supplier has a set of supplied products, SP_x. For each supplied product the supplier maintains an available quantity and base value. This is shown in Definition 4.4 and Table 4.4.

$$\begin{aligned} SP_x &= \{SP_{x^1}, .., SP_{x^{spn_x}}\} \\ SP_{x^y} &= \{PS_{x^y}, BaseValue_{x^y}, Quantity_{x^y}\} \end{aligned} \tag{4.4}$$

The $BaseValue$ and $Quantity$ are considered here to be fixed values but could be replaced by functions or a list of base values and quantities for each time step and so increasing the model's complexity. While this is certainly a possibility it must be reinforced that the SSCM is intended to capture only the starting state of a supply

Table 4.4: SP_x, Supplied Products

SP_x	Products supplied by supplier x ranging from 1 to spn_x
spn_x	The total number of products supplied by x
For each supplied product x^y	
PS_{x^y}	The product supplied $PS_{x^y} \in P$
$BaseValue_{x^y}$	The base value of a product to the supplier
$Quantity_{x^y}$	The quantity of product available for each time step

chain, not describe the participant behaviour in that market that this form of added complexity might represent.

Definition 4.5 and Table 4.5 describe the representation of the participants known by a supplier.

$$\begin{aligned} SKMM_x &= \{SKMM_{x^1}, .., SKMM_{x^{skmmn_x}}\} \\ SKC_x &= \{SKC_{x^1}, .., SKC_{x^{skcn_x}}\} \\ SKS_x &= \{SKS_{x^1}, .., SKS_{x^{sksn_x}}\} \end{aligned} \tag{4.5}$$

4.1.4 The SSCM - Middlemen

Middlemen, M, are defined in terms of the other participants they know and how many outbound communications they are able to make in each time step. This is shown in Definition 4.6 and Table 4.6.

$$\begin{aligned} M &= \{M_1, .., M_{mn}\} \\ M_x &= \{KS_x, KC_x, KM_x, MComOut_x\} \end{aligned} \tag{4.6}$$

The sets of participant known to the middleman divided by type. This is shown in Definition 4.7 and Table 4.7.

Table 4.5: $SKMM_x$, SKC_x & SKS_x, Suppliers, Known Participants

Middlemen	
$SKMM_x$	The set of middlemen known by supplier x ranging from 1 to $skmmn_x$
$skmmn_x$	The total number of middlemen known by supplier x
$SKMM_{x^y}$	The known middleman, $SKMM_{x^y} \in M$
Customers	
SKC_x	The set of customers known by supplier x ranging from 1 to $skcn_x$
$skcn_x$	The total number of customers known by supplier x
SKC_{x^y}	The known customer, $SKC_{x^y} \in C$
Suppliers	
SKS_x	The set of suppliers known by supplier x ranging from 1 to $sksn_x$
$sksn_x$	The total number of suppliers known by supplier x
SKS_{x^y}	The known supplier, $SKS_{x^y} \in S$

Table 4.6: M, Middlemen

M_x	A numbered middleman 1 to mn
mn	The total number of middlemen in the model
For each middleman x	
KS_x	The set of known suppliers
KC_x	The set of known customers
KM_x	The set of known middlemen
$MComOut_x$	The number of outbound communication allowed in one time-step

$$
\begin{aligned}
KS_x &= \{KS_{x^1}, .., KS_{x^{ksn_x}}\} \\
KC_x &= \{KC_{x^1}, .., KC_{x^{kcn_x}}\} \\
KM_x &= \{KM_{x^1}, .., KM_{x^{kmn_x}}\}
\end{aligned}
\tag{4.7}
$$

Table 4.7: KS_x, KC_x & KM_x, Middlemen, Known Participants

Suppliers	
KS_x	The set of suppliers known by middleman x ranging from 1 to ksn_x
ksn_x	The total number of suppliers known by middleman x
KS_{x^y}	The known supplier, $KS_{x^y} \in S$
Customers	
KC_x	The set of customers known by middleman x ranging from 1 to kcn_x
kcn_x	The total number of customers known by middleman x
KC_{x^y}	The known customer, $KC_{x^y} \in C$
Middlemen	
KM_x	The set of middlemen known by middleman x ranging from 1 to kcm_x
kmn_x	The total number of middlemen known by middleman x
KM_{x^y}	The known middleman, $KM_{x^y} \in M$

4.1.5 The SSCM - Customers

A supply chain customer, C_x, is defined in terms of its requirements, known participants and its restriction on communication use. See Definition 4.8 and Table 4.8. The desire to fulfill its requirements should be a prime motivation for customers in the supply chain.

$$\begin{aligned} C &= \{C_1, .., C_{cn}\} \\ C_x &= \{Req_x, CKMM_x, CKS_x, CKC_x, CComOut_x\} \end{aligned} \tag{4.8}$$

The participants known to customers divided by type. This is shown in Definition 4.9 and Table 4.7.

Table 4.8: C, Customers

C_x	An individual customer 1 to cn
cn	The total number of customers
For each customer x	
Req_x	The specification of the customers requirements
$CKMM_x$	The set of known middlemen
CKS_x	The set of known suppliers
CKC_x	The set of known customers
$CComOut_x$	The number of outbound communications allowed per time step

$$
\begin{aligned}
CKMM_x &= \{CKMM_{x^1}, .., CKMM_{x^{ckmmn_x}}\} \\
CKS_x &= \{CKS_{x^1}, .., CKS_{x^{cksn_x}}\} \\
CKC_x &= \{CKC_{x^1}, .., CKC_{x^{ckcn_x}}\}
\end{aligned}
\tag{4.9}
$$

Table 4.9: $CKMM_x$, CKS_x & CKC_x, Customers, Known Participants

Middlemen	
$CKMM_x$	The set of middlemen known by customer x ranging from 1 to $ckmmn_x$
$ckmmn_x$	The total number of middlemen known by customer x
$CKMM_{x^y}$	The known middleman, $CKMM_{x^y} \in M$
Suppliers	
CKS_x	The set of suppliers known by customer x ranging from 1 to $cksn_x$
$cksn_x$	The total number of suppliers known by customer x
CKS_{x^y}	The known supplier, $CKS_{x^y} \in S$
Customers	
CKC_x	The set of customers known by customer x ranging from 1 to $ckcn_x$
$ckcn_x$	The total number of customers known by customer x
CKC_{x^y}	The known customer, $CKC_{x^y} \in C$

Req_x defines the set of requirements for the given customer x. The customer requirement needs to specify what a customer desires, the limits on possible variation

from this and the funds available to obtain it. The way in which customer requirements are structured is likely to be strongly dependant on the specific supply chain problem being considered. In consequence it is difficult to provide a mechanism that is sufficiently generic such that it may capture customer requirements from a wide variety of problems. This difficulty is tackled here by splitting customer requirements in to a series of name/value pairs. The nature of these pairs and their meaning to the participants will be specific to a particular class of supply chain and must be defined along side the problem. While the names used may be constant for a class of problem the associated values would vary from problem instance to problem instance. A named value could be a numeric quantity or something more complex such as a list. This representation mechanism provides a considerable degree of flexibility for defining a customer requirement at the expense of some additional complexity when modelling a specific problem. See Definition 4.10 and Table 4.10.

$$Req_x = \{ReqName_{x^1}, ReqValue_{x^1}, ..., ReqName_{x^{reqn_x}}, ReqValue_{x^{reqn_x}}\} \quad (4.10)$$

Table 4.10: Req_x, Customers, Requirements

Req_x	Requirement set for customer x
$ReqName_{x^y}$	The name of the requirement y for customer x
$ReqValue_{x^y}$	The value of the requirement y for customer x
$reqn_x$	The number of requirements for customer x

The name/value pair formulation may be used to represent complex requirement information. In the example below, see Definition 4.11 and Table 4.11, a simple representation is used to specify a customer budget and a specific product being sought. In this case a customer, n, has a two part requirement (the budget and

product) in a supply chain with at least one product, P_1.

$$Req_n = \{ReqName_{n^1}, ReqValue_{n^1}, ReqName_{n^2}, ReqValue_{n^2}\} \tag{4.11}$$

Table 4.11: Simple Customer Requirement Example, Product P_1 is sought for a budget of 10.0

$reqn_n$	2
$ReqName_{n^1}$	BUDGET
$ReqValue_{n^1}$	10.0
$ReqName_{n^2}$	PRODUCT
$ReqValue_{n^2}$	P_1

4.1.6 The SSCM - Communication Between Participants

The communication scheme, Com, defined as part of the SSCM specifies how the various supply chain participants are able to interact. This interaction is the only channel through which participants are able to meet their objectives. Therefore, while the communication scheme does not directly determine participant behaviour it is likely to have some bearing on their operation.

The communication scheme, like the customer requirements above, is liable to be problem-specific and in modelling a particular class of problem the way in which the participants interact is likely to have an affect on the resolution of the supply chain. To aid the specification of a communication scheme it is first broken down in to the sets of communication utterances available between the different participant types within the supply chain, see Definition 4.12 and Table 4.12.

$$Com = (ComCtoMM, ComMMtoC, ComCtoS, ComStoC, \quad (4.12)$$
$$ComMMtoS, ComStoMM, ComCtoC, ComMMtoMM, ComStoS)$$

Table 4.12: Com, The Basic Communication Break Down

$ComCtoMM$	Communication from Customers to Middlemen
$ComMMtoC$	Communication from Middlemen to Customers
$ComCtoS$	Communication from Customers to Suppliers
$ComStoC$	Communication from Suppliers to Customers
$ComMMtoS$	Communication from Middlemen to Suppliers
$ComStoMM$	Communication from Suppliers to Middlemen
$ComCtoC$	Communication between Customers
$ComMMtoMM$	Communication between Middlemen
$ComStoS$	Communication between Suppliers

The above decomposition is used since the available communication between participants may be asymmetric and we may wish to consider the option of communication between participants of the same type. In consequence, the utterances available between a pair of participant could be identical, similar or entirely different depending on the problem being modelled. Having broken the communication scheme down between participant type pairs, the communication between these pairs can then be further decomposed. Each element of Com comprises a set of communication utterances available to the particular pair of supply chain participant types.

The number of utterances types available to each pair of participants is shown in Table 4.13.

The structure of individual communication sets then take the form shown in Definition 4.13.

Table 4.13: *Com*, The Number Of Communication Utterance Types For Each Participant Type Pair

Number of communication utterance types from	
$cCtoMMn$	Customers to Middlemen
$cMMtoCn$	Middlemen to Customers
$cCtoSn$	Customers to Suppliers
$cStoCn$	Suppliers to Customers
$cMMtoSn$	Middlemen to Suppliers
$cStoMMn$	Suppliers to Middlemen
Number of communication utterance types between	
$cCtoCn$	Customers
$cMMtoMMn$	Middlemen
$cStoSn$	Suppliers

Com, Structure Of Communication Utterance Sets

$$
\begin{aligned}
ComCtoMM &= \{CCtoMM_1, ..., CCtoMM_{cCtoMMn}\} \\
ComMMtoC &= \{CMMtoC_1, ..., CMMtoC_{cMMtoCn}\} \\
ComCtoS &= \{CCtoS_1, ..., CCtoS_{cCtoSn}\} \\
ComStoC &= \{CStoC_1, ..., CStoC_{cStoCn}\} \\
ComMMtoS &= \{CMMtoS_1, ..., CMMtoS_{cMMtoSn}\} \\
ComStoMM &= \{CStoMM_1, ..., CStoMM_{cStoMMn}\} \\
ComCtoC &= \{CCtoC_1, ..., CCtoC_{cCtoCn}\} \\
ComMMtoMM &= \{CMMtoMM_1, ..., CMMtoMM_{cMMtoMMn}\} \\
ComStoS &= \{CStoS_1, ..., CStoS_{cStoSn}\}
\end{aligned}
\tag{4.13}
$$

Along with the symmetric and asymmetric communication scenarios above, by using this mechanism it is possible to assert that communication between certain participant types is disallowed by specifying that the related communication set is empty.

The specification of individual utterances makes use of a name/value pair arrangement, see Definition 4.14 and Table 4.14, similar to that applied to customer requirements in Section 4.1.5. Under this arrangement utterances are defined in a series of name/value pairs specific to the class of problem being considered. When specifying the type of problem the number and meaning of the utterance name/value pairs must also be considered. A value could be simple, such as a numeric value, or more complex, such as a list. Within a class of problems names would have consistent meaning and values would vary across specific problem instances.

$$\begin{aligned} ComUtterance &= \{(ComName_1, ComValue_1), ..., \\ &\quad (ComName_{comn}, ComValue_{comn})\} \end{aligned} \tag{4.14}$$

Table 4.14: *ComUtterance*, Communication Utterance Description (e.g. for $CCtoMM_1$)

$ComName_x$	A particular utterance specifying name
$ComValue_x$	The value associated with that name
$comn$	The number of name/value pairs in the utterance

4.1.7 The SSCM - Summary

The SSCM defines the start position, length in time and mechanisms for interaction for a supply chain containing three different participant types (customers, middlemen and suppliers).

A customer's start position is defined in terms of its requirements, known participants and limitations on communication. The customer requirement is defined using a flexible name/value pair mechanism, the use of which would be specific to the

modelling of a particular class of supply chain problem. The aim of a customer is to fulfill its requirement.

The supplier start position is defined in terms of the products it is able to supply, known participants and limitations on communication. The set of supplied products includes information on the quantities available and their base value to the supplier. Suppliers aim to sell products profitably.

Middlemen are defined in terms of the other supply chain participants they are aware of and their outbound communication limitation. Middlemen aim to leverage a profit by fulfilling requirements on the behalf of multiple customers through trade with multiple suppliers.

The communication scheme used by participants is defined in terms of the set of communication utterances that are valid between two participant types in a particular direction. A specific utterance is defined in terms of name/value pairs the meaning of which will be specific to either the communication mechanism chosen and/or the class of problem being investigated. The set of viable utterances between pairs may be empty indicating no communication is allowed between those types in that direction.

Different classes of problem may be considered by making use of different product sets and requirement representations and adjusting which participants are aware of each other and how these may communicate. Situations may be modelled in which customers deal with middlemen to fulfill their requirements, the middlemen themselves obtaining products from one or many suppliers. Under these conditions how the different participants interact and what strategies can be deployed to tackle the challenges faced by each could be investigated. Various other scenarios can also be

modelled. For example, customers may make use of middlemen to fulfill their requirements but also have the option to communicate directly with suppliers. In this case the incentives that drive customers to make use of a middleman might be investigated. Overall, the SSCM is able to capture situations that contain a great degree of uncertainty and so more closely resemble real world problems than many problems traditional approached using Game Theoretic techniques. Section 4.2 demonstrates how the SSCM can be used to capture two familiar problems for study.

4.2 From Business Model to SSCM

This section discusses how the SSCM can be used to model different businesses scenarios by considering two familiar problems.

The first of these is a travel agent scenario similar to that used for the original Trading Agent Competition game (TAC-Classic, see Section 3.3). In this scenario customers desire to travel to a known destination for some duration and with the option of different entertainment once there. This scenario is used as the basis of SSCM-V1, Section 4.3, the SSCM variant investigated in this thesis.

The second problem is a bandwidth trading scenario in which customers requiring some amount of network access may obtain a composite of network links from several possible suppliers via a middleman.

In considering these scenarios, details of customer requirement representation, the naming of products and the specification of a communication mechanism and its restrictions will be considered.

Sections 4.2.1 and 4.2.2 discuss the travel agent and bandwidth trading scenarios respectively. Section 4.2.3 provides an explicit comparison of the representation of these scenarios through instantiated SSCM instances of simple problems of each type.

4.2.1 The Travel Agent Scenario

The travel agent scenario considered here is similar to the original Trading Agent Competition game (TAC-Classic, see Section 3.3) in that customer requirements consist of the desire to travel to a remote location within some time frame, for some duration and with the expectation of accommodation and entertainment while there. For simplicity, and unlike the bandwidth trading scenario (Section 4.2.2 below), only two locations are considered the home and away locations. The travel agent scenario is illustrated in Figure 4.1.

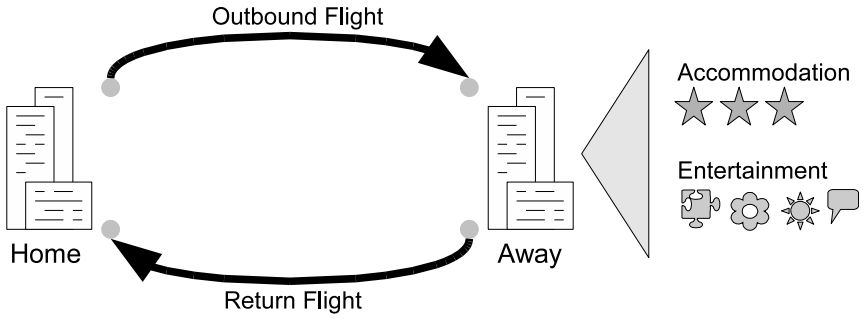

Figure 4.1: The Travel Agent Scenario

This section details how this scenario may be captured by the SSCM for further study. A simple, fully defined, travel agent problem of this type is shown in Section 4.2.3.

4.2.1.1 Products And Time In The Travel Agent Scenario

Time-steps within the travel agent scenario are at the granularity of days since, in reality, customers are unlikely to require travel packages at a resolution less than this. Further, accommodation is usually booked per night and regular flights tend to be made each day between two destinations. Keeping the granularity of time-steps within the model the same same as the considerations of the supply chain participants helps reduce the rest of the models complexity when capturing the problem. Within this in mind we may capture one month of time, thirty days, in which customer requirements should be met by specifying $T_{active} = 30$. To allow a reasonable time for trips early in this period to be obtained a further seven days is added in which deals for this earlier period can be made. This leads to setting $T_{total} = 37$.

Products within the travel agent scenario comprise three basic types, flights, accommodation and entertainment. Flights are required to get customers between the home and away locations. Accommodation is required for the duration of the customers stay at the away location. Entertainment is something a customer may desire while away. Flights between home and away locations could, potentially, be at different times during the day with different quality of seats being available. Each standard flight time and seat quality must be captured as a separate product within the SSCM. Similarly accommodation at the away location might be of varying qualities. Each variation would again need to be a separate product. Finally, each type of entertainment a customer may desire while away must be represented. With the above in mind Table 4.15 demonstrates how a set of the products might be mapped on to SSCM. This product set defines twelve product types, $pn = 12$, including early and

Table 4.15: The Travel Agent Scenario, Example Products

Product	*Name Mapping*
With a $pn = 12$	
P_1	$FlightOutEarlyLowQuality$
P_2	$FlightOutEarlyHighQuality$
P_3	$FlightOutLateLowQuality$
P_4	$FlightOutLateHighQuality$
P_5	$FlightBackEarlyLowQuality$
P_6	$FlightBackEarlyHighQuality$
P_7	$FlightBackLateLowQuality$
P_8	$FlightBackLateHighQuality$
P_9	$AccommodationLowQuality$
P_10	$AccommodationHighQuality$
P_{11}	$EntASightSeeingTour$
P_{12}	$EntBFairground$

late outbound and return flights at two seat qualities, two qualities of accommodation at the away location and two types of entertainment.

With the set of available products defined how these are related to customers and suppliers can then be considered.

4.2.1.2 Customers And Suppliers In The Travel Agent Scenario

Customers within the travel agent scenario wish to travel to the away location within a certain time frame, for some duration and having an expectation about flights, accommodation, entertainment and cost. For example, a customer may wish to travel to the away location for five days within a two week period expecting high quality flights and accommodation but be ambivalent to entertainment while away. Along

with these requirements it is also reasonable to assume that the customer would require some notice in advance of the trip in order to prepare to leave. This notice period would be the latest time before the beginning of a trip that the customer would be willing to accept notification of the requirements being fulfilled. Continuing the example we can specify that the customer is willing to pay 1000.0 for the travel package, wishes to depart no earlier than the seventh active day (total day fourteen), and requires at least five days notice to prepare to leave. With reference to Section 4.1.5 this customer's requirements could be represented using the name/value pair arrangement shown in Table 4.16, for customer one, C_1, of the model. For a specific instance of the travel agent scenario many customers, with their associated requirements, would exist.

Suppliers within the travel agent scenario are able to provide some set of the products specified in Table 4.15 for each time-step during the models active period. Unlike the bandwidth trading scenario suppliers are likely (but not forced) to be distinguished by the the types of products they are able to supply. Airlines would supply flights, hotels supply accommodation and different venues provide entertainment. Under this arrangement and for the products specified six suppliers, $sn = 6$, would be reasonable, two each for flights and accommodation and one each for the types of entertainment. Each supplier, S_x, is able to supply some set of products, SP_x. For each product y that can be supplied the supplier has an associated base value, $BaseValue_{x^y}$, and quantity available, $Quantity_{x^y}$, for each of the active time steps. While the base value of products may be unaffected by time, the availability of products under the travel agent scenario may be subject to time effects. This is reasonable for airlines which may run different flights on different days. For this to

Table 4.16: Example Of A Customer's, (C_1), Requirement In The Travel Agent Scenario

$reqn_1$	9
$ReqName_{1^1}$	BUDGET
$ReqValue_{1^1}$	1000.0
$ReqName_{1^2}$	EARLIEST_START
$ReqValue_{1^2}$	14
$ReqName_{1^3}$	LATEST_END
$ReqValue_{1^3}$	28
$ReqName_{1^4}$	MIN_DURATION
$ReqValue_{1^4}$	5
$ReqName_{1^5}$	MAX_DURATION
$ReqValue_{1^5}$	5
$ReqName_{1^6}$	FLIGHT_QUALITY
$ReqValue_{1^6}$	$HIGH$
$ReqName_{1^7}$	ACCOMMODATION_QUALITY
$ReqValue_{1^7}$	$HIGH$
$ReqName_{1^8}$	DESIRED_ENTERTAINMENT
$ReqValue_{1^8}$	{} (none)
$ReqName_{1^9}$	MIN_NOTICE
$ReqValue_{1^9}$	5

be modelled within the SSCM the quantity associated with a supplied product would not be a single value representing the quantities available for each day but, instead, would be a list of quantities for each active time step.

Both customers and suppliers are aware of middlemen within the travel agent scenario but need not be aware of any other participants to fulfill their aims. Restrictions on how the participants interact are considered in Sections 4.2.1.3 and 4.2.1.5 below.

4.2.1.3 Middlemen In The Travel Agent Scenario

Middlemen in the travel agent scenario aim to leverage a profit from fulfilling requirements on behalf of multiple customers by obtaining products from several different suppliers. Middlemen start with no knowledge beyond that of the other participants

they are aware of and must wait for, or instigate, a dialog with customers in order to discover what opportunities are present. Middlemen must also, as part of their problem-specific behaviour not specified by the SSCM, have the ability to decompose customer requirements such that they can attempt to find viable product sets to fulfill them.

The aim of the middlemen is complicated by the interaction of customer requirements, product price and availability and how these vary over time. A middleman must make decisions about which customer requirements might be reasonably and profitably fulfilled, which suppliers to deal with, what specific products to obtain and what to do if any of these decisions appear erroneous as time progresses. The middleman faces a two-sided problem. On one side a decision must be made about which customers to deal with and a product set acceptable to those customers must be found. On the other side choices must be made about which suppliers to use and how to achieve the lowest price for products. These problems interact to make the task faced by middlemen more difficult. Customer budgets affect the flexibility with which middlemen may negotiate for products and the availability and price of products at the supplier affects the travel packages the middlemen are able to offer. The problem is compounded by the middleman having no prior knowledge defined with regards to product availability, price or likely demand.

The starting knowledge of middlemen is defined primarily in terms of the other supply chain participants they are aware of, this information would strongly affect their behaviour and effectiveness within the supply chain. A middleman that is unaware of any suppliers is likely to face considerable difficulties unless suppliers are aware of it and have reason to start a dialog. Without suppliers a middleman

will not be able to fulfill any customer requirements and so fail to make any profit. Suppliers may, or may not, have an incentive to initiate a dialog with middlemen depending on the level of demand they experience and how willing they are to expend communication resources for this exploratory purpose. Middlemen are likely to be known by some set of customers for the service they are able to offer. If, however, the middlemen are aware of some set of customers from the outset their may be an incentive to open a dialog in an attempt to dissuade customers from competitors. The position of middlemen may become more difficult if variations on the basic scenario are considered. If, for instance, customers were able to deal directly with suppliers, middlemen would need to offer some incentive to customers in order to be profitable. Trade between middlemen might also be considered, allowing middlemen the chance of mitigating the cost of products that were purchased in error.

With the above in mind a basic travel agent scenario would consist of customers that are aware of one or more middlemen and middlemen that are aware of all suppliers or enough to fulfill most customer requirements. Other interactions between participants are not considered. For a customer x this would mean the set of known middlemen, $CKMM_x$, is not empty but no other participants are known, CKS_x and CKC_x are empty. Suppliers would not be aware of any other participant, $SKMM_x$, SKC_x and SKS_x are empty. A middleman, x, while unaware of other middlemen and potential customers, would be aware of many suppliers, enough to fulfill most customer requirements, and so KC_x and KM_x would be empty while KS_x would not.

Section 4.2.1.4, below, discusses communication within the travel agent scenario. The interaction between participants being key to their different objectives being met and so the resolution of the supply chain.

4.2.1.4 Communication In The Travel Agent Scenario

Participants in SSCM supply chains interact through the specified communication scheme, Com. This scheme contains the set of utterance types each pair of participant types is able to use to communicate. Each individual participant has an upper limit imposed on the number of outbound utterances allowed per time-step in order to provide an incentive to use the scheme more carefully. In the travel agent scenario customers and middlemen must be able to enter a dialog as must middlemen and suppliers. Other interactions, between customers and suppliers for example, are not allowed. To specify this the sub-components of Com, $ComCtoMM$, $ComMMtoC$, $ComMMtoS$ and $ComStoMM$, are not empty while the remaining components, $ComCtoS$, $ComCtoC$, $ComMMtoMM$, $ComStoC$ $ComStoS$, are empty and so prevent the unwanted interactions.

The customer/middleman interaction in the travel agent scenario requires that, at a minimum, customers are able to indicate some need to the middleman and the middleman reply if it has been able to fulfill this. Extending this further, it would be beneficial if customers and middlemen entered a dialog in which the customer is able to indicate a need to the middleman and the middleman is able to offer potential alternative travel packages to the customer. These alternatives could then be accepted or rejected and, if accepted, the middleman indicate if it is has been able to obtain them. For this communication scheme $ComCtoMM$ requires at least three utterance types. One requests help from the middleman to fulfill some requirement and one each are needed to accept or reject possible alternatives. The reverse component, $ComMMtoS$, also requires at least three utterance types. One to offer potential alternatives to the customer, one to report that an accepted alternative has been

obtained for the customer and one to inform the customer that the middleman is unable to help. Additional utterance types could be be added to increase the richness of the dialog, perhaps allowing the customer to provide further information to the middlemen to help in the process of alternatives formation.

In discussing the travel agent scenario the middleman/supplier interaction has been characterised as a negotiation to find an acceptable price for both parties over some set of products. A minimum requirement is for middlemen to be able to request a set of products from a supplier at a specified price and the supplier to accept or reject this offer. Extending this further, it would be beneficial if the supplier was able to make a counter offer specifying a different price to which the middleman would be able to respond. The ability to indicate that some products are not available at all would also be helpful. As with the bandwidth trading scenario, see Section 4.2.2.4, an alternating offers negotiation protocol would suit this purpose. Under such a protocol middlemen and suppliers would be able to make counter offers for some set of products at a price and accept or reject the offers accordingly. In this case only middlemen would be able to make initial opening offers. Counter offers from a supplier excluding certain items specified by the middleman could be used to indicate that those items were unavailable. In terms of the communication scheme, Com, the components $ComMMtoS$ and $ComStoMM$ would contain utterances for specifying a counter offer, accepting and rejecting offers. In addition $ComMMtoS$ would contain an utterance type for making an opening offer. A alternating offers protocol for use with the SSCM is specified more precisely in Section 4.3.4.

4.2.1.5 The Travel Agent Scenario - Conclusions

The travel agent scenario comprises customers seeking to make a trip to a remote location, suppliers able to provide specific elements of a travel package to that location and middlemen able to negotiation on behalf of multiple customers with multiple suppliers to help realise acceptable, complete travel packages.

The SSCM can be used to capture this problem by first defining the products dealt with in the supply chain. These products relate to the different travel package elements, flights, accommodation and entertainment, required by a customer at various quality levels. Suppliers tend to supply only products of a certain type but for each product there may be multiple suppliers. The quantities available of each product may vary for each active time-step in the model. Customer requirements are in terms of a desired travel duration within a certain time frame with expectations about the quality of accommodation and flights and the types of entertainment on offer. Middlemen attempt to fulfill customer requirements by obtaining acceptable travel packages through negotiation with multiple suppliers. The communication scheme defined allows customers and middlemen to reach an agreement on acceptable alternative travel packages and middlemen and suppliers to negotiate and reach agreement for bundles of products.

A fully instantiated travel agent problem based on the example values discussed above is shown in Section 4.2.3.

With the travel agent scenario captured in this way it would then be possible to investigate the problem further, for instance examining how participant strategies affect the outcome of the supply chain. Section 4.3 discusses SSCM Variant

One (SSCM-V1) a version of the captured travel agent scenario used as the basis of experimental investigation of middleman supply chain strategies.

4.2.2 The Bandwidth Trading Scenario

A bandwidth trading scenario, as described in Section 3.4, can be modelled by the SSCM with middlemen acting as intermediaries between network operators (suppliers) and bandwidth consumers (customers). Within such a scenario a number of interconnected sites, or nodes, exist. Customers wish to obtain bandwidth between separate nodes which may or may not be directly connected. If nodes are not directly connected bandwidth must be obtained between a series of node pairs that connect the two remote sites. Suppliers are able to provide some amount of bandwidth between a subset of node pairs for discrete amounts of time. See Figure 4.2. Middlemen help customers to fulfill their requirements by building the necessary set of links needed by obtaining products from one or more suppliers. Time-steps in the model may represent different granularities of real time. The actual meaning of a time step would determine if the modelled problem was concerned with longer term bandwidth issues (months or quarters) or shorter term more immediate needs (hours or days). Further, how products are represented might also relate to this time granularity, see Section 4.2.2.1 below.

A fully define bandwidth trading problem of this type is shown in Section 4.2.3.

4.2.2.1 Products And Time In The Bandwidth Trading Scenario

With reference to Figure 4.2, products within the bandwidth trading scenario are the smallest amount of tradable bandwidth for the shortest available discrete time available between two nodes. If different levels of quality of service are available

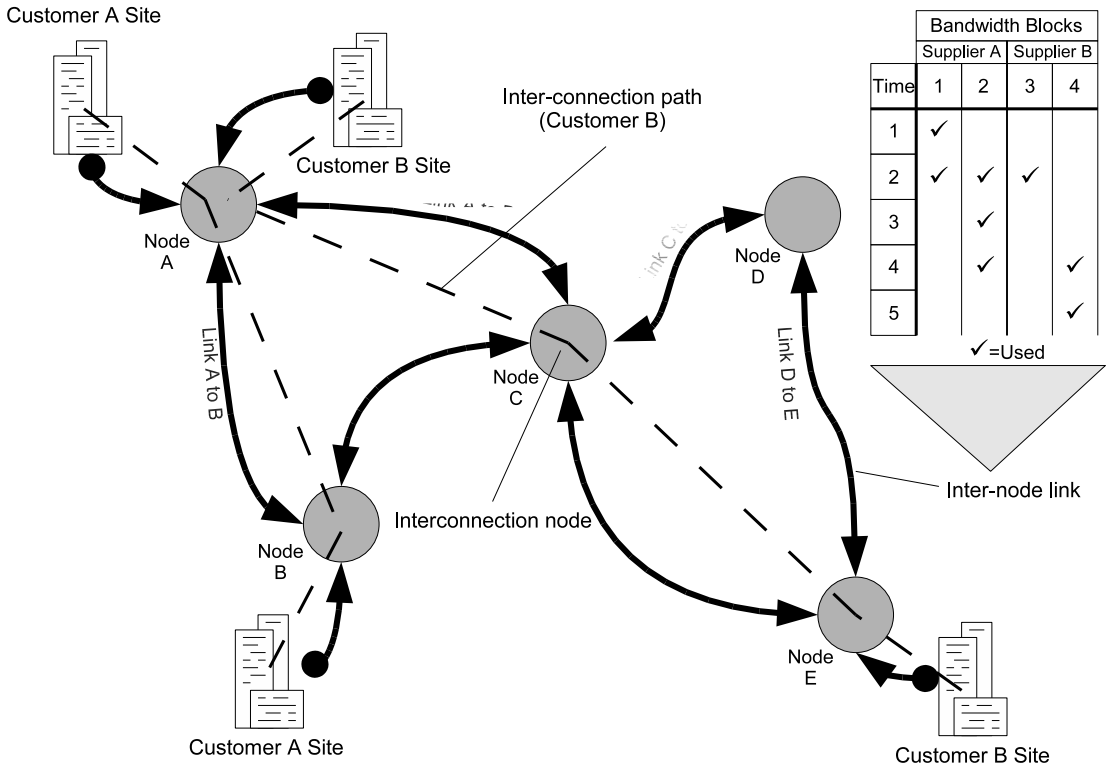

Figure 4.2: The Bandwidth Trading Scenario

these would be represented as separate products available between those nodes. For simplicity, the time-steps discussed within the model here correspond to the same minimum time granularity as the bandwidth for trade. It would, however, be possible to consider time-steps of greater or lesser length than the bandwidth granularity by adjusting the product naming convention and supplier availability used.

Assuming a situation similar to that shown in Figure 4.2 with five nodes not all of which are connected. First, we name the nodes A to E. A link between the nodes can be named for the nodes it connects, so $LinksAtoB$ for example. For the purposes of this example links are considered to be symmetric and bidirectional so the reverse link, $LinkBtoA$, does not need to be represented[1]. These link names provide our set of product names. A product being a specific minimum amount of bandwidth at a certain quality of service available over the link for a specific minimum time. The product quantities available over time from the suppliers determine the total amount of bandwidth available over time on the specific link. More than one supplier may be able to provide bandwidth between two nodes but at least one supplier must be able to provide some amount of bandwidth for the link to exist. Following this convention the SSCM products could be defined as shown in Table 4.17.

If we assume a short term bandwidth trading problem is being considered, bandwidth might be being traded in blocks one hour in duration. A model representing one week of active time will therefore contain 168 hours providing a T_{active} of 168, assuming some time in advance of this would be available to make initial trades, T_{total}

[1] A more complex rendering of the problem might consider differences in up-stream and down-stream bandwidth between the remote sites and so require both link directions to be represented.

Table 4.17: The Bandwidth Trading Scenario, Example Products

Product	*Name Mapping*
With a $pn = 6$	
P_0	$LinkAtoB$
P_1	$LinkAtoC$
P_2	$LinkBtoC$
P_3	$LinkCtoD$
P_4	$LinkCtoE$
P_5	$LinkDtoE$

might be 180 in this instance providing half a day for customers to deliver their requirements to middlemen and middlemen to begin to act on those requirements. This disparity between total and active time is intended to give customers and middlemen a chance to fulfill requirements that appear early in the active time and for which there may not otherwise be sufficient time to make arrangements.

4.2.2.2 Customers and Suppliers In The Bandwidth Trading Scenario

Continuing with the above example, suppliers are able to provide bandwidth over one or more links between nodes in some quantity for each of the 168 active time-steps in the model. Let us assume that in this instance there are three competing suppliers within the supply chain. In some cases all three suppliers will be able to provide bandwidth over a link, in others only one or two may be able to do so. With reference to Section 4.1.3, the three suppliers translate to $sn = 3$ and are represented as S_0 to S_2 in the model. Each of these suppliers are defined in terms of the products they supply, SP_x, the middlemen they are initially aware of, $SKMM_x$, and their outbound communication allocation per time-step, $SComOut_x$.

Each supplier, x, may be able to supply one or more products, spn_x, that is,

may be able to provide bandwidth over one or more links. The information about this set of supplied products is represented in SP_x. Each product, y, in the SP_x relates to a product in the set of products, P. Within SP_x this is specified along with the base value of that product to the supplier, $BaseValue_{x^y}$, and the quantity available for each active time-step, $Quantity_{x^y}$. Following the representation shown in Section 4.1.3 we assume that over a given link maintained by a given supplier, the supplier is able to deliver the same amount of bandwidth in each active time step. This is not unreasonable given that the link is already establish however, if necessary, it would not be difficult to adjust the model to provide a specific time-step specific representation of available bandwidth. The base value assigned to each product by a supplier is somewhat arbitrary, however these values are likely to be similar for suppliers of bandwidth over the same link. For a customer's budget to be viable it would need to be at least as great as the combination of the minimum link base values it requires and more than this if the middlemen are to make a profit or suppliers are not to sell at a loss.

Customers within the bandwidth trading scenario desire to connect sites at remote nodes using some amount of bandwidth at some start time and for some duration. As an example, a customer with sites associated with nodes A and E might desire to connect these sites together using one standard division of bandwidth for five hours and further, continuing the main example, wish that the bandwidth be available from active time-step 12 (total time, 24). Using the customer requirement composition mechanism described in Section 4.1.5, this requirement can be rendered as a series of name/value pairs. With the assertion that this customer is customer C_1 in the model, this is demonstrated in Table 4.18.

Table 4.18: Example Of A Customer's, (C_1), Requirement In The Bandwidth Trading Scenario

$reqn_1$	5
$ReqName_{1^1}$	BUDGET
$ReqValue_{1^1}$	100.0
$ReqName_{1^2}$	SITE_ONE
$ReqValue_{1^2}$	A
$ReqName_{1^3}$	SITE_TWO
$ReqValue_{1^3}$	E
$ReqName_{1^4}$	DURATION
$ReqValue_{1^4}$	5
$ReqName_{1^5}$	START_TIME
$ReqValue_{1^5}$	24

Within a fully defined bandwidth trading problem many customers would exist each with their own set of requirements to fulfill. The customer requirements may be fulfilled through the acquisition of products from the various suppliers. To accomplish this customers enter a dialog with middlemen who are able to build a set of suitable products. Since customers intend to obtain products via the middlemen and suppliers to sell them, the set of middlemen known to the customers and suppliers or, vice versa, the middlemen that are aware of them has an important affect on the resolution of the supply chain. This is considered in Section 4.2.2.3.

4.2.2.3 Middlemen In The Bandwidth Trading Scenario

The aim of middlemen within the bandwidth trading scenario is to fulfill requirements on behalf customers through trade with suppliers. Unlike the customers and suppliers, middlemen have no starting knowledge defined in the SSCM beyond that of the participants they are aware of and their communication limitations. In the

bandwidth trading scenario, however, middlemen need knowledge of how to connect remote sites together, the network topology, in order that they can effectively build the necessary set of links to fulfill customer requirements. If middlemen are aware of all the available products no additional knowledge is required since the topology can be inferred from the product information. If this knowledge was not available middlemen would require additional domain knowledge beyond that captured as part of the SSCM. This knowledge might be considered part of the middleman behaviour.

Middlemen face a complex situation in which they must respond effectively to challenges presented to them from both the customer and supplier side of the problem. Customer requirements might be unfulfillable due to the unavailability of products or their inability to afford them. Suppliers will likely seek to obtain the best price they can for their bandwidth. Further as time progresses middlemen may have to respond to potentially more profitable clients appearing or unexpected highes or lows in demand. Ultimately middlemen face a more difficult challenge than the customers and suppliers in the supply chain due to the uncertainty about what it is they will be doing and the conditions under which they will have to do it.

The challenge middlemen face is connected to the communication scheme being used and its restrictions. While other participants types have a clear idea of what they are attempting to achieve from the outset, for middlemen to achieve anything within the supply chain they must first either initiate or wait for contact with other participants. For example, a middleman with no known customers but several known suppliers would either have to wait for customers to enter a dialog about their requirements before doing anything or consider purchasing some products with a view to selling these on to customers at a later time. Alternatively a middleman with known

customers and suppliers could consider opening a dialog with customers preemptively to gain an idea of the products it will likely require more quickly. A middleman that is unaware of any suppliers would be in a difficult position how ever many customers it is aware of.

Within the bandwidth trading scenario being considered customers make use of middlemen to obtain the collection of required products from suppliers. To represent this within the SSCM directly, a customer's set of known middlemen, $CKMM_x$, must contain one or more middlemen if it is to be able to fulfill its requirements. A customer need not be aware of any other participants and, with reference to Section 4.2.2.4, may have no mechanism through which to communicate in any case. A supplier need not initially be aware of any other participant provided some set of middlemen are aware of it. If a supplier is aware of middlemen ($SKMM_x$ is not empty) however, it may wish to initiate a dialog. In order for middlemen to function they must be aware of some suppliers or suppliers must be aware for them and inclined to open a dialog. In general therefore the set of suppliers a middleman is aware of, $MMKS_x$, would not be empty. While middlemen may be aware of customers they need not be if some set of customers are aware of them, since this is specified as necessary as above the set of customers known by a middleman, $MMKC_x$, is likely to be empty or rather restricted.

The interaction of customer and suppliers with middlemen provides each participant a chance to fulfill its objectives. This interaction is, however, governed by the communication scheme, Com, defined as part of the model. This is discussed in Section 4.2.2.4.

4.2.2.4 Communication In The Bandwidth Trading Scenario

Within the bandwidth trading scenario being considered here customers attempt to fulfill their requirements via dialog with middlemen. Middlemen meanwhile attempt to fulfill multiple customer requirements through dialog with the suppliers. Under this scheme customers, middlemen and suppliers do not talk to other participants of their type nor do customer deal directly with the suppliers[2]. With this in mind communication within the bandwidth trading scenario needs to fulfill a number objectives. Firstly, customers must be able to communicate their requirements to the middlemen. Middlemen in return must be able to signal that they have done or are able to obtain products to fulfill these requirements. Secondly, middlemen must be able to negotiate with suppliers for bandwidth at given times over the specific links.

To accommodate these communication restrictions and objectives the model's *Com* component will need utterances defined for the sets $ComCtoMM$, $ComMMtoC$, $ComMMtoS$ and $ComStoMM$ while the sets $ComCtoS$, $ComStoC$, $ComCtoC$, $ComMMtoMM$ and $ComStoS$ will be empty.

The customer and middleman interaction is defined in sets $ComCtoMM$ and $ComMMtoC$. Set $ComCtoMM$ will need to contain at least one utterance type in order to deliver a requirement fulfill request to the middleman. In the reverse direction set $ComMMtoC$ will need to contain at least two utterance types, one to report a successful fulfilling of the request and one to report that it that it hasn't been possible. These sets could be further extended to include utterances for middlemen seeking requirements from customers and more involved negotiation and acknowledgement

[2]In a more complex rendition of the problem these links might well be considered but are dispensed with here for simplicity.

procedures.

The middleman and supplier interaction is defined in the utterance sets $ComMMtoS$ and $ComStoMM$. A simple solution to the definition of middleman/supplier interaction is to use a basic alternating offers protocol ([53]) for negotiations between both parties. Under this scheme either party can open negotiations by offering to buy or sell some set of products for some value. The recipient of such an offer may accept the offer, paying/receiving the specified amount to obtain/supply the products and ending the negotiation, reject the offer, simply ending the negotiation, or make a counter offer. A counter offer may, or may not be, for the same products but almost certainly at a different value. Using this mechanism middlemen and suppliers can reach an agreement over the transfer of goods for some price. In terms of $ComMMtoS$ and $ComStoMM$ utterance types must defined for making an initial offer, making counter offers and accepting and rejecting offers. This symmetric mechanism would allow middlemen to attempt to purchase bandwidth from suppliers or suppliers to preemptively attempt to sell bandwidth to middlemen. A specific version of this is detailed in Section 4.3.4, in relation to the SSCM Variant One (SSCM-V1) environment

Each participant in the supply chain has a limit imposed on the maximum number of outbound communication utterances allowed per time-step. This limit is intended to provide a way to model communication restrictions that results either from the channel being used or some other source. In providing such a limit the objective is to provide a mechanism that would encourage more efficient use of communication channels. A result of this is that participants face a disincentive from sending lots of

messages without good reason and must instead consider the dialogs they are involved in more carefully.

4.2.2.5 The Bandwidth Trading Scenario - Conclusions

Bandwidth trading is seen as a way of improving the utilisation and profitability of network operators and service providers. This scenario can be viewed as a supply chain with bandwidth consumers acting as customers and network operators acting as suppliers. Middlemen can help bring these two parties together assisting customers in obtaining the bundles of goods necessary to fulfill their requirements.

Using the SSCM, a representation of the bandwidth trading scenario can be constructed such that the available products, starting knowledge of the participants and communication scheme are captured. A product in this model is a specific amount of bandwidth at a given quality for a certain amount of time between two remote nodes. A customer's requirement is in the form of required bandwidth between two nodes for some start time and duration. This is represented by a series of name/value pairs. Suppliers provide products, bandwidth between a set of nodes. Each product has an associated available quantity and base value. Middlemen have no starting knowledge beyond the participants they are aware of. Domain specific knowledge of network topology is known, or inferred, as part of their behaviour. The specified communication scheme defines the way in which the participants in the supply chain can interact.

A fully instantiated bandwidth trading problem based on the example values discussed above is shown in Section 4.2.3.

Having captured the bandwidth trading scenario in this fashion it would then be possible to consider carrying out an investigation of this problem further. Cases of the SSCM-defined bandwidth trading scenario could be examined as part of a simulation and the effect of different participant strategies on the outcome of the supply chain investigated. The simulation could also form the basis of efforts to develop effective strategies for each of the participant types, their operation and interaction being tested in the simulation with starting conditions captured through the SSCM.

4.2.3 Instantiated Travel Agent and Bandwidth Trading Scenario Problems, A Comparison

Sections 4.2.1 and 4.2.2 introduce the travel agent and bandwidth trading scenarios respectively, and discuss how these can be modelled using the SSCM. This section explicitly demonstrates how simple problems of both of these types would be instantiated in the SSCM. In each case the communication scheme, Com, has not been specified. A more detailed description of a communication component is provided in Section 4.3.4 in relation to SSCM-V1.

Tables 4.19 and 4.20 show how a travel agent scenario problem would be instantiated given the example parameters discussed in Section 4.2.1. This includes 12 products of various types, 6 suppliers of these products, 1 customer, a total time of 37 and an active time (in which requirements can exist) of 30. In addition two middlemen are shown and each participant has a communication budget of 10 messages per time step. In this instance suppliers know of no other participants, the middlemen know of all suppliers and the customer knows about both middlemen. Also, suppliers supplying the same product types have different quantities available and different base valuations of those products.

Table 4.19: Travel Agent scenario, problem instantiation, Products and Suppliers

Component	*Travel Agent Scenario*
	{
Products P	{
P_1	$FlightOutEarlyLowQuality$,
P_2	$FlightOutEarlyHighQuality$,
P_3	$FlightOutLateLowQuality$,
P_4	$FlightOutLateHighQuality$,
P_5	$FlightBackEarlyLowQuality$,
P_6	$FlightBackEarlyHighQuality$,
P_7	$FlightBackLateLowQuality$,
P_8	$FlightBackLateHighQuality$,
P_9	$AccommodationLowQuality$,
P_{10}	$AccommodationHighQuality$,
P_{11}	$EntASightSeeingTour$,
P_{12}	$EntBFairground$
	},
Suppliers S	{
S_1	{{{$FlightOutEarlyLowQuality$, 100, 100}, {$FlightOutEarlyHighQuality$, 500, 20}, {$FlightOutLateLowQuality$, 100, 100}, {$FlightOutLateHighQuality$, 500, 20} {$FlightBackEarlyLowQuality$, 100, 100}, {$FlightBackEarlyHighQuality$, 500, 20}, {$FlightBackLateLowQuality$, 100, 100}, {$FlightBackLateHighQuality$, 500, 20} }, {}, {}, {}, 10}
S_2	,{{{$FlightOutEarlyLowQuality$, 80, 50}, {$FlightOutEarlyHighQuality$, 600, 40}, {$FlightOutLateLowQuality$, 80, 50}, {$FlightOutLateHighQuality$, 600, 40} {$FlightBackEarlyLowQuality$, 80, 50}, {$FlightBackEarlyHighQuality$, 600, 40}, {$FlightBackLateLowQuality$, 80, 50}, {$FlightBackLateHighQuality$, 600, 40} }, {}, {}, {}, 10}
S_3	,{{{$AccommodationLowQuality$, 20, 50}, {$AccommodationHighQuality$, 75, 5} }, {}, {}, {}, 10}
S_4	,{{{$AccommodationLowQuality$, 30, 25}, {$AccommodationHighQuality$, 150, 30} }, {}, {}, {}, 10}
S_5	,{{{$EntASightSeeingTour$, 10, 40} }, {}, {}, {}, 10}
S_6	,{{{$EntBFairground$, 20, 200} }, {}, {}, {}, 10}
	},

Table 4.20: Travel Agent scenario, problem instantiation, Middlemen, Customers and Time

Component	*Travel Agent Scenario*
Middlemen M	{
M_1	{ $\{S_1, S_2, S_3, S_4, S_5, S_6\}$, {}, {}, 10},
M_2	{ $\{S_1, S_2, S_3, S_4, S_5, S_6\}$, {}, {}, 10}
	},
Customers C	{
C_1	{ { $BUDGET, 1000.0$, $EARLIEST\_START, 14$, $LATEST\_END, 28$, $MIN\_DURATION, 5$, $MAX\_DURATION, 5$, $FLIGHT\_QUALITY, HIGH$, $ACCOMMODATION\_QUALITY, HIGH$, $DESIRED\_ENTERTAINMENT, \{\}$, $MIN\_NOTICE, 5$ }, $\{M_1, M_2\}$, {}, {}, 10}
	},
Time T_{total}, T_{active}	$37, 30$,
Communication Com	{...}
	}

Table 4.21 shows how a bandwidth trading scenario problem would be instantiated. With reference to the example values discussed in Section 4.2.2, 6 products of a similar type are available, 3 suppliers provide a different subset of the available products but in the same quantities and at the same base evaluation of their worth. There is a total supply chain time of 180 with an active time (in which requirements exist) of 168. In addition, 2 middlemen are shown, both of whom are aware of all the suppliers. One customer is shown and knows about both middlemen. The communication budget for each participant is again 10 with suppliers knowing of no other participants, middlemen only know of suppliers and the customer only knowing of middlemen.

Comparing the two instantiated problems it can be seen that the overall representation is similar despite the different domains being considered. The most significant visible variation is in the customer requirement representations where a different set of name/value pairs is used in each case.

4.2.4 Conclusions

The SSCM captures supply chains situations in terms of the length of time to be considered, products traded, the starting knowledge of different participants (customers, suppliers and middlemen) and how the participants are able to interact. Using this representation different types of problems may be captured. In considering the appropriate time granularity to use it may often make sense to use time-steps that coincide readily with those needed to express requirements or the availability of products. Different granularities might be considered at the expense of more complex product representations. For both the travel agent and bandwidth trading scenarios discussed

Table 4.21: Bandwidth Trading scenario, problem instantiation

Component	*Bandwidth Trading Scenario*
	{
Products P	{
P_1, P_2 and P_3	$LinkAtoB$, $LinkAtoC$, $LinkBtoC$,
P_4, P_5 and P_6	$LinkCtoD$, $LinkCtoE$, $LinkDtoE$
	},
Suppliers S	{
S_1	{{ $\{LinkAtoB, 20, 10\}$, $\{LinkAtoC, 20, 10\}$, $\{LinkBtoC, 20, 10\}$, $\{LinkCtoD, 20, 10\}$, $\{LinkCtoE, 20, 10\}$ }, {}, {}, {}, 10 },
S_2	{{ $\{LinkAtoC, 20, 10\}$, $\{LinkBtoC, 20, 10\}$, $\{LinkCtoD, 20, 10\}$, $\{LinkCtoE, 20, 10\}$, $\{LinkDtoE, 20, 10\}$ }, {}, {}, {}, 10 },
S_3	{{ $\{LinkAtoB, 20, 10\}$, $\{LinkAtoC, 20, 10\}$, $\{LinkBtoC, 20, 10\}$, $\{LinkCtoE, 20, 10\}$, $\{LinkDtoE, 20, 10\}$ }, {}, {}, {}, 10 }
	},
Middlemen M	{
M_1	{ $\{S_1, S_2, S_3\}$, {}, {}, 10},
M_2	{ $\{S_1, S_2, S_3\}$, {}, {}, 10}
	},
Customers C	{
C_1	{ { $BUDGET, 100.0$, $SITE\_ONE, A$, $SITE\_TWO, E$, $DURATION, 5$, $START\_TIME, 24$ }, $\{M_1, M_2\}$, {}, {}, 10}
	},
Time T_{total}, T_{active}	$180, 168$,
Communication Com	{...}
	}

a natural granularity was apparent, days for travel packages, hours for the short term bandwidth trading problem. Products within the supply chain must represent the total set of different type components that may be needed by customers. In the travel agent scenario different types of product were required to build up a bundle acceptable to customers. For the bandwidth trading scenario the product types were essentially identical but were geographically disparate. Representation of customer requirements is important and problem type specific. For the travel agent scenario the time frame for the trip, duration and expected quality of elements had to be represented. For bandwidth trading a representation of the needed start time, duration and level of bandwidth was required. For the suppliers starting knowledge, consideration must be given to the representation of product availability and base value. Within the bandwidth trading scenario these were simple and constant for each timestep. For the travel agent scenario flights might not be available on some days so requiring a slightly more complex representation. Middlemen's starting knowledge is limited to the participants they are aware of. However, what participants are aware of what other participants may strongly influence how the supply chain is resolved. In the scenarios presented above customers obtain products from multiple suppliers via middlemen without recourse to direct contact with those suppliers. This need not be the case. Further, participant types might consider dialog amongst themselves for information sharing or trading of unwanted products. The communication mechanism specified in the model restricts how participants are able to interact. Even if participants are aware of one another, without a defined communication mechanism they won't be able to interact directly. The definition of the communication mechanism is thus a potent way of directing how the supply chain can be resolved without directly

specifying participant behaviours. Richer dialog options may act to improve what happens within the supply chain but could also simply increase complexity with no perceivable benefit. Alongside the defined communication scheme each participant has a restriction placed upon their outbound communication use. This restriction on communication (that need not be used) is intended to provide an incentive to participants to use their available communication options more carefully.

Having considered two supply chain problems Section 4.3, below, introduces SSCM Variant One (SSCM-V1), a version of the travel agent problem used as the basis of experimental work in this thesis.

4.3 Investigating Strategies For The SSCM, SSCM-V1

This research aims to investigate the evolution of middlemen strategies within simple, but non-trivial, supply chain situations. This section introduces SSCM Variant One (SSCM-V1) a variation on the travel agent scenario, discussed in Section 4.2.1, that is used as the basis of experimentation.

SSCM-V1 provides a simple but non-trivial travel agent scenario within which the evolution of middlemen strategies can be investigated. The aim of SSCM-V1 is to provide a standard problem type representation that may then be used to describe specific problem instances to be tackled. Within SSCM-V1 customers desire to travel to a remote location for some duration within a certain time-frame. While away accommodation will be required and some form of entertainment might also be sort. Suppliers within SSCM-V1 are able to provide products to fulfill these requirements but this is only possible via middlemen. These should act to obtain products on the

behalf of multiple customers. The communication scheme used within SSCM-V1 is an alternating offers negotiation protocol.

Section 4.3.1 discusses how middlemen are at the heart of SSCM-V1 and why the strategies employed by these participants is the focus of study. Section 4.3.2 describes the products traded and their suppliers within SSCM-V1 while Section 4.3.3 details how customer requirements are represented and the time-step granularity used. The communication scheme used by the SSCM-V1 participants is specified in Section 4.3.4.

While SSCM-V1 provides a representation to describe specific problems it also encompasses a wide variety of possibilities. To further aid the study of middlemen strategies a series of SSCM-V1 restricting scenarios are discussed in Section 4.4. These scenarios place constraints on how problems within SSCM-V1 should be instantiated and also place expectations on participant behaviour.

4.3.1 Middlemen, At The Heart Of The Problem

Middlemen are at the heart of the SSCM and SSCM-V1. Twin problems of satisfiability and optimisation overlap at the middleman and make the strategies used by these participants to be of the greatest interest. While the customers and suppliers can make do with relatively simple strategies to succeed in their objectives of fulfilling their requirements and selling products for a good price, the middlemen are in a far more precarious and uncertain situation. Middlemen face choices about which customer requirements they should attempt to fulfill from perspectives of profitability, obtainability of products and the amount of time and communication available. They must also consider which suppliers to approach and how to go about negotiating with them for products. The decisions middlemen face are complicated by a lack of

starting knw=owledge knowledge about all of these things. Middlemen start with no intrinsic knowledge of product prices, obtainability of products or how these may vary with time. Middlemen must instead infer this information from interaction with participants and adjust their behaviour accordingly. The satisfiability problem that exists on the customer side of the middlemen involves attempting to find product bundles that the customer may find acceptable. On the supplier side an optimisation problem exists in which negotiations must be entered in to in order to obtain needed products at a good price. These two problems and their related choices interact. The amount customers are willing to pay affects the flexibility with which middlemen may negotiate with suppliers, the attempt to fulfill the associated requirements specifying what will be negotiated over. In return the interaction with suppliers determines which customer requirements are feasible in terms of both availability of products and the potential profitability of different bundles.

The level of uncertainty and the interaction of different elements in the problem make the strategy used by middleman of interest. For this reason these strategies are the primary focus of the work undertaken here. A middleman strategy devised for tackling SSCM-V1 problems must be capable of behaving effectively in the face of a dynamic and uncertain environment that is more complex than those traditionally faced in Game Theoretic problems. A mechanism that allows these strategies to be formed and investigated is therefore of interest. In the following chapters a framework is developed for representing middlemen strategies and evolving them within a market simulation system. Within this environment the customer and supplier participants make use of fixed strategies. This step removes additional uncertainty from the situation and is taken to aid the effective study the middleman behaviour and

how this responds to the supply chain environment.

SSCM-V1, and its related restricting scenarios, aid the process of investigating the middleman behaviour by describing the specific supply chain problems to be tackled. The description of this representation is continued below.

4.3.2 What Products Are Traded Under SSCM-V1

Within SSCM-V1 customers desire flights to and from a remote location, expect accommodation while away and may also require entertainment. Section 4.2.1.1, above, introduces one representation for products in a similar situation. Under SSCM-V1 a less complex representation is, however, made use of. Products, P, in SSCM-V1 comprise flights at standard times and quality to and from the remote location, a single quality of accommodation and some number of possible entertainments. This reduced complexity is intended to make later analysis of the supply chain easier. Issues of time-step granularity and its effects on product representation are briefly discussed in Section 4.3.3, below. Table 4.22 describes this set of products.

Table 4.22: SSCM-V1, Products

Product	*Name Mapping*
P_1	$Outbound\_Flight$
P_2	$Return\_Flight$
P_3	$Accommodation$
P_4	Ent_1
P_pn	Ent_pe
pe, The number of entertainments ($pn - 3$)	

Suppliers within SSCM-V1 are able to provide one or more of the above products. For a supplier x and supplied product y, the supplied product SP_{x^y} exists in the set

of products specified above, P. For each of these the supplier's base value for the product $BaseValue_{xy}$ and available quantity $Quantity_{xy}$ are specified as fixed values for the duration of the active period being considered.

4.3.3 How Customer Requirements Are Formed Under SSCM-V1

Customers in the SSCM-V1 travel agent scenario wish to travel to a remote location for some duration within a certain time frame and with some expectation of entertainment while there. Unlike the scenario discussed in Section 4.2.1 the quality and time of flights and type of accommodation is not considered. This reduced complexity is reflected in the SSCM-V1 product set described above. A customer's requirement is, therefore, primarily concerned with the timing of the trip. As with the previously mentioned scenario, the trips represented by SSCM-V1 are reasonably considered to be in terms of days, with customers flying out one day and returning one or more days later. For simplicity the time-step granularity of the model is thus considered to be in days. This allows customer requirements to use time-steps directly in representing their requirements and also removes any need to extend the product representation described above in Section 4.3.2.

An SSCM-V1 customer requirement is based on a trip being required for some approximate duration within a certain time-frame. Furthermore, the customer has some expectation of entertainment while away, the amount of notice given before the trip begins and a limit on the amount they are willing to pay. The requirement representation must therefore consider the following. Firstly, a customers budget represents a fixed upper limit on the amount they are willing to pay for the trip.

Second, customers require notification a certain minimum time ahead of the trip in order to prepare. Third, the duration of the trip must be specified between some limits. Trips of a duration between these limits would be acceptable to the customer. Fourth, the time-frame within which the customer wishes to travel is represented by an earliest possible start time and latest possible return time, specified in time-steps. Finally, the customer has limits on their expectation of entertainment while away, the type is unimportant in this case but quantity is relevant. With this is mind Table 4.23 demonstrates how a possible SSCM-V1 customer requirement would be represented.

Table 4.23: SSCM-V1, Customer Requirement Representation Example

SSCM Element	*Name/Example Value*
$reqn_x = 8$, Eight Requirement Representation Name/Value Pairs	
$ReqName_{x^1}$	BUDGET
$ReqValue_{x^1}$	400.0
$ReqName_{x^2}$	MIN_NOTICE
$ReqValue_{x^2}$	3
$ReqName_{x^3}$	MIN_DURATION
$ReqValue_{x^3}$	5
$ReqName_{x^4}$	MAX_DURATION
$ReqValue_{x^4}$	5
$ReqName_{x^5}$	EARLIEST_START
$ReqValue_{x^5}$	7
$ReqName_{x^6}$	LATEST_END
$ReqValue_{x^6}$	17
$ReqName_{x^7}$	MIN_ENTS
$ReqValue_{x^7}$	1
$ReqName_{x^8}$	MAX_ENTS
$ReqValue_{x^8}$	2

4.3.4 How Participants Communicate Under SSCM-V1

Communication in the SSCM-V1 is achieved through an alternating offers negotiation protocol. Unlike the scheme discussed in Section 4.2.1.4 this protocol is supplied symmetrically to both the customer/middleman and middleman/supplier interaction. The rationale behind this decision is that both sets of required interactions can be achieved using this mechanism and that using one mechanism for both of these sets may reduce the complexity of implementing participant strategies. Interactions between customers and suppliers are prevented as are interactions between participants of the same type. The components of the communication scheme, Com, that represent the sets of utterances between these types are therefore empty.

Customers within SSCM-V1 must communicate requirements to middlemen in some way and middlemen must be able to offer possible alternative product sets that the customers, in turn, are able to respond to. The alternating offers protocol enables this interaction by allowing customers to describe specific travel packages they desire. Middlemen may then use the same mechanism to propose possible alternatives or to signal that an acceptable package has been obtained. For the middlemen the same mechanism can also be used for the negotiation of products from suppliers. Middlemen are able to specify sets of products they would like to obtain and can enter a dialog with suppliers over the price. The availability of products can be signalled to middlemen by how the set being negotiated over is modified by the supplier. The elements of the communication scheme, Com, that define the interaction between customers and middlemen and middlemen and suppliers thus each contain four utterance types. The ability to open negotiations, make counter-offers and accept or reject

the last offer made. These four utterance types, that are common to $ComCtoMM$, $ComMMtoC$, $ComMMtoS$ and $ComStoMM$, are described in Table 4.24 with further details elaborated upon in Table 4.25.

Table 4.24: SSCM-V1, Communication Utterances

Type	*Contained Named Elements*
NEW	NegID, Buying/Selling, Amount, Product Set, Time-Out
COUNTER_OFFER	NegID, Amount, Product Set, Time-Out
FINISH_ACCEPT	NegID
FINISH_REJECT	NegID

Table 4.25: SSCM-V1, Utterance Element Details

Components	*Explanation*
NegID	A unique identifier for the negotiation thread
Buying/Selling	If the initial request wishes to buy or sell the suggested product set
Amount	The amount the buyer/seller wishes to pay/receive
Product Set	A set of tuples. Each one consisting of a product name, time and (non negative) quantity
Time-Out	The time-step before which a response to this utterance must be received. If a response is not received the sender will assume the receiver implicitly rejects the proposal so ending the dialog

4.3.4.1 Customer Generation Of Specific Requirements For Communication To Middlemen

The communication scheme above is intended to be used by customers to rely requirements to middlemen. To do this a customer must generate and send a specific product set that relates to their requirements. This product set can then act as the starting point for further dialog in finding acceptable alternatives. A specific product set may be generated from a customer's requirements in the following manner.

The first step in generating a specific product set is to determine a set of core parameters defining that set. These are the duration of the trip and its start time. A duration may be selected, randomly or otherwise, from within the range of durations specified by the limits MIN_DURATION and MAX_DURATION. A start time for the trip can likewise be selected from within the range EARLIEST_START to the LATEST_END minus the selected duration. The selected start time may need further adjustment in light of the MIN_NOTICE period the customer requires to ensure time would be available to a middleman to have a chance of obtain the desired trip.

From these basic parameters it is then possible to determine most of the trip information. The start time becomes the time required for the OUTBOUND_FLIGHT product type, the RETURN_FLIGHT is based on the start time and duration. ACCOMMODATION types are required for every time unit from (and including) the OUTBOUND_FLIGHT time and up to (and excluding) the RETURN_FLIGHT time. For entertainment types a number may again be selected from between the limits MIN_ENT and MAX_ENT. This number of entertainments may then be distributed throughout the trip, starting the day after the OUTBOUND_FLIGHT and including the day of the RETURN_FLIGHT. Placement of entertainment types within the trip can be accomplished in biased random manner with a view to a relatively even distribution in time and of types.

This fully trip defining product set having been produced, customers may communicate this to the middlemen as the starting point in a dialog about acceptable travel packages.

4.4 SSCM-V1 Scenarios - Working Towards Strategies, Constraining SSCM-V1

The SSCM-V1 Scenarios restrict how SSCM-V1 is used to specify individual problem instances in order to facilitate the development of a middleman strategy representation. SSCM-V1 provides a way to describe travel agent-based supply chains. The problems specified may be comparatively broad in scope with no restrictions imposed on customer requirements, the quantities of products available, how the communication scheme is used or the knowledge each participant has of the others. Further, the SSCM is not intended to specify participant behaviour, something that must be considered in devising a viable middleman strategy. Three SSCM-V1 Scenarios are defined that restrict how SSCM-V1 is used to define problem instances and how participants behave while operating in the supply chain. Later scenarios are less restrictive than earlier ones, the aim being to aid the development of middlemen strategies that are able to deal with broader sets of circumstances. Sections 4.4.1, 4.4.2 and 4.4.3 introduce these three scenarios in turn.

The design of a middleman strategy framework capable of tackling SSCM-V1 defined problems is discussed in Chapter 5 with reference to the scenarios being considered here. This strategy framework is ultimately used for the evolution of middlemen strategies within a market simulation system, Chapter 6. Problems within that system being defined in terms of the SSCM-V1 and mindful of the scenario restrictions.

4.4.1 Scenario One - Basic Restrictions

This scenario provides a basic situation for consideration.

Under Scenario One (SSCM-V1S1) SSCM restrictions are as follows. Customers know of only one middleman and have a fixed requirement. Suppliers know of no middlemen and supply only one product. Middlemen know of all suppliers but no customers.

SSCM-V1S1 specifies the behaviour of participants as follows. Customers initiate negotiations with middlemen and will wait for a response as long as possible. They will not negotiate for alternative requirement sets. Suppliers will wait for middlemen to initiate negotiations, they will negotiate on a unit price and try to supply a sub set of the products requested. A supplier will only reject a middleman negotiation if the products are unavailable. Middlemen will not try to renegotiate with customers for different requirement sets.

In more specific terms:

$\forall C_x \in C$, $ckmmn_x = 1$

$sn = pn$

$\forall S_x \in S$, $skmmn_x = 0$, $spn_x = 1$

$\forall M_x \in M$, $kcn_x = 0$, $KS_x \supset S$

For each customers requirement set:

MIN_DURATION = MAX_DURATION

LATEST_END = EARLIEST_START + MIN_DURATION - 1

This scenario presents the middlemen with the task of attempting to determine if they can satisfy customer requirements and do so profitably. The question is simplified

since customers have a definite requirement and no renegotiation is possible. Further, there is no worry about dealing with multiple suppliers, if a product appears to be unavailable it will remain so and there is no recourse to other possible suppliers. In essence the middleman's problem is one of attempting to estimate the cost of products and using this to accurately determine if incoming customer requirements are likely to be profitable (if probably fulfillable and there is enough time).

Uncertainty exists for the middleman in terms or product price and availability.

4.4.2 Scenario Two - Customer Satisfaction

This scenario is as SSCM-V1S1 with the exception that customers and middlemen are now able to negotiate over alternative requirement sets.

In practice this means the restrictions on MIN_DURATION/MAX_DURATION and LATEST_END/EARLIEST_START for each customer's requirement are relaxed.

This scenario adds a considerable amount of complexity to the middleman's task. With middlemen able to renegotiate with customers there is a chance that any initially unviable product set could be salvaged. This increases the opportunity of making profits at the expense of communication. The problem becomes one of both trying to find a product set the customer and middleman find acceptable and also trying to find the best possible price to fulfill that product set.

4.4.3 Scenario Three - Supplier Selection

This scenario builds upon SSCM-V1S2 allowing multiple suppliers of products and suppliers that may supply more than one product.

This adjusts the restrictions such that:

$sn \geq 1$

$\forall S_x \in S,\ pn \geq spn_x \geq 1$

Under this scenario further uncertainty is introduced. Multiple suppliers may exist for a single product so middlemen will need to make a choice between them. This may influence earlier decisions about customer requirement viability as well as providing fall back options if current negotiations appear to be going badly.

4.5 Characterising An SSCM Instance

While the SSCM provides a way to effectively capture an individual supply chain situation it is not practical to list the full SSCM information when discussing a particular SSCM-defined problem or a set of related problems. This section describes some basic characteristics of an SSCM problem that are used in this thesis as a means to describe and discuss sets of SSCM problems.

The nine characteristics used for the description are listed below:

- Number Of Middlemen (NOM)
- Customers Per Middleman (CPM)
- Lowest Customer Balance (LCB)
- Mean Customer Balance (MCB)
- Highest Customer Balance (HCB)
- Number Of Products (NOP)

- Number Of Suppliers Per Product (NOSPP)
- Product Availability (PA)
- Communication Budgets (CB)

The Number Of Middlemen (NOM) is identical with mn in the SSCM, it is the number of middlemen present in the chain and so relates to the amount of competition there will be for resources.

The Customers Per Middleman (CPM) is the mean number of customers that know about each middleman. For SSCM-V1, this number is the mean number of distinct customers. CPM should be approximately cn/mn in the model.

The Lowest, Highest and Mean Customer Balance (L, H and MCB) relates to the amount of wealth available in the system from customers. The values are essentially problem-specific. For SSCM-V1 they relate as follows. LCB is the cost of the longest possible trip (calculated from the most expensive supplier costs) being financed by the minimum possible budget. The difference between these to is the LCB and may well be negative. Likewise the HCB is the cost of the longest possible trip financed by the highest possible budget. The longest trip is used since the customer budget relates to the trip length. The MCB is based on the mean trip duration, entertainments and costs. A positive MCB indicates there are likely to be more customers in the system with more wealth to share than customers with insufficient budget to finance a trip. A negative MCB indicates most customers have an insufficient budget to finance a trip.

The Number Of Products (NOP) relates directly to pn in the model. For SSCM-V1 this number is never less than 4.

The Number Of Suppliers Per Product (NOSPP), this is the mean number of suppliers of each product type. For SSCM-V1-S1/2 this will be 1. For SSCM-V1-S3 this may not be equal to one.

Product Availability (PA) is a rough measure of how difficult products are to obtain in the market being represented. It is based on the mean availability of all products in the market. The availability of an individual product is based on the total number in the market for each active time-step divided by the total number of customers averaged over every active time-step. So, for instance, a PA of 1.0 (or 100%) would mean there are sufficient products to supply one to every customer for every active time-step. This is a very gross measure and further elaboration for specific situations may be needed.

Communication Budgets (CB) are the number of outbound communications each type of participant is able to make in each SSCM time-step. This is represented by three numbers in sequence (e.g. $5 - 25 - 25$) to indicate the amount of outbound communications available to customers, middlemen and suppliers respectively. These values correspond to the means of $CComOut_x$, $MComOut_x$ and $SComOut_x$ for all customers, middlemen and suppliers in the instantiated model.

While these measures provide a good indication of the nature of the SSCM instantiation being considered, they fail to take in to account how the behaviour of participants may change their meaning. See Section 6.4.2.1 for a description of additional measures used in relation to the experimental systems employed in this work.

4.6 Conclusions

The SSCM provides a mechanism for describing supply chain situations. These economic problems are interesting since the interaction of participants in the supply chain is non-trivial and difficult to analyse using traditional Game Theory approaches. The SSCM's ability to capture interesting problems is Claim One in Section 1.4 of the Introduction.

The SSCM defines a supply chain problem in terms of the set of products that may be traded, the time in which these trades must take place, participants starting knowledge (including requirements, supplied products and awareness of one another), the communication scheme used by participants to interact and the restrictions placed on each participant for this schemes use. The SSCM may be used to capture different business scenarios by careful consideration of each of these elements and how they relate to the problem to be modelled.

SSCM-V1, a specific SSCM-defined travel agent scenario partially inspired by the TAC game (Section 3.3), has been defined for use as the basis of experimentation.

The strategies of middlemen within an SSCM-defined problem are of the greatest interest to study since these participants face a more complex and uncertain situation than either the customers or suppliers. This work, therefore, focuses on these strategies.

To aid the development of effective middlemen strategies that can operate within the SSCM-V1 environment, SSCM-V1 Scenarios that restrict participant behaviour and how individual problems are instantiated have been defined. The three SSCM-V1 Scenarios provide decreasing restrictions on both participant behaviour and model

instantiation with that aim of guiding strategy development from less complex to more complex, and interesting, environments.

Chapter 5

Defining Middlemen Strategies - The SSCM Strategy Framework (SSF)

This chapter introduces the SSCM Strategy Framework (SSF). The SSF provides a middleman strategy representation that encompasses a broad range of possible behaviour and is conducive to the evolution of effective middleman strategies.

Following a brief introduction, this chapter first reiterates and elaborates upon the problems faced by middlemen in an SSCM environment, Section 5.1. The suitability of different evolutionary approaches to generating middlemen strategies are then discussed along with the reasons for using a framework based approach, Section 5.2. Following this the basic concepts of the SSF and the core decision points within it are introduced, Section 5.3. How these decision points are elaborated upon and used in response to the SSCM-V1 environment is discussed in Section 5.4. The Chapter concludes with an overview of the SSF objectives, contributions made and relationship to other work, Section 5.5. A more detailed algorithmic description of the SSF is provided in Appendix A.

The SSCM Strategy Framework provides the structure through which Middleman strategies are specified. The SSF achieves this by providing a flexible operational structure of a middleman strategy the behaviour of which is varied via a series of parameter settings. The SSF consists of this basic structure, related data structures and the control points through which middleman behaviour is varied.

The aim of the SSF is to facilitate the evolution of middleman strategies through adjustment of parameter settings rather than requiring the formulation of the entire strategy structure. In terms of the strategies themselves the SSF aims to provide a wide range of behaviour that is able to operate reasonably sensibly under varied conditions and make use of less communication than negotiating for each customer requirement individually.

Flexibility of the SSF is important since it must support a wide range of possible middleman operations such that there is sufficient scope to find an effective strategy given the market conditions. The SSF provides further flexibility in terms of implementation. While the core control structures are defined by the SSF, specific implementation and problem-specific decision mechanisms are set aside for later specification. The specification of these elements and their controlling parameters can therefore be adapted to different types of SSCM problem and different implementation philosophies. For the purposes of this research a single SSF variation, SSF Implementation One (SSF-I1), is used. This implementation provides problem-specific mechanism relating to SSCM-V1 (see Section 4.3) and implementation specific mechanism focused towards the evolutionary learning of strategies.

5.1 The Middleman Problem

A middleman strategy must deal with customer negotiations on one side (the customer requirements) and attempt to fulfill these through negotiation with suppliers on the other side. In between the middleman must make decisions about which customer requirements can be successfully and profitably fulfilled and how to go about the negotiation process with the suppliers. The ultimate objective being to make a profit (see Figure 5.1). Tackling this problem is made considerably more difficult by a lack of information. When the market begins the middleman is unaware what customers it will be contacted by or when, nor how much products may cost or how available they are.

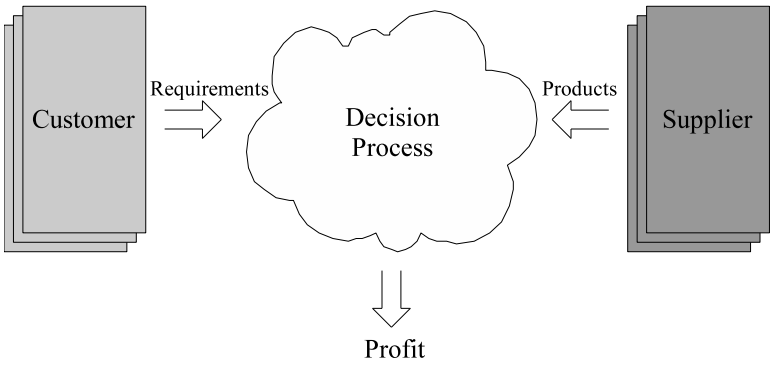

Figure 5.1: The Middleman Problem

The customer-middleman problem is essentially one of *satisfiability*, can the middleman fulfill the customers initial requirement or find a viable alternative that the customer finds satisfactory. With respect to SSCM-V1, satisfiability from the customer perspective is in terms of trip duration, earliest start/latest end, number of entertainments and budget. From the middleman perspective three objectives must be met, there must be enough time to negotiate for the products, the estimated cost

of the products must be less than the amount the customer is willing to pay and it must be possible to obtain those products. Answers to all three middleman questions must be based on estimation since no information about the time taken to negotiate, the cost or availability of products is available to begin with and must instead be inferred from past interactions.

The middleman-supplier problem is one of *optimisation*. Can the middleman obtain products at a good price. This question resolves into two separate elements, which supplier should the middleman negotiate with and how should the middleman negotiate. Selecting a supplier, or set of suppliers, to negotiate with is important as some supplier may be easier to deal with than others or have a greater abundance of required products. Since this information is not provided to the middleman it must be inferred from past interactions. For a strategy tackling a SSCM-V1 Scenario 1 or 2 problem this first question is moot since only one supplier is available for each product. Under Scenario 3 conditions, however, multiple suppliers for each product may be available and so the question becomes relevant. How the middleman should negotiate with suppliers is a combination of what price the middleman should be willing to pay, how that price is used to make an initial offer and how that offer should be adjusted over time. Since no price information is available directly this must again be based on past interactions.

The customer-middleman and middleman-supplier problems strongly interact and, coupled with the inherent uncertainty, make determining a solution to either more difficult. The received customer requirements restricts the amount the middleman should be willing to pay for products and how long negotiations can last with suppliers (as well a determining exactly what is required). Interaction with the suppliers

informs the customer satisfiability problem in terms of how long negotiations are likely to take, what product prices can be expected and what products are or are not available. The way in which this interaction is conducted will affect the estimates made for each of these. Ultimately, the middleman problem is one of tackling these interactions and attempting to make the most of the available information to behave in a sensible way to leverage a profit.

5.2 The Evolution Of Middleman Strategies

The ultimate aim of this research is to generate middleman strategies to investigate or solve interesting economic problems. To this end, the SSCM (Chapter 4) allows us to describe a broad range of supply chain problems for investigation. The next challenge is to describe a representation for the middlemen strategies used to tackle these problems.

Various approaches could be taken to the generation of strategies but in this instance Evolutionary Computation (EC) has been selected. Evolutionary computation is effective for finding or optimising solutions in many domains ([69, 10]) and has been applied successfully to strategy generation for other, less complex, economic problems ([6, 29, 30]). Due to its success and wide applicability it is again used here.

Various forms of Evolutionary Computation exist. Two broad approaches are Genetic Algorithms (GAs, [69]) and Genetic Programming (GP, [10]). While Genetic Algorithms are effective at optimising pre-defined sets of parameters in order to find a solution, they are unable to evolve algorithms and more complex structures. Genetic Programming on the other hand is able to evolve more general algorithms provided a set of primitive operational and terminal symbols is provided.

A middleman strategy operating within an SSCM market place must take an algorithmic form in order to handle the complexity of interactions taking place. This includes decisions about which customer requirements to attempt to fulfill and how to go about it. This would tend to suggest the use of a GP approach in applying evolutionary computation.

To apply GP it would be necessary to define a set of operational and terminal symbols to be used by the algorithm. This presents a problem in that it is difficult to define the symbols necessary for handling ongoing negotiations and other operations required for an SSCM middleman strategy. A further problem in applying GP is that even if a set of symbols is defined the range of possible solutions is sufficiently large that finding a strategy that will behave in a reasonable way becomes unlikely.

To alleviate both problems and make finding a solution more tractable, a framework can be defined that ensures the basic operation of a middleman strategy is reasonable. This includes all of the basic behaviour necessary for the selection of requirements and supplier negotiations without specifying all of the details of these operations precisely.

Having defined a basic framework GP could then be used to evolve elements that control how that strategy behaves more specifically without having to evolve the entire structure from scratch.

A different approach is to further define the structure behind those control elements. These control decisions while fully defined in terms of their structure still require the specification of parameters that adjust their behaviour. In this way the strategy of a middleman can be fully specified in terms of the instantiation of control parameters for the framework rather than in terms of a whole or partial algorithmic

structure. Having done this a GA or similar approach would be used to evolve a solution.

Of the two variations of the framework above, the first results in a larger search space than the second. Reduction of the search space is important to speed up the learning process by reducing the number and complexity of evaluations that need to be carried out. A strategy evolution system based on the first approach would prove more time consuming to evaluate. Since both the search space presented and the time required for evaluation is greater, the second approach is taken. To this end the SSF defines the greatest part of a middleman's strategy, leaving only certain behaviour controlling decisions for specification and parameterisation. A middleman strategy can then, in total, be defined by the combination of the SSF, the specific decision mechanisms used and the parameters used to control the behaviour of those decision mechanisms. With this framework in place, the evolution of a middleman strategy becomes a matter of evolving the parameter set needed to fully define the strategies behaviour. This can be undertaken by a GA or similar evolutionary mechanism. Within this research Population Based Incremental Learning with Guided Mutation is used for the evolution of middlemen strategies. This approach being able to make use of a GA like representation while making effective use of a small population size for evaluations.

5.3 Summary Of The SSF

The SSF provides a basic operational framework for middlemen strategies to follow in an SSCM-V1 environment. The framework consists of an approach based on the grouping together of similar customer requirements and negotiating to fulfill these as an amalgam. The process of sorting, grouping and fulfilling customer requirements is based on a control logic with specific parameterized decision points. By adjusting the parameters of these decisions points the overall behaviour of the strategy may be changed.

The grouping together of customer requirements aims to allow the middleman to negotiate with suppliers for a smaller number of large bundles of goods rather than a large number of small bundles of goods thus reducing the communication burden on all parties. The disadvantage of this approach is the danger of being unable to fulfill some customer requirements within a group, due to product unavailability, for which the middleman may have already obtained other products elsewhere. In parameterising a middleman's strategy, therefore, control over the grouping mechanism may well prove important to its success.

The basic SSF algorithm ensures reasonably sensible operation of a middleman strategy. This is conducive to evolution since parameterisation of the algorithm allows a range of possible behaviours to be explored while reducing the overall search space and increasing the likelihood of viable strategies being formed. If the SSF's algorithm and related data structures were not in place similar elements to these would have to be evolved from scratch considerably slowing the evolutionary process and quite possibly making the problem intractable.

Section 5.3.1, below, describes the assumptions in the SSF about customer and supplier communication behaviour. Following this Section 5.3.2 describes how an SSF-based middleman strategy operates within the supply chain it is situated. Section 5.3.3 goes on to discuss the limits of the SSF in relation to its development to tackle SSCM-V1 environments. Sections 5.3.4 and 5.3.5 discuss the problem and implementation specific decisions that must be made in applying the SSF to tackle a specific situation.

5.3.1 Expected Customer And Supplier Behaviour

The development of the SSF is mainly concerned with the definition of middlemen strategies however, in order to form this framework, it is necessary to make some assumptions about the behaviour of the other SSCM participants. These assumptions are made based upon the customer and supplier behaviour defined for SSCM-V1 within the bounds of the SSCM-V1 Scenarios described in Section 4.4. These definitions provide some indication of how participants will act but do not specify broader behaviour that affects the operation of the SSF.

One of the primary assumptions of the SSF is the use of an alternating offers negotiation protocol being used for communication, this is defined as part of SSCM-V1 in Section 4.1.6. Under this scheme any product set offered by a supplier will be acceptable to that supplier, likewise a requirement set requested by the customer will be acceptable. In attempting to find an alternative requirement set for a customer, middlemen will offer alternative sets that they may not yet be able to fulfill, this is only possible given the assumptions about customer behaviour in relation to a middleman.

Based on the SSCM-V1 Scenarios, customers are assumed to open all negotiations

with middlemen. They may, or may not, respond to an attempt to find an alternative requirement set but they will always start any negotiation. As a result the SSF makes no provision for opening negotiations with customers even if some were to be known from the initial SSCM position. Customers will wait for as long as possible for a response to their initial or latest request up until their notice period before the earliest product in their current requirement set. This provides the SSF with a known latest response time for the customers. Customer will not negotiation with other middlemen to fulfill their requirement. This means the SSF does not need to consider how to mitigate the effects of a customer suddenly finding its requirement elsewhere. Customers will not accept an offer from a middleman for an alternative requirement set, instead, to signal that the proposed set is acceptable, a counter offer containing the same set is returned. This allows middlemen to make suggestions for alternative product sets without having the products available to fulfill them. An initial customer requirement supplied to the middleman is a new negotiation consisting of a product set and price - no other preference information is supplied. While no further information is provided explicitly, implicit information is provided in terms of the negotiation time-out, this providing a maximum amount of time available to the middleman to fulfill the customers requirements.

Suppliers, based on the SSCM-V1 Scenarios, are passive, they will always wait for a negotiation to be started by the middlemen. Thus the SSF does not need to consider negotiations being started by the suppliers even if those middlemen were to know of them. Suppliers will always try to fulfill a middleman's request for products as far as is possible given its available products and current commitments, they will only reject a request if products are entirely unavailable. This behaviour does not

provide a great deal of information to the SSF beyond the assurance that if products are currently available and requested the supplier will respond accordingly. Products may be unavailable either because the supplier has none remaining or because they are tied up with other negotiations. The second of these is a possibility since the suppliers may only make counter offers if they are able to provide the resources specified, as a result products may become available that were previously unavailable if a negotiation fails or changes in someway.

While the above comprises the basic customer and supplier behaviour further details must also be considered. For the customer this includes how an initial product set is determined for communication to the middleman and how to respond to a offered alternative set. For suppliers, the exact negotiation strategy leading to an offered price for the required products must be specified. While these details are not strictly part of the SSF they are specified as part of SSF-I1, the SSF implementation used for experimentation in this research. These details can be found in Section 5.4.3.

5.3.2 How An SSF-Based Middleman Strategy Works

The aim of a middleman strategy operating within an SSCM-V1 defined environment is to make a profit. A middleman may achieve this by selling on bundles of products, travel packages, composed of elements obtained through trade with suppliers. Customers wish to obtain these bundles to fulfill their own requirements. Middlemen initially have no knowledge about what travel packages customers may be interested in and are unaware of any customers. Customers, however, must make use of middlemen to fulfill their requirements and will contact them stating an acceptable package.

Once this has occurred middlemen may attempt to leverage a profit by obtaining the requested bundle of goods, or one similar, from the suppliers for less than the amount the customer is willing to pay. In this manner the operation of a middleman strategy is driven by its associated customers.

With this in mind, the operation of the SSF can be described both by considering how it processes customer requirements, see Section 5.3.2.1, and algorithmically, see Section 5.3.2.2. Section 5.3.2.3 discusses the information that is maintained by the SSF to aid in its operation.

5.3.2.1 The SSF, A Customer Focused Strategy

The SSF deals with customer requirements in groups. These groups form the basis of operation of the SSF, either accumulating new customer requirements that will need to be fulfilled, actively negotiating with suppliers to fulfill them or informing customers of the outcome of negotiations.

In essence an SSF strategy is customer driven, all action being motivated by the attempt to leverage a profit from the fulfilment of customer requirements. The SSF can therefore be considered in terms of how it processes those customer requirements.

Figure 5.2 provides an overview of how the SSF strategy operates and shows how this is tightly bound to the customer requirements received.

With reference to Figure 5.2, newly received requirements (i.e. new negotiations begun by customers), A, are filtered by the SSF into three basic categories, B. Firstly, those requirements that seem unlikely to be profitable or for which insufficient time is available to complete will be discarded, C, rejection messages sent to the customers. Second, requirements that may be profitable but that the SSF determines can not

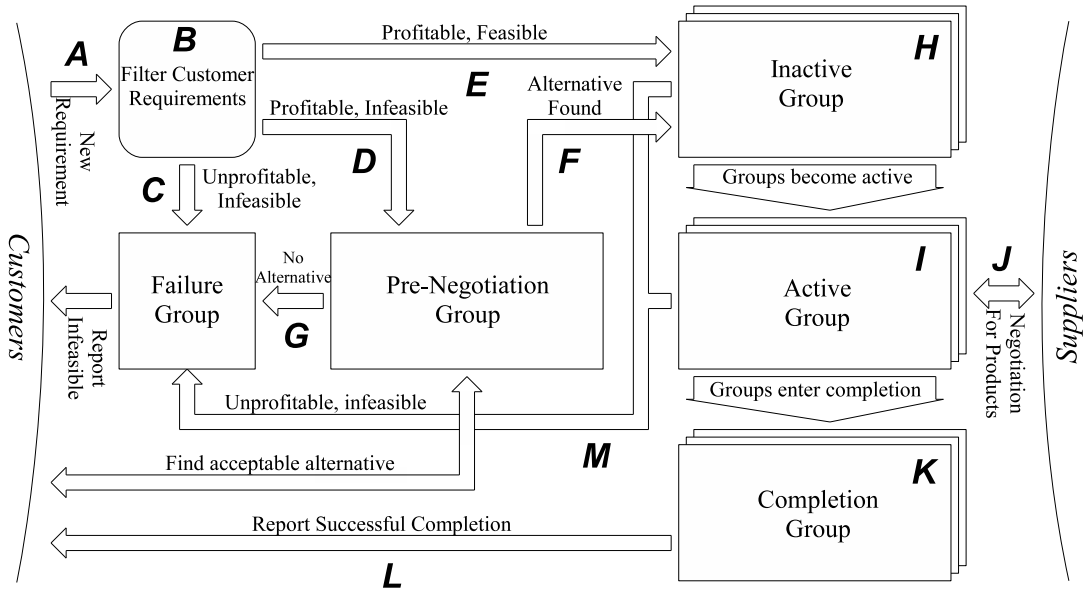

Figure 5.2: SSF Overview, Treatment Of Customer Requirements

be fulfilled because the required products are unavailable are channeled in to a Pre-Negotiation group, D. Third, requirements that appear to be profitable and possible to fulfill will be assigned to an inactive Basic Group, E. If no viable inactive group exists in the set of inactive groups a new one will be created to accommodate the requirement.

For requirements in the Pre-Negotiation group an attempt will be made to find an alternative that the customer finds acceptable and which the SSF believes to be achievable. If an acceptable alternative can be found the requirement will be assigned to an inactive Basic Group, F, if not, it will be discarded, G.

Inactive Basic Groups, H, are used to collect together a set of customer requirements to be fulfilled. An inactive group will, at an appointed time, become active, I, and negotiations with suppliers may begin, J. These negotiation aim to obtain products for the amalgamated set of customer requirements represented by the group.

Multiple groups may be active simultaneously each negotiating in parallel with the suppliers.

Active groups, again at an appointed time, eventually enter a completion phase, K, in which the success of negotiations with suppliers is reported back to the individual customers that are associated with the amalgamated group of requirements, L.

Throughout this process inactive and active groups may discard a customer requirement if it becomes evident that fulfilling it would prove impossible or unprofitable, M. An active group may enter a failed completion state if all requirements are removed but the response to some supplier negotiations is still outstanding.

In making use of the SSF for the basis of a middleman strategy decisions must be made about the operation of certain mechanisms. Some of these mechanisms are specific to the SSCM problem being investigated, SSCM-V1 in this case. These problem-specific decision mechanisms relate to how the middleman uses information from customer offers to help generate possible alternatives for consideration. These mechanisms are discussed further in Section 5.3.4. Other, implementation specific, mechanisms relate to how the SSF negotiates with suppliers and estimates the value of products. These are discussed further in Section 5.3.5. For each of these mechanisms parameterisation of the mechanisms leads to parameterisation of the overall strategy through which its behaviour can be adjusted.

5.3.2.2 The SSF, An Algorithmic View

While the description in Section 5.3.2.1 of the SSF is instructive, the consideration of the SSF as an algorithm is also valuable. The SSF algorithm essentially operates in a two step process, first updating the SSF's world knowledge and then generating

a relevant negotiation message as an output. This process is outlined in Figure 5.3 and discussed further below. A detailed algorithmic account of the SSF is provided in Appendix A.

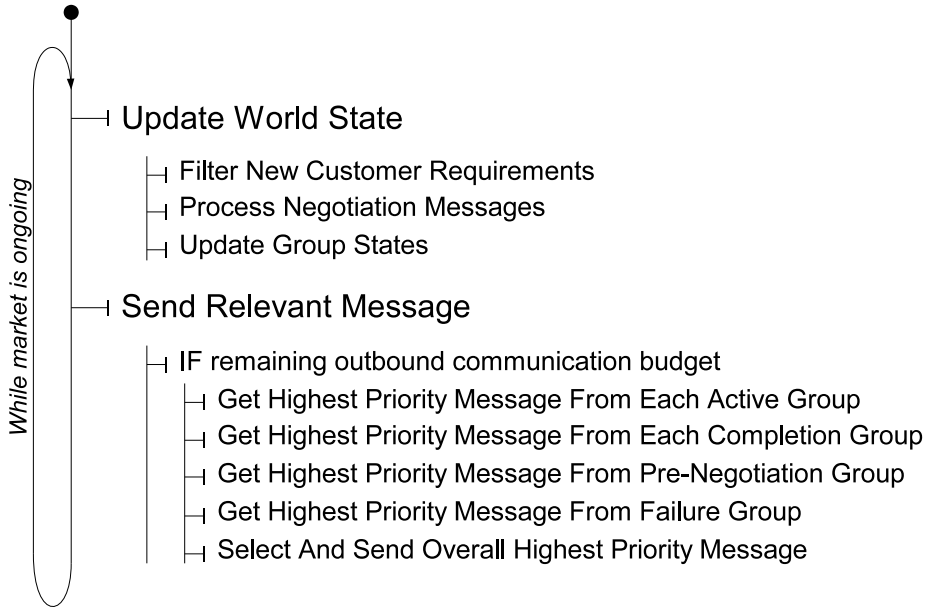

Figure 5.3: SSF Algorithm Overview

In the first algorithmic step all the middleman's knowledge of the world is updated, this knowledge consists of the state of all new, ongoing and completed negotiations, how these relate to the different requirement groups and what may be derived from that information.

This update mechanism primarily involves processing received negotiation messages and dealing with the consequences. In the normal course of events this will mainly involve updating the ongoing state of supplier negotiations to see if any have been accepted or what price is now requested for the required sets of products. It also includes handling newly received customer requirements and filtering these to

find those that the middleman believes may be fulfilled or not.

Along with new negotiation messages, timing effects must also be considered. As discussed above, negotiation for the amalgamated requirements of a group is ordered in time. During its lifetime a group is first inactive, collecting new requirements, becomes active to negotiate with suppliers and then enters a completion phase to inform customers of the result. In updating the world state the algorithm must instigate these state changes.

The second step in the algorithm involves finding the current highest priority negotiation message to send and sending it. This will only be undertaken if sufficient communication budget remains for the current time step, the upper limit being SSCM specified.

To achieve this all active and completion phase groups are polled to generate the message most relevant given their current state. This may be to begin a new supplier negotiation, continue with an ongoing negotiation or inform a customer of an outcome.

Likewise, for customers that the middleman wishes to satisfy but believes products currently to be unavailable for (those in the Pre-Negotiation group), the highest priority alternative suggestion will be generated.

From this set of messages the highest priority message will be selected and sent to the relevant market participant. Messages from the Failure Group terminate negotiations with rejection and will only ever be sent if no other message is available. The SSCM-V1 communication scheme implicitly rejects any negotiation to which a response is not received before the specified Time-Out value.

While providing the main framework of a middleman strategy, the SSF defers specification of some components to provide additional flexibility in implementation. These components are decision points within the framework that will directly affect the strategies behaviour. These decision points fall in to two categories. The first are problem-specific decisions are those which relate specifically to the problem being considered. Specification of decision mechanisms must be based upon some knowledge of the problem under consideration. These are discussed in Section 5.3.4. The second are decisions that are implementation specific, do not require knowledge of the specific problem being tackled but will affect how the strategies behave. These are discussed in Section 5.3.5. While the SSF is intended to provide a flexible framework for tackling SSCM problems it has been motivated by the desire to tackle SSCM-V1 specifically. This places some restrictions on its wider applicability and is discussed in Section 5.3.3.

5.3.2.3 Information Maintained By The SSF

For the SSF to be able to operate effectively it must maintain a wide range of information about its current state and past interactions with other participants. This information can be used by the strategy to help make decisions such as how to filter new customer requirements for profitability and feasibility and which suppliers to make use of. The set of information maintained by the SSF is collectively known as its $World\_State$ and is illustrated in Figure 5.4.

With reference to Figure 5.4, the SSF world state comprises five basic types of information.

The first of these, the $BasicStateInformation$, describe the strategies immediate

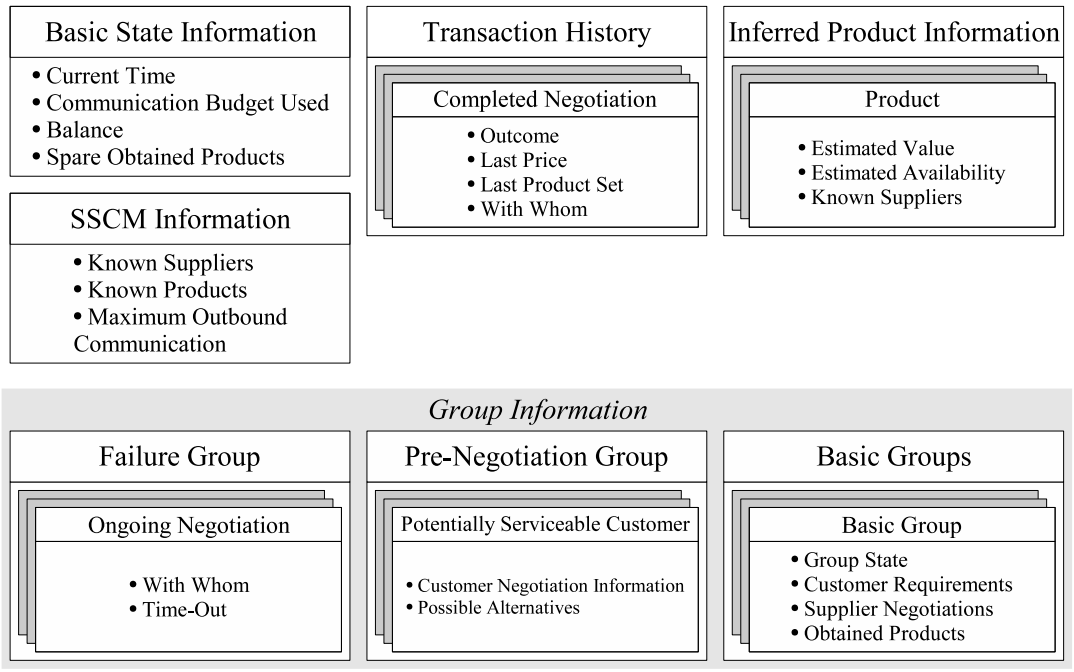

Figure 5.4: Overview Of The Information Maintained By The SSF

circumstance with regards to time, ability to communicate and free resources that might be used to help fulfill customer requirements and as such is used throughout the SSF.

The $SSCMInformation$ relates directly to information provided to the SSF as part of the instantiated SSCM currently being tackled. This information acts as a basic reference about the products and suppliers the strategy will be dealing with.

The SSF's $TransactionHistory$ provides a record of all past interactions with other supply chain participants, both customers and suppliers. This set of information is intended to act as a resource to help strategy mechanisms mediate current middleman behaviour based on past experience.

$InferredProductInformation$ is intended to act as a central store for information derived about products from the $TransactionHistory$ and other, current, interactions. Maintaining a central store in this way allows the information to be updated and used more readily.

Finally, information is maintained about the different groups operating within the SSF. The $GroupInformation$ maintains the association between ongoing customer and supplier negotiations and their related groups, possible sets of alternatives for customers requirements that have yet to be assigned to a group and the state and activity of the various basic groups.

Maintaining this information effectively is an important part of the SSF since it essentially governs how the customer and supplier side of the middleman problem are able to interact and the resources available for decision making.

5.3.3 Limits Of The SSF, Designs With SSCM-V1 In Mind

The SSF has been developed with SSCM-V1 in mind. While adaptable to other SSCM problems this does mean a number of assumptions have been made during its formation.

Firstly, a negotiation based communication mechanism is assumed, this is described briefly in Section 4.3.4 and elaborated on for the SSF in Section A.1.1 below. Communication within the SSF is restricted to negotiations based upon an alternating offers protocol. Timing synchronisation is also assumed. No messages will be sent or received relating to a negotiation that has timed-out and all participants update their timing synchronously to ensure this. Finally, messages will not be lost or garbled, the assumption being that delivery is unhindered.

Secondly, under SSCM-V1, unlike SSCM in general, customer and supplier behaviour has certain restrictions placed upon it. Customers act to initiate contact with middlemen while suppliers wait to respond to them. Further, to some extent, the negotiating behaviour of customers and suppliers is specified and assumed by the SSF. This allows the SSF definition to ignore certain situations that could arise in SSCM problems more generally. In restricting the SSF's applicability to other SSCM-based problems the primary culprit relates to customer negotiation time-outs being used to provide additional information to middlemen about the total time available to fulfill their requirements. While other assumptions relating to the expected behaviour of customers and suppliers allow the simplification of the SSF definition, this assumption is key to the operation of the SSF. In applying the SSF to other problems where this assumption could not be made careful consideration would be required to bypass this mechanism. The customer and supplier behaviour expected by the SSF is described in more detail in Section 5.3.1 below.

5.3.4 Problem-Specific Decisions

The SSF is an approach that is applicable to middlemen in many different SSCM problems not just those that form SSCM-V1. Providing the assumptions about customer and supplier behaviour (Section 5.3.1 above) hold true, the SSF can be used as the core of a middleman strategy.

The SSF provides a basic middleman strategy framework with certain decisions points that must be specified. The way in which these decisions are tackled is mostly a matter of implementation preference (within this research guided by the desire to evolve solutions) however, in some cases these mechanisms must be informed by the

specific problem being tackled. The Table 5.1, below, lists these elements in the form of SSF algorithm components.

All of the listed mechanism relate to the way in which alternative product sets are presented to the customer. Alternative product set generation is problem-specific since the interaction between products may vary from problem to problem. Under SSCM-V1, time steps are viewed as days. Inbound and outbound flights are required, with accommodation being necessary on each day up to but excluding the return flight day. Entertainment of some sort may or may not be required during this time. These characteristics are specific to SSCM-V1 and would be unlikely to hold true in other domains. Generating an alternative product set is thus a matter of generating a set with similar characteristics as that offered by the customer.

To apply the SSF to another SSCM problem these mechanism would therefore need to be specified in accordance with domain knowledge consistent with that problem. The way in which these problem-specific mechanisms are handled in the case of SSF-I1 is discussed in Section 5.4.1.

Other mechanisms that are not specified entirely by the SSF, are not problem-specific but must be defined in any full implementation are discussed in Section 5.3.5, below.

5.3.5 Implementation Specific Decisions

Implementation specific decisions are those that must be made by a middleman strategy that is based on the SSF but which SSF itself does not full specify. Mechanisms for resolving these decisions must be decided upon for the SSF to be used. While the

Table 5.1: SSF Problem-Specific Mechanisms

Procedure Name	*Output*	*Input*	*Description*
Create-AlternativesFor	Alternatives	CusNeg, Value	Generates a set of *Alternatives* to be used by the *PreNegotiationGroup*
Generate-Customer-Alternative	Message	CusNeg, Alternatives	Given a customer negotiation and an alternatives set, generate a customer counter offer
Remove-AlternativeFor	-	Negotiation	Find and remove the attempted alternative from the set of alternatives for a customer negotiation in the *PreNegotiationGroup*

SSF provides the basic algorithm used by middlemen and the primary control logic, decisions within that logic are based upon mechanisms that must be specified by any particular implementation. The specified mechanism could be as simple as a default answer (with or without parameterisation) or some random or other process. These mechanisms may or may not require parameterisation. In the case of this work the aim of learning effective middlemen strategies ultimately is a matter of learning good values for the parameters of the implemented mechanisms.

Tables 5.2 and 5.3 contain lists of mechanisms, described as SSF algorithm components, that must be specified by any particular SSF implementation. The mechanisms chosen for the implementation considered in this research, SSF-I1, are specified and discussed in Section 5.4.1.

The implementation specific mechanisms presented here fall broadly into four categories. Firstly, mechanisms relating to the selection of customers to fulfill. Secondly,

mechanisms that help determine when to start and stop negotiating. Thirdly, mechanisms that determine how to negotiate with suppliers. Finally, mechanisms which deal with the effects of completed negotiations.

Since the SSF is effectively driven by the customer requirements it receives, the mechanisms that specify how these are filtered are extremely important. A good set of mechanisms should be able to reliably select customer requirements that will prove profitable, possible to fulfill and for which sufficient time remains.

In the descriptions of the SSF presented in Sections 5.3.2.1 and 5.3.2.2 the activation and completion phases of groups are described as being triggered according to time. This need not be the case, other mechanisms might be considered and used in place of or to augment this.

The effectiveness of negotiations with customers is important to the success of a middleman strategy. Without effective mechanisms to aid negotiation products may be obtained at higher prices than necessary. Higher prices would lead to a reduction in profitability, potentially making the middleman unviable in the supply chain. These mechanisms include decisions about what price to offer suppliers for product bundles but also which suppliers to try and make use of to begin with and what time constraints to place upon offers.

The completion of a negotiation may have many effects within the SSF depending on the participant involved and how the negotiation ends. The completion of negotiations provides and opportunity to update product value estimates but may also give rise to a need for conflict resolution. A need for conflict resolution arises when insufficient products are available to fulfill all the requirements of an active group, this may

Table 5.2: SSF Implementation Specific Mechanisms, Part One. Sorting Customer Requirements and Group State Transitions

Procedure Name	*Output*	*Inputs*	*Description*
Sorting Customer Requirements			
Sufficient-TimeTo-Negotiation	Boolean	CusNeg	Determine if there is sufficient time to negotiate with suppliers to fulfill this customers requirements
ProductSet-Feasible	Boolean	CusNeg	Determine if the latest product set of the negotiation is probably possible to obtain from suppliers
GroupWill-Take-Negotiation	Boolean	BasicGroup, CusNeg, Value	Determines if an *Inactive*, *BasicGroup*s should/could take this negotiation taking in to account its estimated value
BestBasic-GroupToJoin	BasicGroup	CusNeg, Value	Assuming there is an existing *Inactive*, *BasicGroup* able to take the customer negotiation, find and return the best possibility
Group State Transitions			
ShouldBecome-Active	Boolean	BasicGroup	True if an *Inactive* basic group should become *Active*
Enter-CompletionDue-ToTiming	Boolean	BasicGroup	True if the *BasicGroup* should enter completion due to other (timing) reasons. Responses from suppliers must be set to time-out by the time this mechanism would trigger completion

Table 5.3: SSF Implementation Specific Mechanisms, Part Two. Supplier Negotiations, Handling Negotiation Failure, Handling Negotiation Success

Procedure Name	*Output*	*Inputs*	*Description*
Supplier Negotiations			
SelectSupplier	Participant	Profile	Select a supplier from the set available in the $Profile$
Determine-StartTimeOut	Time	-	Determine the timeout the middleman should use in making an initial offer
Determine-StartOffer	Real	ProductSet	Determine the amount the middleman would initially be willing to offer a supplier for the specified set of products
Determine-TimeOut	Time	Negotiation	Determine a new timeout that should be used in responding to this negotiation
Determine-Offer	Real	SupNeg, ProductSet	Determine the amount the middleman is willing to pay for these goods at this time
Handling Negotiation Failure			
FindConflicting-Customer-Requirements	Set of CusNegs	BasicGroup, ProductSet	Find a set of customer requirements from an active group that should be removed as a result of the inability to obtain the specified $ProductSet$
Negotiation Success - Updating Product Price Estimates			
UpdatedProduct-ValueEstimates	-	-	Update the estimated value of products in the $Product\_Info$ of the $World\_State$

occur either in ongoing negotiations or as a result of a negotiation ending. When this occurs a decision must be made about which requirements to abandon with consideration for their potential profitability, products already obtained and other ongoing supplier negotiations.

In summary, while problem-specific decision mechanisms are concerned with the issues relating to the structure of the problem and how to accommodate them, ultimately both problem and implementation specific decisions mechanisms are concerned with how to best leverage the strategies available information most effectively. Most of this information being derived from the interaction with other supply chain participants.

5.4 Tackling SSCM-V1 Specifically, SSF-I1

The SSF as discussed so far provides the basic framework for a middleman strategy. This framework is, however, incomplete and requires problem (see Section 5.3.4) and implementation (see Section 5.3.5) specific mechanisms to be defined before it can be used.

This section discusses SSF Implementation One (SSF-I1). SSF-I1 completes the SSF definition by providing the problem and implementation specific mechanisms required. Problem-specific mechanisms are considered that fit SSCM-V1 while both problem and implementation specific mechanisms are designed to complement evolutionary processes (see Section 5.4.1). These mechanisms are controlled through a series of parameters shown in Section 5.4.2.

Having fully defined the operation of the SSF, SSF-I1 further specifies the operation of Customer and Supplier participants (see Section 5.4.3) and list their control parameters (see Section 5.4.4).

5.4.1 Problem And Implementation Specific Decisions

This section discusses the set of SSF problem and implementation specific mechanisms that have been decided upon for SSF-I1. These mechanism essentially complete the more general SSF and allow us to consider more precisely the behaviour of SSF-based middleman participants. The problem and implementation specific decisions that must be specified are described in Section 5.3.4 and Section 5.3.5 respectively.

The problem and implementation specific decisions fall into five broad categories. Firstly, mechanisms that help in the selection of which customer requirements to fulfill. Second, mechanisms that help find alternative requirements for customers. Third, mechanisms that effect group state transitions and so determine when negotiations start and end. Fourth, mechanisms that aid in supplier negotiations. Finally, mechanisms that help deal with the effects of completed supplier negotiations. These categories are tackled in the following sections. The complete set of controlling parameters for these mechanisms, those that would be subject to evolution, are shown in Section 5.4.2.

5.4.1.1 Customer Selection Mechanisms

The functions to be defined in this section are *SufficientTimeToNegotiation*, *ProductSetFeasible*, *GroupWillTakeNegotiation* and *BestBasicGroupToJoin*.

The *SufficientTimeToNegotiation* mechanism uses a simple fixed minimum required time to determine if there is sufficient time to negotiate for a given customer

requirement. Basic groups become active a certain amount of time before the first contained customer negotiation times-out, see Section 5.4.1.3. This *GroupActivationTime* therefore determines if there is enough time to negotiation. To be accepted by this criteria a customer negotiation time-out must be $\geq CurrentTimeStep + GroupActivationTime$.

The *ProductSetFeasible* mechanism is based on information available from the $TransactionHistory$. For any required product at a given time, a check is made to determine how many times the middleman has been unable to obtain products from each of the possible suppliers. A product is considered to have been unable to be obtained if the supplier has either rejected a product set containing this element or has offered a new product set without it. If all suppliers have done this a number of times beyond a certain threshold (the *FoundUnavailableThreshold*), that product at that time is considered unobtainable. If any product in a specified product set is considered to be unavailable then the product set as a whole is considered unfeasible.

SSF-I1 groups customer negotiations together based on their time-out times. The initial customer negotiation in a group acts as the basis for a time based catchment area. Any other customer with a time-out time on or after that first customers time-out time within a certain window (*GroupDuration*) may become part of that group. The *GroupWillTakeNegotiation* mechanism will therefore respond positively if the specified customer negotiations time-out falls within one of these inactive group catchments.

Catchments may overlap if a customer negotiation is placed in a group and has a time-out time that is shortly before a customer in an existing group. If customers negotiations may be placed in one of several groups the group with the smallest

membership is selected. If this fails to make a determination, random selection is used. This is the mechanism behind the *BestBasicGroupToJoin* mechanism.

5.4.1.2 Customer Re-Negotiation Mechanisms

Customer renegotiation mechanisms are those that relate to the generation, selection and removal of possible alternatives. These mechanisms are *CreateAlternativesFor*, *GenerateCustomerAlternative* and *RemoveAlternativeFor*, each of these is specific to the SSCM-V1 problem.

An alternative for the purposes of SSF-I1 is represented as a tuple containing a inbound and outbound flight time and number of entertainments.

The *CreateAlternativesFor* mechanism creates a set of alternatives based on the customer's initial offer. Possibilities are generated for all trip durations that match that of the customer's initial suggestion and below (down to the minimum possible, 2). This is done for all remaining possible outbound trip start times. The number of entertainments is counted and this is added to each tuple, except where it exceeds the trip duration. All alternatives are then pruned according to whether any of the flights or accommodation in between are unfeasible. These alternatives are then ordered according to priority. Prioritisation is achieved through simple Euclidean distance, the duration and outbound flight time being the two metrics. If a distance is tied, the closest outbound flight time is preferred.

The *GenerateCustomerAlternative* mechanism takes the highest priority alternative in the created set. A product set containing outbound and inbound flights and accommodation may be formed. To this the number of specified entertainments are added. Those that match the customer's original suggestion, are considered feasible

and fit within this trip are added first. The remaining number of entertainments are added at random but ensuring feasibility. This then forms the basis for the negotiation message.

The *RemoveAlternativeFor* removes the highest priority alternative, the one that would have been suggested for sending.

In addition SSF-I1 places a limit on the number of attempts that will be made to renegotiate with customers. If this limit is exceeded without an acceptable alternative being found the attempt to satisfy this customer is abandoned and the related negotiation removed to the failure group. This limit is specified by the *PreNegotiationTries* parameter.

5.4.1.3 Group State Transition Mechanism

Mechanisms that affect the group state transitions are *ShouldBecomeActive* and *EnterCompletionDueToTiming*. The determination of when an *Inactive* group should become *Active* and when an *Active* group should enter *Completion* and begin information customers of success or failure is based on timing criteria.

The *ShouldBecomeActive* function will indicate an *Inactive* group should become *Active* if the $CurrentTimeStep$ becomes equal to the time-out of the earliest customer of the group minus the $GroupActivationTime$.

The *EnterCompletionDueToTiming* is based on the *ActiveProportion* parameter. This variable helps to determine the amount of time a basic group will spend negotiating with suppliers for products versus the amount of time set aside for contacting customers to report the successful acquisition of acceptable packages. The total amount of time to achieve both of these is determined by the $GroupActivationTime$, the time

before the earliest required customer response that an inactive group will become active and start supplier negotiations. The *ActiveProportion* parameter specifies how much of this time should be spent in negotiations with suppliers. The remaining time to be used for contacting customers in the groups completion phase. To ensure that some time is available for contacting customers a minimum of one time-step is set aside for this purpose potentially overruling the specified *ActiveProportion* value. The *EnterCompletionDueToTiming* takes account of these factors to determine if an active basic group should now enter the completion phase.

5.4.1.4 Negotiating With Suppliers

The mechanisms that aid negotiation with suppliers perform three jobs. The selection of a supplier to deal with, the determination of when a response is required by and the evaluation of the required product set value, the price that should be aimed for.

The *SelectSupplier* mechanism operates by selecting a supplier randomly from the available set. This evenly distributes attempts to obtain products but at the expense of deeper reasoning about which supplier may be better to deal with.

The *DetermineStartTimeOut* and *DetermineTimeOut* mechanisms make use of an arbitrarily specified time-out time ahead of the $CurrentTimeStep$. This is specified by the parameter $NegotiationTime-Out$. The calculated negotiation time-out must be adjusted if it would exceed the point in time at which the associated basic group would enter its completion phase. This is necessary to ensure that when the group enters completion all supplier negotiations are under the middleman's control. To this end if a calculated time-out does exceed this point it is instead set to coincide with the group entering completion.

The *DetermineStartOffer* and *DetermineOffer* functions are responsible for determining what value should be specified for a given product set when making and offer or counter-offer to a supplier. These mechanisms make use of the inferred product information about the probable per unit value of products as well as the history of the current negotiation being considered. The mechanisms are controlled by the $UpperPriceMultiplier$, $LowerPriceMultiplier$ and $TacticValue$ parameters associated with the product types being sort.

The mechanism used to determine the offer price is based upon that boulware and conceder bargaining tactic proposed by Matos ([67]). Under a revised version of this mechanism an acceptable per unit price is sought for the required product types. The acceptable value is specified between a maximum and minimum value based upon the estimated product per unit value. The limits are determined by multiplying this value by the *UpperPriceMultiplier* and *LowerPriceMultiplier* respectively. These multipliers are specified separately for each product type.

The amount of time taken so far for the negotiation is taken in to consideration and adjusts the per unit value that is currently acceptable, or will be offered, between the above limits.

At the beginning of a negotiation the minimum possible unit price will be used. This allows easy determination of an offer by the *DetermineStartOffer* mechanism. The *DetermineOffer* mechanism makes use of a time based function to determine the unit cost. This function is controlled by a tactic value ($TacticValue$) specified for each product individually. This mechanism is shown in Definition 5.4.1.

Definition 5.4.1. Determine Product Unit Price

$$UnitPrice = MinPrice + (PriceGap * PriceMultiplier)$$

$PriceMultiplier = 1.0 - (TimeUsedSoFar/NegotiatingTime)^{1.0/TacticValue}$
$TimeUsedSoFar = CurrentTime - GroupActivationTime$
$NegotiatingTime = GroupCompletionTime - GroupActivationTime$
$PriceGap = MaxPrice - MinPrice$
$MinPrice = EstimatedPrice * LowerPriceMultiplier$
$MaxPrice = EstimatedPrice * UpperPriceMultiplier$

EstimatedPrice	-	The estimated product price
GroupCompletionTime	-	When the group to which the negotiation belongs would enter completion
GroupActivationTime	-	When the group to which the negotiation belongs would become active

5.4.1.5 Completed Negotiations

These are mechanisms that come in to effect when negotiations are completed. *FindConflictingCustomerRequirements* is used to determine which customers will be dropped in response to an unobtainable product set. *UpdatedProductValueEstimates* determines how the estimated product values are updated.

The *FindConflictingCustomerRequirements* mechanism essentially uses a greedy algorithm to remove customers. Customers with requirements that conflict with the product set are removed according to their value. Each customer requirement within the active group has an associated estimated value based on the amount the customer is willing to pay and the estimated cost of the products needed to fulfill it. Requirements with the lowest estimated value are removed first. This process continues until the complete set of unobtainable products is covered by the total set of products encompassed by the removed customer requirements.

The *UpdatedProductValueEstimates* mechanism makes use of the SSF *TransactionHistory* to update estimated product values. A product's value is taken as the mean of unit prices achieved for that product for all accepted negotiation within

a specified window. That window is determined by the parameter *ProductInformationWindow*. If no information is available from the $TransactionHistory$ an arbitrary *GuessPrice* is used. The *GuessPrice* is specified for each product individually.

5.4.2 Evolvable Middleman Parameters

This section, see Table 5.4, lists the set of parameters required by the SSF-I1 specific mechanisms discussed in Section 5.4.1, above, along with the standard parameters needed by any SSF implementation. These parameters will be the focus of evolution in the SSCM Market Simulation System described in Chapter 6.

5.4.3 Specific Customer And Supplier Mechanisms

The SSF-I1 provides a description of the specific mechanisms need to fully specify an SSF strategy framework for its use by middlemen in a supply chain.

However, the behaviour of customers and suppliers has not yet been fully defined. To provide a more specific context to the operation of the SSF-I1 middlemen the definition of customer and supplier behaviour is discussed in the Sections 5.4.3.1 and 5.4.3.2 and a list of controlling parameters is provided separately in Section 5.4.4.

While the operation of SSF-I1 is not contingent upon these specific mechanisms being used by customers and supplier they are included here for simplicity.

5.4.3.1 Customer Behaviour

Customers under SSF-I1 aim to fulfill a requirement from within a set range. This requirement is specified via the SSCM-V1 specific parameters shown in Section 4.3.3. Under these conditions the customer desires a trip of some minimum and maximum

Table 5.4: Evolvable Middleman Strategy Parameters

Parameter Name	*Usage*
GroupActivationTime	A group consists of multiple customer requirements. Each requirement must be responded to by a certain time. This parameter specifies how many time steps before the earliest requirement must be responded to the group should become active
FoundUnavailableThreshold	How many times a specific product at a specific time must be unavailable from a particular supplier for it to be considered unavailable from that supplier in general.
GroupDuration	In this SMSS implementation groups are responsible for customer negotiation over a set time period. This parameter specified how long that time period is
PreNegotiationTries	S2/3 only. This is the number of times the SSF will attempt to find an acceptable alternative to the customers initial product set. If exceeded the attempt will be abandoned
Active Proportion	As a base half of the time from group activation to the earliest required customer response is set aside for supplier negotiations. This parameter determines how much of the remaining time is used for negotiations
NegotiationTime-Out	How long is given for suppliers to respond to a negotiation
ProductInformationWindow	The number of past transactions involving a particular product that should be used to estimate its value
	For Each Product
GuessPrice	The price used as a default estimate of a products unit value when there is otherwise insufficient information available
$UpperPriceMultiplier$	Determines the maximum price the middleman will pay per unit for a product. This is a multiple of the current estimated product value
$LowerPriceMultiplier$	Determines the lowest price the middleman will offer per unit for a product. This is a proportion of the current estimated product value
$TacticValue$	Value that controls the curve of the offered price over time.

duration within a certain time frame (or the market active time) with a minimum number of entertainments provided. Further, this requirement should be met according to a specified budget.

Customer behaviour under SSF-I1 is partially governed by the SSCM Scenario being investigated however in all cases the customer must make the first approach to the middleman.

Before an initial offer can be made the trip specifically to be requested must be formed from the requirement specification. In the discussion of value selection from a range, even probability is always used. To do this a trip duration is first select from the range of possible durations specified by the requirement ($MIN\_DURATION$ to $MAX\_DURATION$). A outbound flight time can then be selected from within the range of desired travel times ($EARLIEST\_START$ to $LATEST\_END$) taking into account the need to return before the specified latest possible end of the trip. The amount of required entertainments is then determined from the range specified (between MIN_ENTS and MAX_ENTS) but restricted to being no more that of the trip duration. The specific types of entertainment and their distribution throughout the duration of the trip then proceeds randomly. No entertainment is allowed on the outbound flight day but may occur on the return flight day. The budget of the trip is taken directly from the specifications $BUDGET$ value.

Having determined an initial requirement from the specification this must the be presented to the middleman. The timing of this offer is the next customer decision that must be made. The customer could request its requirement as soon as the market begins but must have its requirement fulfilled MIN_NOTICE time-steps before the

outbound flight day of the requested trip (the latest response time). This places and upper bound on the latest time the customer can make the request. The specific time is based on the $NEGOTIATION\_TIME$ parameter listed below (see Table 5.5). This indicates that the initial request will be sent $NEGOTIATION\_TIME$ time-steps prior latest response time, giving the middleman time to negotiate. If this is before the beginning of the market, the request must be sent immediately. In sending the request, the latest response time is used as the negotiations time-out value.

Under Scenario One conditions, the customer is unwilling to budge from this initial requirement. Any counter-offer that is received, will be met with a customer counter-offer matching the initial request. Under Scenario Two and Three conditions, the customer is willing to negotiate on its initial offer within the confines of its requirement specification. If a middleman's counter-off falls within the bounds of the requirement specification, the customer will respond by matching that offer exactly in its own counter-offer. If, however, the middleman's counter-offer fails to fall within the requirement specification, the customer will respond with the best possible match to that counter-offer that does lie within the specification.

Matching a middleman's counter offer begins by matching the duration as closely as possible. Following this the outbound flight time will be matched taking in to consideration the latest possible return time. Entertainments will the be selected. Any offered by the middleman that fall within this new requirement will be kept and, if more are required, these will be placed randomly in the manner of the initial offer generation. The budget will be kept as in the initial offer and this counter-offer sent to the middlemen.

With this process in place, the middleman's task becomes one of attempting to fulfill the customer's requirement without direct knowledge of what that requirement is.

5.4.3.2 Supplier Behaviour

Suppliers under SSF-I1 operate as specified the SSCM Scenarios (Section 4.4. Under these conditions, suppliers wait for middleman requests for products and will attempt to supply any possible subset of that request, only rejecting outright if the supplier is entirely unable (at this time) to fulfill it.

Availability of products for an individual supplier is based on the initial allocation, fulfilled prior requests and current negotiations. Thus, the amount of a given product available for new negotiations is at a given time based on the total amount remaining to the supplier minus any amount currently committed to through previously made counter-offers in other negotiations. The supplier will provide a subset of the request requirement or reject that requirement if no products to fulfill it are currently available. It is, however, possible that products to fulfill that requirement may at a later stage become available if other ongoing negotiations change or are rejected for some reason.

The price of a product set requested from the supplier is based on a calculated per unit price for the products of that set. The mechanism for the calculation of this price is based on a similar system as used for middlemen (see Section 5.4.1). This mechanism uses as the minimum possible price, the SSCM-defined $BaseVaue$ associated with this product for this supplier. An upper price is generated by multiplying this value by the $BASE\_VALUE\_MULTIPLIER$ for this supplier. The value sort

at any given time is then calculated based on the formula in Definition 5.4.2 and controlled by the $SUPPLIER\_TACTIC$. If the price offered by the middleman equals or exceeds the suppliers calculated unit price, the offer will be accepted. If not a counter-offer making use of this unit price will instead be sent.

Definition 5.4.2. Supplier, Product Unit Price

$$SupplierUnitPrice = BaseValue + (PriceGap * PriceMultiplier)$$

PriceMultiplier $= (TimeUsedSoFar/NegotiatingTime)^{1.0/SUPPLIER\_TACTIC}$
$PriceGap = MaxPrice - MinPrice$
$MinPrice = BaseVaue$
$MaxPrice = BaseVaue * BASE\_VALUE\_MULTIPLIER$

$NegotiatingTime$	-	The earliest requested product time minus the time the request was initially received
$TimeUsedSoFar$	-	The amount of time that has passed since the initial request was made

By varying the $SUPPLIER\_TACTIC$ suppliers may become more or less difficult to negotiate with for the middleman. Values less than one indicate the supplier will be stubborn and will only give ground slowly. Values greater than one indicate a willingness to trade quickly and so give ground quickly over price.

5.4.4 Customer And Supplier Parameters

The SSF-I1 specifies customer and supplier behaviour as discussed in Section 5.3.1. The parameters that control this behaviour are listed here in Table 5.5.

The customer and supplier parameters make up part of the environment to which a SSF strategy would be exposed. While the SSCM provides the definition of most of the market conditions the customer and supplier parameters strongly effect how difficult the middleman task will be for any given SSCM instantiation. A stubborn

Table 5.5: SSF-I1, Customer and Supplier Parameters

Parameter Name	*Usage*
Customer Parameters	
$NEGOTIATION\_TIME$	The time interval the customer will give the middleman to fulfill its requirement. This helps define when the customer will make its initial requirement request to the middleman
Supplier Parameters	
$BASE\_VALUE\_MULTIPLIER$	Determines the upper unit price sort for products by a supplier. This multiplies the SSCM-defined $BaseValue$
$SUPPLIER\_TACTIC$	Value that defines how the supplier will behave in negotiation

supplier, or one with high initial product unit costs, will make acquiring products at a reasonable price difficult. Likewise a customer with low $NEGOTIATION\_TIME$ will make a middleman's task much more difficult by restricting the time available to it for attempt to obtain products at a reasonable cost.

The consideration of these parameters in characterising SSCM market used for evolutionary purposes is discussed further in Section 6.4.2.1.

5.5 Conclusions

The SSCM Strategy Framework (SSF) provides the core structure for a middleman strategy operating within an SSCM environment. This structure is used as the basis for a fully defined, parameterised, set of middlemen strategies, SSF-I1, that is conducive to evolutionary learning. A specific middleman strategy is defined by instantiation of the SSF-I1 parameters.

Designed to tackle the problems of SSCM-V1, the SSF provides a feasible framework that may exhibit different behaviour depending on the instantiation of its control mechanisms and parameters.

The SSF, while providing the core strategy framework, requires decision mechanisms for certain situations to be specified in order that a middleman strategy be fully defined. The specification of these mechanism and the way in which they are parameterised control the overall behaviour of the middleman strategies.

For the purposes of experimentation within the bounds of SSCM-V1, SSF Implementation One (SSF-I1) has been defined. SSF-I1 provides a fully defined SSF framework that is conducive to the evolution of middleman strategies. Specific negotiation decisions within this framework are based on mechanisms used by Matos, [67], for successful evolutionary learning of negotiation strategies.

The SSF itself operates by first updating its world state, taking in to account the change in state of negotiations with customers and suppliers and how these are grouped, before generating and sending at most one message to another supply chain participant. The SSF repeats this process until the supply chain ends.

The flexibility in behaviour of SSF-based middlemen strategies operating within SSCM-defined situations, and their ability to be generated and improved through the application of evolutionary computation, ties in with Claim Two of the thesis made in the introduction, see Section 1.4.

Chapter 6

Evolving Middleman Strategies - The SSCM Market Simulation System (SMSS)

The SSCM Market Simulation System (SMSS) is a platform for the evolution of middlemen strategies within supply chain environments. This platform provides a mechanism for the evolution of SSF-I1 based middlemen strategies for supply chain situations primarily defined in terms of the SSCM. The evolutionary component of the SMSS is based on Population Based Incremental Learning with Guided Mutation (PBIL+GM, [29, 9]).

This chapter introduces the SMSS, describing what it is attempting to achieve, how it goes about accomplishing this and what it finally produces.

Implementation details of the SMSS software are not discussed here, for an overview of these please see Section B.1 of the Appendix B.

6.1 What Are We Trying To Achieve

The objective of the SMSS is to provide the capability to experiment within the context of the SSCM, more specifically SSCM-V1 Scenario's 1 to 3.

The aim of experimentation with the SMSS is to find good middleman strategies through a process of evolution in response to a specified set of market conditions. This can be encapsulated thus:

$F(E) \longrightarrow M$

Where E is the market environment we specify and M is the resultant middleman strategy. The SMSS aims to provide the mapping function F. The SMSS also provides a mechanism for describing E and capturing M.

Middleman strategies are embedded within an SSF-I1, see Section 5.4, implementation and evaluated within a market simulation.

The evolved middlemen strategies are represented in the form of a Population Based Incremental Learning with Guided Mutation (PBIL+GM) probability distribution. This probability distribution provides an indication of how SSF-I1 based strategies should be instantiated such that they operate effectively and obtain reasonable results for the specified supply chain conditions.

6.2 Learning Middlemen Strategies, An Overview

The aim of the SMSS is to learn effective middlemen strategies for a given set of supply chain conditions. This section discusses these objectives and describes what, specifically, the SMSS is trying to learn, Section 6.2.1, and how it goes about this, Sections 6.2.2 and 6.2.3.

6.2.1 What are we trying to learn?

The SMSS attempts to learn a good middleman strategy for a given set of supply chain conditions.

Within the SMSS middleman strategies are defined through SSF-I1, introduced in Chapter 5. SSF-I1 based middlemen strategies require a number of parameters to be set (see Table 5.4), thus learning a good middleman strategy is a matter of learning a good set of parameters that allow SSF-I1 to work effectively under the specified conditions.

While finding individual middlemen strategies is useful, the SMSS is ultimately trying to learn how to instantiate those strategies. Therefore the result of an SMSS run experiment is not an individual strategy instantiation but rather a mechanism for performing successful instantiation of strategies.

A set of parameters used to instantiate SSF-I1 constitute the strategy of an individual middleman. Learning a good middleman strategy for the environment requires the SMSS to learn how to pick those parameters for effective operation of SSF-I1. To this end the SMSS maintains a probability distribution that represents what values are more, or less, likely to yield good results for each parameter. Starting with a probability distribution that assigns equal quality to all values of all parameters the SMSS aims to refine this model to select good values for parameters.

The SMSS, while aiming to learn good middlemen strategies, is in practice refining a probability distribution that can be used to effectively instantiate SSF-I1 to work within the specified environment. In relation to the description above, the SMSS (F) is attempting to learn a good strategy (M) that constitutes a probability distribution

of values for the parameters specified in Table 5.4. The probability distribution provides the final output of an SMSS experiment, a mechanism for the instantiation of successful middleman strategies.

The process of refining the probability distribution is achieved through the repeated evaluation of the performance of middlemen strategies within SSCM-defined markets. These markets, the process of evaluation and its use in updating the probability distribution are discussed in more detail in the sections below. Section 6.3 provides details about the SMSS probability distribution and how it is used to instantiate (or generate) middleman strategies. The way in which the probability distribution is used is based upon Population Based Incremental Learning, its use here is briefly discussed in Section 6.5.4

6.2.2 What are we trying to learn from?

As discussed above the SMSS learns effective middlemen strategies, that are instantiations of SSF-I1 (see Section 5.4), by refining a probability distribution. The probability distribution ideally represents good sets of parameter values with which to instantiate SSF-I1, so fully defining a middleman strategy.

The mechanism that refines the probability distribution is important since its capability determines the effectiveness of the resultant middlemen strategies.

In order to achieve this refinement many possible middlemen strategies must be evaluated and used to provide feedback for learning. The evaluation of a set of middlemen strategies is accomplished by having them operate within an SSCM-defined situation, interacting with other supply chain participants while trying to make a

profit. The performance of individual strategies within the specified supply chain is then used as the basis for evaluation and learning.

The SSCM markets used for the evaluation of middlemen strategies within SMSS are constrained by the SSCM-V1 Scenario's 1 to 3. These scenarios place certain constraints both on the starting conditions of the market and also the behaviour of participants. In defining the markets middlemen strategies will be exposed to, the scope of possible SSCM-V1 instantiations and participant behaviour will be further restricted. This restriction constitutes the environment, E, to which good strategies, M, are to be found.

The restriction of possible SSCM-V1 instantiations comprises elements such as availability of products, customer budgets and the limitation of outbound communication.

Restrictions on participant behaviour are primarily in the form of how stubborn suppliers will be during negotiations and the delay customers exercise before requesting requirements. The parameters that control these restrictions are discussed in Section 5.4.4.

Further details about the instantiation of SSCM markets and participant restrictions are found in Section 6.4.

6.2.3 Reinforcement learning for SSCM

The aim of the SMSS is to learn good middlemen strategies for a specified market environment. To this end a reinforcement learning system based upon evolutionary computation is used. Evolutionary computation is appropriate for this task and has been used effectively in other strategy learning situations, [30]. Other potential learning mechanism are not considered here and are beyond the scope of this thesis.

The SMSS's reinforcement learning system makes use of a feedback loop, the mechanism that is used to generate the middlemen strategies ultimately being updated (and hopefully improved) by the evaluation of those strategies.

Within the SMSS, reinforcement learning is based upon Population Based Incremental Learning with Guided Mutation (PBIL+GM). This is an evolutionary approach to learning that incorporates learning rules similar to those of neural networks.

In PBIL+GM, and so in the SMSS, an iterative process of refinement is used to gradually learn good solutions, in this case middlemen strategies. An initial probability distribution is set up and used to generate a number of solutions. These solutions are then tested in some way. The evaluation of those solutions is used to reinforce the probability distribution so that those solutions that performed well are more likely to be generated in future. To this end the quality, or probability of the values of the parameters of a successful solutions are increased. Correspondingly, the probability of the values of parameters of solutions that performed badly may be decreased. Details of the specific probability distribution update mechanism used in this work are given in Section 6.5.1.

This process is repeated using the updated probability distribution as a starting point until some stop condition is reached.

The basic process is outlined in Algorithm 1 and shown for the SMSS in Figure 6.1.

Algorithm 1 Basic PBIL+GM/SMSS Reinforcement Learning Feedback Loop

1: Initialise Probability Distribution
2: **repeat**
3: Generate A Solution Set From The Probability Distribution
4: Test And Evaluate All Solutions
5: Decrease Chance Of Worse Solutions In Probability Distribution
6: Increase Chance Of Better Solutions In Probability Distribution
7: **until** Stop Condition Met
8: Report Final Probability Distribution

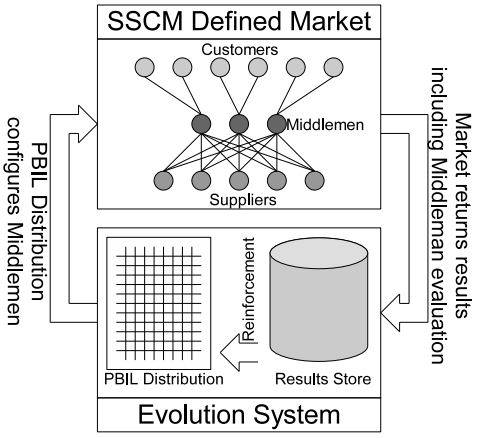

Figure 6.1: The Basic SMSS Feedback Loop

The probability distribution and the process of middleman strategy generation is discussed in more detail in Section 6.3. The running of experiments, operation of the market used to test strategies and the way in which performance is evaluated is discussed in Section 6.4. Section 6.5 discusses the way in which the probability distribution is updated based upon the evaluation.

6.3 Strategies Generation

This section describes how middlemen strategies are represented and generated for evaluation within the SMSS.

6.3.1 Middleman Strategy Representation

Middleman strategies are based upon SSF-I1 defined in Chapter 5. An individual middleman strategy is based upon this framework but its specific behaviour is determined according to how the the set of parameters in Table 5.4 are instantiated.

The SMSS maintains a probability distribution that represents the quality of possible values for each parameter. The quality of a value for a particular parameter is equated to the probability of its use in generated strategies. As such the probabilities of all values for a particular parameter sum to one. A greater probability indicates greater confidence that the value will lead to a more successful middleman strategy.

For parameters that have either symbolic (some fixed set of possibilities) or integral possible values the representation within the probability distribution is quite simple, this is shown in Figure 6.2.

Probability Distribution

	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value 9	Value 10
Variable A	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

	Value 1	Value 2	Value 3	Value 4	Value 5
Variable B	0.2	0.2	0.2	0.2	0.2

	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value 9	Value 10
Variable A'	0.125	0.25	0.5	0.25	0.125	0.125	0.325	0.125	0.125	0.125

	Value 1	Value 2	Value 3	Value 4	Value 5
Variable B'	0.1	0.7	0.1	0.75	0.25

Figure 6.2: Probability Distribution - Symbolic Representation

Parameters may have differing numbers of values and those values may be entirely different, the underlying representation is however always the same. When initilised the probabilities for values of a parameter are all equal. After learning has taken place the probabilities of the values for each parameter should reflect good choices in values for those parameters. In Figure 6.2 above variable A and B have different numbers of values and both start with an even set of probabilities for those values. After learning A' and B' show how the probabilities have converged on good values. For variable A' values 3 and 7 have acted as a focus, there need not be only one. For variable B' the focus is clearly on value 2 which has accrued 70% of the probability (quality) for that parameter. Note that in both the pre and post learning cases the probabilities for each value in a variable sum to one.

Parameters that require a continuous range of values are more difficult to represent than their symbolic counterparts. Within the SMSS's probability distribution they are represented by an upper and lower limit with the range between those limits being broken in to a number of blocks. Each block in the range has a corresponding probability associated with it. This probability represents the likelihood of a good value existing within that block. One block wraps around the range limits representing probability at both the top and bottom end of the range. The boundaries between blocks are able to shift up and down the range so increasing or decreasing the size of blocks they define and adjusting their positions. While doing this the boundaries may wrap across the limits and 'wrap around' to the other end of the range. The continuous range representation is demonstrated in Figure 6.3.

In Figure 6.3 variable C is shown to have a range divided in to ten blocks (A to J). In the initial position all of variable C's blocks are of even size, with even

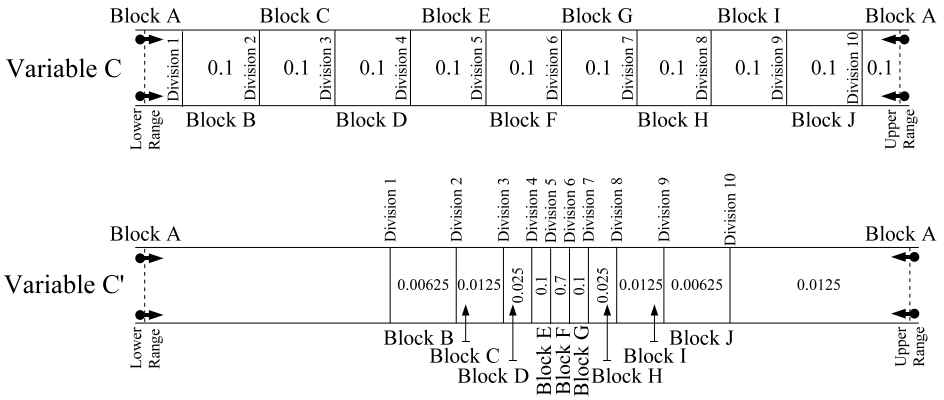

Figure 6.3: Probability Distribution - Continuous Representation

spacing of divisions, and have identical probabilities that sum to one. In the case of block A it acts to 'wrap' around the range limits and the probability for this block covers this wrapping case. This initial position looks similar to that shown for variable A in Figure 6.2. After learning variable C' looks considerably different from it's initial state. Block F has accrued most of the variables probability (70%) with surrounding blocks having acquired decreasing amounts the further they are from the block. Correspondingly the block divisions have converged toward a point in Block F concentrating the probability around that point further. This contracting around Block F has greatly expanded Block A which now covers about half the total range. The probabilities of all blocks still sum to one however. The concentration of blocks around a good point provides a way of apportioning probability more effectively to areas of the range space that appear to be more promising while the remaining space still has some probability assignment rather than being wholey ignored. To prevent excessive convergence to one point in the range space and arbitrary minimum block size may be set.

The parameters of SSF-I1 require a combination of both symbolic and continuous

Table 6.1: SSF-I1 Parameter Representation

Parameter Name	*Continuous (block)/Symbolic*	*Lower Range*	*Upper Range*
Product Information Window	Symbolic	1	50
Group Activation Time	Symbolic	1	30
Active Proportion	Continuous (30)	0.5	0.95
Group Duration	Symbolic	1	30
Negotiation Time-Out	Symbolic	0	10
Found Unavailable Threshold	Symbolic	1	30
For Each Product			
Guess Price	Continuous (50)	0.0	500.0
Upper Price Multiplier	Continuous (50)	1.0	10.0
Lower Price Multiplier	Continuous (30)	0.0	1.0
Tactic Value	Continuous (50)	0.1	10.0

representation, in the case of symbolic representation this relates to parameters that require only integer values rather than numbers in a continuous range. Along with symbolic and continuous representation a choice of upper and lower limits must be made and, in the case of continuous representation, number of blocks. Table 6.1 lists the parameters of SSF-I1 along with their corresponding representation and limits.

For the purposes of this representation parameters are essentially assumed to be independent. No information is gathered or recorded in the probability distribution about how different parameters and their values may relate to one another. Parameters within a middleman strategy are, however, likely to be dependant upon one another to some extent. This representation therefore presents a trade off between representation simplicity for the sake of expediency against attempting to capture more complex information of unknown utility. Provided the evaluation mechanism being used is effective a less complex representation should be sufficient.

6.3.2 Instantiating Strategies

Instantiating middlemen strategies is the first step in the process of refining the SMSS probability distribution during an experiment and is accomplished from the current state of the probability distribution.

The process of instantiation is relatively straight forward. As is described above, each parameter required for SSF-I1 is represented in the probability distribution by either a symbolic or continuous range of possible values.

In instantiating a middleman strategy a value is selected for each of those parameters from the probability distribution. For a discrete parameter this is simply a matter of selecting one of the possible value probabilistically from the available set. This is described in Algorithm 2 and shown in Figure 6.4.

Algorithm 2 Parameter Instantiation For Discrete Values

1: Generate Value p Between 0.0 And 1.0 With Even Probability
2: Value Count $vCount$ Set To Before Start
3: **repeat**
4: Increment $vCount$
5: Reduce p By Probability Of Value $vCount$
6: **until** $p \leq 0.0$
7: Value $vCount$ Selected

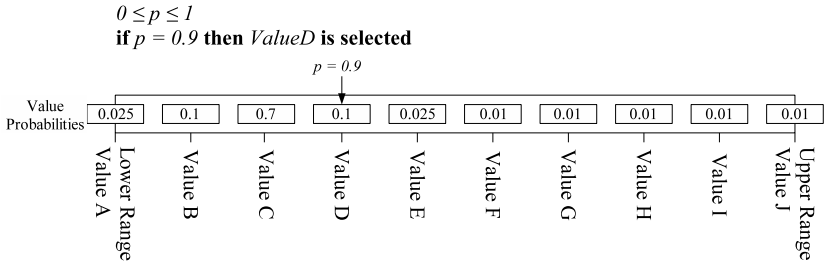

Figure 6.4: Value Selection For Discrete Values

For continuous parameters the process is slightly more complex. In this case one of

the available blocks is selected probabilistically in the same manner as a for a discrete parameter. To instantiate the parameter fully, a specific value is then chosen with even probability from across the range within the selected block (this also applies to the wrapping block case). This is described further in Algorithm 3 and shown in Figure 6.5.

Algorithm 3 Parameter Instantiation For Continuous Values

```
1: Generate Value p Between 0.0 And 1.0 With Even Probability
2: Block Count bCount Set To Before Start
3: repeat
4:    Increment bCount
5:    Reduce p By Probability Of Block bCount
6: until p ≤ 0.0
7: Generate Value vp Between 0.0 And 1.0 With Even Probability
8: Set Selected Value r To 0.0
9: if Block bCount Is Non-Wrapping Block then
10:   Get lowerDivision And higherDivision For Block bCount
11:   r = lowerDivision + ((higherDivision − lowerDivision) ∗ vp)
12: else
13:   Get lowestDivision And highestDivision
14:   lowerRange = lowestDivision − lowerLimit
15:   upperRange = upperLimit − highestDivision
16:   range = lowerRange + upperRange
17:   r = range ∗ vp
18:   if r > lowerRange then
19:      r = r − lowerRange
20:      r = r + highestDivision
21:   else
22:      r = r + lowerLimit
23:   end if
24: end if
25: Selected Point Is r
```

Having selecting a value for a symbolic or continuous range probabilistically a further process is a applied that may or may not adjust that value. This process

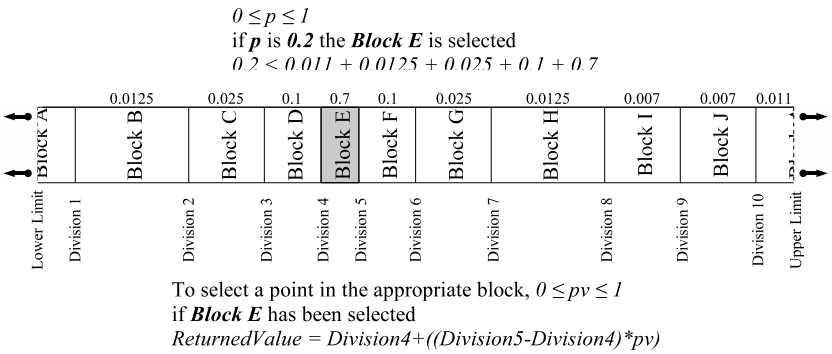

Figure 6.5: Value Selection For Continuous Values

is known as mutation. Two different mutation mechanisms are present within the SMSS.

The first mutation process is that of standard mutation which is applied with some arbitrary probability for all selected parameter values. As a result it may or may not occur in each case. The mutation process replaces the selected value with one from the range of possibilities but with equal probability. This mechanism is designed to allow a greater variety of strategies to be generated when the probability distribution has become converged on a specific solution and so helps search the space of possibilities more effectively. The $MutationRate$ is a value between 0 and 1 that represents the likelihood of a value being replaced. At 1 the value will always be replaced, at 0 it will never be replaced and will be replaced half the time with a value of 0.5.

The second mutation process that is applied is Guided Mutation, this is controlled by the $GuidedMutationRate$. Guided mutation operates in the same way as mutation but instead of replacing a value with one drawn from the set of all possible values for that parameter, it instead replaces the value with one from a known good solution. The process acts to focus the search more specifically on known values of good

Table 6.2: SMSS Strategy Generation Parameters

Parameter Name	*Usage*
Mutation Rate	The rate of mutation applied to PBIL generated strategy string. Evenly distributed values will replace a single strategy parameter at this rate
Guided Mutation Rate	The rate of guided mutation to use, applied as with normal mutation
Guided Mutation Target	Specifies if the best overall strategy so far, or the last best strategy should be used as the guided mutation target

solutions and may lead to more rapid convergence of the probability distribution.

The selection of a known good solution to act as the basis for guided mutation value replacement is discussed in Section 6.5.2. Table 6.2 lists the SMSS experimental parameters responsible for controlling mutation in the strategy generation process.

Having selected a value for each SSF-I1 parameter a middleman strategy is thus fully defined. Since the instantiation process is probabilistic, repeated use of this method is likely to produce different middlemen strategies each time it is used. As the probability distribution converges on a specific, effective pattern of instantiation, the possible variation (if any) will reduce. After instantiating some number of middlemen strategies for testing, the next step is to evaluate those strategies, this is discussed below.

6.4 Strategies Evaluation

This section describes how middlemen strategies are evaluated within the context of running an experiment with the SMSS.

6.4.1 Using A Market To Evaluate Strategies

Within the SMSS middlemen strategies are evaluated as part of an SSCM-defined market. Within this market customers attempt to have their requirements fulfilled, suppliers aim to sell the products they have available and middlemen try to bring the two of these together via an SSF-I1 strategy. The results of a market are recorded and used to evaluate the middlemen strategies taking part and update future strategy generation accordingly. This process constitutes the core operation of the SMSS.

In using an SSCM-defined market for the evaluation of strategies the operation of the market, the data recorded and how that data is used for the evaluation must be considered. Section 6.4.2 discusses the process of market generation in more detail.

The market simulation that is core to the SMSS comprises the aforemention customers, suppliers and middlemen participants rendered as autonomous software agents. These agents interact using the SSCM-V1 defined alternating offers based communication mechanism and are synchronised via a universal clock. The operation of a market begins by initialising the various agents strategies (this primarily refers to the middlemen). Next, the market to be used must be generated and the information relevant to each agent supplied. The market is then run for some period of time with a universal clock maintaining synchronisation in terms of SSCM time steps. Within an individual time step agents will operate as their strategies dictate, receiving inbound communications from others and sending communications up to their defined maximum limit. When the market's final time step has ended, data about each agents performance is output for evaluation.

Evaluating middlemen strategies in this way means that individual strategies will

Table 6.3: Middleman Data For Evaluation

Data Name	*Description*
Wealth/Net Worth	The net currency value of the middleman at markets end
Total Gains	The total amount of currency gained during the course of the market
Total Expenditure	The total amount of currency expended during the course of the market
Net Group Scores	The net value of all completed SSF-I1 basic groups. $NetGroupScore_1$ to $NetGroupScore_{total}$
Negotiation Activity	A record of the state of all negotiations the middlemen took part in at the close of the market. This includes accepted (us and them), rejected (us and them), implicit reject (us and them), new, new offer, new product set and waiting. The last four of these should not occur by the end of the market for any negotiation

not normally be assessed in isolation. Interaction between strategies may occur when the set of known suppliers overlaps (as it will do in SSCM-S1 and 2 at least) and the amount of available products is not large enough to fulfill all possible customer requirements. In this case middlemen must not only tackle the problem presented to them directly by the customers but also cope with potential competition from other middlemen.

While information about all participants is recorded, that relating to the middlemen is of greatest relevance. For evaluation purposes the data described in Table 6.3 is obtained for each middleman and may be made use of.

Since the generation of SSCM market is (ideally) non-deterministic and elements of competition exist within the market place, the evaluation of middlemen may be carried out over multiple markets and the results averaged to obtain a fairer picture

of performance. Thus the values used for evaluation will often be averages of those in Table 6.3 from a number of repeated markets.

Four different evaluation mechanisms are available to the SMSS to asses the effectiveness of middlemen strategies within a market. These are:

- $e1$ - Net Worth
- $e2$ - Expenditure/Income Weighting
- $e3$ - Positive Group Skew
- $e4$ - Negotiation State Scoring

Each of the above is calculated for each middlemen present in the market, multiplied by a weight and summed. So for a middlemen strategy s within the supplied market (or averaged markets) the score is calculated thus: $EvaluationScore(s) = e1_s * weight_{e1} + e2_s * weight_{e2} + e3_s * weight_{e3} + e4_s * weight_{e4}$

The weighting mechanism for each score allows some to be ignored or given greater emphasis if so desired.

The $NetWorth$ evaluation of a middleman is taken directly from the $NetWorth$ data for that middlemen in the market, it is the currency value accumulated by the agent by the end of the market taking in to account all income and outgoings. Along with providing a part of the evaluation mechanism $NetWorth$ also represents the middleman's performance in the market directly. Ideally a middleman will have a strongly positive net worth by the end of the market. This may not be possible if either the market has been unfavorable or the adopted strategy is not good. $e1_s =$

$NetWorth_s$

$Expenditure/IncomeWeighting$ is based on a rough break down of the middleman's $NetWorth$ being a record of the total income and expenditure. The calculation for this evaluation mechanism takes these two values, weights them then sums the result. This can, for instance, be used to reward any behaviour that accrues currency more highly than that which causes its loss. $e2_s = Expenditure_s * expenditure\_weight + Income_s * income\_weight$

$PositiveGroupSkew$ is based on the idea of rewarding groups that perform in a helpful way. An SSF strategy may form multiple groups in an attempt to fulfill customer requirements but not all of these may be successful. $PositiveGroupSkew$ is designed to reward those that are by increasing by a positive factor the value of any positive groups and summing these, negative group scores are then added to this directly.

$$e3_s = \sum_{i=1}^{NetGroupScores_{total_s}} \begin{array}{l} if NetGroupScore_{i_s} > 0, NetGroupScore_{i_s}^{positive\_group\_skew\_weight} \\ if NetGroupScore_{i_s} \leq 0, NetGroupScore_{i_s} \end{array}$$

$NegotiationStateScoring$ for a middleman is calculated from the number of negotiations in each state at the end of the market. Although ten negotiation states are possible (Accepted (us and them), Rejected (us and them), Implicit Reject (us and them), Waiting, New, New Price, New Offer) in practice only six of these should appear at the end of the market as all will have either been decided upon or have timed-out causing an implicit reject. $NegotiationStateScoring$ applies a weighting to the number of negotiations in each state and adds these together then weights the

entire sum

$$
\begin{aligned}
e4_s \;=\; & overall\_neg\_weight * (AcceptedUs_s * acceptedus\_weight \\
& + AcceptedThem_s * acceptedthem\_weight + RejectedUs_s * rejectedus\_weight \\
& + RejectedThem_s * rejectedthem\_weight \\
& + ImplicitlyRejectedUs_s * implicitlyrejectedus\_weight \\
& + ImplicitlyRejectedThem_s * implicitlyrejectedthem\_weight)
\end{aligned}
$$

The full list of evaluation control parameters is shown in Table 6.4.

While the $NetWorth$ of a middleman strategy provides a measure of its effectiveness within a market it is not necessarily ideal for learning. While $NetWorth$ reflects performance it is unable to reward positive behaviour that none the less is swamped by other negative behaviour. The use of the other evaluation mechanisms allows this to be accomplished. For instance, using $PositiveGroupSkew$ allows positive groups to strongly counter the effects of poor performance on the part of other groups. This allows the appearance of some positive behaviour to be rewarded and hopefully reinforced. Likewise $NegotiationStateScoring$ can reward the strategy for behaviour that doesn't lead to explicit rejects. A combination of the evaluation mechanisms allows a more complex evaluation to be performed even while the $NetWorth$ is retained as an overall measure of performance. Section 6.5 discusses how the evaluation of a middleman is used for the purposes of reinforcing the SMSS probability distribution.

Table 6.4: SMSS Evaluation Control Parameters

Parameter Name
Overall and Weights
$e1_s$ $weight_{e1}$ $e2_s$ $weight_{e2}$ $e3_s$ $weight_{e3}$ $e4_s$ $weight_{e4}$ *For e2*
expenditure_weight *income_weight* *For e3*
positive_group_skew_weight *For e4*
overall_neg_weight *acceptedus_weight* *acceptedthem_weight* *rejectedus_weight* *rejectedthem_weight* *implicitlyrejectedus_weight* *implicitlyrejectedthem_weight*

6.4.2 Defining The Scope Of The market

In the discussion of what the SMSS is aiming to achieve it was stated that the SMSS represents F in the function $F(E) \longrightarrow M$ where E is the environment under study and M is the resultant middleman strategy. This subsection concerns the definition and use of that environment E.

As stated, the SMSS operates through a feedback loop process in which multiple middlemen strategies are repeatedly exposed to SSCM-defined markets for evaluation. The evaluation of middlemen ultimately affecting what strategies are subsequently presented for evaluation. Within this process it is important to realise that the markets to which middlemen strategies are exposed are not fixed. Instead the markets vary within some range of possibilities. Further, while the SSCM-defined market largely constitutes the environment the middlemen strategies are exposed to, other factors also play an important part. Variation in customer and supplier control parameters must also be specified in line with their definition in SSF-I1 (see Section 5.4.4) and these will affect the market outcomes. We first consider the instantiation of SSCM markets and then discuss how specification of customer and supplier parameters may adjustment the behaviour of customers and suppliers and lead to a full definition of the environment.

In defining an SSCM market, the range of possibilities considered is not the total range of possibilities covered by the SSCM but is instead constrained by the Scenarios defined in Chapter 4. The range of possibilities within this constraints set markets is still large however. For the purposes of investigating more specific situations through experimentation further restrictions are made.

Within the SMSS the Model Generator component is responsible for the instantiation of valid SSCM markets within a set of boundary conditions for experimentation. Table 6.5 specifies the Model Generator parameters. These parameters control the range of possible SSCM markets that are used by the SMSS for partially specifying environments for experimentation and thus for evaluating individual middleman strategies.

The Model Generator uses these parameters to guide a random process of SSCM instantiation.

The process of instantiation begins by defining the total and active time of the market from within the boundaries specified by the parameters. The product set is defined directly from the specified parameters before suppliers are instantiated. Supplier instantiation is primarily a matter of selecting the product base cost and quantity available from within the bounds specified. The number of middlemen in the system is specified by the particular experiment and is independent of the actual SSCM definition. Middlemen within the instantiated SSCM are created to match the number existing in the experiment and are informed of the suppliers. A number of customers (from within the set bounds) is the generated for each middleman. SSCM Customer instantiation is somewhat more complex. Initially a earliest start and latest end point are selected for each customer. A minimum and maximum duration is selected within the specified boundaries and checks ensure that these are viable within the earliest start and latest end times for the customer. The minimum notice period is determined and checks ensure this will allow at least one time step for negotiation (even if this would prove impractical). The minimum number of entertainments is then determined. Checks ensure that the minimum entertainment allocation will not

Table 6.5: Model Generator Parameters

Parameter Name	*Usage*
Total Time	Min and max values for the total time of the SSCM market (T_{total})
Active Time	Min and max values for the active time of the SSCM market (T_{active})
Customer Per Middleman	Min and max values for the number of customers for each middleman in the model
Earliest Start	Min and max values for the earliest starts possible within the active period for viable customer requirements
Latest Return Offset	Min and max values for the latest return times possible for customer requirements
Shortest Duration	Min and max values for the shortest duration customer acceptable requirements
Longest Duration	Min and max values for the longest duration customer acceptable requirements
Entertainment	Min and max values for the amount of entertainment expected by customers
Notice Period	Min and max values for the amount of notice (prior to the requirement) needed by the customer
Communication Budget	The min and max values for customer, supplier and middleman outbound communication budgets
Number Of Products	The number of products (at least 3 in SSCM-S1-3)
For Each Product Type	
Product Name	The products name (these are determined by the scenarios in Chapter 4
Base Cost	Min and max values for supplier product base costs
Customer Budget	Min and max values for the customer budget for this type of product
Availability	Min and max values for the amount of this product supplied by a supplier

be greater than the duration of the trip. The budget of each customer is calculated by first determining a mean trip. One with the mean duration and the requisite number of entertainments for that duration (this may be less than the minimum), the amount of required accommodation, inbound and outbound flight and number of entertainments can then be multiplied by the minimum budget allocation for each, added to this is a pre determined proportion of the range between this minimum and the maximum budget allocation and all are finally added together. $CustomerBudget_{m_i}$ Budget for customer i of middleman m.

$0 \leq CustomerWealth_{m_i} \leq 1$ The wealth proportion for the customer.

$$CustomerBudget_{m_i} = \sum_{p=1}^{pn} (CustomerBudgetMin_p + (CustomerBudgetMax_p - CustomerBudgetMin_p) * CustomerWealth_{m_i})$$

Each customer is informed of the middleman for which it was generated. Having generated all participants and their links to each other the maximum communication budget per time step is calculated for each. This process forms the SSCM model for an individual market used for testing middlemen strategies, the relevant parts of which are communicated to the correct parties. In the case of middlemen, knowledge of which suppliers supply which products may also be communicated (for expediency) although is not necessary. With the SSCM market generated and sent all customers are aware of their requirements, all suppliers of their products, cost and availability and middlemen know of the suppliers.

Table 6.6: Customer Parameters

Parameter Name	*Usage*
Minimum Delay	How soon from the beginning of a market should a customer delay making a request (minimum)
Maximum Delay	How soon from the beginning of a market should a customer delay making a request (maximum)

Table 6.7: Supplier Parameters

Parameter Name	*Usage*
Price Upper Multiplier ($MaxMultiplier$)	Used to determine the maximum unit price of a product for sale. Multiplies the product base value supplied by the SSCM
Tactic Value ($TacticValue$)	Value that controls the curve of the offered price over time.
Negotiation Time-Out Delay	How soon a supplier expects a reply from the middleman

For the market environment to be fully defined the customer and supplier behaviour must be further specified, this is achieved via a small set of controlling parameters. These parameters are shown in Table 6.6 and Table 6.7.

The customer minimum and maximum delay time are used to calculate a delay time for each customer in the market. Instead of each customer immediately requesting their requirements at the beginning of the market the delay time subtracts from the customer minimum notice period to determine when the request should be sent. This mechanism helps to ensure that middlemen have a stream of new requirements for varying future times as well as helping to ensure that some amount of time is available for negotiation.

The supplier parameters relate to their tactics in negotiation. The $PriceUpperMultiplier$ acts to provide an upper limit to the amount that suppliers

will request for products. The $TacticValue$ controls the rate at which suppliers are willing to give way to middlemen and lower their price from the maximum. The $NegotiationTime - OutDelay$ specifies how long suppliers are willing to wait for a response before considering a negotiation terminated by implicit reject. Increasing the $PriceUpperMultiplier$ acts to make suppliers start from a higher price point in negotiations making products potentially more expensive. Likewise, decreasing $TacticValue$ to below 1.0 makes the suppliers negotiating behaviour more stubborn meaning they will give ground on price less easily over time. A $TacticValue$ above 1.0 indicates suppliers that are willing to give ground on price more easily potentially leading to lower product prices for the middlemen.

The specification of the SMSS Model Generator parameters along with those for customer and supplier behaviour is sufficient to fully define the environment E.

The use of these parameters allows for the definition of a wide range of possible environments within the context of SSCM-S1 to 3. In a simple case the difficultly with which middlemen may come to agreements with suppliers can be adjusted through the supplier $TacticValue$ or $PriceUpperMultiplier$. In a tough market the suppliers will only give ground slowly forcing the price middlemen must pay for products up. Increasing the level of competition between middlemen is possible by reducing the availability of products. The number of customers per middleman maybe adjusted to affect how much work the middlemen must do per time step and how wisely the communication budget must be used as a result. The adjustment of customer budgets and supplier costs directly effects the potential for profitability within the market.

6.4.2.1 Measuring Markets

Section 4.5 introduces some ways in which SSCM markets may be characterised. Unfortunately, when taking in to account the behaviour of customers, suppliers and the SSF, these measures fail to fully capture the situation and can prove misleading. In particular, the profitability gap determined by the Mean Customer Balance fails to take into account the difficulty of obtaining products at a reasonable price. In order to fully understand theses shortcomings it is first necessary to consider the problem faced by a middleman within an SMSS market simulation.

The total budget available to the middleman for negotiation is determined by the customer requirements it is currently dealing with. More specifically, the active group currently being dealt with has its maximum possible negotiation budget determined by its set of customer requirements, thus all of the required products must be bought using the same budget pool for the group to be successful. The amount of time available for negotiation is determine by the structure and instantiation of the SSF-I1 based middleman strategy. When this time begins is determined by the earliest customer negotiation time-out in the group under consideration. From the SSF description it can be seen that the amount of time available for negotiation found thus:

Definition 6.4.1. SMSS, Middlemen time to negotiate

$$TimeToNegotiate \quad = \quad GroupActivationTime - TimeToReply$$

$$
\begin{aligned}
TimeToReply \;=\; & \textbf{if} \\
& GroupActivationTime * (1.0 - GroupActiveProportion) \geq 1.0 \\
& \textbf{then } GroupActivationTime * (1.0 - GroupActiveProportion) \\
& \textbf{else } 1.0
\end{aligned}
$$

GroupActivationTime -	SSF-I1 Middleman parameter, see Section 5.4
GroupActiveProportion -	SSF-I1 Middleman parameter, see Section 5.4

A group becomes active *GroupActivationTime* before the time-out of the earliest customer negotiation that it contains. It then has *TimeToNegotiate* time-units to negotiate with suppliers to obtain the required products before entering a completion phase in which customer replies will be issued. Given the SSF-I1 middleman negotiation mechanism (see Section 5.4.1.4) this means the highest acceptable price to the middleman will be paid at the end of the available negotiation time, just prior to entering the completion phase of the group.

Suppliers consider the end point of a negotiation to be just prior to the earliest required product in that negotiation. Given the supplier negotiation mechanism (see Section 5.4.3 this means the lowest price will only be offered just prior to the earliest product required. The time-outs issued with customer negotiations ensure that there is a notice period between the completion of the negotiation and the earliest required product. The parameters controlling these effects are described in Section 5.4.4. Middlemen, to ensure there is sufficient time to respond, begin replying to customers sometime before the negotiation Time-Outs take effect (this is the *TimeToReply* shown above).

Given the customer, middleman and supplier behaviour prior to and during negotiation, it can be seen that the minimum price of a product will never be achieved by

the middleman. The less time available for negotiation the worse a middleman will tend to do against the suppliers. The instantiation of the middleman SSF-I1 strategy may exacerbate the problem, leading to overestimation of the price that should be paid and a poor negotiation strategy. While evolutionary learning aims to avoid this, it may still prove a problem early on making it difficult for middlemen to leverage a profit at all.

To take account of these problems and provide effective measures of the SSCM markets middlemen are exposed two new measure are proposed. The Adjusted Mean Trip Cost ($AMTC$) complements the earlier measures by determining the mean trip price for a given environment taking in to account the way in which the market participants operate. $AMTC$ determines the likely cost of securing each product type given the full set of parameters that affect the negotiating decisions. A companion value to the $AMTC$ is the Mean Adjusted Agreement Time ($MAAT$). This value is the mean of the times at which negotiations forming the $AMTC$ conclude (assuming a common start time at group activation). $MAAT$ is intended to provide an indication of how market conditions may affect agreement times. $MAAT$ is measured in SSCM time-step units and is considered infinite if agreement for all elements of the mean trip is impossible.

The $AMTC$ is calculated by determining the likely cost of every trip for the set of trips encompassed by the customer minimum and maximum allowed trip duration. Entertainments are assumed to fulfill the minimum entertainment requirement except where duration is too short. The likely price of a unit of accommodation and outbound flights is calculated by determining the approximate intersection of the suppliers and middleman's negotiation curve. For the suppliers these curves are based on

the base cost, tactic value and price upper multiplier specified in Table 6.7. For the middleman the curve is based on a combination of the mean customer notice period (Table 6.5), mean delay time (Table 6.6) and an estimation of the likely SSF parameters. For each relevant SSF-I1 parameter the a value is taken based upon those in Table 6.1. Those relevant are the Group Activation Time, Active Proportion and, for each product, the Guess Price, Upper and Lower Price Multipliers and Tactic Value. The Group Activation Time and Active Proportion combine with the customer delay time and minimum notice to provide an estimate of the amount of time available to the middleman for negotiation. The Guess Price, Upper and Lower Multipliers and Tactic Control Value control the negotiation curve.

The return flight price calculation is affected by trip duration since the end of that negotiation from the supplier perspective is pushed back and as such will be different for each trip. For entertainments the exact position within the trip is not specified so a point central in the trip duration is selected with potentially similar results. This effectively increases the likely cost of return flights and entertainments to the middleman. A summation of likely product costs with the amounts required yields a estimated trip cost, these costs can then be averaged for each duration. This provides the $AMTC$. The $MAAT$ is the mean of the negotiation agreement times across all durations and products.

6.5 Learning From Evaluation

Section 6.3 describes the way in which the SMSS probability distribution is used to represent the SSF-I1 parameters needed to define a middleman strategy and how these strategies are generated from that probability distribution. Section 6.4 describes the mechanism by which the generated strategies are evaluated within a market, how that market is defined, the resultant data and how this is used to provide an evaluation score for each strategy.

This section describes the way in which the SMSS probability distribution is updated in response to the evaluation of middlemen strategies and how this process, repeatedly applied, leads to the learning of effective middlemen strategies for the specified environment. The aim of updating the probability distribution is to focus the search for good strategies on to areas that have shown some promise. By increasing the probability of values that have lead to successful strategies in the past we aim to discover more successful strategies in the future. Ultimately the probability distribution should be capable of reliably producing highly effective strategies for the specified environment.

6.5.1 Updating The Probability Distribution

The SMSS probability distribution may be updated in one of two way. The first is to increase the chance of successful strategies occurring in the future, positive reinforcement. The second is to discourage less successful strategies from occurring, negative reinforcement. The SMSS makes use of both positive and negative reinforcement for learning effective strategies. In using both approaches negative reinforcement is always applied first.

For both reinforcement techniques the first step is the selection of the strategy to be used as the basis for updating the probability distribution. Within the SMSS this selection process is achieved through the use of the market simulation. This mechanism allows the SMSS to distinguish effective middlemen strategies (which should received high evaluation scores) from their less effective counterparts (with lower scores). The market simulation allows the evaluation of multiple middlemen strategies simultaneously. Several of these could be used for reinforcement of the probability distribution but in practice only the best and worst are selected. The process of updating the probability distribution occurs individually for each selected strategy. As noted earlier, a number of market evaluations may be used in combination (averaged) to provide a more accurate measure of strategy performance.

Having selected the strategy to use as the basis for learning, the next step is to determine by how much the probability distribution should be adjusted. That is, what Learning Rate (lr) should be used. The learning rate is a value between 0.0 and 1.0 that determines by how much the probability distribution will be adjusted in response to the strategy being learnt from. For positive reinforcement, a learning rate of 1.0 indicates that probability distribution should precisely match the supplied strategy. In this case the probability of all values for all parameters that are not part of the strategy would be set to 0.0, those that are part of the strategy would be given a probability of 1.0. This fully converged probability distribution would therefore only produce the selected strategy in the future, except where mutation caused some deviation. A learning rate of 0.0 would indicate that no learning should take place leaving all probabilities of all values for all parameters as they were before reinforcement occurred. Negative reinforcement is more complex, redistributing some of the

probability associated with the selected strategy to other possibilities proportionate to their current probability. The amount redistributed is the proportion specified by the learning rate. For a learning rate of 1.0 all the probability associated with the strategy would be redistributed to other possibilities meaning those parameter values would no longer occur except under conditions of mutation. As for positive reinforcement a learning rate of 0.0 would leave the probability distribution unchanged. Setting the learning rate effectively is therefore an important part of learning mechanism. The process of calculating a learning rate based upon the evaluation of the selected strategy is discussed in Section 6.5.2 below.

Having selected a strategy and calculated an appropriate learning rate, the positive or negative reinforcement of the probability distribution is carried out. This is achieved on a parameter by parameter basis, using the specified learning rate and the associated strategy value for that parameter as a focus to adjust probabilities accordingly.

Positive reinforcement for symbolic (integral value) parameters is a matter of reducing the probabilities for all non focus values by the proportion represented by the learning rate and increasing the probability of the focus value directly by the learning rate. The update mechanism for non focus values is shown in Definition 6.5.1. The focus value update mechanism is shown in Definition 6.5.2. These mechanisms are applied to each value, v_n, in the probability distribution accordingly.

Definition 6.5.1. Update of probability v_n where value v_n is not the focus value (positive reinforcement)

$$probability'_{v_n} = probability_{v_n} * lr$$

Definition 6.5.2. Update of probability v_n where value v_n is the focus value (positive reinforcement)

$$probability'_{v_n} = (probability_{v_n} * lr) + lr$$

Positive reinforcement for continuous value parameters operates in part like that for symbolic parameters. While probability values are updated in the same way the divisions that delimit blocks also adjust in response to learning. The divisions for the block in which the focus value reside adjust towards the focus point reducing the size of the block (down to a lower block size limit) and so concentrate more closely on the area of interest. The adjustment of of the block divisions is by the proportion of the distance from the boundary to the focus point inside the block multiplied by the learning rate. So, for a higher learning rate, the focus towards the point of interest is increased and for lower rates it is less marked. For the lower division of a block ($lowdiv$) and a given focus within that block ($focus$) the lower division is increased by $(focus - lowdiv) * lr$. The upper division ($highdiv$) is decreased by $(highdiv - focus) * lr$.

Negative reinforcement operates in essentially the same way for both symbolic and continuous parameters. The focus value's probability is reduced to the proportion $1 - lr$, this amount being redistributed to all other values in the parameter proportional to their current probability. The same mechanism is used for both value symbols and continuous range block probabilities. For continuous value parameters the block divisions are not adjusted in response to the negative learning. Definition 6.5.3 shows how the amount to reduce the focus values probability by is calculated and thus the amount of probability that need be redistributed to the remaining symbols or blocks. Definition 6.5.4 shows the reduction of the focus values probability. Definition 6.5.5

shows the calculation of remaining (non focus value probability), this value being used in Definition 6.5.6 which shows the update of non-focus value probabilities.

Definition 6.5.3. Negative reinforcement, probability reduction and redistribution amount where v_{focus} is the focus value

$$redist = probability_{v_{focus}} * (1 - lr)$$

Definition 6.5.4. Negative reinforcement, adjusted probability of the focus value, v_{focus}

$$probability'_{v_{focus}} = probability_{v_{focus}} - redist$$

Definition 6.5.5. Negative reinforcement, the total non-focus value probability, $rest$

$$rest = \sum_{n=1}^{v_{total}} \begin{array}{ll} \textbf{if } v_n \neq v_{focus} & probability_{v_n} \\ \textbf{else} & 0 \end{array}$$

Definition 6.5.6. Negative reinforcement, adjusted probability values of non-focus values, v_n

$$probability'_{v_n} = probability_{v_n} + (probability_{v_n} * (rest/redist)$$

A strategy acting as a focus of positive reinforcement may also have an effect on the guided mutation mechanism that comprises part of the strategy generation process (Section 6.3.2. Guided Mutation makes use of a set of target values that may be applied to individual parameters of a new strategy if it is invoked. These target values are drawn from either the last best strategy or the overall highest scoring strategy according to the Guided Mutation Target parameter (see Table 6.2). Therefore a newly evaluated strategy selected for positive reinforcement will be used as the Guided Mutation target if either the parameter indicates the last best strategy should be used or if the strategy has yielded the highest score so far from the market simulations. The

two approaches offer different potential benefits, either directing the search towards a overall known good solution but one which may now be less effective in the current context or towards a strategy known to be effective in the current context but that may, overall, be less effective than some strategy instantiated in the past.

6.5.2 Making Use Of The Evaluation

Section 6.5.1, above, describes the mechanism by which the SMSS probability distribution is updated. This description does not however specify how the learning used by this mechanism is calculated. This section describes how appropriate learning rates are calculated based on the evaluation scores achieved by middlemen in the market simulation.

Within the SMSS the learning rates used are based upon the evaluation scores received by middlemen strategies participating in market simulations. The SMSS is able to take advantage of both positive and negative reinforcement learning. In making use of these learning mechanism, the best strategy from the recent market (or set of markets) is used for positive reinforcement while the worst is used for negative reinforcement. Negative reinforcement is applied before positive reinforcement in the SMSS.

The learning rate used for positive reinforcement is selected from between upper and lower boundaries (*min_positive_learning_rate* and *max_positive_learning_rate*) specified for the individual experiment. The precise learning rate is based upon the past performance of other strategies used for positive reinforcement, the best and worst scores for these having been recorded. A strategy with a score equal to or higher than the best historical score will use the maximum learning rate. A strategy

with a score equal or lower than the worst historical score will use the minimum learning rate. Scores that fall between these historical boundaries use a learning rate scaled linearly between *min_positive_learning_rate* and *max_positive_learning_rate* accordingly.

If *current_score* represents the score of the strategy currently being used for positive reinforcement, *lowest_best_score* represents the lowest score achieved by a strategy selected for positive reinforcement and *highest_best_score* the highest score achieved, the Algorithm 4 may be applied to find the positive learning rate to be used in this instance. If the score of the strategy under consideration is higher or lower than the historical minimum or maximum scores, these boundaries will be adjusted appropriately. This may mean that a strategy considered later will receive a greater or lesser calculated learning rate.

Algorithm 4 Find Positive Reinforcement Learning Rate

1: **if** $current\_score \geq highest\_best\_score$ **then**
2: **return** $max\_positive\_learning\_rate$
3: **else if** $current\_score \leq lowest\_best\_score$ **then**
4: **return** $min\_positive\_learning\_rate$
5: **else**
6: $learning\_gap = max\_positive\_learning\_rate - min\_positive\_learning\_rate$
7: $score\_gap = highest\_best\_score - lowest\_best\_score$
8: $score\_prop = (current\_score - lowest\_best\_score)/score\_gap$
9: $learning\_amount = learning\_gap * score\_prop$
10: **return** $(min\_positive\_learning\_rate + learning\_amount)$
11: **end if**

The learning rate used for negative reinforcement is calculated in the same way as that of positive reinforcement. The calculation makes use of its own set of minimum and maximum learning rates and an independent set of highest scoring and lowest

Table 6.8: SMSS Learning Rate Parameters

Parameter Name	*Usage*
Maximum Positive Learning Rate	The maximum positive learning rate that will be applied
Minimum Positive Learning Rate	The minimum positive learning rate the will be applied
Maximum Negative Learning Rate	The maximum negative learning rate that will be applied
Minimum Negative Learning Rate	The minimum negative learning rate the will be applied

scoring strategies used for negative reinforcement. In calculating the learning rate to be used, a strategy with a score greater than or equal to the historically highest score will yield the minimum negative reinforcement learning rate. Correspondingly, a score equal of less than the historical minimum will receive the maximum learning rate.

Scaling of the learning rate is applied in this way to allow greater or lesser learning to take place dependant upon the performance of strategies in the market. This allows some learning to take place even if the strategies have performed worse than might be expected from historical performance. Since the maximum learning rates will only be applied in situations where performance at least equal to the historically best (or worst) strategy score these will seldom be applied. Table 6.8 lists the SMSS experimental parameters that directly affect the learning mechanism.

6.5.3 Learning Through Repetition

Updating the probability distribution as the result of middleman strategy evaluation forms the basis of learning within the SMSS. However a single instance of updating the probability distribution is not sufficient on its own to yield an effective mechanism

Table 6.9: SMSS Experimental Parameters

Parameter Name	*Usage*
Iterations	The number of iterations of repeated markets that should be carried out to learn the middleman strategy
Repeats	The number of markets that should be run and averaged to determine an overall winning strategy out of all the middleman strategies provided

for instantiating middlemen strategies for the supplier environment. Instead learning takes place over many iterations of instantiating strategies, evaluating those strategies and using the evaluation to update the probability distribution. The repeated application of this process gradually directs a search for good strategies toward more profitable areas of the search space, providing focus upon potential solutions while allowing a reasonable degree of variation to be considered. Using a mechanisms of repetition also helps mitigate the potential for fluke results to upset the process.

Within the SMSS repetition is used in one of two ways. The first is the number of learning *iterations* applied, that is the number of times the generate, evaluate, reinforce process is repeated. The second is the *repeat*ed application of markets to strategies. The iterations essentially determines the length of an experiment, and thus the scope for exploration of strategies. The market repeats, provides a way of delivering a more effective evaluation of middleman performance by averaging their performance across different markets from the same environment.

6.5.4 Population Based Incremental Learning + Guided Mutation

Population Based Incremental Learning (PBIL, [9]), is a statistical approach to evolutionary learning. PBIL comprises a probability distribution that is used to instantiate solutions for evaluation. The evaluation of these solutions is used as the basis for reinforcing the probability distribution. The process is iterated and, over time, better solutions will emerge providing the evaluation mechanism is effective. As should be apparent from the descriptions provide in previous sections the learning mechanism employed by the SMSS is based on PBIL with the addition of a Guided Mutation operator (PBIL+GM). The PBIL+GM learning mechanism has been shown to be effective in other strategy learning domains ([29]), albeit ones of less complexity. Its use here is motivated by its ability to leverage small evaluated population sizes over repeated iterations to come up with good strategy solutions.

6.6 Conclusions

This chapter has introduced the SSCM Market Simulation System (SMSS), a mechanism that is able to evolve middleman strategy representations within a specified market environment.

The final product of the SMSS is a probability distribution that represents good value selections for the parameters of the SSF-I1 described in Chapter 5. Using this probability distribution as the basis for selecting SSF-I1 parameters should yield an effective middleman strategy for the specified environment. This is claim Three in Section 1.4 of the Introduction.

The SMSS specifies a market environment in terms of the SSCM (specifically

SSCM-V1) described in Chapter 4 along with additional customer and supplier details.

The SMSS is able to find good middlemen strategies through the use of evolutionary learning based on the Population Based Incremental Learning ([9]) approach. In this approach middlemen strategies are repeatedly evaluated against a market simulation allowing them to be selected as the basis for updates to the probability distribution.

Chapter 7

Experimentation

The previous chapter introduced the SMSS, a system for evolving SSF-I1 (see Chapter 5) based middlemen strategies in a variety of environments.

The environments considered by the SMSS are defined in terms of supply chain instances, constrained by SSCM-V1 (see Chapter 4), and parameters adjusting the behaviour of customers and suppliers.

This chapter discusses the aims and objectives of experimentation with the SMSS, the set of experiments proposed, how these relate to the stated objectives and how data recorded by the SMSS may be used to show these objectives have been met.

7.1 Aims And Objectives

The introduction to this thesis provides two overall objectives for experimentation under SMSS conditions. These were: First, establish that the evolution platform is able to evolve effective strategies and do so under a variety of conditions. Secondly, establish how the evolved strategies are affected by the environments in which they form and how they react when placed in a different environment.

Overall therefore, the aim is to demonstrate that the SMSS is an effective means for evolving middlemen strategies in a complex environment, that the SSF provides the necessary flexibility for this to be achieved and that the specific environmental conditions have an affect on the final outcome. This aim helps to reinforce the claims that the SMSS provides a useful evolutionary platform for the investigation of complex environments, that the SSF aids this process by providing a feasible strategy framework and that the SSCM is able to capture environments of some interest in which variation in strategy may be important.

To be more specific about these aims and to help guide experimentation, we wish to show the following.

1. The SMSS is able learn SSF-I1 based middlemen strategy.
2. The learnt middlemen strategies are effective, providing middlemen reasonable performance within the supply chain.
3. The SMSS is able to learn effective strategies under a variety of SSCM-V1 conditions.
4. Strategies within a given environment will tend to converge on a solution.
5. Strategies are specific to their environment. Strategies evolved under a set of conditions, E_1, will do less well in an environment, E_2, when compared with strategies evolved under E_2.

Objectives one and two reinforce the claim that the SMSS is useful for the evolution of middlemen strategies in complex environments. This can be demonstrated

for a given experiment by observing a drop in the entropy of the SMSS's probability distribution coupled with an increase in the mean wealth achieved by middlemen within the supply chain, see Section 7.2.

Objective three further demonstrates the usefulness of the SMSS for evolving SSF-I1 based strategies since the ability to do so over a wider range of environments is preferable. Further, it demonstrates that SSF-I1 based middlemen strategies can adapt to these environments.

Objectives four and five seek to provide additional evidence for the flexibility of SSF-I1 in representing middleman strategies. This is achieved by demonstrating that SSF-I1 based strategies are able to adapted specifically to the environments in which they evolved. Objective four shows this through a combination of measuring the similarity of strategies and testing the significance of differences between evolved strategies. Objective five directly compares strategies evolved from one environment under conditions of another and makes use of similar measure to those used for objective two.

Section 7.2, below, describes the techniques used to interpret the SMSS experimental results for the purposes of demonstrating these objectives have been met.

7.2 Data Analysis Techniques For The Demonstration Of The Experimental Objective

The experimental objectives described in Section 7.1, above, must be supported by the analysis and interpretation of results data generated by the SMSS. This section describes the techniques used for this purpose and how these relate to demonstrating that the stated objectives have been met.

Data generated by the SMSS falls in to two categories, PBIL data and the market data.

PBIL data relates to the probability distribution used by the SMSS for the generation of SSF-I1 base middlemen strategies. The final outcome of any experiment being a probability distribution that should be capable of instantiating effective middlemen strategies for the environment presented.

Market data is all other data relating to the experiment and the performance of participants within the example supply chains used for strategy evaluation. This includes participant wealth and the number of different negotiation outcomes.

To help demonstrate that the stated objectives have been met a combination of both information types is used.

7.2.1 Strategy Definitions, PBIL Based Information

This section discusses the PBIL data recorded by the SMSS and how it can be used. This data is in the form of the probability distribution used by the SMSS to instantiate SSF-I1 based middlemen strategies.

The PBIL data primarily relates to understanding the strategies evolved by the SMSS but also contributes information about the overall learning process.

PBIL data is recorded for each iteration of an experiment after changes due to reinforcement learning have been made to the probability distribution. The final set of PBIL data forms the primary output of the SMSS, a probability distribution that should lead to effective middlemen strategy instantiations for the given environment.

Section 7.2.1.1 shows how the entropy of the probability distribution can be used to

help demonstrate learning. Section 7.2.1.2 describes a method for measuring the similarity between strategies, the Sum Of Probability Differences (SOPD). Section 7.2.1.3 discusses the mean probability of parameter values as a means for investigating how strategies are instantiated. Finally, Section 7.2.1.4 discusses the use of significant difference testing in determining how strategies vary between environments.

7.2.1.1 Demonstrating Learning, Entropy

The entropy of the probability distribution is important for showing that learning is occurring in the system.

The entropy is a measure of the information content of a system. The greater the entropy the larger the amount of information present. Greater entropy implies greater disorder within a system, thus a chaotic, random system will have high entropy and a fixed, static one will have a low (or no) entropy. For the SMSS's probability distribution maximum entropy occurs when the system is initialised and all the probabilities for all the groups of values of each variables are even. Entropy is at maximum as any set of values is as likely as any other when instantiating a strategy. During the course of an SMSS experiment learning will adjust some of the value probabilities, focusing more importance on some areas of the search space than others. This reduces the entropy of the probability distribution as strategy instantiation becomes increasingly ordered.

A drop in the probability distribution's entropy therefore shows that something is being learnt. The reduction in entropy on its own however only shows that something is being learnt, not that what is being learnt is useful.

While the probability distribution as a whole can be considered, the entropy of

individual parameters may also be examined. Looking at the entropy of parameters is useful for understanding where learning is occurring fastest. This may therefore show which parameters are most important to get right for a given environment.

7.2.1.2 Similarity Between Strategies, The Sum Of Probability Differences

The sum of probability differences (SOPD) is used as a measure of convergence between two different probability distributions. SOPD may be considered either for the distributions as a whole or individually between matched parameters.

SOPD is a measure of how similar the distribution of probability is between two probability distributions. A score of zero indicates that there are no differences, while a score of two indicates that the two probability distributions are completely different.

SOPD is calculated separately for each matching parameter in the distributions, the mean of all matched parameters being found to obtain the score for the probability distributions as a whole.

For a detailed description of how SOPD is calculated see Section C.1 of the Appendix C.

SOPD is used in two ways. Firstly, it allows the comparison of variability between strategies evolved within the same environment. The mean and standard deviation of the SOPD score is found as an indicator of how closely converged strategies have become.

Secondly, it allows the direct comparison of strategies evolved under different environments to determine by how much these vary. Again, by making multiple

comparisons, a mean and standard deviation of the variability between strategies evolved under the different environments can be produced.

While SOPD has been used for the analysis of experimental results in this thesis an alternative approach would be the use of the Kullback-Leibler (KL) divergence, [22, 56, 57]. In a similar manner to SOPD, the KL divergence measures the distance from one probability distribution to another. The introduction of a *log* term helps to discriminate between the relative size of differences between variable probabilities. In using KL divergence for the continuous range variables considered here a similar range break up method as used for SOPD would need to be applied - this method is described in Appendix C.

7.2.1.3 Similarity Between Strategies, Mean Probability Value

The probability distributions produced by the SMSS represent a way to instantiate the parameters of an SSF-I1 based middleman strategy. To gain a further perspective on the strategies produced by the SMSS, the mean probability value of each parameter is calculated.

The mean probability value is calculated for each parameter individually. This value represents the point within the represented value range at which half the probability occurs, i.e. the total amount of probability contained within the range either side of this point is equal. This point provides an indication of where most of the probability is focused and therefore what values are likely to be produced for the parameter.

For discrete value parameters, the mean will correspond to one of the possible discrete values. For continuous value parameters the mean will correspond to the

mid-point in the range of one of the parameters dividing blocks. See Section 6.3.1 for details of the probability distribution's representation.

Determining the parameter values produced as a result of a probability distribution is important since it allows consideration to be given to how the instantiated middleman strategy behaves in a supply chain.

7.2.1.4 Significance Testing

To demonstrate that strategies evolve differently under different environments, a Students t-test is used to verify the significance of the variation in parameter value means. Probable parameter values are calculated using the mechanism shown above in Section 7.2.1.3.

If the mean value of a parameter varies significantly between strategies evolved under two different environments it is reasonable to conclude that evolution in those environment has led to varying strategies.

The Student t-test, [20], requires that the values under test are not strictly bound within a range. Parameter mean information is bounded by the range of the parameters, to overcome this and make use of the t-test, parameter means are first normalised (by dividing by their upper range values) and then have arcsin applied. This provides the t-test values with which it can more effectively work.

7.2.2 Market Based Information

During the course of an experiment each participant returns information about their performance in the simulated markets. This information is important as it shows how middlemen are performing in the supply chain and what the consequences of learning are on that performance.

Information obtained from the SMSS's probability distributions is used to understand the middlemen strategies that evolve as a consequence of the presented environment, see Section 7.2.1. The market information is used to compliment this analysis and show how the performance of those strategies has improved over time.

To achieve this, market information is combined with information about probability distribution entropy over time to demonstrate how increased strategy performance is associated with reduced entropy, so implying that useful learning has occurred.

7.2.2.1 Market Scores, Mean, Median, Upper And Lower Values

The most important piece of information returned by the SMSS market is that of the middleman scores. These are averaged across each iteration's repeats to provide a better picture of how the middlemen are performing.

While entropy may drop whether anything useful is being learnt or not, the middleman scores indicate what is happening in the markets themselves. For a market in which something useful is being learnt by the middlemen a drop in entropy should be accompanied by a rise in the mean middlemen scores. The highest and lowest scores for a market can provide an indication of the stability of strategies in the environment in which they reside.

Two scores are of particular importance. Mean middleman wealth indicates the amount of 'money' middlemen are able to make within the market simulation. The amount of wealth middlemen are able to accrue is a combination of the difficulty of the environment in which they are situated and the effectiveness of their strategy. Mean middlemen wealth is therefore useful as an indicator of strategy performance, more effective strategies being those able to leverage more wealth from the given

environment. The mean middleman score is the value used to assess a middleman's performance in the market for the purposes of evolution. This score is a combination of factors (including wealth) that determines which middleman is strategy is used as the basis of learning on each experimental iteration. This score is discussed in more detail in Section 6.4.1.

7.3 The Experiments

This section discusses a series of SMSS experiments designed to meet the objectives introduced in Section 7.1.

For each experiment run with the SMSS a number of parameters must be specified. These parameters control both the evolutionary learning process used to generate middlemen strategies and the supply chain environment those strategies will be exposed to.

The same set of parameters that control the SMSS's evolutionary process are used consistently for all experiments introduced here. These parameters are discussed in Section 7.3.1, below.

The parameters that define the supply chain environment in which middlemen strategies are evolved are discussed in Section 7.3.2. These parameters vary between experiments to provide different evolutionary environments.

To provide a clearer picture of performance, ten experiments of each type are run. The resultant probability distributions can then be compared to determine the similarity of strategies within an environment, the resultant market information is averaged.

Except for some elements of the baseline experiments described in Section 7.3.3, experiments were run under SSCM-V1 Scenario Three restrictions as discussed in Section 4.4.

7.3.1 Basic Experimental Parameters

Each experiment conducted using the SMSS requires a number of controlling parameters to be specified. These parameters affect the way in which evolutionary learning takes place in the SMSS. Table 7.1 lists the default set of parameters values used by the SMSS for experiments run during the course of this research.

The number of Iterations (default *7500*) is the number of individual learning steps that take place during an experiment. The selection of the number of Iterations to use is based on the observation that middlemen performance tends to have stabilised by this point.

The number of Repeats selected (default *5*) was found to provide a reasonable indication of relative performance of middleman during one Iteration, see Section C.3 of the Appendix C. The performance of an individual strategy is averaged over the set of repeats within one iteration.

The selection of the Guided Mutation rate is intended to mean the likelihood of the operator being used once per iteration. The Target is selected as the most recent best to prevent a past high scoring strategy from acting as a focal point despite what may have been favorable conditions in terms of competing strategies. The Mutation rate is set arbitrarily low to cause some minor variation. An explanation of the guided mutation rate and target can be found in Section 6.3.2.

The evaluation parameters are chosen such that positive group skew and negotiation based scoring are used for evaluation process. Since the other mechanisms are unused (weighted *0*) their parameters are not specified. The positive skew value is set to *2.0* to provide a strong incentive toward positive performance and one that should mitigate the bad performance of some groups. The use of negotiation based scoring helps to improve middleman strategy performance over positive skew alone (see Section C.2 of the Appendix). The weighting of the negotiation based scoring is set up to encourage the acceptance of negotiations and marginally penalise any negotiations failing to be responded to leading to implicit rejects.

The learning rates used by the PBIL update mechanism are scaled between the upper and lower values specified according to the strategy evaluation scores (see Section 6.5.2). The upper learning rate can afford to be so high since a reinforcement will seldom achieve this value. Asymmetric positive and negative learning rates are used since equal rates tend to prevent convergence on a solution.

7.3.2 Experiment List

This section provides a list of numbered experiments and discusses how they relate to the experimental objectives.

Experiment One (CTRL1-3) is intended to provide a baseline against which variation of the environment defining parameters may be judged. In terms of the experimental objectives discusses in Section 7.1, this set of experiments is intended to provide evidence that the SMSS is able to evolve reasonable middlemen strategies, objectives one and two. The experiment comprises three parts, one for each of the SSCM Scenarios discussed in Section 4.4. These experiments are intended to show

Table 7.1: Experiment Parameters

Parameter	*Value*
Overall Control Parameters	
Repetitions of each experiment	10
Repetition Parameters (Table 6.9)	
Iterations	7500
Repeats	5
Strategy Generation Parameters (Table 6.2)	
Mutation Rate	0.001 (0.1%)
Guided Mutation Rate	0.025 (2.5%)
Guided Mutation Target	Last Iteration Winner
Evaluation Parameters (Table 6.4)	
$weight_{e1}$	0.0
$weight_{e2}$	0.0
$weight_{e3}$	1.0
$weight_{e4}$	1.0
positive_group_skew_weight	2.0
overall_neg_weight	10.0
acceptedus_weight	1.0
acceptedthem_weight	1.0
rejectedus_weight	0.0
rejectedthem_weight	0.0
implicitlyrejectedus_weight	-0.001
implicitlyrejectedthem_weight	0.0
Learning System Parameters (Table 6.8)	
Maximum Positive Learning Rate	0.01 (1%)
Minimum Positive Learning Rate	0.0001 (0.01%)
Maximum Negative Learning Rate	0.001 (0.1%)
Minimum Negative Learning Rate	0.00001 (0.001%)

Table 7.2: Experiment Reference List. For referenced objectives see List 7.1

Num	*Name*	*Objectives*	*Usage*
1_{a-c}	CTRL1-3	1, 2	Control, establishes a baseline.
2	LOWBUDGET	3 & 4	Decreased customer budgets.
3	LOWAVAIL	3 & 4	Decreased product availability.
4	STUBBORN	3 & 4	Increased supplier negotiation stubbornness.
5	LOWCOMA	3 & 4	Reduced communication (asymmetric).
6	LOWCOMS	3 & 4	Reduced communication (symmetric).
7	CTwithLCA	5	CTRL-3 environment with LOWCOMA evolved middlemen strategies. No learning.
8	LCAwithCT	5	LOWCOMA environment with CTRL-3 evolved middlemen strategies. No learning.
9	LAwithLCA	5	LOWAVAIL environment with LOWCOMA evolved middlemen strategies. No learning.
10	LCAwithLA	5	LOWCOMA environment with LOWAVAIL evolved middlemen strategies. No learning.
11	STwithLCA	5	STUBBORN environment with LOWCOMA evolved middlemen strategies. No learning.
12	LCAwithST	5	LOWCOMA environment with STUBBORN evolved middlemen strategies. No learning.

how the different Scenario conditions affect the performance and evolution of middlemen strategies. The Scenario Three component of this experiment (1_c, CTRL-3) provides the baseline against which all other experiments are compared - the variation in performance and effects on learning being noted and considered. The characteristics of this market are discussed in Section 7.3.3.

Experiments Two to Five (LOWBUDGET, LOWAVAIL, STUBBORN and LOWCOMA) provide a variety of environments that deviate from the baseline case, CTRL-3. Each of these experiments increases the difficulty of the environment faced by middlemen strategies in some way and are described in Section 7.3.4. These experiments demonstrate that objectives three and four are met, that the SMSS is able to evolve middlemen strategies for a variety of environments and that strategies evolved within different environments are different. Experiment Two (LOWBUDGET) adjusts the funds available to customers and so the potential profit margins available for middlemen to exploit. Experiment Three (LOWAVAIL) varies the amount of products available from suppliers, so increasing competition between middlemen. Experiment Four (STUBBORN) adjusts the negotiating behavior of suppliers making it potentially more costly for middlemen to obtain the required products and again effecting their profit margins. Experiment Five (LOWCOMA) reduces the number of outbound communication utterances middlemen are able to make per time-step, this aims to increase the pressure for communication to be used more carefully.

Experiment Six (LOWCOMS), like experiments Two to Five, aims to increase the difficulty of the environment faced by middlemen strategies. This is achieved by symmetrically reducing the outbound communication budgets of all participants as opposed to the asymmetric reduction of LOWCOMA conditions. While providing

additional evidence for objectives three and four, this experiment is intended to put the results of the LOWCOMA environment in context. This experiment is described in more detail in Section 7.3.5, its use for putting the LOWCOMA experiments in to context is discussed in Section 8.2.5.

Experiments Seven to Twelve are intended to demonstrate objective five has been met, the specialisation of middleman strategies to the environments in which they were evolved. That is to say, these experiments aim to show that strategies evolved under one set of environmental conditions will tend to perform worse under different conditions. A strategies performance is, in this case, measured against both how it performed in its home environment and how it compares to strategies evolved in this other environment. Since these experiments are based on environments already described in Sections 7.3.4 and 7.3.5 no additional details, beyond those given here, need be provided.

To compare the effectiveness of strategies under different environments some of the strategies resulting from Experiments One to Six are tested in environmental conditions different to those under which they evolved. This involved taking the resulting probability distributions from those environments and using them to instantiate middlemen in the new environment. For instance, to examine middlemen strategies from the LOWCOMA environment in the LOWAVAIL environment the following process is followed. First, set up the LOWAVAIL environment in line with the description of Experiment Three (see Section 7.3.4). Second, for each of the ten middlemen in that environment ensure each will be configured from a separate, predetermined and static (non-evolving) probability distribution. Third, ensure that the set of probability dis-

tributions used for this purpose correspond to those that resulted from the running of experiments under the Experiment Five (LOWCOMA) environment conditions.

This set up provides the basis for comparison between the two environments, for a full comparison the reverse situation must also be considered (LOWAVAIL instantiated middlemen in a LOWCOMA environment). Since comparison, not evolution, is the aim, the total number of iterations required for the experiments is reduced. To this end *500* iterations are used under these conditions rather than the usual *7500*.

The environments considered in Experiments Seven to Twelve were selected based on a comparison of the resulting strategies from each. See Section 8.3.1. This selection is based around strategies from the asymmetrically low communication, LOWCOMA, environment as these were found to deviate the most from other strategies.

Experiments Seven and Eight compare strategies evolved under the baseline environment, CTRL-3, with those from the asymmetrically low communication (LOWCOMA). As with the example above probability distributions evolved under one environment are used to configure middlemen in the other and vice versa. Experiment Seven maintains the CTRL-3 baseline environment while Experiment Eight maintains a asymmetrically low communication, LOWCOMA, environment.

Experiments Nine and Ten compare strategies from the environments of Experiments Three and Five, that is strategies evolved under asymmetrically low communication (LOWCOMA) and low product availability (LOWAVAIL). Experiment Nine maintains a low availability (LOWAVAIL) environment while Experiment Ten maintains a asymmetrically low communication (LOWCOMA) environment.

Experiments Eleven and Twelve compare strategies from the environments of

Experiments Four and Five. The STUBBORN environment is used by experiment Eleven while the LOWCOMA environment is used by experiment Twelve.

7.3.3 Control SSCM Market Description

The control experiments conducted using the SMSS are intended to provide a baseline for comparison when varying environment affecting parameters. These experiments are also intended to provide an initial proof that effective learning is taking place.

The following market description (Table 7.3) relates to these control experiments. Under this environment middlemen exist in an SSCM market in which customers are mostly profitable but not by very large margins. Products are not fully available but suppliers are not very stubborn during negotiation. Further market information common to all experiments is shown in Table 7.4. This information includes the individual product base values used by suppliers and the number of time-steps over which the market is run.

With the exception of Supplier Harshness and Adjusted Mean Trip Cost, the properties of the market description are explained in Section 4.5 of the SSCM chapter. Supplier Harshness and Adjusted Mean Trip Cost relate to how difficult suppliers are to negotiate with. These measures are described in Section 6.4.2.1. The supplier negotiation mechanism is discussed in Section 5.4.3.2.

Based on the Mean Trip Budget and Adjusted Mean Trip Cost, customer are generally unprofitable. For a middleman strategy to be able to make a profit it must either filter the customer requirements effectively and/or negotiation more aggressively with suppliers.

Table 7.3: SSCM Market Description, Control

Property	*Value*
Number Of Middlemen (NOM)	10
Customers Per Middleman (CPM)	100
Lowest Customer Balance (LCB)	-618.75
Mean Customer Balance (MCB)	1142.19
Highest Customer Balance (HCB)	4125
Number Of Products (NOP)	5
Number Of Suppliers Per Product (NOSPP)	1 (2 for SSCM-V1-S3)
Product Availability (PA)	25
Communication Budgets (CB)	5-25-25
Supplier Harshness (SH)	1.0
Mean Trip Cost	537.5
Mean Trip Budget	1679.69
Adjusted Mean Trip Cost (AMTC)	1699.33
Mean Adjusted Agreement Time (MAAT)	0.0198

Table 7.4: Further Controlling Parameters (see Table 6.5)

Parameter	*Value*
Market Length	
Market Length (T_{total})	30
Market Active Time (T_{active})	20
Product Base Values	
$Outbound_F light$	100.0
$Return_F light$	100.0
$Accommodation$	50.0
Ent_x	25.0
Trip Duration And Notice Period Generation	
Min Shortest Duration	5
Max Longest Duration	10
Min Notice Period	1

7.3.4 Demonstration Of Effective Learning

The demonstration of effective learning within the SMSS is related to experimental objectives one and two and therefore directly to claim two of the thesis.

Effective learning can be demonstrated in a two part process. Observing the change in entropy of the SMSS probability distribution over time establishes whether anything is being learnt over the course of the experiment. A reduction in entropy shows that learning of some form is taking place albeit this may not be useful. To show that what is being learnt is useful, that the middlemen strategies being evolved are effective, the mean middleman wealth is also observed over time. Ideally, if learning is effective, an entropy drop will correspond to an increase in mean middleman wealth. This mean middleman wealth measure is independent of the score used for evaluation since factors other than total wealth prove important to evolve middlemen strategies. Wealth does, however, indicate the actual middlemen performance in the markets presented. Under adverse conditions, in which leveraging a profit may not be possible, a negative mean wealth value should be minimised and preferably reduced to zero.

While results from the control experiments are sufficient to establish that effective learning takes place under some conditions, it is important to establish that this is possible under a broader set of circumstances. This is necessary to reinforce the claim of flexibility within the SSF and SMSS. To this end further experiments under other conditions must be considered.

Table 7.5, below shows the market descriptions for experiments two, three and four. The parameter settings for these experiments each deviate from the control case by varying one of the available parameters. This is intended to demonstrate that

Table 7.5: Experiment Two to Five, Description

Property	*LOWBUDGET*	*LOWAVAIL*	*STUBBORN*	*LOWCOMA*
NOM	10	10	10	10
CPM	100	100	100	100
LCB	-618.75	-618.75	-618.75	-618.75
MCB	**873.44**	1142.19	1142.19	1142.19
HCB	**3300**	4125	4125	4125
NOP	5	5	5	5
NOSPP	2	2	2	2
PA	25	**20.0**	25	25
CB	5-25-25	5-25-25	5-25-25	5-**12**-25
SH	1.0	1.0	**0.25**	1.0
Mean Trip Cost	537.5	537.5	537.5	537.5
Mean Trip Budget	**1410.94**	1679.69	1679.69	1679.69
AMTC	1699.33	1699.33	**1700**	1699.33
MAAT	0.0198	0.0198	**0.02**	0.0198

learning may take place under a variety of different market conditions.

Experiment Two varies from the control case by reducing the funds available to customers for obtaining their desired product sets. This is reflected in the highest and mean customer balance as well as in the average trip budget. The reduced mean trip budget is now represents only *83%* of the adjusted mean trip cost as opposed to *99%* in the control case. This deviation means that the learnt strategy will have to be that much more effective at either estimating the cost of products or negotiating more harshly.

Experiment Three reduces the availability of products from *25%* to *20%* so increasing competition between middlemen for resources.

Experiment Four increases the stubbornness of suppliers during negotiations by reducing their tactic control value to *0.25* from *1.0*. Within the measures available this translates in to small increase in the adjusted mean trip cost and mean adjusted

agreement time.

Experiment Five asymmetrically reduces communication budgets within the market. For middlemen only, the number of outbound communications per time-step are reduced to under half their default value.

7.3.5 Further Demonstration Of Effective Learning

Section 7.3.4 discusses experiments to determine the effectiveness of learning under the SMSS. In the course of these experiments the effects of asymmetric reduction in middleman communication budgets proved interesting (see Section 8.2.5).

To place these results in context this experiment reduces communication symmetrically for all participants in the supply chain, rather than asymmetrically for middlemen only. Table 7.6 provides details of this reduced communication environment.

7.4 Conclusions

This chapter provides the motivation and description of a series of experiments conducted using the SMSS introduced in Chapter 6.

The overall aims of these experiments is to demonstrate that the SMSS is an effective means for evolving middlemen strategies in a complex environment, that the SSF (in the form of SSF-I1) provides the necessary flexibility for this to be achieved and that the specific environmental conditions have an affect on the final outcome.

These aims are broken in to five experimental objectives that are described in Section 7.1.

Table 7.6: Experiment Six, Description

Property	*Exp 6, LOWCOMS*
Number Of Middlemen (NOM)	10
Customers Per Middleman (CPM)	100
Lowest Customer Balance (LCB)	-618.75
Mean Customer Balance (MCB)	1142.19
Highest Customer Balance (HCB)	4125
Number Of Products (NOP)	5
Number Of Suppliers Per Product (NOSPP)	2
Product Availability (PA)	25
Communication Budgets (CB)	**2-12-12**
Supplier Harshness (SH)	1.0
Mean Trip Cost	537.5
Mean Trip Budget	1679.69
Adjusted Mean Trip Cost (AMTC)	1699.33
Mean Adjusted Agreement Time (MAAT)	0.0198

The five experimental objectives are used as the motivation for twelve sets of experiments discussed in Section 7.3.

Of these twelve experimental sets, one acts as a baseline case against which the results of other experiments can be considered. Five of the remaining experiments vary the environmental conditions faced by middlemen strategies along different lines. The remaining six experiments examine how strategies evolved under one set of environmental conditions fare in other environments.

To provide a clearer picture of SMSS performance individual experiments are repeated ten times and the results data compared or averaged as appropriate.

Table 7.7 shows how these experiments relate to the experimental objectives stated in Section 7.1 and the overall thesis claims laid out in the introduction, Section 1.4.

Chapter 8, below, describes the results from these experiments and discusses their interpretation.

Table 7.7: Experiment Quick Reference Table

Experiment	*Experiment Objectives*	*Claims*	*Comments*
1_{a-c} (CTRL)	1, 2	3	Baseline for comparison
2 (LOWBUDGET)	3, 4	2, 3	Reduced Customer budgets
3 (LOWAVAIL)	3, 4	2, 3	Reduced Product availability
4 (STUBBORN)	3, 4	2, 3	Increased Supplier stubbornness
5 (LOWCOMA)	3, 4	2, 3	Reduced Middleman communication (asymmetric)
6 (LOWCOMS)	3, 4	2, 3	Reduced communication (symmetric)
7 (CTwithLCA)	5	2, 3	Baseline environment with asymmetric reduced communication evolved Middlemen
8 (LCAwithCT)	5	2, 3	Asymmetrically reduced communication environment with baseline evolved Middlemen
9 (LAwithLCA)	5	2, 3	Reduced availability environment with asymmetric reduced communication evolved Middlemen
10 (LCAwithLA)	5	2, 3	Asymmetrically reduced communication environment with reduced availability evolved Middlemen
11 (STwithLCA)	5	2, 3	Increased supplier stubbornness environment with asymmetric reduced communication evolved Middlemen
12 (LCAwithST)	5	2, 3	Asymmetrically reduced communication environment with increased supplier stubbornness evolved Middlemen

Chapter 8

Results

This chapter discusses the results of experimentation with the SMSS described in Chapter 6. The SMSS evolves SSF-I1 based middleman strategies, see Chapter 5, within a supply chain environment that is defined primarily in terms of SSCM-V1, see Chapter 4.

The experimental configurations used with the SMSS are described in Chapter 7. These experiments have been designed to fulfill the objectives listed in Section 7.1.

For each of the experiments listed in Table 7.2, a basic set of control parameters was used as outlined in Section 7.3.1.

A baseline set of experiments is described in Section 7.3.3. The results from these experiments act as a reference point against which the other experimental results may be compared. Other experimental configurations, described in Sections 7.3.4 and 7.3.5, deviate from the baseline configuration to increase the difficulty of the supply chain environment faced by the middlemen strategies being evolved.

Experiments with each configuration were repeated ten times, the results compared or averaged to provide a clearer picture of SMSS and middleman strategy performance.

The results of the control experiments that provide a baseline for comparison in the remainder of this chapter are discussed in Section 8.1, below.

The SMSS experimental software was implemented in Java using a distributable, multi-threaded approach (see Appendix B for more details). The primary computers used for experimentation were based on AMD Athlon-2400 (2.0GHz) processors with 256MB of PC2700 memory. These machines were able to process a single SSCM reference problem in about 1 second and a whole experiment, including $7500 * 5$ problems, PBIL+GM updates and the recording of results, in around 12 hours.

8.1 The Control Experiments

This section examines the results obtained from running the control experiments characterised in Section 7.3.3.

These experiments were run under the control conditions using the restrictions of SSCM-V1 Scenarios One, Two and Three discussed in Section 4.4.

The restrictions placed on the SSCM-defined problem and agent operations are reduced for later scenarios. Thus, the problem faced by middleman strategies is more complex for later scenarios and is likely to affect how they evolve.

The aim of applying different Scenario restrictions under the control configuration was to gauge the effect of complexity on the evolution and performance of the middleman strategies.

The experiments fulfill experimental objectives one and two about the demonstration of effective learning, see List 7.1.

The results from the control conditions run under Scenario Three restrictions act

as the baseline for comparison in all subsequent experiments. All these experiments also being run under SSCM-V1 Scenario Three restrictions.

The configurations of the SMSS for the purposes of these experiments are shown in Table 7.1 and Table 7.3 of Chapter 7.

8.1.1 Demonstration Of Learning

The demonstration of learning corresponds to experimental objectives one and two discussed in Section 7.1.

Figure 8.1 compares the change in entropy and mean middleman wealth over time (in iterations) for the control configuration under SSCM-V1 Scenario One, Two and Three restrictions. This figure demonstrates that effective learning is taking place. As the probability distribution converges towards a solution the mean middleman wealth increases.

For entropy it can be observed that from a high of roughly *170.0* the entropy drops slowly to around *167.0* by iteration *300*. For approximately *250* iterations after this the rate of entropy drop increases reducing entropy to around *160.0* by iteration *550*. As the rate of entropy drop increases, deviation between the three experiments is observed. For Scenario Three restrictions, entropy continues to drop at an increased, but steady, rate until approximately iteration *1100* reaching a level of about *142.0*. Beyond this point the rate of change slowly decreases beginning to level off by the end of the experiment at iteration *7500* at approximately the *89.0* level. For Scenario Two restrictions, approximately the same curve is observed but with increasing deviation from Scenario Three evident. This results in a final level of approximately *92.0*. Like Scenario Two, Scenario One restricted experiments deviate

from the Scenario Three observations following roughly the same curve. This results in a final level around *96.0*.

Observing the mean middleman wealth we see that under each set of Scenario restrictions the scores begin around the *-10,000.0* level. The score appears to be rising steadily from this level to a final, stable score, with a flattening of the rise and greater divergence of scores evident as the *0.0* point is reached. For all Scenarios scores begin to stabilise around iteration *2750* and have stabilised by iteration *4500* only a modest rise beyond this being evident by the end of the experiments. Under Scenario Three restrictions deviation from the observations of Scenario Two and One experiments is evident by around iteration *2250*. At this point the score rises from around the *7,500.0* level to stabalise at about the *12,500.0* point. For Scenario One and Two, the mean middleman wealth follows a similar but exaggerated curve, stabilising around the *20,000.0* level.

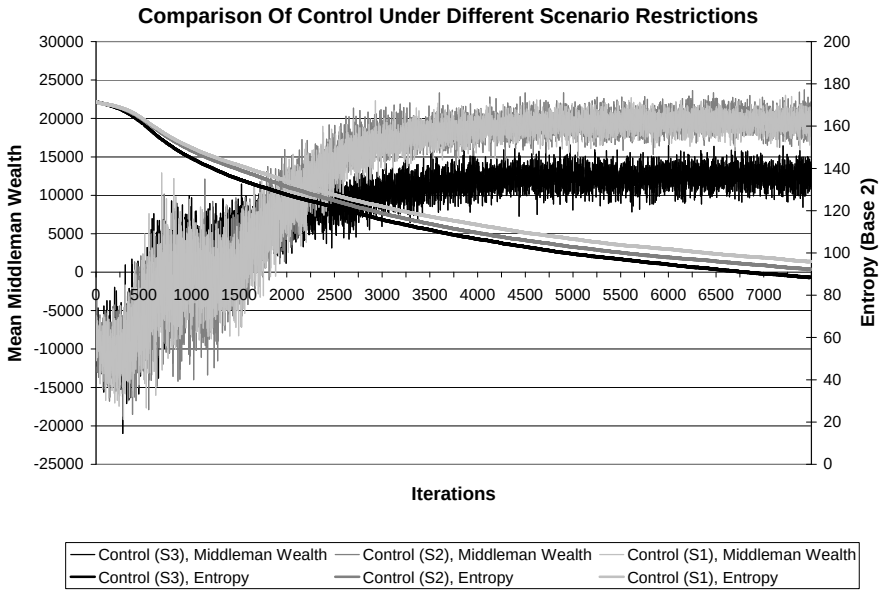

Figure 8.1: Learning, Control Configuration, S1-3

The reduction of entropy during the course of the experiments show that the probability distribution of the SMSS is learning something. Since there is a corresponding increase in the mean middleman wealth this demonstrates that what is being learnt is useful for the middlemen strategies being instantiated.

Increased scenario complexity appears to lead to more rapid learning. In the case of scenario three restrictions, the increased complexity also translates in to lower achieved mean middleman wealth. This would tend to suggest that more adverse conditions tend to promote more rapid learning.

8.1.2 Variation In Learnt Strategies

This section discusses the differences found between the strategy generating probability distributions evolved under SSCM-V1 Scenario One, Two and Three restrictions using the control configuration.

8.1.2.1 Evolved Strategy Descriptions

Table 8.1 shows the mean strategy values and their associated standard deviation for each evolved SSF-I1 parameter. The evolvable SSF-I1 parameters are described in Section 5.4.2.

With reference to Table 7.4 the SSCM-V1 Scenario Three restricted strategy shows the following. The evolved Group Duration is close to the active period of the market. A result of this is that many requirements will be dealt with in one bundle. The Group Activation Time and Group Active Proportion are quite tightly converged, this would leave a middlemen with approximately *3* time units in which to negotiate for the requirements of a group. In light of the amount of negotiation time available,

Table 8.1: Evolved SSF Parameters For Control Configuration Under Different Scenario Conditions

	Scenario 3		*Scenario 2*		*Scenario 1*	
Parameter Name	*Mean*	*StdDev*	*Mean*	*StdDev*	*Mean*	*StdDev*
Main Control Parameters						
Product Information Window	29.7	6.717	30.5	4.767	30.7	8.883
Group Activation Time	6.1	0.568	6.3	0.483	6.4	0.516
Group Active Proportion	0.631	0.057	0.606	0.037	0.602	0.044
Group Duration	17.2	5.116	14.7	5.832	14.5	5.796
Negotiation Time-Out	2.3	1.767	5.4	1.35	6.2	1.229
Pre-Negotiation Tries	25.8	8.351	26.2	4.158	25.3	7.086
Unavailable Threshold	26	1.826	28.8	0.789	29.1	0.568
Outbound Flight						
Guess Price	93.488	30.58	232.074	82.187	173.65	54.742
Upper Multiplier	3.49	1.303	4.931	1.832	5.794	2.665
Lower Multiplier	0.6	0.255	0.488	0.264	0.542	0.219
Tactic Value	1.869	0.985	1.601	0.847	1.845	1.109
Return Flight						
Guess Price	164.091	42.858	206.069	105.661	195.414	72.498
Upper Multiplier	2.56	0.508	5.159	1.848	5.379	2.235
Lower Multiplier	0.402	0.248	0.417	0.156	0.491	0.335
Tactic Value	2.154	0.84	1.732	1.161	1.907	0.927
Accommodation						
Guess Price	56.072	32.367	19.756	20.108	32.585	37.834
Upper Multiplier	3.401	1.679	2.303	0.525	2.052	0.124
Lower Multiplier	0.531	0.216	0.391	0.276	0.419	0.275
Tactic Value	3.125	1.239	1.31	0.518	1.845	2.564
Entertainment One						
Guess Price	52.491	34.311	89.135	58.712	58.774	43.592
Upper Multiplier	4.859	1.624	7.205	1.346	7.447	1.014
Lower Multiplier	0.542	0.139	0.49	0.263	0.556	0.242
Tactic Value	2.853	1.023	1.901	0.603	1.873	0.621
Entertainment Two						
Guess Price	62.384	37.986	95.829	67.928	93.931	70.515
Upper Multiplier	5.052	1.309	7.116	1.746	7.07	1.007
Lower Multiplier	0.409	0.284	0.501	0.274	0.562	0.284
Tactic Value	2.613	0.904	2.431	0.794	2.196	0.799

the mean Negotiation Time-Out appears to be quite large, allowing a comparatively long delay for responses. The number of attempts at pre-negotiation for any one requirement seems reasonably generous at *25*, this is not very tightly converged but never the less suggests it is quite important to make use of the available potentially profitable requirements. The Found Unavailable Threshold also seems quite generous and is more tightly converged. This would seem to suggest that it is important to give suppliers a reasonable chance to supply products when they have previously appeared to be unavailable. In relation to products, while quite spread out, Guess Prices appear, except in the case of Entertainments, to have clustered around their respective base values as given to suppliers at the beginning of the market. In the case of Entertainments the Guess Prices appear to be about twice the market level. A large Product Information Window means that price estimates will be made over many transactions (around *30*) although this isn't very tightly converged. Along with this it appears that the starting offer middlemen should work with is around half the estimated value with Lower Multipliers clustered around *0.5*. Upper Multiplier also seem reasonably well clustered suggesting an upper value of about three times the estimated price would be reasonable. The tactic values listed are subject to transformation from a *0.0-10.0* range to one that corresponds in equal parts to an actual tactic value in the ranges *0.0-1.0* and *1.0-10.0*. The Tactic Value's while quite spread all translate in to more stubborn negotiating practices controlled within the *0.0-1.0* range, roughly clustering around *0.5*.

The strategy resulting from the control configuration run using SSCM-V1 Scenario Two restrictions appears as follows. For the main control parameters the primary deviation from Scenario Three evolved parameters appears to be the Negotiation

Time-Out value. In the case of Scenario Two restrictions this value is greater indicating that negotiations would overrun the required response time if not for the SSF's built in restriction of such timing (see Section 5.4.1.4). In other respects the evolved parameters are similar to those under Scenario Three restrictions. A long Group Duration indicates that most requirements will be dealt with in large bundles, the Group Activation Time and Active Proportion will allow roughly *3* time-steps for negotiations. The Pre-Negotiation Tries and Unavailability threshold are again similar providing suppliers with more chance to provide products and customers with a larger number of alternatives to choose from. For the product specific parameters Guess Prices do not appear to estimate those of the market very effectively, for the most part being both greater and exhibiting a wide degree of variation. Tactic Values again have evolved to exhibit stubborn negotiating behaviour. Lower Multipliers appear similar to those under Scenario Three restrictions, clustering around the *0.5* level, while Upper Multipliers seem rather higher. For the Accommodation product type the above does not fully hold. The Guess Price is much below the market level and the Upper Multiplier is lower and far more constrained than for other products. The Tactic Value is also evolved to be lower and is more tightly constrained. This would seem to suggest that obtaining Accommodation products is more important to strategy success than obtaining any other product.

Under Scenario One restrictions the following is observed. The main control parameters are similar to those evolved under Scenario Two restrictions. Variation from Scenario Three evolved strategies is seen in the Negotiation Time-Out parameter, this being greater still than under Scenario Two restrictions. Product Guess Price's seem to have converged very slightly closer to those values provided to the market than

those evolved under Scenario Two restrictions. Upper and Lower Multipliers are similar to those observed under Scenario Two restrictions and Tactic Values again indicate stubborn negotiation behaviour.

8.1.2.2 Differences In Strategies

In order to better understand the differences between the evolved strategies, the Students t-test, [20], was run to compare the differences between individual parameters.

Table 8.2 shows the t-test values derived from the mean data for each parameter of the compared control probability distributions. With reference to Table 7.1, 10 distributions from each of the 10 experimental repetitions are compared. This requires, in each case, a t-test value greater than *2.8982* to indicate a significance difference in the means at the *99%* level.

This is summarised pictorially in Figure 8.2 that shows for which parameters a statistically significant difference was detected and, if so, how the associated means vary.

The set of parameters evolved under Scenario One restrictions is shown to vary from those evolved under Scenario Three restrictions in the following way. For the main control parameters, The Negotiation Time-Out and Unavailable Threshold are different, both being less under Scenario Three restrictions. For the Negotiation Time-Out the mean is about half that under Scenario One while the Unavailable Threshold is similar but exhibits less deviation from the mean. For product specific control parameters some variation from Scenario Three is observed, in all cases values are greater under Scenario One, indicating a higher estimated product value or higher accepted prices. The Guess Price is observed to be different for Outbound Flights

Table 8.2: Comparison t-test Values For Evolved Parameters, Control Configurations

Parameter Name	t-test Values *S1 Against S3*	*S2 Against S3*	*S1 Against S2*
Main Control Parameters			
Product Information Window	0.376	0.22	0.241
Group Activation Time	1.235	0.847	0.447
Group Active Proportion	1.267	1.176	0.204
Group Duration	1.125	0.966	0.111
Negotiation Time-Out	**5.362**	**4.262**	1.325
Pre-Negotiation Tries	0.172	0.036	0.266
Unavailable Threshold	**5.172**	**4.228**	0.554
Outbound Flight			
Guess Price	**4.033**	**4.6**	1.863
Upper Multiplier	2.534	2.051	1.0
Lower Multiplier	0.668	1.065	0.488
Tactic Value	0.043	0.651	0.56
Return Flight			
Guess Price	1.205	1.247	0.336
Upper Multiplier	**3.757**	**4.264**	0.331
Lower Multiplier	0.83	0.041	0.897
Tactic Value	0.617	0.906	0.357
Accommodation			
Guess Price	1.485	**3.015**	0.95
Upper Multiplier	2.485	1.962	1.47
Lower Multiplier	1.02	0.963	0.061
Tactic Value	0.993	**4.217**	0.724
Entertainment One			
Guess Price	0.363	1.711	1.319
Upper Multiplier	**4.395**	**3.537**	0.351
Lower Multiplier	0.338	0.405	0.603
Tactic Value	2.582	2.528	0.102
Entertainment Two			
Guess Price	1.261	1.371	0.059
Upper Multiplier	**3.827**	**2.932**	0.331
Lower Multiplier	1.263	0.759	0.547
Tactic Value	1.091	0.485	0.658

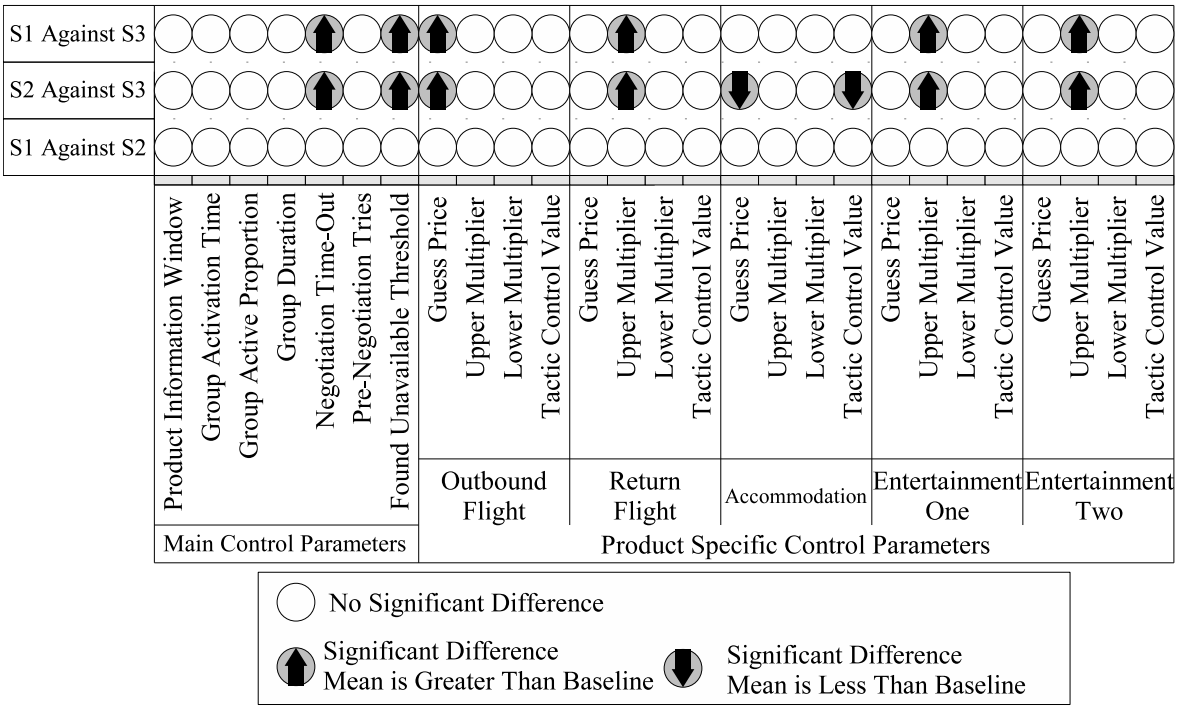

Figure 8.2: Significant Differences Between Evolved Parameters, Control Configurations

while the Upper Multiplier is different for the Return Flights, Entertainment One and Entertainment Two.

Comparing the evolved Scenario Two restricted strategy against that evolved under Scenario Three restrictions the following is observed. For the main control parameters the Negotiation-Time Out and Unavailable Threshold are seen to be significantly different. Under Scenario Two restrictions the mean of the Negotiation Time-Out parameter is roughly double that evolved under Scenario Three. For the Unavailable Threshold the mean is a little higher than that under Scenario Three but the standard deviation is lower. For product specific parameters a difference is found in some instances for the Guess Price, Upper Multiplier and Tactic Control Value. The Guess Price is found to be greater under Scenario Two restrictions for the Outbound Flight the mean being more than double the market supplied value. For Accommodation the Guess Price is found to be lower under Scenario Two restrictions, the mean being roughly half the supplied market value. The Upper Multiplier is different for the Return Flight, Entertainment One and Entertainment Two product types. In each case the multiplier is greater for Scenario Two restrictions the mean being almost double the values found under Scenario Three. The Tactic Control Value is found to be significantly different for the Accommodation product type, the mean value being roughly half that under Scenario Three restrictions and with a much lower standard deviation.

The comparison between the strategies evolved under Scenario One and Two restrictions indicates there is no statistically significant difference between them when compared at the *99%* level.

8.1.2.3 Variation In Control Experiment Strategies, Conclusions

The comparison of strategies evolved under the control conditions using different Scenario restrictions would appear to support the following.

The evolution of product Guess Price's close to the supplier values is important under the more difficult Scenario Three conditions. Under all conditions these prices are within the lower part of the possible range. Since the Guess Price is responsible for initial offers made by a middlemen, when no other information is available, it is reasonable to assume that evolving an effective value for this parameter is important

Evolving an effective Negotiation Time Out appears to be important. Under both Scenario One and Two restrictions this value would allow respondents to reply at the last possible moment available to the middlemen. Under Scenario Three restrictions this value is lower potentially providing middlemen with greater time to respond themselves.

8.1.3 Control Experiments, Conclusions

The control experiments discussed above show that the SMSS is able to evolve effective middlemen strategies for these environments and demonstrate that objective one and two (from Section 7.1) have been met.

In examining the increase in complexity of the environment it is observed that as complexity increases so does the rate of learning. This suggests that the complexity of the environment offers an incentive for more rapid learning.

Strategies evolved under the different SSCM-V1 Scenario conditions are quite similar with differences only appearing between the most complex scenario, SSCM-V1S3, and the other two scenarios.

In the cases where a difference is observed these relate to parameters associated with product negotiation.

8.2 Variation In Environments

In this section we examine the effects of deviating from the baseline configuration. Middlemen strategies are evolved under SSCM-V1 Scenario Three restrictions, providing maximum complexity to the situation, and individual environment affecting parameters are adjusted. These experiments are described in Section 7.3.4.

The evolution of middlemen strategies under different environmental conditions relates to experimental objective three in Section 7.1.

8.2.1 Reduced Customer Wealth

Under conditions of reduced customer wealth middlemen are faced with a situation in which many of the potential clients are likely to be unprofitable, this configuration is described in Section 7.3.4.

Figure 8.3 plots the probability distributions entropy and mean middleman wealth over the time of the experiment compared against that for the control case. Table 8.3 shows the mean parameter values from the evolved strategy, their standard deviations and places this in context with a Student's t-test against the same parameters from the control.

With reference to Figure 8.3, at the beginning of the experiment mean middleman wealth seen to be around the *-5,000.0* level. After some initial fluctuations this gradually increases reaching the break even point by approximately iteration *3750*. Beyond this point mean wealth continues to increase but begins to level off. By

Table 8.3: Reduced Customer Wealth, Evolved Strategy and Deviation From Control

Parameter Name	*Evolved Strategy*		*t-test Comparison Against Control*
	Mean Value	*Standard Deviation*	
Main Control Parameters			
Product Information Window	31.7	5.458	0.678
Group Activation Time	6.4	0.843	0.936
Group Active Proportion	0.632	0.044	0.046
Group Duration	11.4	4.088	2.84
Negotiation Time-Out	4.8	1.549	**3.333**
Pre-Negotiation Tries	25	7.86	0.221
Unavailable Threshold	24	2.828	1.591
Outbound Flight			
Guess Price	110.956	68.175	0.769
Upper Multiplier	4.261	1.536	1.223
Lower Multiplier	0.464	0.22	1.289
Tactic Value	1.428	0.496	1.269
Return Flight			
Guess Price	165.325	70.523	0.103
Upper Multiplier	3.005	0.777	1.523
Lower Multiplier	0.463	0.216	0.508
Tactic Value	1.705	1.067	1.023
Accommodation			
Guess Price	41.25	42.477	0.866
Upper Multiplier	3.037	1.606	0.458
Lower Multiplier	0.485	0.232	0.484
Tactic Value	1.239	0.521	**4.373**
Entertainment One			
Guess Price	63.497	58.971	0.524
Upper Multiplier	4.885	2.035	0.09
Lower Multiplier	0.482	0.253	0.444
Tactic Value	2.335	1.18	1.041
Entertainment Two			
Guess Price	71.906	43.208	0.524
Upper Multiplier	5.55	1.652	0.813
Lower Multiplier	0.618	0.297	1.707
Tactic Value	2.201	1.127	0.89

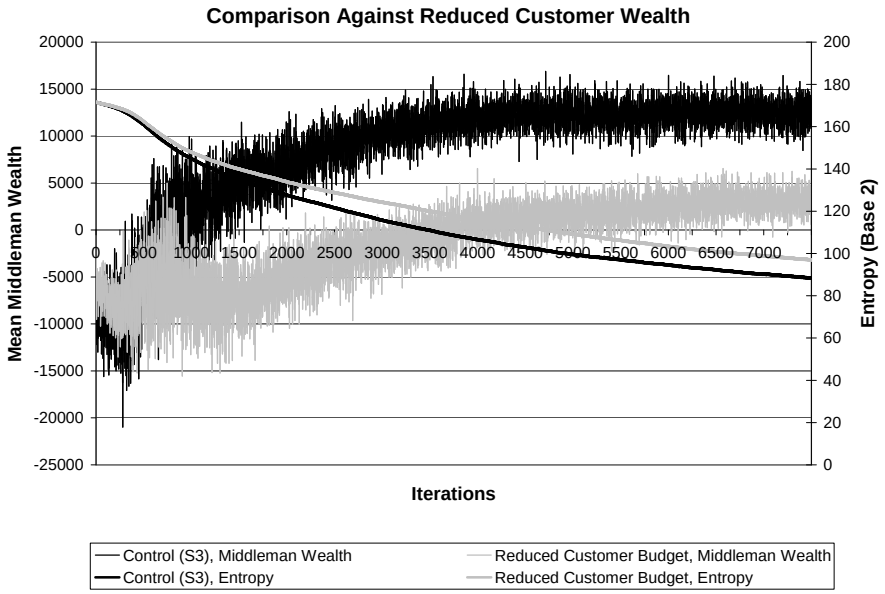

Figure 8.3: Learning, Reduced Customer Budgets

the end of the experiment the mean wealth is around the *2,500.0* level. During the course of the experiment entropy reduces with time roughly matching the curve of the control case. Deviation from the control line begins around generation *600* resulting in a final level around the *97.0* level, about *8.0* points different.

With reference to Table 8.3 the strategy evolved under conditions of reduced customer budget is mostly similar to that evolved under the control case. Two parameters exhibit a significant difference. Firstly, the Negotiation Time-Out has increased to a similar level seen under Scenario Two restrictions allowing a greater time for supplier responses. Second, the Tactic Value used for negotiation of the Accommodation product is much less and has a lower standard deviation than that of the control case.

8.2.2 Reduced Product Availability

In this section we look at the effects of reducing the availability of products and how this impacts the evolution of middlemen strategies. Figure 8.4 shows the probability distribution entropy and mean middleman wealth plotted against the experimental iterations. Table 8.4 shows the evolved middleman strategy and compares this against the control case using a Students t-test.

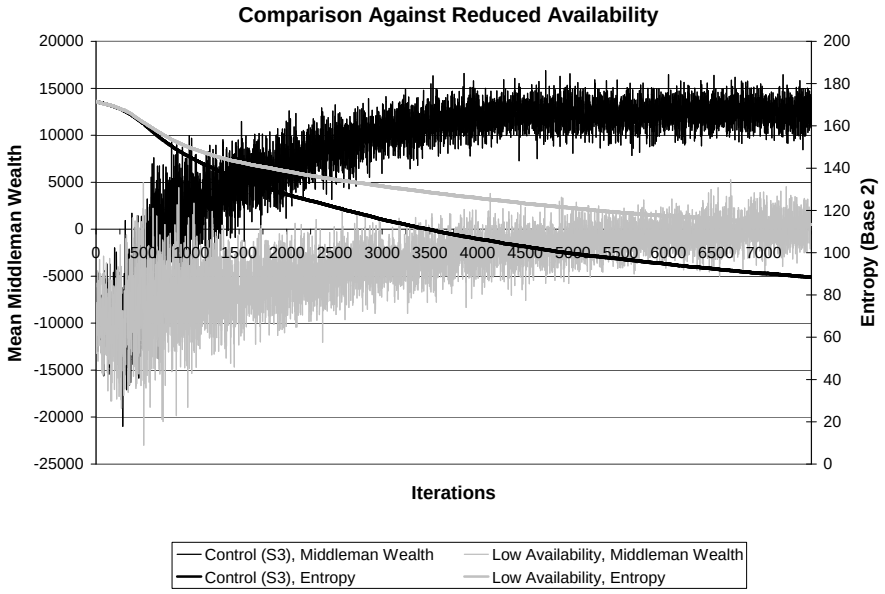

Figure 8.4: Learning, Reduced Product Availability

With reference to Figure 8.4 it can be observed that both the probability distribution entropy and mean middleman wealth deviate strongly from the control case. Entropy follows a similar curve to the control case beginning deviation around iteration *400* and finally finishing at approximately the *113.0* level, approximately *24.0* points higher. For mean middleman wealth the score begins at approximately the *-10,000.0* level rising steadily to break even by around iteration *6500*. Little progress at increasing the score is made beyond this point.

Table 8.4: Reduced Product Availability, Evolved Strategy and Deviation From Control

Parameter Name	*Evolved Strategy*		*t-test Comparison*
	Mean Value	*Standard Deviation*	*Against Control*
Main Control Parameters			
Product Information Window	31.4	4.452	0.573
Group Activation Time	7.4	0.516	**5.355**
Group Active Proportion	0.584	0.04	2.102
Group Duration	14.5	3.206	1.506
Negotiation Time-Out	2.7	2.359	0.463
Pre-Negotiation Tries	26	5.249	0.004
Unavailable Threshold	25.9	1.37	0.255
Outbound Flight			
Guess Price	154.631	66.931	2.627
Upper Multiplier	4.1	1.446	1.0
Lower Multiplier	0.685	0.253	0.735
Tactic Value	1.757	0.92	0.262
Return Flight			
Guess Price	206.535	61.817	1.796
Upper Multiplier	2.849	1.015	0.824
Lower Multiplier	0.556	0.282	1.338
Tactic Value	2.005	0.932	0.37
Accommodation			
Guess Price	49.045	47.96	0.374
Upper Multiplier	2.86	1.397	0.775
Lower Multiplier	0.45	0.235	0.777
Tactic Value	2.162	0.981	1.927
Entertainment One			
Guess Price	101.908	72.368	1.968
Upper Multiplier	4.377	2.059	0.513
Lower Multiplier	0.473	0.259	0.557
Tactic Value	1.938	0.953	2.06
Entertainment Two			
Guess Price	92.593	64.598	1.288
Upper Multiplier	6.638	1.179	2.871
Lower Multiplier	0.368	0.35	0.126
Tactic Value	2.164	0.908	1.1

From Table 8.4 it can be seen that the evolved strategy under these conditions is largely the same as for the control case. A significant deviation is however present in relation to the Group Activation Time, this being centered and tightly focused on *7.4*. This increased length from the control case would provide an additional one or two time unit for negotiation, *4* or *5* in total.

8.2.3 Increased Supplier Stubbornness

This section describes results from experiments in which the stubbornness of supplier negotiation is increased. This makes it more difficult for middlemen to obtain products at a reasonable price and so leverage a profit. These experiments are described in Section 7.3.4.

The Figure 8.5 shows middleman wealth and probability distribution entropy over the course of the experiments. These are compared against the same values from the control case. Table 8.5 shows the evolved strategy parameter means, their standard deviations from that mean and the result of a Students t-test compared with the same parameters from the control case to determine if there is any significant difference.

From Figure 8.5 it can be observed that the reduction in entropy under conditions of increased supplier stubbornness follows that of the control case closely, ending roughly *2.0* points lower at roughly the *87.0* level. Like entropy the mean middleman wealth roughly follows the control case, tracing a similar but reduced curve, and finishing at approximately the *8,250.0* level by the end of the experiment.

With reference to Table 8.5 it can be seen that under conditions of increased supplier negotiation stubbornness, the evolved strategy exhibits no significant deviation from that evolved under the Scenario Three restricted control case.

Table 8.5: Increased Supplier Stubbornness, Evolved Strategy and Deviation From Control

Parameter Name	*Evolved Strategy*		*t-test Comparison*
	Mean Value	*Standard Deviation*	*Against Control*
Main Control Parameters			
Product Information Window	32.9	4.677	1.159
Group Activation Time	6.5	0.527	1.633
Group Active Proportion	0.62	0.05	0.444
Group Duration	16.7	4.572	0.211
Negotiation Time-Out	1.8	1.317	0.726
Pre-Negotiation Tries	22.9	9.219	0.719
Unavailable Threshold	25.2	2.44	0.771
Outbound Flight			
Guess Price	83.903	26.796	0.749
Upper Multiplier	4.012	2.026	0.748
Lower Multiplier	0.491	0.286	0.836
Tactic Value	1.647	0.917	0.522
Return Flight			
Guess Price	150.846	47.54	0.641
Upper Multiplier	2.73	1.139	0.46
Lower Multiplier	0.697	0.224	2.764
Tactic Value	1.598	0.814	1.494
Accommodation			
Guess Price	52.37	51.764	0.171
Upper Multiplier	4.146	2.019	0.909
Lower Multiplier	0.388	0.244	1.369
Tactic Value	2.715	1.616	0.604
Entertainment One			
Guess Price	45.917	27.218	0.477
Upper Multiplier	4.468	1.938	0.455
Lower Multiplier	0.493	0.214	0.528
Tactic Value	2.822	1.229	0.049
Entertainment Two			
Guess Price	82.646	37.714	1.193
Upper Multiplier	4.471	2.305	0.528
Lower Multiplier	0.376	0.245	0.344
Tactic Value	2.298	1.026	0.721

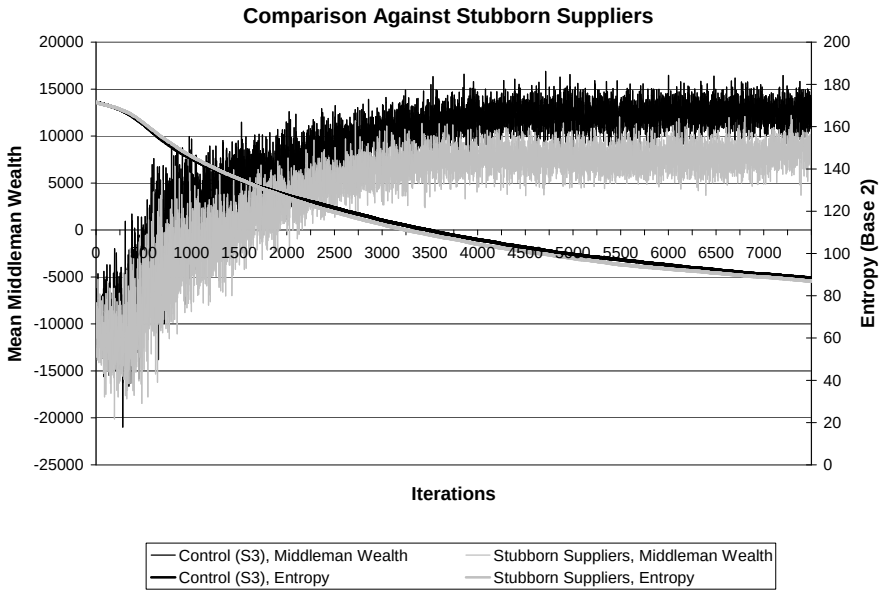

Figure 8.5: Learning, Stubborn Suppliers

8.2.4 Decreased Middleman Communication

This section describes results from experiments in which the communication budget available to middlemen is reduced. These experiments are described in Section 7.3.4.

The Figure 8.6 shows middleman wealth and probability distribution entropy over the course of the experiments. These are compared against the same values from the control case. Table 8.6 shows the evolved strategy parameter means, their standard deviations from that mean and the result of a Students t-test compared with the same parameters from the control case to determine if there is any significant difference.

From Figure 8.6 it can be observed that under conditions of reduced outbound communication for the middlemen, the performance of middlemen is much improved over that of the control case. For the probability distribution entropy, the observed curve follows a similar but exaggerated path to the control case. Starting at a level around *170.0* the initial drop in entropy is somewhat slower than for the control.

Table 8.6: Reduced Middleman Communication, Evolved Strategy and Deviation From Control

Parameter Name	*Evolved Strategy*		*t-test Comparison Against Control*
	Mean Value	*Standard Deviation*	
Main Control Parameters			
Product Information Window	24.8	8.324	1.416
Group Activation Time	4.8	0.422	**5.811**
Group Active Proportion	0.75	0.048	**4.941**
Group Duration	14.7	6.325	0.962
Negotiation Time-Out	5.8	1.229	**5.006**
Pre-Negotiation Tries	25.2	12.246	0.013
Unavailable Threshold	15.0	6.515	**5.736**
Outbound Flight			
Guess Price	245.514	101.704	**4.019**
Upper Multiplier	2.649	1.08	1.571
Lower Multiplier	0.504	0.246	0.988
Tactic Value	2.63	2.897	0.885
Return Flight			
Guess Price	230.301	72.667	2.43
Upper Multiplier	3.859	2.048	1.949
Lower Multiplier	0.377	0.189	0.344
Tactic Value	2.752	1.72	1.028
Accommodation			
Guess Price	43.573	15.905	1.102
Upper Multiplier	8.201	0.874	**8.144**
Lower Multiplier	0.423	0.229	1.09
Tactic Value	4.195	1.305	1.895
Entertainment One			
Guess Price	112.934	49.379	**3.174**
Upper Multiplier	5.673	1.892	1.093
Lower Multiplier	0.521	0.292	0.014
Tactic Value	3.377	1.21	1.052
Entertainment Two			
Guess Price	86.941	42.619	1.356
Upper Multiplier	6.358	1.449	2.166
Lower Multiplier	0.474	0.148	0.457
Tactic Value	3.339	0.953	1.736

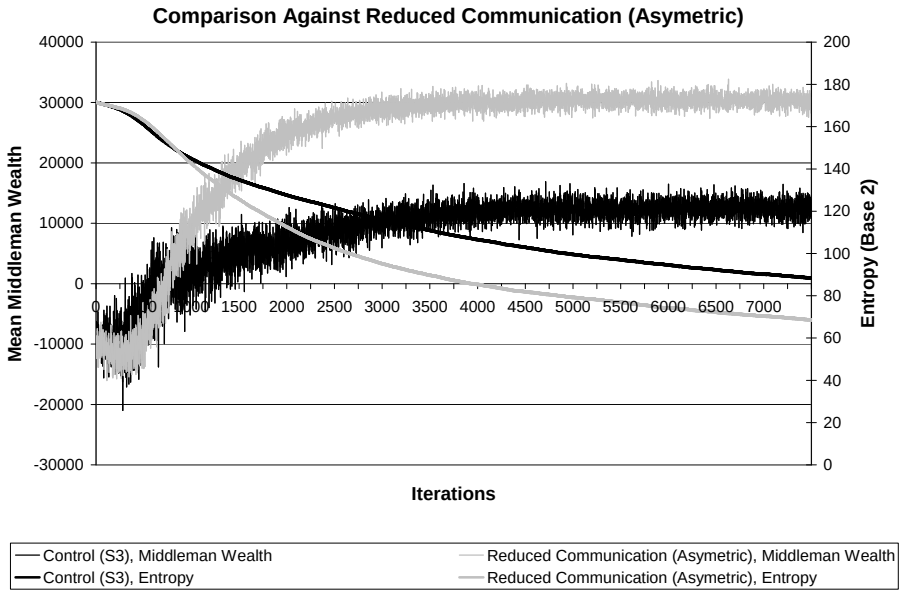

Figure 8.6: Learning, Reduced Middleman Communication

Around iteration *550* entropy begins to drop rapidly and at a greater rate than for the control. The two paths cross around iteration *850* at about the *148.0* level. Beyond this point entropy continues to drop more rapidly than for the control until around iteration *5000*. By the end of the experiment the rate of change in entropy appears somewhat less than for the control but a level of about *69.0* is achieved, *20.0* points below the control case.

With reference to Table 8.6 it can be seen that under conditions of reduced customer communication a number of parameter means deviate significantly from the control case. For the main control parameters four are found to be different, the Group Activation Time, Group Active Proportion, Negotiation Time-Out and Unavailable Threshold. The Group Activation Time and Group Active Proportion are quite strongly converged and would tend to provide middlemen with *3* time units in which to negotiate with suppliers. Thus, while the values deviate quite strongly

from the control, the overall effect on the amount of time available for negotiation with suppliers appears to be similar. However, because of this difference groups will become active earlier than under the control and have less time to provide responses to customers, *1* time-step in this case as opposed to *3* in the control case. Unlike the control case the Negotiation Time-Out value would force the middleman strategy to use the latest possible response time available given the total amount of time to negotiate. The Unavailable Threshold is much lower than in the control case, meaning less time will be spent trying to negotiate with suppliers when a product has appeared to be unavailable in the past. For the product control parameters, differences are seen in the Guess Price for both the Outbound Flight and Entertainment One products. In both cases the means now provide an initial estimate around double the supplied market value. For the Accommodation product the Upper Multiplier is seen to be different, having a value more than double its control counter-part. This difference leads to a higher final price and consequently increases the size of offers earlier on in negotiations.

The increased performance of middlemen strategies relative to the baseline experiments was an unexpected outcome. Section 8.2.5, below, places these results in context and discusses their explanation.

8.2.5 Symmetrically Decreased Communication

To help understand the results in Section 8.2.4 a second set of reduced communication experiments are run.

For this set of experiments outbound communication budgets are reduced symmetrically for all supply chain participants rather than asymmetrically, for middlemen

participants only.

This experimental configuration corresponds to Experiment 6 (LOWCOMS) described in Section 7.3.5.

Figure 8.7 shows how mean middleman wealth and the probability distribution entropy varied over the course of the experiment in comparison to both the control case and the asymmetric reduced communication case. Table 8.7 shows the parameter means, and standard deviations and the Students t-test values against the control and the asymmetric reduced communication case.

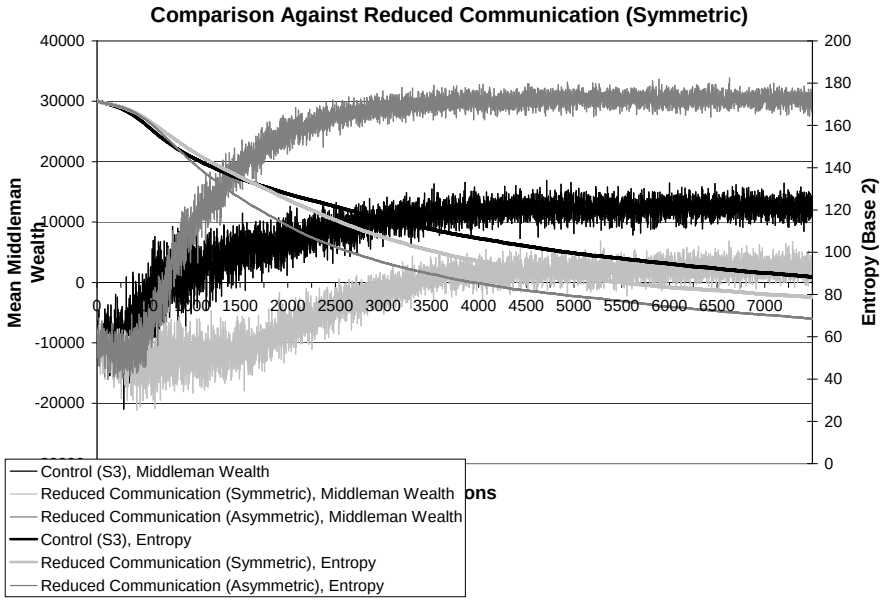

Figure 8.7: Learning, Symmetrically Reduced Communication

Figure 8.7 shows how the mean middleman wealth and probability distribution entropy change over the time under *LOWCOMS* conditions. Mean middleman wealth is seen to start at around the *-10,000* level continuing in this vein until around iteration *1500*. Beyond this point mean middleman wealth rises gradually, breaking

Table 8.7: Symmetrically Reduced Communication (LOWCOMS), Evolved Strategy and Deviation From Control And Asymmetrically Reduced Communication

Parameter Name	*Evolved Strategy*		*t-test Against*	
	Mean Value	*Standard Deviation*	*CTRL-3*	*LOWCOMA*
Main Control Parameters				
Product Information Window	27.3	6.600	0.843	0.698
Group Activation Time	4.4	0.843	**5.283**	1.336
Group Active Proportion	0.676	0.026	2.184	**4.164**
Group Duration	16.3	5.498	0.299	0.617
Negotiation Time-Out	5.9	1.197	**5.19**	0.18
Pre-Negotiation Tries	26.6	8.566	0.219	0.185
Unavailable Threshold	22.1	4.818	2.43	2.865
Outbound Flight				
Guess Price	185.436	47.941	**5.086**	1.698
Upper Multiplier	2.417	0.581	2.379	0.625
Lower Multiplier	0.595	0.338	0.186	1.01
Tactic Value	3.581	0.956	**3.911**	0.688
Return Flight				
Guess Price	178.521	73.152	0.583	1.537
Upper Multiplier	2.713	0.789	0.527	1.675
Lower Multiplier	0.525	0.217	1.117	1.712
Tactic Value	1.711	0.731	1.257	1.792
Accommodation				
Guess Price	51.92	31.533	0.289	0.752
Upper Multiplier	3.541	1.951	0.211	**6.871**
Lower Multiplier	0.43	0.297	0.55	0.297
Tactic Value	4.3	1.283	2.104	0.18
Entertainment One				
Guess Price	39.198	30.301	0.918	4.006
Upper Multiplier	4.127	0.775	1.366	2.43
Lower Multiplier	0.648	0.216	1.454	1.049
Tactic Value	2.0	1.094	0.309	0.737
Entertainment Two				
Guess Price	39.858	26.908	1.528	**2.941**
Upper Multiplier	4.338	1.041	1.384	**3.544**
Lower Multiplier	0.422	0.18	0.001	0.632
Tactic Value	3.329	1.317	1.425	0.011

even around iteration *3500* and stabalising around iteration *4000* at approximately the *2500.0* level. This reduced performance is below that seen in the control case and much below that shown in the *LOWCOMA* asymmetrically reduced communication case. The curve of the entropy drop associated with this experiment corresponds well with that seen under *LOWCOMA* experimental conditions deviating early on to end about *10.0* points higher at around *78.0*. This reduced entropy drop is still around *10.0* points greater than that seen under the control conditions (*CTRL-3*).

With reference to Table 8.7 it can be seen that of the main control parameters the Product Information Window, Group Duration and Pre-Negotiation Tries parameters do not vary significantly from either the *CTRL-3* or *LOWCOMA* results. The Group Activation Time and Negotiation Time Out parameters vary from those evolved under *CTRL-3*, the former being about a third of that value (*4.4* as opposed to *6.1*) and less tightly converged, the latter being greater at *5.9* as opposed to *2.3* but similarly converged. This leads the strategy to have approximately *2-3* time-units in which to negotiate leaving only *1* time-unit for confirmation of product acquisition with customers. The increased negotiation timeout ensures that the latest possible response time is specified to suppliers such that there may be no further opportunity for negotiation when a reply is received. Relative to the strategy evolved under *LOWCOMA* conditions the Group Active Proportion exhibits being about *10%* less and more tightly converged. For the remaining product specific parameters the evolved strategy shows a difference relative to the control for only the Outbound Flight product. In this case the Guess Price is seen to be close to double that evolved under control conditions and is similarly converged. The Tactic Value is greater than that evolved under the control conditions, giving ground more easily in

negotiations, but still acts as a stubborn negotiator. Relative to the strategy evolved under *LOWCOMA* the product parameters vary for the Accommodation, Entertainment One and Entertainment Two products. For the Accommodation product the Upper Multiplier exhibits a significant difference being about half that under *LOWCOMA* conditions. For Entertainment One, the Guess Price has evolved to be almost *1/3* that under *LOWCOMA* conditions. For the Entertainment Two product both the Guess Price and Upper Multiplier are shown to be significantly different. The Guess Price being approximately half that evolved under *LOWCOMA* conditions and the Multiplier being about *2/3* that evolved under *LOWCOMA* conditions.

In the baseline configuration, the communication budgets of middlemen and suppliers are symmetric. Both participant types are allowed *25* utterances per time-step. Under these conditions middlemen strategies are able to perform reasonably effectively

The *LOCOMA* environment reduces the communication budget for middlemen only. Under these conditions the performance of middlemen strategies is shown to improve considerably. When communication budgets are reduced symmetrically, as for the *LOWCOMS* environment, performance reduces below that of the baseline.

To explain this, first assume that middlemen will use all their communication budget if possible. Under baseline conditions ten middlemen exist within the supply chain each capable of using *25* utterances per time-step, this translates to a possible *25*10=250* messages being sent in total by middlemen per time step. For each supply chain investigated five suppliers are used, under baseline conditions these have *25* utterances translating to a maximum *25*5=125* utterances per time-step. Assuming most middleman communication is directed towards suppliers for the acquisition of

products, suppliers will find themselves swamped and unable to reply in a timely fashion to middlemen requests.

When the middlemen communication budget is reduced without a reduction being made to other participant communication budgets, the supplier swamping effect is removed. This allows middlemen to more reliable obtain product bundles, so improving their performance.

Further, when replies from suppliers are unreliable, the evolution of earlier Negotiation Time Outs and lower Unavailable Thresholds would be less beneficial for middlemen. A longer wait for replies may increase the chance of a positive outcome and the information on which the Unavailable Threshold is based becomes unreliable. When suppliers are not swamped by messages a quicker response is likely and beneficial and the Unavailable Threshold can be used more effectively to judge if a product is truly unavailable. These affects are observed for the baseline configuration, shown in Section 8.1.2.1, and the reduced communication environments above.

When communication budgets are reduced symmetrically the supplier swamping problem re-emerges with the added problem of less communication being available to resolve the chain all round. This would be reasonably expected to reduce middlemen performance below that of the baseline case as is observed.

8.2.6 Conclusions

The results of experiments in the *LOWBUDGET*, *LOWAVAIL*, *STUBBORN*, *LOWCOMA* and *LOWCOMS* environments show that the SMSS is able to evolve middlemen strategies under a variety of different supply chain conditions. This fulfills experimental objective three discussed in Section 7.1.

Evolving middlemen strategies under tougher environmental conditions than the baseline (shown in Section 8.1) leads to lower middlemen performance and some variation in the resultant strategies.

Reducing communication between middlemen asymmetrically increases the performance of middlemen and allows improved learning to take place. This appears to be due to reducing the problem of swamping suppliers with communication utterances they are then unable to respond to in a timely fashion.

8.3 Environment Specialisation Of Middleman Strategies

Experimental objectives four and five, see Section 7.1, seek to demonstrate that SSF-I1 based middlemen strategies evolved using the SMSS will specialise to the environment in which they are evolved.

This section examines the specialisation of middlemen strategies in two ways. Section 8.3.1, below, discusses how strategies evolved within one environment are more similar to each other than to strategies evolved under different environments. Section 8.3.2 reports on the performance of middlemen strategies evolved under one environment being exposed to different conditions.

8.3.1 The Convergence Of Strategies Within An Environment

Figure 8.8 compares strategies evolved under each of the experimental conditions using the Sum Of Probability Differences (SOPD) score discussed in Section 7.2.1.2.

The SOPD indicates how similar two probability distributions are by comparing

the distribution of probability across parameter ranges. To obtain the 'SOPD Score Against Self' shown in Figure 8.8, all probability distributions evolved under the same experimental conditions were compared. The remaining values show the mean variation in SOPD when those strategies are then compared against strategies from other environments.

	SOPD Score Against Self	CTRL-3	CTRL-2	CTRL-1	LOWBUDGET	LOWAVAIL	STUBBORN	LOWCOMA	LOWCOMS
CTRL-3	**1.853**	**0.000**	0.018	0.023	0.015	0.009	0.008	0.085	0.063
CTRL-2	1.837	0.033	0.000	***-0.009***	0.022	0.037	0.048	0.091	0.062
CTRL-1	**1.824**	0.052	0.004	**0.000**	0.033	0.044	0.063	0.105	0.079
LOWBUDGET	1.857	0.010	0.001	***-0.001***	0.000	0.000	0.020	0.065	0.041
LOWAVAIL	**1.813**	0.049	0.061	0.055	0.045	**0.000**	0.043	0.121	0.097
STUBBORN	1.859	0.002	0.026	0.028	0.019	***-0.003***	0.000	0.086	0.064
LOWCOMA	**1.883**	0.055	0.045	0.046	0.039	0.051	0.063	**0.000**	0.027
LOWCOMS	**1.873**	0.043	0.026	0.030	0.026	0.037	0.050	0.037	**0.000**

Value	Most similar evolved strategies (self)
Value	Most similar evolved strategies (not self)

Figure 8.8: Comparison of SOPD scores between strategies evolved under different experimental environments

In most cases strategies evolved under one set of conditions are most similar to other strategies evolved under the same conditions.

Strategies evolved under the *CTRL-2*, *LOWBUDGET* and *STUBBORN* conditions do not however follow this trend. In each of these cases while similar to themselves the probability distributions from other environments appear to more similar.

Strategies evolved under the *LOWCOMA* set of experimental conditions appear to be most dissimilar to strategies from other environments. The difference of *LOWCOMA* strategies from others acted as the determining factor when selecting strategies for comparison in alternative environments.

Broadly, the strategies evolved under one set of conditions tend to be most similar to other strategies evolved under those same conditions. This supports the fulfillment of experimental objective four in Section 7.1.

8.3.2 How Middlemen Strategies Perform In Unfamiliar Environments

Experiments seven to twelve discussed in Section 7.3 aim to demonstrate that strategies evolve to tackle the environment they are exposed to explicitly. This relates directly to experimental objective five discussed in Section 7.1.

Since the strategies resulting from the *LOWCOMA* environment deviated most strongly from strategies evolved under other environments this is used as the basis for comparison. Strategies from this environment are examined under *CTRL-3*, *STUBBORN* and *LOWAVAIL* environments and vice versa.

To act as a guide to normal performance, a static mean middleman wealth score is obtained for strategies evolved under each of the environmental conditions compared. These scores are calculated by averaging the mean middleman wealths obtain in the last *2000* iterations of the related experiments.

8.3.2.1 Comparison Of LOWCOMA And CTRL-3 Evolved Strategies

Figure 8.9 shows the results of comparing middlemen strategies evolved in the *CTRL-3* and *LOWCOMA* environments.

As can be seen, the *CTRL-3* based middlemen perform slightly worse than normal when exposed to the *LOWCOMA* environment. They also perform far less effectively than the *LOWCOMA* evolved strategies native to that environment.

LOWCOMA evolved middlemen strategies perform worse than they do in their native environment. However, these strategies are still able to do *better* than the strategies native to the *CTRL-3* environment.

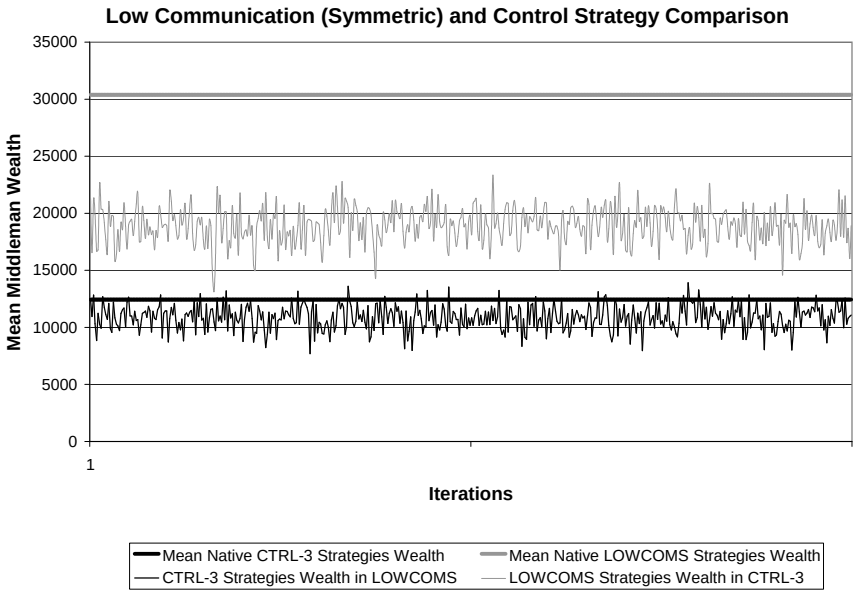

Figure 8.9: Strategy Cross Environment Comparison, CTRL-3 And LOWCOMA

8.3.2.2 Comparison Of LOWCOMA And STUBBORN Evolved Strategies

Figure 8.10 shows the results of comparing middlemen strategies evolved in the *STUBBORN* and *LOWCOMA* environments.

Strategies evolved under the *STUBBORN* environment perform somewhat worse in the *LOWCOMA* environment compared with their normal *STUBBORN* environment performance. They are also far less effective than the strategies evolved in the *LOWCOMA* environment.

Strategies from the *LOWCOMA* environment perform far worse under *STUBBORN* conditions. However, these strategies are still able to perform a little better than the strategies that evolved under those same *STUBBORN* conditions.

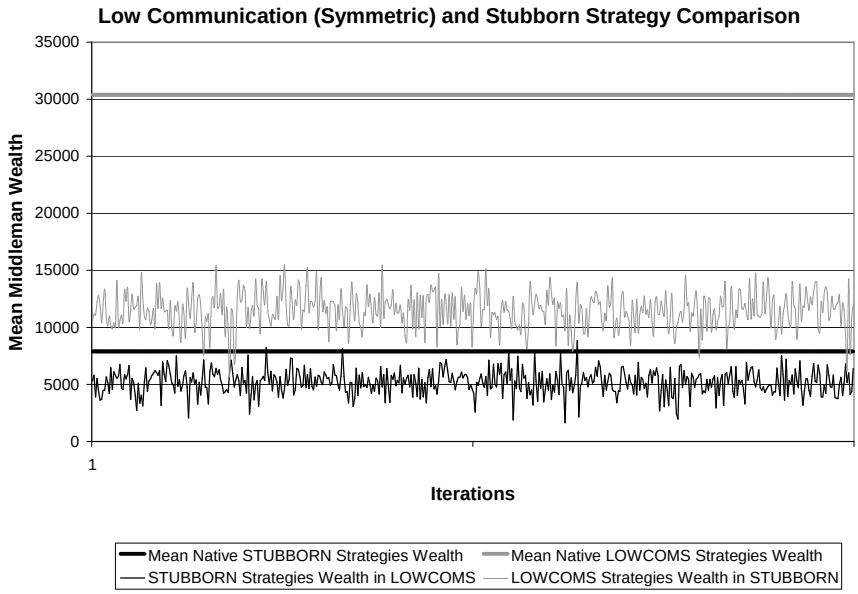

Figure 8.10: Strategy Cross Environment Comparison, STUBBORN And LOWCOMA

8.3.2.3 Comparison Of LOWCOMA And LOWAVAIL Evolved Strategies

Figure 8.11 shows the results of comparing middlemen strategies evolved in the *LOWAVAIL* and *LOWCOMA* environments.

Strategies evolved within the *LOWAVAIL* environment do not perform very effectively (in terms of the middleman wealth score) even within their native environment. Under *LOWCOMA* conditions they perform far worse and are entirely unable to make a profit making losses of around *5000.0*.

The strategies evolved under *LOWCOMA* conditions also have difficulty with this environment and perform far worse than in their native environment. They do, however, perform better than strategies evolved within this environment and are able to leverage a profit at around the *10000.0* level.

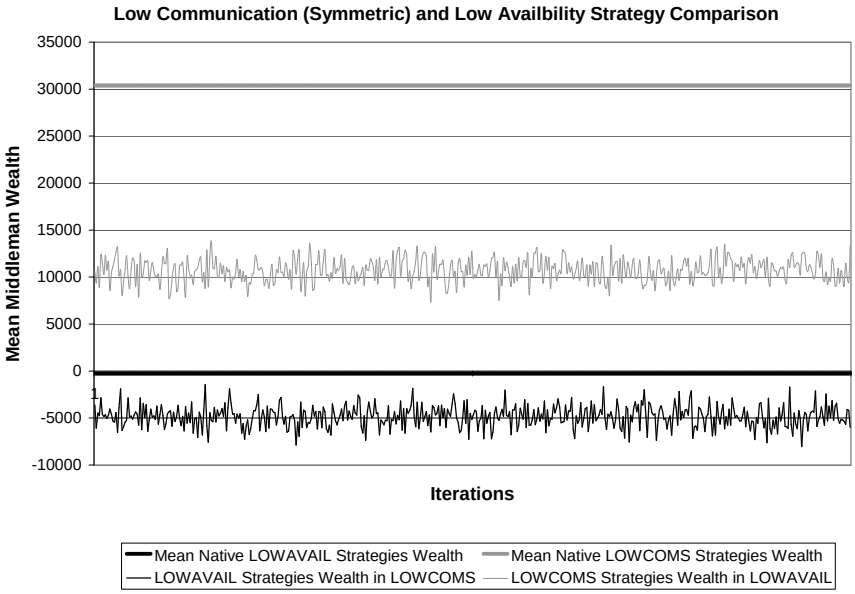

Figure 8.11: Strategy Cross Environment Comparison, LOWAVAIL And LOWCOMA

8.3.2.4 Comparison Of Strategies In Different Environments, Conclusions

In all the cases shown above strategies perform less well in environments that they were not evolved in. This supports the assertion that strategies evolve to specialise in their home environment, fulfilling experimental objective five (see Section 7.1).

While all strategies performed best in their own environments, strategies evolved under *LOWCOMA* conditions demonstrated an ability to perform very effectively in other environments. In non-native environments, *LOWCOMA* evolved strategies were able to perform better than strategies evolved within those environments.

This suggests that the *LOWCOMA* environment is more conducive to good strategy evolution. This is supported by the lower entropy and higher middleman wealth observed for strategy evolution under those conditions, see Section 8.2.4.

8.4 Conclusions

This chapter has reported results from twelve sets of experiments aimed at demonstrating the five experimental objectives introduced in Section 7.1.

Experiments were run using the SMSS described in Chapter 6 using the configurations outlined in Section 7.3.

Experimental objectives one and two seek to establish that the SMSS is able to evolve effective middlemen strategies based on the SSF-I1 introduced in Section 5.4. The results described in Section 8.1 show that middlemen operating within supply chains improve their performance over time as learning occurs. This is supported by a reduction in SMSS probability distribution entropy that correlates with increase mean middleman wealth.

Experimental objective three requires that the SMSS is able to evolve effective middlemen strategies under a variety of environmental conditions. This is demonstrated in Section 8.2 where results are report from experiments run using environments that vary from a set of baseline conditions.

Experimental objectives four and five seek to establish the adaptability of SSF-I1 by demonstrating that strategies will tend to specialise to the environments in which they evolve. Section 8.3 shows that this occurs through a combination of strategy difference testing and cross environment strategy testing.

Overall, the results of experimentation show that the SMSS, in combination with the SSF-I1 middleman strategy representation, provides an effective platform for the evolution of middlemen strategies. Different supply chain situations may be examined with variations in customer requirements, supplier configuration and assumptions

about the availability and use of the communication scheme being possible. Within these different environments successful middlemen strategies are able to evolve.

Chapter 9

Summary

This thesis has considered the problem of strategy formation for complex economic problems in the context of simple but non-trivial supply chains. Within these supply chains middlemen have been the focus of study since strategies for these participants face the greatest challenge in pursuing their own objectives while integrating the opportunities presented by both customer and suppliers.

The Simple Supply Chain Model (SSCM) has been introduced to describe supply chain situations, the SSCM Strategy Framework (SSF) to represent middlemen strategies and the SSCM Market Simulation System (SMSS) to evolve SSF-based strategies for SSCM-defined environments.

This chapter provides a summary of the presented work, lists the main contributions and discusses limitations and future research.

9.1 Summary Of The Presented Work

Chapter 1 discusses the motivation behind this work, a desire for techniques to study complex economic problems that are beyond the scope of traditional Game Theory to analyse with a view to strategy formation. This is important as automated strategies

for these types of environments are likely to become increasingly significant as trading electronically becomes more prevalent.

Chapter 2 discusses issues concerning this work in the context of other related research. Existing architectures for operating within supply chains are considered along with other economic problems and multi-agent scenarios. Consideration is given to existing communication schemes for negotiation between software agents and on evolutionary learning mechanisms in the context of strategy generation.

Chapter 3 provides a context-light introduction to concepts important to this work, including Game Theory, Evolutionary Computation and a review of the Trading Agent Competition.

Chapter 4 introduces the Simple Supply Chain Model (SSCM), a language for capturing simple non-trivial supply chain situations. The SSCM captures the start state of a supply chain in terms of participants' starting knowledge, tradable products and the communication scheme in place for interactions. Examples of the SSCM being used to model two familiar business models are provided and SSCM Variant One (SSCM-V1) is described. SSCM-V1 is a specific travel agent scenario designed to provide the basis of experimentation for this work. The SSCM-V1 Scenarios provide additional definition to the problems that will be tackled by partially specifying supply chain participant behaviour.

Chapter 5 describes the SSCM Strategy Framework (SSF), a middleman strategy framework capable of operating in SSCM-defined supply chain conditions. The SSF provides the basic operation framework for middlemen strategies, removing the need to evolve this from scratch. SSF Implementation One (SSF-I1) provides a complete SSF implementation that defines the mechanism for handling specific problems related

to SSCM-V1. The behaviour of an SSF-I1 based middleman strategy may be changed by adjusting its parameters. This adaptability allows middlemen strategies based on SSF-I1 to be improved through evolution of the controlling parameters.

Chapter 6 introduces the SSCM Market Simulation System (SMSS), a strategy evolution platform based on Population Based Incremental Learning with Guided Mutation (PBIL+GM). The SMSS is able to evolve SSF-based middlemen strategies for a given environment by operating a reinforcement learning feedback loop. In this feedback loop middlemen strategies are instantiated from a probability distribution, evaluated inside a supply chain simulation and the evaluations used as the basis for updating the same probability distribution. In this way improved middlemen strategies may be learnt over time. The primary output of the SMSS is a probability distribution capable of instantiating a reasonable SSF-based middleman strategy for the specific SSCM-defined environment.

Chapter 7 describes a set of twelve experiments conducted with the SMSS. These experiments seek to demonstrate that the SMSS is capable of evolving reasonable middlemen strategies under a variety of conditions and that these strategies are able to specialise to the environments in which they are formed. Demonstrating these capabilities is crucial to showing that the SMSS provides an effective strategy evolution platform and that the SSF is feasible as a framework for middlemen strategies.

In Chapter 8 the results from the experiments described in Chapter 7 are described and discussed. This chapter is broken in to three parts. In the first a baseline set of results is described and provides evidence for the capability of the SMSS to evolve effective strategies. The second part shows how middlemen strategies may be evolved under a range of different conditions affecting the supply chain start conditions and

the behaviour of participants. The final part demonstrates how middlemen strategies evolved under one set of conditions tend to specialise and operate less effectively when placed in other environments. The chapter concludes that experimentation has demonstrated the effectiveness of the SMSS as an evolutionary platform for middlemen strategies and the adaptability of the SSF for this task.

9.2 Contributions

The contributions of this work are threefold, a language for the specification of supply chain problems, a strategy framework for representing middlemen strategies operating within these supply chains and a strategy evolution platform for investigating the evolution of these strategies within different supply chain environments. This is novel and significant because:

The Simple Supply Chain (SSCM) can capture complex economic problems that:

- Contain considerable uncertainty for the participants.
- Does not impose behavioural constrains on participants.
- Model real world problems of current interest.
- Are difficult to consider using traditional Game Theoretic techniques.

The SSCM Strategy Framework (SSF) is a feasible framework for strategies operating within SSCM supply chains and provides:

- The flexibility to adapt to different supply chain conditions.
- A system conducive to evolution by providing the core strategy structure so that it does not need to be recreated and,

- Supplying parameters that define the actual behaviour of the strategy.

The SSCM Market Simulation System (SMSS) is an effective strategy evolution platform that:

- Produces reasonable middlemen strategies.
- Allows investigation of strategies under a wide range of supply chain conditions.

9.3 Limitations

The SSCM provides a means to capture interesting economic problems for study. The SSF and SMSS provide an effective way to represent and evolve successful middlemen strategies for these modelled economic problems.

Although the SSCM, SSF and SMSS have been shown to be effective they have some limitations, these are considered below:

The SSCM provides an effective way to capture a wide variety of supply chain situations, however:

- Longer chains including several layers of middlemen are not represented by the SSCM in its current form.
- There is no provision for the representation of problem specific supply chain information that may have a bearing on participant behaviour or performance. This form of information has been considered part of a participants behaviour here.

- The representation of communication schemes does not capture the inter-dependencies of utterances, only the utterance types themselves.
- Relationships between products are not modelled, this again being considered as part of the participant behaviour.

The SSF is a feasible and flexible framework for representing middlemen strategies within supply chains, however:

- The design assumes an alternating offers protocol for negotiation with both customers and suppliers. The SSCM may represent other communication schemes, the efficacy of the SSF to be adapted to these schemes is unclear.
- Assumptions about how customers will use the negotiation protocol to communicate their requirements have been made. More generally, the SSF does not attempt to mitigate the possibility of customers finding an alternative supplier, something that may be necessary in other problem domains.

The SMSS is an effective platform for the evolution of middlemen strategies, however:

- The PBIL+GM mechanism assumes the independence of the parameter variables that it is evolving. This simplifying assumption reduces the complexity of the probability distribution needed for the representation of middlemen strategies at the expense of this potentially important source of information.

9.4 Future Research

This thesis introduces the SSCM, a method for capturing interesting economic problems for study, the SSF, a framework for representing strategies in those environments, and the SMSS, a platform for the evolution and investigation of such strategies.

While each of these has been shown to be effective, this section summarises a number of further research directions in this important and challenging area:

In relation to the SSCM :

- The SSCM may be used to model other business scenarios. The bandwidth trading scenario, discussed in Section 4.2.2, could be further extended and investigated with greater consideration given to the representation of product attributes, such as quality of service and how participants interact. Other problems, such as the allocation of resources for wireless network access or grid computing, might also be considered.
- The SSCM defines supply chain with one layer of middlemen, this could be extended allowing the capture of longer chains.
- The SSCM could be extended to capture more information about supply chain participants' starting knowledge, helping to formalise the specification of problem-specific information.

For the SSF :

- The SSF could be extended to help mitigate the affects of customers looking for and finding alternative suppliers elsewhere.

- Extensions that consider alternative communication schemes might also prove advantageous.
- SSF-I1 provides a problem-specific implementation of the SSF. Alternative implementations could be considered that make use of different mechanisms for product price estimation, customer requirement grouping and supplier selection. Different negotiation mechanisms could be considered, a full implementation of the combined tactics approach presented by Matos, [67], for instance.

With regards to the SMSS :

- More complex customer and supplier behaviour could be considered, increasing the realism of the supply chains investigated and the challenge faced by evolving middlemen strategies.
- In parallel with this, customer and supplier strategies might also be evolved. What strategies emerge for each participant type under these conditions, their stability, interdependencies and effectiveness, would prove interesting.

Appendix

The following appendix provide additional supportive material to main body of the thesis.

Appendix A contains additional information about the operation of the SSCM Strategy Framework discussed in Chapter 5.

Appendix B provides information about the structure, operation and implementation of the SSCM Market Simulation System described in Chapter 6.

Appendix C supplies further details of analytical techniques applied to the results from the SMSS as well as providing supporting evidence for the configuration options selected for experimentation. This relates to the Chapter 7 of the main thesis body.

Further material relating to the thesis is provided on the associated DVD. This includes the SMSS source code, configuration files and raw results data.

Appendix A

SSF Details

This Appendix contains additional details about the SSCM Strategy Framework, introduced in Chapter 5, that was not included in the main text for editorial reasons. These details comprise a discussion of the core SSF strategy and include information on the representation of world state and the algorithms used.

The core of the SSF strategy is designed to tackle the problems presented by the SSCM-V1 scenarios. To this end the strategy must decide which customer requirements it should attempt to fulfill, negotiate with suppliers for the products necessary to do this and report success or failure of those negotiations back to the customers at the earliest opportunity. The decisions of the SSF are therefore motivated and lead by the customer requirements.

The SSF algorithm is conceived as an ongoing process that only terminates when the market ends. During each iteration the SSF updates its perceived world state before generating and sending at most one outbound communication. Ideally the SSF algorithm will iterate many times within a single SSCM market time-step, preferably enough times to potentially send the number of messages specified by the upper outbound communication limit imposed via the SSCM.

In between iterations of the SSF, communications may be received that will affect subsequent behaviour. These messages, including new requirements from customers and responses from suppliers, are stored for processing at the beginning of the next iteration.

The SSCM time-steps are considered to be applied external, updating the strategies perceived time and the amount of outbound communication budget remaining.

The top-level of the SSF is shown in Algorithm 5 below.

Algorithm 5 SSF Algorithm Top Level

```
1: while Market Continues do
2:     Update Perceived World State
3:     Generate And Send A Message
4: end while
```

The first step in the SSF algorithm takes all inbound communications and and changes in SSCM time-step into account to update the perceived world state (see Section A.2). The second step uses this information to generate and send at most one outbound communication - if one is sent, the communication will itself update the perceived world state (see Section A.10).

To discuss the SSF further it is first necessary to define the SSF world state more specifically, this is done in Section A.1 below.

A.1 Representing The Perceived World State

The SSF world state provides all the information from which decisions are made and, ultimately, messages generated and sent. The state comprises the initial information

provided from the SSCM, information derived about products, group information (including what customer requirements and supplier negotiations are collected together), transaction histories and the middleman currency balance.

The world state from the perspective of an SSF strategy can be considered as follows (Definition A.1.1).

Definition A.1.1. SSF, Strategy Perceived World State (Top Level)

$$MiddlemanWorldState = (SSCM\_Info, Product\_Info, Group\_Info, Transaction\_History, FreeProducts, Balance, CommsBudget, CurrentTimeStep, Scanning, Parameters)$$

$SSCM\_Info$	-	The information supplied to the middleman from as part of the SSCM.
$Product\_Info$	-	Further information about products
$Group\_Info$	-	The information based on the collection of customer and supplier negotiations.
$Transaction\_History$	-	The record of all past negotiations no longer playing any other role.
$FreeProducts$	-	This is a $ProductSet$ (see Definition A.1.9). This set contains all products that are unattached to any group and thus could be used to help fulfill customer requirements without the need for negotiation
$Balance$	-	The simple currency balance of the strategy at this point (a real value).
$CommsBudget$	-	The amount of outbound communications still allowed
$CurrentTimeStep$	-	The current SSCM time-step
$Scanning$	-	A boolean, determines the mode of the SSF algorithm. When in scanning mode the SSF is attempting to fill $Product\_Info$
$Parameters$	-	The set of implementation specific parameters required for the strategy to operate.

The SSCM-based information $SSCM\_Info$ is defined below (Definition A.1.2). This is taken directly from the SSCM definition for this middleman strategy within the supply chain. Because the SSF is designed to tackle SSCM-V1S1-3 there is no need to consider known customers as part of this.

Definition A.1.2. SSF, World State, SSCM Information

$$SSCM\_Info = (KnownSuppliers, KnownProducts, MaxCommsBudget)$$

$KnownSuppliers$	-	Equates to KS_x in the SSCM where this is the strategy for middleman x
$KnownProducts$	-	Equates to P in the SSCM
$MaxCommsBudget$	-	Equates to $MComOut_x$ in the SSCM where this is the strategy for middleman x

The $Product\_Info$ is derived from interactions with suppliers, it complements the information provided initially as part of the SSCM. To this end it specifies the known suppliers for each product and an estimated product value.

Definition A.1.3. SSF, World State, Product Information

$$Product\_Info = \{(Product, KnownProductSuppliers, EstimatedProductValue), ...\}$$

$Product$	-	The product (taken from $KnownProduct$ that this information refers to
$KnownProductSuppliers$	-	The suppliers know to be able to provide this product $KnownProductSuppliers \subseteq KnownSuppliers$
$EstimatedProductValue$	-	The estimated value, or cost, of obtaining this product from suppliers

$Product\_Info$ contains one tuple for each in $KnownProducts$, no tuples have matching $Product$s. $EstimatedProductValue$ is determined by some SSF implementation specific mechanism.

$Group\_Info$ is the collection of information about ongoing negotiations with customers and suppliers. This is elaborated on further below.

Definition A.1.4. SSF, World State, Group Information (Top)

$$Group\_Info = (FailureGroup, PreNegotiationGroup, BasicGroups)$$

$FailureGroup$	-	Information about the group that handles all failing negotiations
$PreNegotiationGroup$	-	Information about the group that handles customer requirement renegotiation
$BasicGroups$	-	The collection of groups handling all supplier negotiations for fulfilling customer requirements

The $Transaction\_History$ is the collection of past negotiations with both customers and suppliers. All negotiations that are currently ongoing or have a direct bearing on current decision making are held as part of the $Group\_Info$ above. The set of past transactions is important as a source of information to help inform current decision making.

Definition A.1.5. SSF, World State, Transaction History

$$Transaction\_History = \{Negotiation_1, Negotiation_2, ..., Negotiation_{pnn}\}$$

$Negotiation_x$	-	A past negotiation
pnn	-	The number of historical transactions

This high-level view of the SSF's world state information is illustrated in Figure A.1 below.

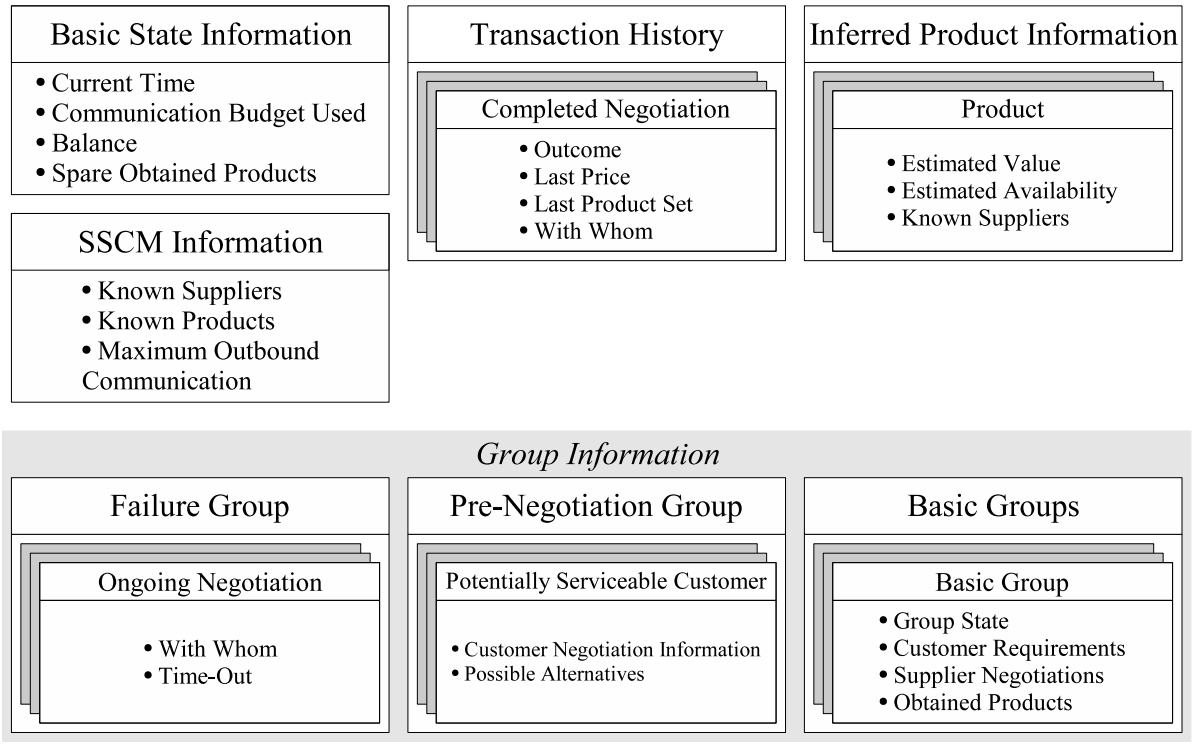

Figure A.1: Overview Of The SSF World State Information

A.1.1 Representing Negotiations

Negotiations are one of the building blocks of the SSF. Based on the form described in Section 4.3.4, negotiations act as the mechanism through which customers, middlemen and suppliers interact. Customer negotiations are often referred to as customer requirements throughout the thesis.

Definition A.1.6. SSF, World State, Negotiation

$$Negotiation = (Buyer, Seller, State, Time - Out, Offer\_History, NegID)$$

$Buyer$	-	The buyer
$Seller$	-	The seller
$State$	-	The negotiation state drawn from the set $\{Not\_Waiting$, $Waiting$, $Accepted(Us)$, $Accepted(Them)$, $Rejected(Us)$, $Rejected(Them)$, $Implicit\_Reject(Us)$, $Implicit\_Reject(Them)\}$
$Time - Out$	-	The SSCM market time-step the negotiation will be considered to have been implicitly rejected
$Offer\_History$	-	The set of offers and counter offers made
$NegID$	-	A unique negotiation identifier

The $Buyer$ and $Seller$ determine between whom the negotiation is being held. The $State$ specifies the current condition of the negotiation, a $Waiting$ or $Not\_Waiting$ negotiation is one that is still ongoing, all other states indicate that the negotiation has finished and no further actions can be taken or should be expected. If $Waiting$ a response is being waited for until the $Time - Out$ time is reached. If $Not\_Waiting$ a response is expected by the other party before $Time - Out$. A negotiation must have a response before the $Time - Out$ time-step of the SSCM market or be implicitly rejected. A customer negotiation $CusNeg$ is a negotiation in which the $Seller$ is the participant using this strategy. A supplier negotiation $SupNeg$ is one in which $Buyer$ is the participant using this strategy. Customer negotiation are also referred to as

requirements. Strictly speaking a customer requirement is the latest offer made by the customer to the middleman as part of a $CusNeg$, the requirement comprising the required product set and the value ascribed by the customer to this.

The $Offer\_History$ of a negotiation is a record of each offer and counter offer made by both parties. If a negotiation is concluded successfully the last offer in the history represents the deal ultimately struck specifying both the currency and products transferred.

Definition A.1.7. SSF, World State, Negotiation Offer History

$$Offer\_History = \{Offer_1, Offer_2, ..., Offer_{ohn}\}$$

$Offer_x$	-	The offer made/sent
ohn	-	The number of offers in this history

An individual offer in an $Offer\_History$ is tuple comprising a value, and product set. The value is either the amount the buyer is willing to pay for the product set or the amount the seller is seeking to obtain for the same.

Definition A.1.8. SSF, World State, Negotiation Offer

$$Offer = (Value, ProductSet)$$

$Value$	-	The value ascribed by one part to the specified product set
$ProductSet$	-	The set of products currently under consideration

The $ProductSet$ of an $Offer$ comprises the total product set under consideration for this offer. It need not be the same as previous product sets in previous offers, though this would normally be the case. The set comprises number tuples specifying the type of product, amount and time required

Definition A.1.9. SSF, World State, Negotiation Product Set

$$ProductSet = \{(Type, Time, Amount), (Type, Time, Amount), ...\}$$

$Type$	-	The product type, on of those available as part of the SSCM
$Time$	-	The market time-step for which this product is required
$Amount$	-	The amount required

Within a $ProductSet$ only one tuple of a specific Type and Time are allowed, if multiple amounts of the same product at the same time are required these must be amalgamated in to a single tuple.

A.1.2 The Failure Group

Referring back to the $Group\_Info$ (Definition A.1.4), the $FailureGroup$ information is simply the set of current negotiations to which the strategy requires a reject message be sent. Negotiations in the failure group must be of the status $Not\_Waiting$ (i.e. the strategy is able to respond), but may be of either customer or supplier type.

Definition A.1.10. SSF, World State, Failure Group Information

$$FailureGroup = \{Negotiation_1, Negotiation_2, ..., Negotiation_{fgn}\}$$

$Negotiation_x$	-	A negotiation due for reject message send
fgn	-	The number of negotiations in the failure group

A.1.3 The Pre-Negotiation Group

The $PreNegotiationGroup$ information relates to the customer negotiations for which the strategy is attempting to find an alternative product set. Here customer negotiations are coupled with an estimated value and set of possible alternatives.

Definition A.1.11. SSF, World State, Pre-Negotiation Group Information

$$\begin{aligned} PreNegotiationGroup &= \{(CusNeg, EstimatedValue, Alternatives), \\ &\quad (CusNeg, EstimatedValue, Alternatives), ...\} \end{aligned}$$

$CusNeg$	-	A customer negotiation for which the strategy is attempting to find an alternative product set
$EstimatedValue$	-	The estimated value of the customer to the middleman
$Alternatives$	-	A set of possible alternatives to the original customer requirement

Customer negotiations may be of status $Waiting$ or $Not\_Waiting$ depending on how the renegotiation is going. The $EstimatedValue$ of a customer negotiation is determined by some external SSF mechanism.

The *Alternatives* for Pre-Negotiation Group information tuple is a set of possible alternatives to the product sets originally specified by the customer.

Definition A.1.12. SSF, World State, Alternatives

$$Alternatives = \{Alternative_1, Alterantive_2, ..., Alternative_{can}\}$$

$Alternative_x$	-	An alternative to the original customer requirement
can	-	The number of alternatives

The expression of alternatives is SSF implementation specific. It could, for instance, simply be a $ProductSet$ which deviates from that originally specified by the customer or something more complex.

A.1.4 The Basic Groups

The basic groups maintained by the SSF are more dynamic than those of the failure or pre-negotiation groups. The *BasicGroups* information maintained by the SSF relates to the strategies attempts to fulfill the requirements of customers through negotiations with suppliers. Multiple basic groups may exist within the strategy at any given time dealing with different sets of requirements separately. The basic groups can be subdivided down three lines. These are Inactive groups, Active groups and

Completion groups. Inactive groups act as the collection points for new customer requirements, Active groups are involved in negotiations with suppliers and Completion groups are responsible for ultimately reporting to customers. Active groups can be further subdivided into Active and Waiting groups, an Active group being actually involved in negotiations, a Waiting group holding off on negotiations for the time being. Completion groups likewise can be considered as either Successful or Unsuccessful. Successful groups Completion groups are attempting to respond positively to customers, Unsuccessful ones are trying to clear any outstanding negotiations to allow for the reporting of failure. The transition between basic group states is shown in Figure A.2.

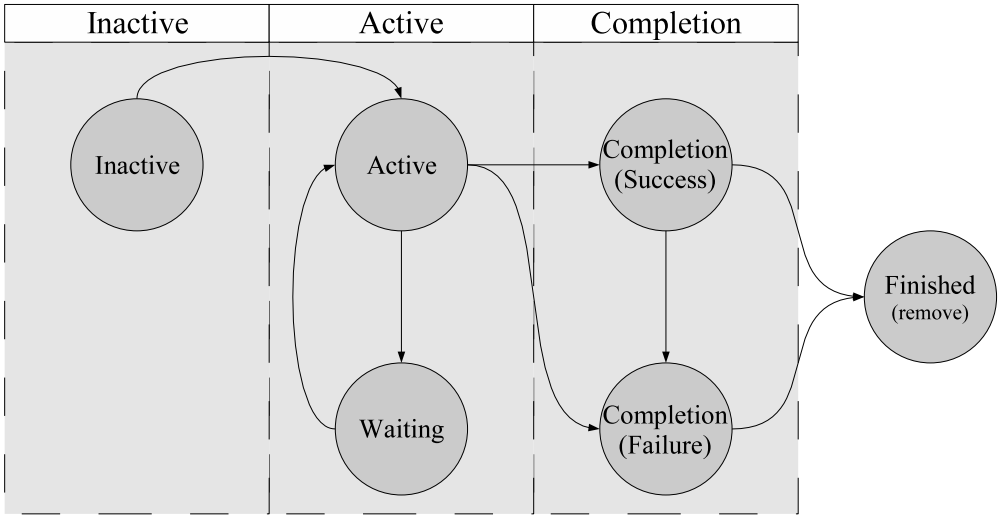

Figure A.2: SSF Group State Transitions

Definition A.1.13. SSF, World State, Basic Groups

$$BasicGroups = \{BasicGroup_1, BasicGroup_2, ..., BasicGroup_{bgn}\}$$

$BasicGroup_x$	-	A basic group
bgn	-	The current number of basic groups

Definition A.1.14. SSF, World State, Basic Group

$$BasicGroup = (CustomerInfo, SupplierInfo, GroupStatus)$$

$CustomerInfo$	-	Information relating to the customer negotiations
$SupplierInfo$	-	Information relating to negotiating with the supplier
$GroupStatus$	-	The status of the group drawn from the set $\{Inactive, Active, Waiting, Completion\_Success, Completion\_Failure\}$

A basic group in the $Inactive$ state has no $SupplierInfo$ and all $FoundFor$ and $Remove$ variables are false. As such, an Inactive group is purely a mechanism for collecting together customer requirements before negotiations start. A basic group in the $Active$ and $Waiting$ states has transitioned from being $Inactive$, the $SupplierInfo$ being built. New customer negotiation may not be added but can be lost.

The customer information maintained by a basic group is similar to that of a Pre-Negotiation group but enhanced to take in to account the ongoing supplier negotiations.

Definition A.1.15. SSF, World State, Customer Information

$$\begin{aligned} CustomerInfo \quad = \quad & \{(CusNeg, EstimatedValue, FoundFor, Remove), \\ & (CusNeg, EstimatedValue, FoundFor), ...\} \end{aligned}$$

$CusNeg$	-	Customer negotiation
$EstimatedValue$	-	Estimated value of the customer negotiation (Real)
$FoundFor$	-	If products have been obtained for this customer (Boolean)
$Remove$	-	Is this customer negotiation marked for convenient removal (Boolean)

The supplier information maintained by a basic group.

Definition A.1.16. SSF, World State, Supplier Information

$$SupplierInfo = (CashPool, ObtainedProducts, ProductProfiles)$$

$CashPool$	-	The funds available for negotiation. This is initially the sum of differences between the $EstimatedValue$ of a $CusNeg$ and the $Value$ ascribed by the customer to it's required $ProductSet$. The value should always be positive
$ObtainedProducts$	-	This is a $ProductSet$. This set contains all products obtained so far in the process of attempting to fulfill the customer negotiations
$ProductProfiles$	-	The product profiles collect together the information used in the attempts to negotiate for the required customer product sets

The $CashPool$ is initially derived from the sum of differences as expressed above. In the course of the strategies operation this value is depleted to fund the individual negotiations but should ultimately become positive over time as less funding becomes required. This is discussed in more detail later. The $CashPool$ is independent of the strategy $Balance$ and is used for decision making purposes only. The $ObtainedProducts$ are those that will ultimately be used to fulfill customer requirements as far as possible. When the group finally ends and is removed any remainder is returned to the $FreeProducts$ shown in Definition A.1.1.

The $ProductProfiles$ of the $SupplierInfo$ contain information about the attempts to obtain products for customers. Each profile is tied to a specific product type, to this end there will be no more profiles than the number of product types, possibly less.

Definition A.1.17. SSF, World State, Product Profiles

$$ProductProfiles = \{Profile_1, Profile_2, ..., Profile_{npp}\}$$

$Profile_x$	-	A product profile
npp	-	The number of product profiles

Definition A.1.18. SSF, World State, Product Profile

$$\begin{aligned} Profile \;=\; & (StillToFind, ProfileBudget, UsedBudget, \\ & NeedMoreFunds, SupplierNegotiations, UntriedSuppliers) \end{aligned}$$

$StillToFind$	-	A $ProductSet$. The products that still need to be found and aren't currently being sort as part of existing $SupplierNegotiations$
$ProfileBudget$	-	The amount of funds assigned to this profile to negotiate with suppliers
$UsedBudget$	-	The amount of the $ProfileBudget$ currently being used
$NeedMoreFunds$	-	If this profile currently needs more funds for ongoing negotiations
$SupplierNegotiations$	-	Information about ongoing supplier negotiations
$UntriedSuppliers$	-	The set of suppliers that have not been approached by this group to help fulfill the customer requirements. Initially this is filled from the $KnownSuppliers$, see Definition A.1.2

The $SupplierNegotiations$ information is a set of tuples containing the negotiation being undertaken and the product set being sort. The product set being sort need not be the one last requested by the strategy if some external effect has caused the requirements to change - in these circumstances the set represents that which the strategy now deems necessary and will try to obtain when the time comes to make a response.

Definition A.1.19. SSF, World State, Supplier Negotiations

$$SupplierNegotiations = \{(SupNeg, SortSet), (SupNeg, SortSet), ...\}$$

$SupNeg$	-	A supplier negotiation
$SortSet$	-	A $ProductSet$. The set of products this negotiation should be aiming to secure

A.2 Updating The World State

The SSF World State is the basis for deciding what, if any, outbound communication should be made. Before this can be decided the World State must be brought up-to-date taking in to account all communication received from other participants and any other external effects. To bring the world state up to date the inbound message queue

Table A.1: Update Perceived World State

Function	*Output*	*Input*
ProcessInboundMessageQueue	-	-
Empty the inbound message queue updating the world state in the process. See Section A.3.2.		
DoGroupStateChangeCheck	-	-
Reconcile the World State then determine if any group statuses need to change. See Algorithm 27.		

must be incrementally emptied, updating the state accordingly for each message. Having made updates to the world state according to the individual communications the overall state is reevaluated to find further changes that should be made in light of the information. Algorithm 6, below, shows this overall update process. Algorithm 7 and Algorithm 27 in Section A.3.2 and Section A.9 respectively elaborate on this process further. The message queue from which messages are drawn and the format of those messages is elaborated upon in Section A.3.

Algorithm 6 Update Perceived World State

1: ProcessInboundMessageQueue
2: DoGroupStateChangeCheck

A.3 Relating To The Outside World, The Inbound Message Queue

The SSF World State allows the SSF to make a decision about what to do next. Part of that world state information is provide as initial conditions from the SSCM being used. The rest of this information must come from the interaction with other participants. As discussed earlier, communication between participants is handled via a negotiation based mechanism. Any inbound communication must be added to the

inbound message queue and used to update the world state on the next algorithm iteration. Along with participant negotiation messages the queue also handles inbound timing change information. As mentioned in Section 5.3.3 timing synchronisation is one assumption on which the SSF is built, this ensures no messages for a timed out negotiation will ever be sent.

A.3.1 Defining The Inbound Message Queue

The inbound message queue ($InboundMessageQueue$) is defined below. The queue is a set of time ordered received messages

Definition A.3.1. SSF, Inbound Message Queue

$$InboundMessageQueue = \{Message_1, Message_2, Message_{imn}\}$$

$Message_x$	-	An inbound message
imn	-	The total number of messages in the queue at this time

Messages themselves take two basic forms, either participant to participant (negotiation) messages or those related to timing. A message comprises two parts a header specifying where the message originated and a contents part.

Definition A.3.2. SSF, Message

$$Message = (Header, Contents)$$

$Header$	-	Origination information, comprises $(From, FromType)$, the originator of the message and if they are a market participant or not
$Contents$	-	The actual message contents

The message $Header$ allows the SSF to determine what type of message the contents will contain. Any message from a market participant must relate to a negotiation, a non-market participant message will relate to timing. If the $Contents$ of a message is negotiation related the following information is contained.

Definition A.3.3. SSF, Message Contents, Inter-Participant

$$Contents = (NegID, MesssageType, Offer, TimeOut)$$

$NegID$	-	Uniquely identifies the negotiation to which this message relates
$MesssageType$	-	Type drawn from the set $\{Offer$, *Accept*, *Reject*$\}$
$Offer$	-	A negotiation offer, if necessary (i.e. $MessageType = Offer$)
$TimeOut$	-	If an offer, the time-step on which it will become invalid

A $NegID$ that does not relate to any existing negotiation indicates that this a new negotiation being opened. In this case the $MessageType$ must be $Offer$.

Definition A.3.4. SSF, Message Contents, Timing

$$Contents = (NewTimeStep)$$

$NewTimeStep$	-	The new SSCM time-step

Having defined the message queue and the messages contained therein it is possible to further define the helper functions used as part of the SSF World State update process.

A.3.2 Processing The Inbound Message Queue

The first stage of the SSF World State update process is the processing of all received communications, this is elaborated upon here.

For participants, received communication is either new customer negotiations (customer requirements) being received or ongoing customer or supplier negotiations. In the case of new customer negotiations these must be filtered for those the middleman believes can be fulfilled profitably. Ongoing customer negotiations, those to find alternative product sets may result in the requirement being moved to the filtration process. Supplier negotiation messages may have more complex implications. In the

normal course of events a supplier message would simply update the world state to reflect the latest offer however, it may indicate that products have been obtained or that some of a required product set is unobtainable. In this last case, if an alternative can not be found, the implication for the motivating customer requirements must then be considered.

Non participant communication relates to changes in timing being applied. A new SSCM time step refreshes the available communication budget and may also cause some negotiations to go in to implicit-reject (or time-out). Further, the change in basic group status from Inactive through to Completion is affected by the timing.

Algorithm 7 shows the outline of the SSF World State message processing procedure, the message queue elements of this procedure are elaborated on further in Table A.3.2. Table A.2 lists the sections that discuss World State message updated relevant to Customers, Suppliers and Timing.

Algorithm 7 Processing The Inbound Message Queue

```
1: while InBoundMessagesRemaining do
2:    msg = RemoveOldestMessageFromInboundQueue
3:    if IsCustomerMessage (msg) then
4:       if NewNegotiation (msg) then
5:          NewCustomerNegotiationHandling (msg)
6:       else
7:          AlternativeRequirementHandling (msg)
8:       end if
9:    else if IsSupplierMessage (msg) then
10:      SupplierResponseHandling (msg)
11:   else
12:      DoTimeStepHandling
13:   end if
14: end while
```

Table A.2: SSF World State, Customer, Supplier and Timing updates

Update Source	*Section*
Customer	NewCustomerNegotiationHandling, for sorting of new negotiations see Section A.4 AlternativeRequirementHandling, for response handling of alternatives response see Section A.5
Supplier	SupplierResponseHandling, for all the consequences of a supplier negotiation message see Section A.6
Other	DoTimeStepHandling, performs all the SSCM time-step changes necessary. See Section A.7.1

Table A.3: Message Queue Helper Functions

Function	*Output*	*Input*
InBoundMessagesRemaining	Boolean	-
True if $imn > 0$		
RemoveOldestMessageFromInboundQueue	Message	-
Removes the oldest message in the message queue and returns it for processing, reducing imn by 1		
IsCustomerMessage	Boolean	Message
True if the $Header$, $From$ is not part of the $KnownSupplier$ set		
NewNegotiation	Boolean	Message
True if $NegID$ is not recorded within the SSF World State $Group\_Info$		
IsSupplierMessage	Boolean	Message
True if the $Header$, $From$ is a part of the $KnownSupplier$ set		

A.4 Selecting Profitable Customer Requirements

The objective of the middleman strategy is to leverage a profit from customers by fulfilling their requirements through negotiation with suppliers. The first step in this process is to determine which, if any, of the received customer negotiations are likely to prove profitable. To this end all new customer negotiations go through a three step filtration process. To attempt to fulfill a requirement the strategy must believe that it is profitable to do so, that the products are available to do so and that there is sufficient time to undertake the necessary negotiations.

The aim of the filtration process is to assign new negotiations ($CusNeg$s) to the relevant part of the SSF World State for a response. A customer negotiation that is considered to be unprofitable or for which it seems there would be insufficient time to fulfill will be assigned to the Failure Group. A customer negotiation that is considered likely to be profitable, for which there is time to negotiation but for which products maybe unavailable is assigned to the Pre-Negotiation group. The logic here being that a similar, alternative, requirement is likely to be acceptable to the customer and still profitable for the middleman. A customer negotiation that fulfills all the requirements is assigned to a Basic Group. In this case the negotiation may either be assigned to an existing Inactive basic group or a new Inactive basic group may be created to accommodate it.

The mechanisms that help determine potential profitability, time requirements and the availability of products are implementation specific and are listed in Section 5.4.1. While the mechanism that determines the estimated value is defined here, the mechanism that establishes the values on which that mechanism is based

Table A.4: NewCustomerNegotiationHandling, Helper Functions

Function	*Output*	*Input*
CreateNegotiationFromMessage	Negotiation	Message
Create a new negotiation based upon this message, filling in the buyer, seller, timeout and offer history as appropriate		

is problem-specific, this mechanism is. In assigning a customer negotiation to the Pre-Negotiation group possible alternatives must be generated. This mechanism is necessarily problem-specific as the way in which the products in a $ProductSet$ interact may vary. This is listed as part of Section 5.3.4.

Algorithm 8 shows the layout of the NewCustomerNegotiationHandling function of the SFF Perceived World State Update procedure. The procedure first sets up a basic Negotiation tuple based upon the message and then passes this to the filtration process shown in Algorithm 9.

Algorithm 8 NewCustomerNegotiationHandling (msg)

1: neg = CreateNegotiationFromMessage (msg)
2: FilterNegotiation (neg)

The filtration process uses a number of helper functions to complete the process, these are described in further detail in Table below.

The AssignToBasicGroup function must determine to which $BasicGroup$ in $Group\_Info$ a negotiation should be assigned. Only basic groups in the $Inactive$ state may be considered for this purpose. Alternatively a new basic group can be created and the negotiation assigned to this - this allows basic groups to come in to existence to begin with and allows different collections of customer requirements to be fulfilled independently. The mechanisms that decide which group the negotiation should be

Table A.5: Negotiation Filtration Helper Functions

Function	*Output*	*Input*
SufficientTimeToNegotiation	Boolean	CusNeg
An implementation specific mechanism see Section 5.3.5		
LatestOfferValue	Real	Negotiation
The value associated with the latest offer in this negotiation		
LatestProductSet	ProductSet	Negotiation
Given a negotiation, return the latest product set		
DetermineEstimatedPrice	Real	ProductSet
Given the product set determine an estimated value. Do this by the summation of estimated values of each tuple in the $ProductSet$. An individual tuple value is determined by multiplying the amount by the product types $EstimatedProductValue$ found in $Product\_Info$		
ProductSetFeasible	Boolean	CusNeg
An implementation specific mechanism to determine if the latest product set of the negotiation is probable possible to obtain from suppliers. See Section 5.3.5		
AssignToBasicGroup	-	CusNeg, EstimatedValue
Assigns the negotiation to a basic group, detailed further below		
AssignToPreNegotiationGroup	-	CusNeg, EstimatedValue
Assigns the negotiation to the pre-negotiation group, detailed further below		
AssignToFailureGroup	-	Negotiation
Assigns the negotiation to the $FailureGroup$ information, the negotiation tuple is added to the set of negotiations in the $FailureGroup$ set, incrementing fgn		

Algorithm 9 FilterNegotiation (*neg*)

```
1: if SufficientTimeToNegotiation (neg) then
2:    value = LatestOfferValue(neg) - DetermineEstimatedPrice
      (LatestProductSet(neg))
3:    if value > 0.0 then
4:       if ProductSetFeasible (neg) then
5:          AssignToBasicGroup (neg, value)
6:       else
7:          AssignToPreNegotiationGroup (neg, value)
8:       end if
9:    else
10:      AssignToFailureGroup (neg)
11:   end if
12: else
13:   AssignToFailureGroup (neg)
14: end if
```

applied to are implementation specific but are brought together in the manner show in Algorithm 10 and Table A.6.

Algorithm 10 AssignToBasicGroup (*neg*, $EstimatedValue$)

```
1: if {x∃BasicGroups : x.GroupStatus = Inactive AND
   GroupWillTakeNegotiation(x, neg, EstimatedValue)} ≠ {} then
2:    BasicGroup = BestBasicGroupToJoin (neg, EstimatedValue)
3:    AddNegotiationToBasicGroup (neg, EstimatedValue, BasicGroup)
4: else
5:    CreateBasicGroupFor (neg, EstimatedValue)
6: end if
```

The AssignToPreNegotiationGroup function involves the creation of a new tuple for the $CusNeg$ in the $PreNegotiaionGroup$. The complication in this process is the need to create generate a series of possible alternatives to the original $ProductSet$ proposed by the customer. This mechanism, CreateAlternativesFor, shown in Algorithm 11 and Table A.7, is problem-specific since the relationship between products in

Table A.6: AssignToBasicGroup Function Elements

Function	*Output*	*Input*
GroupWillTakeNegotiation	Boolean	BasicGroup, CusNeg, Value
Determines if an *Inactive*, *BasicGroup*s should/could take this negotiation taking in to account its estimated value. Implementation specific, see Section 5.3.5		
BestBasicGroupToJoin	BasicGroup	CusNeg, Value
Assuming there is an existing *Inactive*, *BasicGroup* able to take the customer negotiation, find and return the best possibility. Implementation specific, see Section 5.3.5		
AddNegotiationToBasicGroup	-	CusNeg, Value, BasicGroup
Add a new tuple to the $Customer\_Info$ of the *BasicGroup* comprising the supplier negotiation, its estimated value and a boolean $FoundFor$ value of false		
CreateBasicGroupFor	-	CusNeg, Value
Create a new *BasicGroup* tuple with empty $Supplier\_Info$ and a *GroupStatus* of *Inactive*. A single tuple will be added to the $Customer\_Info$ comprising the negotiation supplier, the associated value estimate and a boolean $FoundFor$ setting of false. This new *BasicGroup* will be added to the *BasicGroups* set of $Group\_Info$ and the counter, *bgn* incremented as a result		

Table A.7: AssignToPreNegotiationGroup Function Elements

Function	*Output*	*Input*
CreateAlternativesFor	Alternatives	CusNeg, Value
Generates a set of *Alternatives* to be used by the *PreNegotiationGroup*. Problem-specific, see Section 5.3.4		
AddToPreNegotiationGroup	-	CusNeg, Value, Alternatives
Adds a tuple of the *PreNegotiationGroup* of $Group\_Info$. The tuple comprises the customer negotiation, its estimated value and a set of supplied possible alternatives		

a customer specified *ProductSet* is likely to be different for different modelled problems. If no alternatives to the proposed set are possible (due to improbable product availability for example) the negotiation will instead be assign to the *FailureGroup*.

Algorithm 11 AssignToPreNegotiationGroup (*neg*, $EstimatedValue$)

```
1: Alternatives = CreateAlternativesFor (neg)
2: if Alternatives = {} then
3:    AssignToFailureGroup (neg)
4: else
5:    AddToPreNegotiationGroup (neg, EstimatedValue, Alternatives)
6: end if
```

A.5 Negotiation With Customers To Find Alternatives

If a customers initial offer in a negotiation is found to profitable, timely but the products needed are probably unavailable the SSF will attempt to renegotiate the requirement to one that is similar but which can be fulfilled.

Within the SSF World State, the *PreNegotiationGroup* holds information about these negotiations. For each negotiation a set of alternatives is generated (see Algorithm 11 and used as the basis for dialogue with the customer. The process of

negotiation involves the middleman selecting its best alternative from the set available and presenting this to the customer. The customer will then respond either with the same *ProductSet*, indicating its acceptance of the offered alternative, or a different *ProductSet* indicating a further alternative the SSF may find acceptable. If an acceptable alternative is found the customer negotiation, with its new requirement, will be added to a *BasicGroup*. If no alternative is found to be acceptable the negotiation will be added to the *FailureGroup*.

A.5.1 Receiving A Customer Response

Having sent an alternative *ProductSet* to a customer for evaluation the middleman expects a response. The response mechanism is show in Algorithm 12 and Table A.8. If a customer response includes a feasible, profitable offer and there is time still to negotiate for the products then reassign the negotiation to a basic group. If it does not and there are no more alternatives or there is insufficient time remaining, reassign the negotiation to the failure group.

A.6 Negotiating With Suppliers For Grouped Requirements

Customer requirements are grouped together in order to be fulfilled. This allows the SSF to use less communication than if handling all the requirements individually. The cost of grouping the requirements is increased complexity in dealing with the negotiation process and consequences of supplier responses.

Supplier negotiation is a two part process. Firstly, what negotiation messages should be sent out by the SSF. Secondly, how should the SSF respond to messages

Algorithm 12 AlternativeRequirementHandling (msg)

```
1: neg = FindRelatedNegotiation (msg)
2: UpdateNegotiationWithMessage (neg, msg)
3: alternatives = FindAlternativesFor (neg)
4: if SufficientTimeToNegotiation (neg) AND alternatives ≠ {} then
5:    value = LatestOfferValue(neg) - DetermineEstimatedPrice
      (LatestProductSet(neg))
6:    if value > 0.0 AND ProductSetFeasible (neg) AND SufficientTimeToNegotia-
      tion (neg) then
7:       RemoveFromPreNegotiationGroup (neg)
8:       AssignToBasicGroup (neg, value)
9:    end if
10: else
11:   RemoveFromPreNegotiationGroup (neg)
12:   AssignToFailureGroup (neg)
13: end if
```

Table A.8: AlternativeRequirementHandling Function Elements

Function	*Output*	*Input*
FindRelatedNegotiation	Negotiation	Message
Find the negotiation that is associated with this message. The message contents contains a $NegID$ that can be used to search the $Group\_Info$		
UpdateNegotiationWithMessage	-	Negotiation, Message
Given the contents of the supplied message, update the negotiation to reflect this. This involves adjusting *State* to either $Not\_Waiting$, *Accepted*($Them$) or *Rejected*($Them$). If changed to $Not\_Waiting$ a new offer is added from the message contents and the $TimeOut$ value updated.		
RemoveFromPreNegotiationGroup	CusNeg	-
Removes the tuple related to this negotiation from the $PreNegotiationGroup$		

from the suppliers.

In the first case this is a matter of determining what value to assign the required product set, the product set itself being determine by the $BasicGroup\ Profile$ to which it is associated. This may lead to acceptance if the suppliers last offer was lower. In the second case this deals with how the SSF World State is updated as a result of supplier message.

A.6.1 Starting Supplier Negotiations

To fulfill customer requirements it is at some point necessary to begin negotiations with suppliers. Supplier negotiations are begun when basic group $Profile$s that still have products left to find (in $StillLeftToFind$) and still have suppliers available to obtain them from are considered. The basic process is to attempt to obtain the entire required product set from the supplier for some price. If the supplier is unable to provide all the required products, the SSF will attempt to obtain the remainder from another supplier and so on, until there are either no suppliers remaining or products can be negotiated for. The basic negotiating process is shown in Algorithm 13 and Table A.9.

A.6.2 Ongoing Supplier Negotiations, How To Respond

Response to a supplier offer can be threefold. A new value counter offer, a new counter off with a different product set (if conditions have changed) or an accept message. This is outlined below in Algorithm 14 and Table A.10.

Determining if the suggested offer is feasible is an important part of maintaining the world state. If in the course of negotiation message generation it is seen

Algorithm 13 StartNewNegotiation ($Profile$)

1: $Offer$ = DetermineStartOffer ($Profile.StillToFind$)
2: $Available = Profile.ProfileBudget - Profile.UsedBudget$)
3: $TimeOut$ = DetermineStartTimeOut
4: $Participant$ = SelectSupplier ($Profile$)
5: **if** $Offer > Available$ **then**
6: SetNeedMoreFunds ($Profile$)
7: **return** GenerateStartingOfferMessage ($Profile.StillToFind$, $Available$, $TimeOut$)
8: **else**
9: **return** GenerateStartingOfferMessage ($Profile.StillToFind$, $Offer$, $TimeOut$)
10: **end if**

Table A.9: StartNewNegotiation, Function Elements

Function	*Output*	*Input*
SetNeedMoreFunds	-	Profile
Set the $NeedMoreFunds$ value of the $Profile$ to true		
DetermineStartOffer	Real	ProductSet
Determine the amount the middleman would initially be willing to offer a supplier for the specified set of products. Implementation specific, see Section 5.3.5		
DetermineStartTimeOut	Time	-
Determine the timeout the middleman should use in making an initial offer. Implementation specific, see Section 5.3.5		
SelectSupplier	Participant	Profile
Select a supplier from the set available in the $Profile$. Implementation specific, see Section 5.3.5		
GenerateStartingOfferMessage	Message	ProductSet, Real, Time
Generate a new negotiation message given the specified set of products, the offer value and timeout. Implementation specific, see Section 5.3.5		

Algorithm 14 GenerateSupplierNegotiationMessage ($SupNeg$)

1: $SortSet$ = DetermineSortSet ($SupNeg$)
2: $Offer$ = DetermineOffer ($SupNeg$, $SortSet$)
3: $FeasibleOffer$ = DetermineFeasibleOffer ($SupNeg$, $Offer$)
4: $SupplierOffer$ = DetermineLatestOffer ($SupNeg$)
5: **if** $SupplierOffer \leq FeasibleOffer$ **then**
6: **return** GenerateAcceptMessage ($SupNeg$)
7: **else**
8: $TimeOut$ = DetermineTimeOut ($SupNeg$)
9: **return** GenerateOfferMessage ($SupNeg$, $SortSet$, $FeasibleOffer$, $TimeOut$)
10: **end if**

Table A.10: GenerateSupplierNegotiationMessage, Function Elements

Function	*Output*	*Input*
DetermineSortSet	ProductSet	SupNeg
Find the *SortSet* associated with this negotiation		
DetermineLatestOffer	Real	Negotiation
Return the latest received offer value of the negotiation		
GenerateAcceptMessage	Message	Negotiation
Generate and accept message for this negotiation		
GenerateOfferMessage	Message	SupNeg, ProductSet, Real, Time
Given the negotiation, an offered product set, the offer value and the required timeout, generate a new offer message		
DetermineFeasibleOffer	Real	SupNeg, Real
Determine the if the suggest offer is possible, if not return the maximum possible and cause the $Profiles$ $NeedMoreFunds$ flag to be set *true*. See Algorithm 15		
DetermineTimeOut	Time	Negotiation
Determine a new timeout that should be used in responding to this negotiation. Implementation specific, see Section 5.3.5		
DetermineOffer	Real	SupNeg, ProductSet
Determine the amount the middleman is willing to pay for these goods at this time. Implementation specific, see Section 5.3.5		

Table A.11: DetermineFeasibleOffer, Function Elements

Function	*Output*	*Input*
DetermineOurLastOffer	Real	SupNeg
Determine the last offer made by the SSF in this negotiation. Returns 0 if not previous offer (which should be impossible)		

as impossible to offer the amount the middleman currently is whiling to pay the $NeedMoreFunds$ flag will be set allowing further allocation of funds if it becomes available later.

Algorithm 15 DetermineFeasibleOffer ($SupNeg$, $Offer$)

```
1: group = FindBasicGroup (neg)
2: profile = FindProfileFor (SupNeg)
3: TotalFunds = profile.ProfileBudget
4: UsedFunds = profile.UsedBudget
5: OurLastOffer = DetermineOurLastOffer (SupNeg)
6: Available = TotalFunds − (UsedFunds − OurLastOffer)
7: if Offer ¿ Available then
8:    SetNeedMoreFunds (profile)
9:    return Available
10: else
11:    return Offer
12: end if
```

A.6.3 Receiving A Supplier Response

The response of the SSF to supplier negotiation messages is handled as part of the Update World State process. The negotiation information must be updated according to the message - if the message contains anything other than a simple counter offer without a new product set more complex actions must be taken. If a new *ProductSet* is proposed this indicates the suppliers inability to fulfill the *ProductSet* originally specified. If another supplier of the required product type is available this is not a

problem, if no other supplier is available then at least one customer requirement is no longer able to be fulfilled. In consequence one or more customer requirements will be removed from the associated *BasicGroup* and moved to the *FailureGroup* and ongoing supplier negotiations within the *BasicGroup* may have to have their *SortSet*s adjusted as a result. If at the end of this process there are no customer requirements remaining within the *BasicGroup* the group will either enter completion (to wait for negotiation responses) or be removed. If a supplier rejects a negotiation outright then the above process for a new *ProductSet* must again be followed but with the likelihood of greater loss of customer requirements. An accepted negotiation must updated the *ObtainedProducts* of the *BasicGroup*. If all required products have been obtained as a result of the acceptance the group will enter completion. This is shown in Algorithm 16 and Table A.12.

A supplier negotiation being accepted is generally positive for the SSF as it means products required to fulfill some customer requirements have been obtained. Since the supplier negotiation has completed successfully information about product prices must be updated according to the implementation specific mechanism selected. The process of handling an accepted supplier negotiation is shown in Algorithm 17 below.

A supplier negotiation terminating in rejection may have serious consequences within the SSF. If alternative suppliers are available for the required products in the affected group then recovery is possible if, however, no other supplier is available then some set of customer requirements in the group are impossible to fulfill and must be removed. This process is elaborated upon in Algorithm 18 and Table A.14.

If a supplier has proposed a new product set it indicates that, for the present

Algorithm 16 SupplierResponseHandling (msg)

```
1: neg = FindRelatedNegotiation (msg)
2: UpdateNegotiationWithMessage (neg, msg)
3: group = FindBasicGroup (neg)
4: if NegotiationEnded (neg) then
5:    profile = FindProfile(group, neg)
6:    productset = RemoveSupplierNegotiation (profile, neg)
7:    if Accepted (neg) then
8:       HandleAcceptedSupplierNegotiation (neg, group, profile, productset)
9:    else
10:      HandleRejectedSupplierNegotiation (neg, group, profile, productset)
11:   end if
12: else
13:   needed = FindSortSetFor (neg)
14:   if needed = {} then
15:      RemoveSupplierNegotiation (profile, neg)
16:      AssignToFailureGroup (neg)
17:   else
18:      if NewProductSetProposed (neg) then
19:         HandleSupplierNewProductSetProposed (neg, profile, group, productset)
20:      end if
21:   end if
22: end if
23: ReconcileWorldState
```

Algorithm 17 HandleAcceptedSupplierNegotiation (neg, $group$, $profile$, $productset$)

```
1: UpdateObtainedProducts (group, productset)
2: AddNegotiationToHistory (neg)
3: UpdatedProductValueEstimates
4: UpdateFoundFor (profile)
```

Algorithm 18 HandleRejectedSupplierNegotiation (neg, $group$, $profile$, $productset$)

```
1: UpdateStillToFind (profile, productset)
2: ReturnUsedFunds (profile, neg)
3: if NOT OtherSupplierAvailable (profile) then
4:    negotiations = FindConflictingCustomerRequirements (group, productset)
5:    RemoveCustomerNegotiations (group, negotiations)
6: end if
```

Table A.12: SupplierResponseHandling Function Elements

Function	*Output*	*Input*
FindBasicGroup	BasicGroup	Negotiation
Returns the basic group that is associated with this negotiation		
NegotiationEnded	Boolean	Negotiation
True iff the negotiation *State* is $Accepted(Us)$, $Accepted(Them)$, $Rejected(Us)$, $Rejected(Them)$, $Implict\_Reject(Us)$ or $Implict\_Reject(Them)$		
Accepted	Boolean	Negotiation
True iff the negotiation *State* is $Accepted(Us)$ or $Accepted(Them)$		
NewProductSetProposed	Boolean	Negotiation
True iff the latest $ProductSet$ in the offer history does not match the previous $ProductSet$ in the $Offer\_History$ of the negotiation		
FindProfile	Profile	BasicGroup, SupNeg
Find the $Profile$ of a *BasicGroup* that contains the supplied negotiation		
RemoveSupplierNegotiation	SortSet	Profile, SupNeg
Remove the supplier negotiation, *SortSet* tuple from the $SupplierNegotiations$ of the $Profile$ and return the *SortSet*		
HandleAcceptedSupplierNegotiation	-	Negotiation, BasicGroup, Profile, ProductSet
Handle the affects of an accepted supplier negotiation. See Algorithm 17 below.		
HandleRejectedSupplierNegotiation	-	Negotiation, BasicGroup, Profile, ProductSet
Handle the affects of a rejected supplier negotiation. See Algorithm 18 below.		
HandleSupplierNewProductSetProposed		Negotiation, BasicGroup, Profile, ProductSet
Handle the affects of a supplier proposing an alternative product set during negotiations. See Algorithm 19 below.		
FindSortSetFor	ProductSet	SupNeg
Find *SortSet* associated with the supplier negotiation supplied. Part of a tuple in $SupplierNegotiations$ of a $Profile$ of the $SupplierInfo$ of a *basicGroup*		
ReconcileWorldState	-	-
Ensure changes in estimated product values are reflected throughout the SSF World State. See Algorithm 26		

Table A.13: HandleAcceptedSupplierNegotiation Function Elements

Function	*Output*	*Input*
	UpdateObtainedProducts	BasicGroup, ProductSet
Add the elements from the specified *ProductSet* to *ObtainedProducts* of the *BasicGroup*		
AddNegotiationToHistory	-	Negotiation
Add the negotiation to the $Transaction\_History$ of the $WorldState$		
UpdatedProductValueEstimates	-	-
Update the estimated value of products in the $Product\_Info$ of the $World\_State$. This is an implementation specific mechanism, see Section 5.3.5		
UpdateFoundFor	-	Profile
Check which customer negotiation in $Customer\_Info$ have products being negotiated for as part of this profile. For these $CusNeg$ have the associated $FoundFor$ value set to *true*, if the *Remove* value is set to *true* reset it to $false$, we do not wish to drop customer for which we have already obtained products		

Table A.14: HandleRejectedSupplierNegotiation Function Elements

Function	*Output*	*Input*
UpdateStillToFind	-	Profile, ProductSet
Add the specified *ProductSet* to the *StillToFind* set of the $Profile$		
ReturnUsedFunds	-	Profile, SupNeg
Takes the last offer made by this participant and removes it from the profiles $UsedBudget$		
OtherSupplierAvailable	Boolean	Profile
True iff $UntriedSuppliers$ of the $Profile$ is not the empty set ({})		
FindConflictingCustomerRequirements	Set of CusNegs	BasicGroup, ProductSet
Find a set of customer requirements from an active group that should be removed as a result of the inability to obtain the specified *ProductSet*. The mechanism is implementation specific, see Section 5.3.5		
RemoveCustomerNegotiations	-	BasicGroup, CusNegs
Remove the set of customer negotiations from the group taking into account effects on supplier negotiations and budgets. See Algorithm 20 below		

Table A.15: HandleSupplierNewProductSetProposed Function Elements

Function	*Output*	*Input*
DetermineUnfoundProducts	ProductSet	Negotiation
	Find elements request in the previous offer of the negotiation that are not present in the new offer	

time, some of the required products are unavailable. If those products might be obtained from other suppliers this does not present a problem. If however there are no other potential suppliers some set of customers within the affected group are no longer possible to fulfill. Under these conditions the SSF must find a subset of customers that conflict with lost products and remove them. This process is outlined in Algorithm 19 and Table A.15.

Algorithm 19 HandleSupplierNewProductSetProposed (*neg*, *profile*, *group*, *productset*)

1: *missing* = DetermineUnfoundProducts (*neg*)
2: UpdateStillToFind (*profile*, *missing*)
3: **if** NOT OtherSupplierAvailable (*profile*) **then**
4: *negotiations* = FindConflictingCustomerRequirements (*group*, *productset*)
5: RemoveCustomerNegotiations (*group*, *negotiations*)
6: **end if**

Removing customer negotiations from a *BasicGroup* has a considerable impact and must be approached carefully to minimise the effects. The effects are two fold, first adjustment of the set of products being sort from suppliers, the second, adjustment of the budgets used in negotiation.

BasicGroup Profiles maintain *ProductSets* (*StillToFind*) that specify what products still need to be negotiated for. Since these products are not currently being negotiated for any adjustment in the required set of products can initially be taken from here. Ongoing supplier negotiations may be in either a *Waiting* or *Not_Waiting*

state. Negotiations in a *Not_Waiting* state have just received a non-terminating response from the suppliers. The associated *SortSet* for these negotiations can safely have elements from the removed customers products set taken away so mitigating the danger of obtaining unrequired products. Secondary adjustments are therefore made here if necessary. If as a result of this the *SortSet* becomes empty, the negotiation is moved to the *FailureGroup*. Those supplier negotiations in a *Waiting* state have recently sent an offer to a supplier prior to the loss of these customers, as a result they may have requested products that are now no longer required. A tertiary procedure is to remove elements from these negotiations associated *SortSet*s in the hope that suppliers will not respond positively, allowing the middleman to reply with a new, reduced, set. If the *SortSet* is emptied a response must still be waited for, only then, if non terminating, will the negotiation be moved to the failure group. Any elements still not removed as a result of this process must have already been obtained.

Adjusting the *BasicGroup* budget in response to the removed customer set is a matter of proportionally adjusting the entire set of budgets to reflect the new available set of funds (i.e. the gap between perceived cost of the products required and the amount the customers are willing to pay).

A.7 Keeping Track Of Time, SSCM Time-Steps

Within the SSF time is primarily considered in terms of SSCM time-steps. These are provided synchronously to all participants in the the market and so ensure no conflicts in terms of negotiation messages sent or received. For the purposes of negotiation, time can be further considered in terms of the amount of outbound communications

Algorithm 20 RemoveCustomerNegotiations (*group*, *negs*)

```
1: removeset = DetermineCombinedProductSetFor (negs)
2: for EACH Profile IN SupplierInfo OF group do
3:    MatchAndRemoveElementsFrom (StillToFind, removeset)
4: end for
5: if removeset ≠ {} then
6:    for EACH Profile IN SupplierInfo OF group do
7:       for EACH tuple IN SupplierNegotiations OF Profile do
8:          if State OF SupNeg IN tuple = Not_Waiting then
9:             MatchAndRemoveElementsFrom (SortSet, removeset)
10:            if SortSet = {} then
11:               RemoveSupplierNegotiation (Profile, SupNeg)
12:               AssignToFailureGroup (SupNeg)
13:            end if
14:         end if
15:      end for
16:   end for
17: end if
18: if removeset ≠ {} then
19:   for EACH Profile IN SupplierInfo OF group do
20:      for EACH tuple IN SupplierNegotiations OF Profile do
21:         if State OF SupNeg IN tuple = Waiting then
22:            MatchAndRemoveElementsFrom (SortSet, removeset)
23:         end if
24:      end for
25:   end for
26: end if
27: removevalue = DetermineTotalCustomerValue (group, negs)
28: if removevalue ≠ group.SupplierInfo.CashPool then
29:   if removevalue < group.SupplierInfo.CashPool then
30:      Set group.SupplierInfo.CashPool = group.SupplierInfo.CashPool −
         removevalue
31:      removevalue = 0
32:   else
33:      removevalue = removevalue − group.SupplierInfo.CashPool
34:      Set group.SupplierInfo.CashPool = 0
35:      ProportionatelyRemoveBudgetFromProfiles (group, removevalue)
36:      removevalue = 0
37:   end if
38: else
39:   Set group.SupplierInfo.CashPool = 0
40:   removevalue = 0
41: end if
42: for EACH CusNeg IN negs do
43:   AssignToFailureGroup (CusNeg)
44: end for
```

Table A.16: RemoveCustomerNegotiations Function Elements

Function	*Output*	*Input*
MatchAndRemoveElementsFrom	-	ProductSet, ProductSet
Match elements from the second *ProductSet* to the first and remove from both sets. Matches are made based on product *Type* and *Time*. If the removing set has a greater quantity then the tuple is removed entirely from the matching set and the quantity reduced by the corresponding amount in the removal set. If the quantities are equal then the tuple is removed from both sets. If matching set tuple has a greater quantity, the quantity is reduced by the amount from the removing set, the tuple in the removing set being removed		
DetermineTotalCustomerValue	Real	BasicGroup, CusNegs
Find the sum of the *EsimtatedValue*s associated with these customer negotiations in this *BasicGroup*		
ProportionatelyRemoveBudgetFromProfiles	-	BasicGroup, Real
Reduce the total currency embodied in the budgets of the profiles of the group by the amount specified. For each profile reduce its budget proportional to the total budget		

remaining within this time-step, the time creeping forward as the outbound communications available are used up.

A.7.1 Response To A Timing Message

Timing messages have a considerable number of effects on the SSF. Negotiations may 'time-out' as a result of a new time step, pushing them into a *Implicit_Reject* state. This may in turn effect the make up of *BasicGroup*s if it is supplier negotiations that have timed-out. The *PreNegotiationGroup* and *BasicGroup*s are also directly effected by timing. In the case of the *PreNegotiationGroup* customer negotiations that no longer meet the time required to negotiate for their products will, if possible, be removed to the *FailureGroup*. For *BasicGroup*s time-step changes are the cause of a group going from the *Inactive* to *Active* state and may cause a group to go from the *Active* to *Completion* state. Algorithm 21 and Table A.17 show the outline of the time-step change update process.

Algorithm 21 DoTimeStepHandling

1: DoGroupStateChangeCheck
2: IncrementTimeStep
3: RefreshOutboundCommunicationBudget
4: DoNegotiationTimeOutCheck
5: DoGroupStateChangeCheck

Checking for negotiation timeouts is a matter of scanning the entire set of active negotiations throughout the World State to see what, if any, negotiations may have timed out as a result of timing changes.

Negotiations within the *FailureGroup* that time-out can simply be removed to the *Transaction_History* as the objective for this group is in any case to reject any

Table A.17: DoTimeStepHandling Function Elements

Function	*Output*	*Input*
IncrementTimeStep	-	-
	Increment the $CurrentTimeStep$ of the world state	
RefreshOutboundCommunicationBudget		-
	Set $CommsBudget$ to $MaxCommsBudget$ in the $SSCM\_Info$	
DoNegotiationTimeOutCheck	-	-
	See Algorithm 22 and Table A.18	
DoGroupStateChangeCheck	-	-
	Reconcile the World State then determine if any group statuses need to change. See Algorithm 27.	

Algorithm 22 DoNegotiationTimeOutCheck

1: DoFailureGroupTimeOutCheck
2: DoPreNegotiationGroupTimeOutCheck
3: DoBasicGroupsTimeOutCheck

Table A.18: DoNegotiationTimeOutCheck Function Elements

Function	*Output*	*Input*
DoFailureGroupTimeOutCheck	-	-
	Handle the effects of negotiation time-outs within the Failure Group. See Algorithm 23	
DoPreNegotiationGroupTimeOutCheck		-
	Handle the effects of negotiation time-outs within the PreNegotiation Group. See Algorithm 24	
DoBasicGroupsTimeOutCheck	-	-
	Handle the effects of negotiation time-outs within BasicGroups. See Algorithm 25	

contained negotiations.

Algorithm 23 DoFailureGroupTimeOutCheck

```
1: for all Negotiations IN FailureGroup do
2:    if Negotiation.Time − Out = CurrentTimeStep then
3:       RemoveFromFailureGroup (Negotiation)
4:       AddNegotiationToHistory (Negotiation)
5:    end if
6: end for
```

Customer negotiations within the *PreNegotiationGroup* may time out if a response is not received rapidly enough. In this case the negotiation and related information must be removed and the negotiation added to the *Transaction_History*. *Not_Waiting* customer negotiations that no longer meet the requirements for timing will be removed to the *FailureGroup*.

Algorithm 24 DoPreNegotiationGroupTimeOutCheck

```
1: for all Negotiations IN PreNegotiationGroup do
2:    if Negotiation.Time − Out = CurrentTimeStep then
3:       RemoveFromPreNegotiationGroup (Negotiation)
4:       AddNegotiationToHistory (Negotiation)
5:    else
6:       if NOT SufficientTimeToNegotiation (Negotiation) then
7:          RemoveFromPreNegotiationGroup (Negotiation)
8:          AssignToFailureGroup (Negotiation)
9:       end if
10:   end if
11: end for
```

Customer negotiations within *BasicGroup*s will never timeout however supplier negotiations may well do so. If a supplier negotiation times out its *SortSet* is returned to its related $Profile.StillToFind$ and the supplier re-added to the $UntriedSuppliers$ set for a further negotiation attempt. If the group is in the *Completion* state this may cause that state to fail

Algorithm 25 DoBasicGroupsTimeOutCheck

```
1: for all BasicGroups IN BasicGroups do
2:   for all SupNegs IN SupplierInfo OF BasicGroup do
3:     if SupNeg.Time − Out = CurrentTimeStep then
4:       profile = FindProfile(BasicGroup, SupNeg)
5:       productset = RemoveSupplierNegotiation (profile, SupNeg)
6:       UpdateStillToFind (profile, productset)
7:       ReturnUsedFunds (profile, SupNeg)
8:       AddSupplierToUntried (profile, SupNeg.Seller)
9:       AddNegotiationToHistory (SupNeg)
10:      DoStatusCheck (group)
11:    end if
12:  end for
13: end for
```

Table A.19: DoBasicGroupsTimeOutCheck Function Elements

Function	*Output*	*Input*
AddSupplierToUntried	-	Profile, Participant
	Add the specified supplier to the set of untried suppliers for this profile	

A.8 Reconciling The World State

Reconciliation of the World State is the process by which wider changes resulting from inbound communication may be considered. Changes of this type will be the result of changes in the estimated value of products, these changes being a possible result of completed supplier negotiations. The estimated value of products is updated via an Implementation specific algorithm as a result of a supplier negotiation being accepted (see Algorithm 16). The estimated product values effect the *PreNegotiationGroup* customers and all *BasicGroup*s. For customers, the estimated effects the SSF's belief about potential profitability. This in turn effects whether a customer requirement should be negotiated for at all and the way in which negotiations proceed with suppliers.

A.9 From Collecting To Fulfilling Requirements - Basic Group Transitions

Basic groups, their states and the transitions between these are outlined in Section A.1.4. The process of checking the group to see if its state should change occurs both as a result of timing effects but also due to supplier negotiation responses. In summary, *Inactive* groups collect new customer negotiations. *Active* groups are in the process of negotiation with suppliers. *Waiting* groups are *Active* groups that are attempting to shed one or more customer negotiations. Groups in the $Completion_Success$ state are attempting to report success back to the customer negotiations. $Completion_Failed$ groups are waiting for supplier negotiations to respond before being removed. *Inactive* groups will become *Active* to begin negotiations with

Algorithm 26 ReconcileWorldState

```
1: for all tuples IN PreNegotiationGroup do
2:    EstimatedValue = LatestOfferValue(CusNeg) - DetermineEstimatedPrice
      (LatestProductSet(CusNeg))
3:    if EstimatedValue ≤ 0.0 AND CusNeg.State = Not_Waiting then
4:       RemoveFromPreNegotiationGroup (CusNeg)
5:       AssignToFailureGroup (CusNeg)
6:    end if
7: end for
8: for all BasicGroups IN BasicGroups do
9:    difference = ReEvaluateAllCustomers (BasicGroup)
10:   if BasicGroup.GroupState = Inactive then
11:      for all tuples IN Customer_Info OF BasicGroup do
12:         if EstimatedValue ≤ 0.0 then
13:            RemoveCustomerTuple (BasicGroup, CusNeg)
14:            AssignToFailureGrouop (CusNeg)
15:         end if
16:      end for
17:   else if BasicGroup.GroupState = Active OR BasicGroup.GroupState =
      Waiting OR BasicGroup.GroupState = Completion_Success then
18:      for all tuples IN Customer_Info OF BasicGroup do
19:         if EstimatedValue ≤ 0.0 then
20:            FlagForRemoval (BasicGroup, CusNeg)
21:         else
22:            UnflagForRemoval (BasicGroup, CusNeg)
23:         end if
24:      end for
25:      if difference ≠ 0.0 then
26:         if difference < 0.0 then
27:            ProportionatelyRemoveBudgetFromProfiles (BasicGroup, difference)
28:         else
29:            AddToCashPool (BasicGroup, difference)
30:         end if
31:      end if
32:   end if
33: end for
```

Table A.20: ReconcileWorldState Function Elements

Function	*Output*	*Input*
ReEvaluateAllCustomers	Real	BasicGroup
Determine $EstimateValue$ of all $CusNeg$ tuples in the *BasicGroup*s $Customer\_Info$. Returns the difference between the original and new total evaluated worth		
RemoveCustomerTuple	-	BasicGroup, CusNeg
Remove the tuple for a specified customer negotiation from an $Inactive$ basic group		
FlagForRemoval	-	BasicGroup, CusNeg
For the $CusNeg$ tuple in the specified *BasicGroup* set the *Remove* value to $true$ UNLESS the $FoundFor$ value is set to $true$		
UnflagForRemoval	-	BasicGroup, CusNeg
For the $CusNeg$ tuple in the specified *BasicGroup* set the *Remove* value to $false$		
AddToCashPool	-	BasicGroup, Real
Adds the supplied amount to the *BasicGroup CashPool*. If any $Profile$s are flagged for additional funds the $CashPool$ is distributed amongst all flagged $Profile$s proportionate to the total worth of the products being negotiated for		

suppliers. *Active* groups will become $Waiting$ groups if a customer requirement is now considered to be be unprofitable and should be dropped. The $Waiting$ state occurs when supplier negotiations that effect the customer are still waiting for a response. The group waits to see if suppliers will respond without an accept. If all related supplier negotiations can be brought to a $Not\_Waiting$ or *Rejected* state the customer negotiation can be safely removed. If suppliers accept an offer the customer will have goods associated with it and should not be dropped ending the $Waiting$ state. If supplier negotiations in a $Not\_Waiting$ state threaten to time-out the group must become *Active* to attempt to prevent this from happening and forcing a complete renegotiation. Any customer negotiation with products associated with it (via $FoundFor$) can not be the cause of a $Waiting$ state. *Active* and $Waiting$ groups may change state to become $Completion\_Success$ or $Completion\_Failed$ groups. $Completion\_Success$ occurs when all products have been obtained for the set of customer negotiations. The state may also occur due to timing effects provided that any unaccepted supplier negotiations are in a $Not\_Waiting$ state and there is sufficient funds to provide an *Accept* response to all. $Completion\_Failed$ occurs when all customer requirements have been removed from the group but some supplier negotiations are still in a $Waiting$ state. A $Completion\_Success$ group may become a $Completion\_Failed$ group if timing effects cause one of the supplier negotiations to time out before a response can be sent.

The outline of the DoGroupStateChangeCheck algorithm is shown in Algorithm 27. This essentially invokes the DoGroupStatusCheck function on all basic groups. This algorithm is elaborated upon in Algorithm 28 and Table A.21. For groups already in a completion state the status check is invoked to perform any further processing

resulting from supplier negotiations having timed-out or had responses, in this case *Completion_Success* group may fail and *Completion_Failure* groups maybe removed entirely from the set of groups.

Algorithm 27 DoGroupStateChangeCheck

```
1: ReconcileWorldState
2: for all BasicGroups IN BasicGroups do
3:    DoGroupStatusCheck (BasicGroup)
4: end for
```

Algorithm 28 DoGroupStatusCheck (*BasicGroup*)

```
1: if BasicGroup.GroupStatus = Inactive then
2:    if ShouldBecomeActive (BasicGroup) then
3:       MakeGroupActive (BasicGroup)
4:    end if
5: else if BasicGroup.GroupStatus = Active then
6:    if ShouldEnterCompletion (BasicGroup) then
7:       MakeGroupCompletion (BasicGroup)
8:    end if
9: else if BasicGroup.GroupStatus = Waiting then
10:   if ShouldStopWaiting(BasicGroup) then
11:      EndWaitState (BasicGroup)
12:   end if
13:   if ShouldEnterCompletion (BasicGroup) then
14:      MakeGroupCompletion (BasicGroup)
15:   end if
16: else if BasicGroup.GroupStatus = Completion_Success then
17:   MakeGroupCompletion (BasicGroup)
18: else if BasicGroup.GroupStatus = Completion_Failed then
19:   MakeGroupCompletion (BasicGroup)
20: end if
```

A basic group should enter completion if there are no customers remaining, no products left to obtain or timing effects come in to play. This is shown in Algorithm 29 and Table A.22.

Table A.21: DoGroupStatusCheck Function Elements

Function	*Output*	*Input*
ShouldBecomeActive	Boolean	BasicGroup
True if an $Inactive$ basic group should become *Active*. This decision is implementation specific, see Section 5.3.5		
MakeGroupActive	-	BasicGroup
Makes an $Inactive$ basic group *Active*. First the group $CashPool$ is calculated from the summation of $EstimatedValue$s in the $Customer\_Info$. The total $ProductSet$ is determined from the customer negotiations and then split down product type lines. One $Profile$ is created for each product type being sort, the total requirement for that product assigned to the $StillToFind$ of that $Profile$. Any $FreeProducts$ elements matching a those in a $Profile$s $StillToFind$ are removed from both and added to the $ObtainedProducts$ set for the $Profile$. The $CashPool$ is then redistributed to the profiles proportionate to the estimated cost of each $StillToFind$ product set. The $GroupStatus$ is then set to *Active*		
ShouldEnterCompletion	Boolean	BasicGroup
True if the group should enter the completion state (either successful or unsuccessful). See Algorithm 29		
MakeGroupCompletion	-	BasicGroup
Changes the status of a group to one of the available completion states. May remove the group entirely if finished. See Algorithm 30.		
ShouldStopWaiting	Boolean	BasicGroup
True if there are no longer any customer tuples with the *Remove* value set *true* or, if all there are tuple with the value set *true*, all $SupNegs$ for all $Profile$s associated with those negotiations are in a $Not\_Waiting$ state. Finally, also true if any $Not\_Waiting$ $SupNeg$ is about to timeout		
EndWaitState	-	BasicGroup
Ends the wait state, changing the group status back to *Active* and possibly removing customer requirements. See Algorithm 33.		
ReconcileWorldState	-	-
Ensure changes in estimated product values are reflected throughout the SSF World State. See Algorithm 26		

Algorithm 29 ShouldEnterCompletion (*BasicGroup*)

```
1: if NoCustomers (BasicGroup) then
2:    return true
3: end if
4: if EnterCompletionDueToTiming (BasicGroup) then
5:    return true
6: end if
7: if NoneLeftToFind (BasicGroup) then
8:    return true
9: end if
10: return false
```

Table A.22: ShouldEnterCompletion Function Elements

Function	*Output*	*Input*
NoCustomers	Boolean	BasicGroup
True if the *BasicGroup* has no customers remaining		
EnterCompletionDueToTiming	Boolean	BasicGroup
True if the *BasicGroup* should enter completion due to other (timing) reasons. Responses from suppliers must be set to time-out by the time this mechanism would trigger completion. Implementation specific, see Section 5.3.5		
NoneLeftToFind	Boolean	BasicGroup
True if for all $Profiles$ in the group $StillToFind = \{\}$ and there are no ongoing supplier negotiations		

A group may enter completion because it has run out of customers, has obtained all products or has been forced to do so due to time. In the first case the group will enter the *Completion_Failure* state. Any supplier negotiations will be removed, if possible. If all supplier negotiations are successfully removed the group will be removed from the set of basic groups. In the second case all needed elements are available and the group can enter the *Completion_Success* state. In this last case the the group may or may not be able to complete successfully. The SSF ensures all suppliers must have responded before timing will cause the state to change. Any suppliers timing out will have done so by this point. If any products must still be negotiated for there is no time remaining and associated customers must be removed. If any customers remain after this process and there is sufficient funds to accept the remaining supplier negotiations then the group may enter the *Completion_Success* state. If not the group enters *Completion_Failure*, all remaining customer and supplier negotiations are removed to the *FailureGroup* and the group itself removed from the set of *BasicGroups*. This is shown in Algorithm 30 and Table A.23.

Algorithm 30 MakeGroupCompletion (*BasicGroup*)

1: **if** NoCustomers (*BasicGroup*) **then**
2: MakeGroupCompletionNoCustomers (*BasicGroup*), see Algorithm 31
3: **else if** NoneLeftToFind (*BasicGroup*) **then**
4: $GroupStatus = Completion\_SUCCESS$
5: **else if** EnterCompletionDueToTiming (*BasicGroup*) **then**
6: MakeGroupCompletionDueToTime (*BasicGroup*), see Algorithm 32
7: **end if**

Basic groups enter a *Waiting* state if customer requirements are considered no longer profitable and are flagged for removal. The state will end under one of three conditions. First, if no customer requirements are still flagged for removal. If this is

Algorithm 31 MakeGroupCompletionNoCustomers (*BasicGroup*)

```
1: GroupStatus = Completion_Failed
2: for all SupNegs IN Profiles OF BasicGroup do
3:    if State OF SupNeg IS Now_Waiting then
4:       profile = FindProfile(group, neg)
5:       productset = RemoveSupplierNegotiation (profile, neg)
6:       AssignToFailureGroup (SupNeg)
7:       UpdateStillToFind (profile, productset)
8:       ReturnUsedFunds (profile, neg)
9:    end if
10: end for
11: if NOT AnySupplierNegotiations (BasicGroup) then
12:    RemoveGroup (BasicGroup)
13: end if
```

Table A.23: MakeGroupCompletion, MakeGroupCompletionNoCustomers and MakeGroupCompletionDueToTime Function Elements

Function	*Output*	*Input*
NoCustomers	Boolean	BasicGroup
True if the *BasicGroup* has no customers remaining		
AnySupplierNegotiations	Boolean	BasicGroup
True if there are any ongoing supplier negotiation within the *Supplier_Info* of the group		
RemoveGroup	-	BasicGroup
Removes the basic group entirely from the set of *BasicGroups* decrementing *bgn* and returning any elements in *ObtainedProducts* to *FreePrroducts*		
CanAcceptSuppliers	-	BasicGroup
True if all supplier negotiations in the *BasicGroup* could be accepted now, taking into account available funds		

Algorithm 32 MakeGroupCompletionDueToTime (*BasicGroup*)

```
1: for all Profiles IN Supplier_Info OF BasicGroup do
2:   if Profile.StillToFind ≠ {} then
3:     negotiations = FindConflictingCustomerRequirements (BasicGroup,
       Profile.StillToFind)
4:     RemoveCustomerNegotiations (BasicGroup, negotiations)
5:   end if
6: end for
7: if NoCustomers (BasicGroup) then
8:   GroupStatus = Completion_Failed
9:   for all SupNegs IN Profiles OF BasicGroup do
10:    profile = FindProfile(BasicGroup, SupNeg)
11:    productset = RemoveSupplierNegotiation (profile, SupNegs)
12:    AssignToFailureGroup (SupNeg)
13:    UpdateStillToFind (profile, productset)
14:    ReturnUsedFunds (profile, SupNegs)
15:  end for
16:  RemoveGroup (BasicGroup)
17: else
18:  if CanAcceptSuppliers (BasicGroup) then
19:    GroupStatus = Completion_Success
20:  else
21:    GroupStatus = Completion_Failure
22:    for all SupNegs IN Profiles OF BasicGroup do
23:      profile = FindProfile(BasicGroup, SupNeg)
24:      productset = RemoveSupplierNegotiation (profile, SupNegs)
25:      AssignToFailureGroup (SupNeg)
26:      UpdateStillToFind (profile, productset)
27:      ReturnUsedFunds (profile, SupNegs)
28:    end for
29:    for all CusNegs IN Custoomer_Info OF BasicGroup do
30:      RemoveCustomerNegotiations (BasicGroup, negotiations)
31:    end for
32:    RemoveGroup (BasicGroup)
33:  end if
34: end if
```

the case the group will revert to being *Active*. Second, if any $Not\_Waiting$ supplier negotiation is in danger of timing out the group must revert to being *Active* to avoid the possibility of losing needed requirements. Third, if all supplier negotiation associated with the $Waiting$ are now in the $Not\_Waiting$ negotiation state the flagged customer requirements may be removed to the $FailureGroup$ and the group continue by being *Active*. This is outlined below in Algorithm 33 and Table A.24 for the EndWaitState function.

Algorithm 33 EndWaitState (*BasicGroup*)

```
1: if NOT AnyRemoveFlagCustomers (BasicGroup) then
2:    BasicGroup.GroupStatus = Active
3: else if NOT NeedToWait (BasicGroup) then
4:    for all CusNegs IN Custoomer_Info OF BasicGroup do
5:       if FlaggedForRemoval (CusNeg) then
6:          RemoveCustomerNegotiations (BasicGroup, negotiations)
7:       end if
8:    end for
9:    BasicGroup.GroupStatus = Active
10: else if NOT SafeToWait (BasicGroup) then
11:    BasicGroup.GroupStatus = Active
12: end if
```

A.10 Acting On The World State

A participant within an SSCM-V1 market interacts with other participants via negotiations. For an SSF base middleman, the decision about what negotiation message to send at any given point is based on the World State information. In Section A.2 the process of updating this World State based on received communication was detailed. This section discusses how that updated World State is used to determine what, if any, message should be sent and how the World State responds to that sent message.

Table A.24: EndWaitState Function Elements

Function	*Output*	*Input*
AnyRemoveFlagCustomers	Boolean	BasicGroup
True if the no $Customer\_Info$ tuples have a *Remove* value set to *true*		
NeedToWait	Boolean	BasicGroup
True if any $Profile$ negotiating for products associated with a $CusNeg$ flagged for removal still has negotiations in a $Waiting$ state		
FlaggedForRemoval	Boolean	CusNeg
True if the $CusNeg$'s tuple has a *Remove* value of *true*		
SafeToWait	Boolean	BasicGroup
True if the no $SupNeg$ is about to time out (has more than one time-step until time-out)		

At any given time the SSF strategy may wish to send messages relating to the $FailureGroup$, $PreNegotiationGroup$ or the $BasicGroups$. $FailureGroup$ messages comprise negotiation rejects to both customers and suppliers. In the case of customers this is to inform them that their desired products either could not be found or are unprofitable to obtain on their behalf. Negotiations within the $FailureGroup$ will time-out without the possibility of messages being received achieving the same results as a reject message. It is still advantageous to send reject messages, particularly to suppliers who will then be able to free up committed resources to other negotiations. $FailureGroup$ messages should however only be sent if there is no other message to send from the $PreNegotiationGroup$ or $BasicGroups$. The $PreNegotiationGroup$ messages comprise alternative product sets being sent to customers in an attempt to find mutually acceptable requirements. The $BasicGroups$ may send messages to both customers and suppliers. In the case of customers, messages are sent to accept their last offer and so confirm the middlemans attempt to fulfill their requirements. These are necessarily the highest priority messages the SSF can send as they result

Table A.25: Generate And Send A Message, Function Elements

Function	*Output*	*Input*
GenerateMessage	Message	Negotiation
Given a negotiation in the $Not\_Waiting$ state generate an appropriate message. See Algorithm 35		
HigherPriority	Message	Message, Message
Return the higher priority of the two messages presented. Prioritisation is based on the scheme shown in Table A.27		
DecrementCommsBudget	-	-
Decrement the $CommsBudget$ by one		
SendMessage	-	Message
Actually send the message		
MessageSendUpdateWorldState	-	Message
Update the world state according to the message sent. See Algorithm 36		
StartNewNegotiation	Message	Profile
Given a $Profile$ with non-empty $StillToFind$, generate a message for starting a new negotiation. See Algorithm 13		

in income and offset the cost of obtained products. Supplier messages comprise offers for products and acceptance of counter offers made. Offers made are the result of attempting to obtain products to fulfill customer requirements but are lower priority than accepting a supply offer and actually obtaining them. Further the SSF must open negotiations with suppliers initially and this is generally more important than other ongoing supplier negotiations. In all cases the likelihood of a negotiation timing-out and leading to a potential loss of income or required products is always significant and must be considered. The SSF's task is to review the World State and determine out of all groups and all negotiations within those groups what message to send. Algorithm 34 and Table A.25 show this process.

Generated messages may take the form of counter offers, accepts or rejects. Which of these to generate for a specific negotiation will depend on what group it is a part

Algorithm 34 Generate And Send A Message

```
1: if CommsBudget > 0 then
2:   bestmessage = null
3:   for all Negotiations IN THE SSFWorldState do
4:     if Negotiation.State = Not_Waiting then
5:       msg = GenerateMessage (Negotiation)
6:       if bestmessage = null then
7:         bestmessage = msg
8:       else if msg = HigherPriority(msg, bestmessage) then
9:         bestmessage = msg
10:      end if
11:    end if
12:  end for
13:  for all Profiles IN THE SSFWorldState do
14:    if Profile.StillToFind ≠ {} then
15:      msg = StartNewNegotiation (Profile)
16:      if bestmessage = null then
17:        bestmessage = msg
18:      else if msg = HigherPriority(msg, bestmessage) then
19:        bestmessage = msg
20:      end if
21:    end if
22:  end for
23:  if bestmessage ≠ null then
24:    DecrementCommsBudget
25:    SendMessage(bestmessage)
26:    MessageSendUpdateWorldState(bestmessage)
27:  end if
28: end if
```

of and the conditions within that group. Algorithm 35 and Table A.26 outline this process.

Algorithm 35 GenerateMessage (*Negotiation*)

```
1: if PartOfFailureGroup (Negotiation) then
2:    return GenerateFailureMessage (Negotiation)
3: else if PartOfPreNegotiationGroup (Negotiation) then
4:    alternatives = FindAlternativesFor (Negotiation)
5:    return GenerateCustomerAlternative (Negotiation, alternatives)
6: else
7:    if IsCustomerNegotiation (Negotiation) then
8:       return GenerateAcceptMessage (Negotiation)
9:    else
10:      return GenerateSupplierNegotiationMessage (Negotiation)
11:   end if
12: end if
```

Prioritisation of negotiation messages is based on the following principles. Negotiation accept messages have priority. If tied, the soonest to time-out has priority. If still tied, the negotiation dealing with the greatest value has priority. If still tied, select a message at random, or pick the first of two. Next in priority are ongoing negotiations. If tied, soonest to time-out has priority. If still tied, if one is the beginning of a new negotiation, this has priority. If still tied, the negotiation dealing with the highest value has priority. Again, if none of these resolve priority select a message at random or the first of two. Lowest in priority are reject messages. If tied, time-out has priority. If still tied, select at random or pick first of two. This is shown in Table A.27 below.

Having selected and sent a message for one the negotiations, the SSF World State must updated to reflect the change. This process if outlined in Algorithm 36 and Table A.28. *FailureGroup* messages cause their related negotiations to be removed to

Table A.26: GenerateMessage, Function Elements

Function	*Output*	*Input*
PartOfFailureGroup	Boolean	Negotiation
True if the negotiation is part of the $FailureGroup$		
GenerateFailureMessage	Message	Negotiation
Generate a failure type message for this negotiation		
PartOfPreNegotiationGroup	Boolean	Negotiation
True if the negotiation is part of the $PreNegotiationGroup$ and thus a $CusNeg$		
GenerateCustomerAlternative	Message	CusNeg, Alternatives
Given a customer negotiation and an alternatives set, generate a customer counter offer. This is problem-specific, see Section 5.3.4		
IsCustomerNegotiation	Boolean	Negotiation
True if the negotiation is a CusNeg, this middleman is the Seller		
GenerateSupplierNegotiationMessage	Message	SupNeg
Generate a supplier negotiation message, this may either be an offer or an accept message. See Algorithm 14		

Table A.27: Message Priorities

Main Priority	*Tie-Break 1*	*Tie-Break 2*	*Tie-Break 3*	*Tie-Break 4*
Accept				
	Time-Out			
		EstimatedValue		
			Random	
Ongoing				
	Time-Out			
		New Negotiation		
			EstimatedValue	
				Random
Reject				
	Time-Out			
		Random		

Table A.28: MessageSendUpdateWorldState, Function Elements

Function	*Output*	*Input*
PartOfFailureGroup	Boolean	Negotiation
True if the negotiation is part of the $FailureGroup$		
PartOfPreNegotiationGroup	Boolean	Negotiation
True if the negotiation is part of the $PreNegotiationGroup$		
IsCustomerNegotiation	Boolean	Negotiation
True if the negotiation is with a customer, this middleman being the seller		
UpdateUsedBudget	-	Profile
Update the $Profile$ to reflect the funds required to accept this negotiation. Take into account previous funds allocated for the middleman last offer (if any)		
RemoveRequiredGoods	-	CusNeg
Remove the latest $ProductSet$ of goods from the related $BasciGroup$s $ObtainedProducts$		
NewNegotiation	-	Negotiation
True if this negotiation has only one offer in its $Offer\_History$		
AssociateWithCreatingGroup	-	Negotiation
Determine the originating $BasicGroup$ and $Profile$. Remove the contacted supplier from the $Profiles$ $UntriedSuppliers$ set. Create a new $SupplierNegotiations$ tuple using the profiles $StillToFind$ and this $SupNeg$, clear the $StillToFindSet$ set		
RemoveAlternativeFor	-	Negotiation
Find and remove the attempted alternative from the set of alternatives for a customer negotiation in the $PreNegotiationGroup$. Problem-specific, see Section 5.3.4		

the $Transaction\_History$. $PreNegotiationGroup$ messages cause the set of possible alternatives to be reduced. $BasicGroup$ messages may have a wide range of effects.

Algorithm 36 MessageSendUpdateWorldState ($Message$)

```
1: if NewNegotiation (Message) then
2:    Neg = CreateNegotiationFromMessage (Message)
3: else
4:    Neg = FindRelatedNegotiation (Message)
5:    UpdateNegotiationWithMessage (Neg, Message)
6:    if PartOfFailureGroup(Neg) then
7:       RemoveFromFailureGroup (Neg)
8:       AddNegotiationToHistory (Neg)
9:    else if PartOfPreNegotiationGroup (Neg) then
10:      RemoveAlternativeFor (Neg)
11:   else
12:      if IsCustomerNegotiation(Neg) then
13:         RemoveRequiredGoods (Neg)
14:         AddNegotiationToHistory (Neg)
15:      else
16:         if NewNegotiation(Neg) then
17:            AssociateWithCreatingGroup (Neg)
18:         end if
19:         BasicGroup = FindBasicGroup (Neg)
20:         Profile = FindProfile(BasicGroup, Neg)
21:         if NegotiationEnded(Neg) then
22:            obtained = RemoveSupplierNegotiation (Profile, Neg)
23:            UpdateUsedBudget (Profile, Neg)
24:            UpdateObtainedProducts (BasicGroup, obtained)
25:            AddNegotiationToHistory (Neg)
26:            UpdatedProductValueEstimates
27:            UpdateFoundFor (Profile)
28:         else
29:            UpdateUsedBudget (Profile, Neg)
30:         end if
31:      end if
32:   end if
33: end if
```

Appendix B

The SMSS, Some Implementation Details

This section contains additional details about the implementation of the SSCM Market Simulation System (SMSS) discussed in Chapter 5.

B.1 SMSS Software Overview

The structure and operation of the SMSS software is somewhat more complex than is immediately apparent from the description provided in Chapter 6. This view of the SMSS is presented to more effectively communicate its intended purpose and core elements rather than to provide an in depth look at the implementation used.

This section of the Appendix is intended to provide a brief overview of the software implemented for the SMSS and provide more details of its capability.

The SMSS is implemented in Java as a series of inter-dependant software agents each of which provides some function to the overall SMSS system. One potential advantage of this break down of the SMSS in to various semi-autonomous subcomponents is the ability to distribute them across multiple computers and so reduce

the load on any one system. Figure B.1 shows the various components of the SMSS software and how they are inter-related.

In order to provide a highly configurable and adaptable experimental environment the SMSS consists of a number of components along side those that provide the core market simulation and evolver elements. These components are outlined in Table B.1 and their basic interaction illustrated in the aforementioned Figure B.1. The system as a whole is implemented in Java and is able to work stand-alone on a single machine or across many machines via a Java Messaging Service (JMS) server.

The Agent Loader acts as the starting point for the SMSS system and provides the initial agent configurations to the other components of the system. Following this it instigates the process by which each agent learns of the others (those both local and remote) and so kick-starts the system. Once this is accomplished the Experimenter takes over as the overall controlling entity of the SMSS. The Experimenter is able to run any number of experiments each with the same or different configurations as desired. The Evolver is responsible for an individual experiment and makes use of the various market elements in the process. The Generator creates individual SSCM instances for the Evolver configuring the Customers, Suppliers and Middlemen as appropriate. The Customers, Suppliers and Middlemen are further configured by the Evolver (in the latter case via the Evolvers PBIL component) and report the results of the market back. Timing in the market is controlled via the Timer component. If the system is operating across multiple machines a separate JMS server is used for message passing, if not messages go via the Agent Loader.

Table B.1: SMSS Software Components

Component	*Role*
Agent Loader	Loads and runs the agents in the system including the supply chain participants
Experiment Controller (Experimenter)	Configures and runs the experiments
Evolution Controller (Evolver)	Houses the evolutionary component of the system and runs specific experiments involving many market repeats
SSCM Generator	Creates SSCM representations for solving in each market
Timer	Provides timing services to the markets being run, counting each market time-step
Customers and Suppliers	The customer and supplier market participants configured directly by the Evolver and Generator
Middlemen	Based on an SSF-I1 implementation and configured via the evolutionary component of the Evolver.
JMS Server	Optionally, an independent Java Messaging Service (JMS) server to allow distribution of the system over multiple machines by providing a message passing service

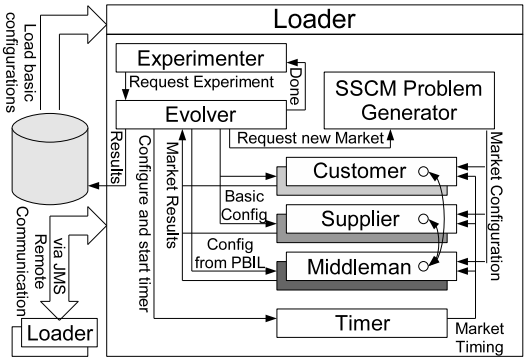

Figure B.1: SMSS Software Components

Appendix C

SMSS Experimentation, Further Information

This appendix contains additional information pertaining to the SMSS experiments discussed in Chapter 7.

This information primarily relates to the specifics of each experimental configuration but, also includes details of the data analysis mechanisms used.

Table 7.1 shows the basic configuration used in all experiments.

C.1 Sum Of Probability Differences (SOPD)

The sum of probability differences is a measure of convergence used for the investigation of probability distributions resulting from SMSS experiments. SOPD compares two probability distributions on a matched parameter basis. The mean of the differences provides a score for the distributions as a whole. A score of zero indicates the compared distributions (or elements) are identical, a score of two that they are entirely different.

Parameters are matched according to their use in the SSF strategy so, for instance, *GroupActivationTime* from both distributions might be compared but not *GroupActivationTime* against *GroupDuration*.

Parameters with discrete and continuous number ranges are represented differently within the distribution and this difference is reflected in how the SOPD is calculated for each.

For a discrete (or symbolic) parameter, the absolute difference in probability of each value is found. These differences are summed to obtain the SOPD score for this parameter. See Figure C.1.

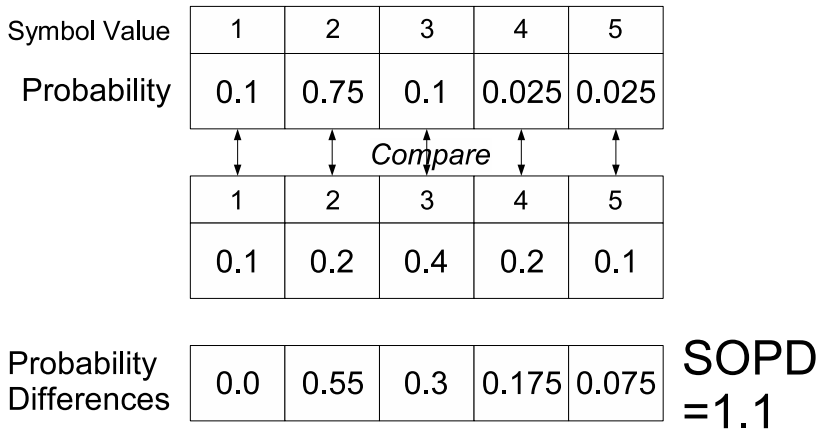

Figure C.1: SOPD Calculation for Symbolic parameter

The representation of continuous parameters makes a comparison more difficult. Since the boundaries of blocks in a continuous range parameter may well be in different places, a direct comparison as for symbolic parameters isn't immediately possible. An initial reconciliation step must be made for the parameters from both distributions first. This is show in Figure C.2

The reconciliation process that allows continuous range parameters to be compared aims to allow the same mechanism used for symbolic cells to be applied. To achieve this both parameters must have identical block sets. Since the continuous parameters are highly unlikely to have matching blocks, a temporary set is found for each that allows the comparison. The reconciliation process starts by combining the block boundaries from both parameters and adding to this the upper and lower range values. With duplicates removed, the resulting set of boundaries spans the full range of both parameters. Taking the boundaries as block definitions, the range is filled with blocks that either exactly match corresponding blocks in one parameter or span a subset of the space of a block in a parameter. Using this block set as a template, probabilities are found for each block for both parameters. To assign a probability to a block in the template for a parameter, the block's boundaries are compared against those of the parameters. Either the template block will exactly match a block in the parameters set or, it will occur as a subset of the range of a block in that parameters set. If the template block matches exactly, that blocks probability may be used directly with the template. If the block is a subset, the probability assigned to the block will be the containing blocks probability multiplied by the proportion of that blocks range that the template block represents.

Having found probabilities for each block in the template for both parameters, the differences between those probabilities can be found and summed in the same manner as for symbolic parameters, yielding the SOPD for the continuous parameters.

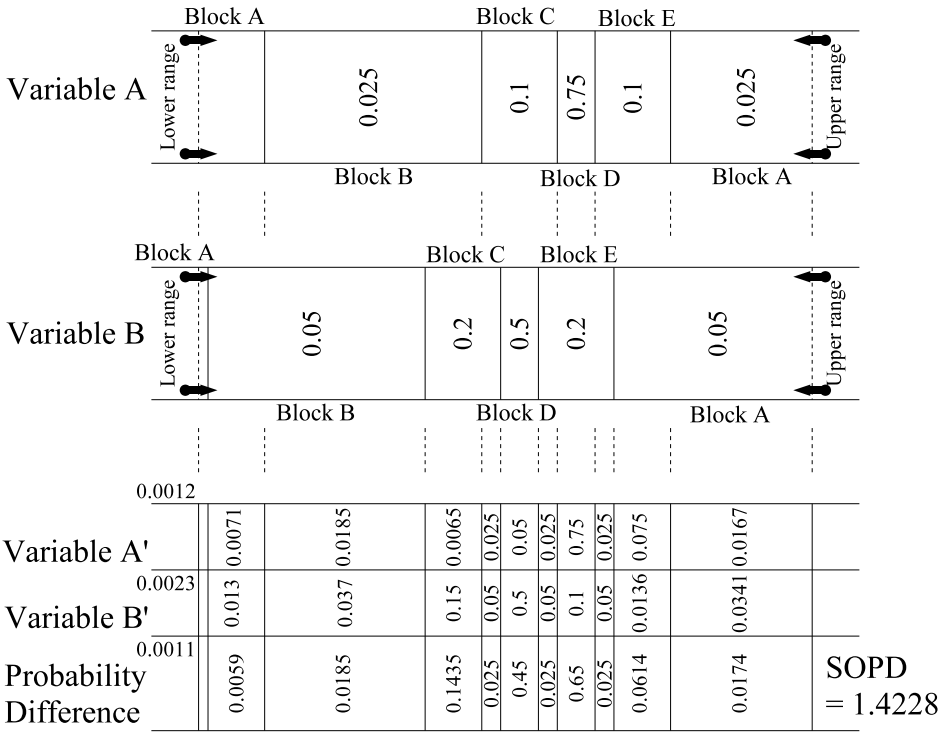

Figure C.2: SOPD Calculation for Continuous parameter

C.2 Evaluating Middlemen, Use of Positive Group Skew

The SMSS provides several mechanisms by which middlemen strategies may be evaluated prior to selection and reinforcement of the probability distribution. In Section 7.3.1 it is asserted that the use of positive group skew is more effective when combined with negotiation based scoring. Figure C.3 shows how performance of middlemen strategies is affected if negotiation based scoring is not used under the control conditions case. As can be seen, learning is less effective leading to a *7.0* point higher score entropy score while mean middleman wealth is reduced by around *2500.0* to approximately *10000.0* by the end of the experiment.

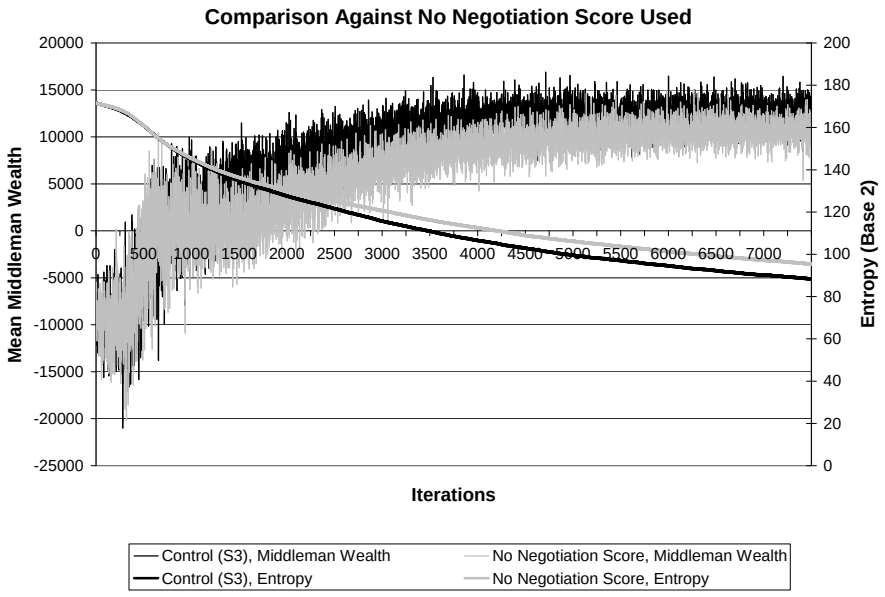

Figure C.3: CTRL-3 Condition Experiment Comparing Use Of Positive Group Skew With And Without Negotiation Based Scoring

C.3 Averaging Market Evaluations, the use of Repeats

While the SMSS is able to determine an evaluation score for a middleman strategy from a single market simulation this would be subject to fluke. To mitigate this problem the SMSS is able to average results for a single strategy over several market simulations and so yield a more accurate evaluation of performance relative to the supplied environment and the other market participants.

In Section 7.3.1 it is asserted that averaging five markets is sufficient for a reasonable estimation of strategy performance, this is supported here by comparing the effects of winner and loser selection from an experiment making use of ten repeats and comparing if the same winner and loser would have been selected if only five of those evaluations had been used.

The baseline experimental configuration (CTRL-3) was run using ten repeats instead of the usual five. Results from this experiment were analysed to see which middleman strategy won in each of the iterations based on both the full set of repeats and a selection of five of the ten repeats. The selected winners could then be compared, five repeats accruing one point for every correct selection. The same process is repeated for losing strategies.

The result of this test is a score between 0 and the maximum number of iterations, 7500, that represents the effectiveness of using five repeats as opposed to ten repeats. The higher the score, the closer to equivalent with ten repeats based evaluation.

For winner selection, five repeats matched ten repeats 3851 times out of the 7500 iterations. For loser selection this figure was 4382. This presents an accuracy of

51% and 58% respectively. Not ideal figures by feasible given the time saving in computation.

Curiously, the five repeats mechanism appears to be more effective earlier on in the experiment. In the first 2000 iterations it achieves accuracies of 68% and 81% for winner and loser selection respectively. This suggests that for the earlier period of an experiment, in which more rapidly learning occurs, the five repeat mechanism will prove more effective than in the general case.

Bibliography

[1] Patricia Anthony and Nicholas R. Jennings, *Evolving bidding strategies for multiple auctions.*, ECAI, 2002, pp. 178–182.

[2] ———, *Developing a bidding agent for multiple heterogeneous auctions*, ACM Transactions on Internet Technology **3** (2003), no. 3, 185–217.

[3] ———, *A heuristic bidding strategy for multiple heterogeneous auctions*, Fifth International Conference on Electronic Commerce (ICEC2003), ACM Press New York, NY, USA, 2003, pp. 9–16.

[4] R. Arunachalam and N.M. Sadeh, *The supply chain trading agent competition*, Electronic Commerce Research and Applications **4** (2005), no. 1, 66–84.

[5] L.M. Ausubel, P. Crampton, and R.J. Deneckere, *Bargaining with incomplete information*, Handbook of Game Theory, vol. 3, Amsterdam: Elsevier Science B.V., 2002, pp. 1897–1945.

[6] R. Axelrod, *The evolution of strategies in the iterated prisoner's dilemma*, Genetic algorithms and simulated annealing (Research notes in AI), Pitman/Morgan Kaufmann, 1987, pp. 32–41.

[7] T. Back, D.B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, IOP Publishing Ltd. Bristol, UK, UK, 1997.

[8] S. Baluja, *Population based incremental learning - a method for integrating genetic search based function optimisation and competitive learning*, Tech. Report CMU-CS-94-163, Pittsburgh, PA: Carnegie Mellon University, 1994.

[9] Shumeet Baluja and Rich Caruana, *Removing the genetics from the standard genetic algorithm*, The Int. Conf. on Machine Learning 1995 (San Mateo, CA) (A. Prieditis and S. Russel, eds.), Morgan Kaufmann Publishers, 1995, pp. 38–46.

[10] Nordin P. Keller R. Kaufmann M. Banzhaf, W., *Genetic programming an introduction*, Morgan Kaufmann, 1998.

[11] K. Binmore, *Game theory and the social contract i, playing fair*, MIT Press, 1994.

[12] ———, *Game theory and the social contract ii, just playing*, MIT Press, 1998.

[13] M.F. Bramlette and E.E. Bouchard, *Genetic algorithms in parametric design of aircraft*, Handbook of Genetic Algorithms (1991), 109–123.

[14] T. Brodmeier and E. Pretsch, *Application of genetic algorithms in molecular modeling*, Journal of Computational Chemistry **15** (1994), no. 6, 588–595.

[15] N.R. Jennings C. Sierra, P. Faratin, *Deliberative automated negotiators using fuzzy similarities*, EUSFLAT-ESTYLF Joint Conference on Fuzzy Logic, 2000, pp. 155–158.

[16] Yoon Chang and Harris Makatsoris, *Supply Chain Modelling Using Simulation*, International Journal of Simulation **2** (2001), no. 1, 24–30.

[17] A. Chavez and P. Maes, *Kasbah: An agent marketplace for buying and selling goods*, First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96) (London, UK), Practical Application Company, 1996, pp. 75–90.

[18] G. Cheliotis, *Structure and Dynamics of Bandwidth Markets*, Ph.D. thesis, IBM Research, 2001.

[19] Ye Chen, Yun Peng, Tim Finin, Yannis Labrou, Scott Cost, Bill Chu, Jian Yao, Rongming Sun, and Bob Wilhelm, *A negotiation-based multi-agent system for supply chain management*, In Working Notes of the Agents '99 Workshop, Third Conference on Autonomous Agents (Agents-99), Seattle, WA, ACM Press, 1999.

[20] G.M. Clarke and D. Cooke, *A Basic Course in Statistics*, Arnold New York, 1998.

[21] J. Collins, M. Tsvetovat, B. Mobasher, and M. Gini, *MAGNET: a multi-agent contracting system for plan execution*, Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice, Albuquerque, NM, August 1998 (1998), 63–68.

[22] T.M. Cover and J.A. Thomas, *Elements of information theory*, Wiley New York, 1991.

[23] Pardoe D. and Stone P., *Tactex-05: A champion supply chain management agent*, Twenty-First National Conference on Artificial Intelligence (2006), 1389–1394.

[24] M.D. Davis, *Game Theory: A Nontechnical Introduction*, Dover Publications, 1997.

[25] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings, *Bargaining with incomplete information*, Annals of Mathematics and Artificial Intelligence **44** (2005), no. 3, 207–232.

[26] H. Furuta, K. Maeda, and E. Watanabe, *Application of genetic algorithm to aesthetic design of bridge structures*, Microcomputers in Civil Engineering **10** (1995), no. 6, 415–421.

[27] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[28] T. Gosling, *The simple supply chain model and evolutionary computation*, Evolutionary Computation, 2003. CEC'03. The 2003 Congress on **4** (2003), 2322–2329.

[29] T. Gosling, N. Jin, and E. Tsang, *Population Based Incremental Learning with Guided Mutation Versus Genetic Algorithms: Iterated Prisoners Dilemma*, Evolutionary Computation, 2005. The 2005 IEEE Congress on **1** (2005), 958–965.

[30] T. Gosling, N. Jin, and E. Tsang, *Games, supply chains, and automatic strategy discovering using evolutionary computation*, Handbook of Research on Nature Inspired Computing for Economics and Management, Idea-Group, 2006, pp. 572–588.

[31] T. Gosling and E. Tsang, *Technical report 1: The simple supply chain model (sscm)*, Tech. Report CSM-392, Department of Computer Science, University of Essex, 2003.

[32] ———, *Tackling the simple supply chain model*, Congress on Evolutionary Computation 2006 (CEC2006), 2006, pp. 2179–2186.

[33] A. Greenwald, *The 2002 trading agent competition: An overview of agent strategies*, AI Magazine **24** (2003), no. 1, 83–91.

[34] A. Greenwald and J. Boyan, *Bidding algorithms for simultaneous auctions: A case study*, Proceedings of Third ACM Conference on Electronic Commerce (2001), 115–124.

[35] GR Harik, FG Lobo, and DE Goldberg, *The compact genetic algorithm*, Evolutionary Computation, IEEE Transactions on **3** (1999), no. 4, 287–297.

[36] M. He and N.R. Jennings, *SouthamptonTAC: Designing a successful trading agent*, Fifteenth European Conference on Artificial Intelligence (2002), 8–12.

[37] M. He, N.R. Jennings, and H. Leung, *On agent-mediated electronic commerce*, IEEE Trans on Knowledge and Data Engineering **15** (2003), no. 4, 985–1003.

[38] I. Inza, M. Merino, P. Larra naga, J. Quiroga, B. Sierra, and M. Girala, *Feature subset selection by population-based incremental learning. a case study in the survival of cirrhotic patients treated with tips*, Tech. Report EHU-KZAA-IK-1/99, University of the Basque Country, Spain, 1999.

[39] A. Iselt, A. Kirstadter, and R. Chahine, *Bandwidth trading-a business case for ASON?*, Telecommunications Network Strategy and Planning Symposium. NETWORKS 2004, 11th International (2004), 63–68.

[40] Andreas Iselt, Andreas Kirstdter, and Rayane Chahine, *Ason for bandwidth trading*, ITG FACHBERICHT (2004), 47–52.

[41] Nicholas R. Jennings Peter McBurney Simon Parsons Liz Sonenberg Iyad Rahwan, Sarvapali D. Ramchurn, *Argumentation-based negotiation*, The Knowledge Engineering Review **18** (2004), no. 4, 343–375.

[42] R. JAIN and P.P. VARAIYA, *COMBINATORIAL EXCHANGE MECHANISMS FOR EFFICIENT BANDWIDTH ALLOCATION*, Journal on Communications in Information and Systems **3** (2004), no. 4, 305–324.

[43] N. Jin and E. Tsang, *Co-evolutionary Strategies for an Alternating-Offer Bargaining Problem*, IEEE 2005 Symposium on Computational Intelligence and Games CIG05, 211–217.

[44] K.L. Judd and L. Tesfatsion, *Handbook of Computational Economics II: Agent-Based Computational Economics*, Handbooks in Economics Series, North-Holland, the Netherlands (2005).

[45] M. Kern, *Parameter Adaptation in Heuristic Search–A Population-Based Approach–*, Ph.D. thesis, University Of Essex, 2006.

[46] A. Kirstädter, A. Iselt, A. Autenrieth, D.A. Schupke, R. Prinz, and B. Edmaier, *Business Models for Next Generation Transport Networks*, Photonic Network Communications **10** (2005), no. 3, 283–296.

[47] H. Kitano, *Designing neural networks using genetic algorithms with graph generation system*, Complex Systems **4** (1990), no. 4, 461–476.

[48] ———, *RoboCup-97: Robot Soccer World Cup I*, Springer, 1998.

[49] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, *RoboCup: The Robot World Cup Initiative*, Proceedings of the first international conference on Autonomous agents (1997), 340–347.

[50] Paul Klemperer, *Auction theory: A guide to the literature*, Journal of Economic Surveys **13** (1999), no. 3, 227–86.

[51] J.R. Koza, M.J. Streeter, M.A. Keane, J. Yu, G. Lanza, and W. Mydlowec, *Genetic Programming IV: routine human-competitive machine intelligence*, ch. 13, Kluwer Academic Pub, 2003.

[52] D. Kraines and V. Kraines, *Learning to cooperate with pavlov - an adaptive strategy for the iterated prisoner's dilemma with noise*, Theory and Decision **35** (1993), 107–150.

[53] S. Kraus, *Strategic negotiation in multiagent environments (intelligent robotics and autonomous agents)*, MIT Press, 2001.

[54] S. Kraus, K. Sycara, and A. Evenchik, *Reaching agreements through argumentation: A logical model and implementation*, Artificial Intelligence **104** (1998), no. 1-2, 1–69.

[55] D.M. Kreps, *Game theory and economic modelling*, Clarendon Press Oxford, 1990.

[56] S. Kullback, *Information theory and statistics*, (1959).

[57] S. Kullback and RA Leibler, *On Information and Sufficiency*, The Annals of Mathematical Statistics **22** (1951), no. 1, 79–86.

[58] Yannis Labrou, Tim Finin, and Yun Peng, *The current landscape of agent communication languages*, IEEE Intelligent Systems **14**, no. 2.

[59] W.B. Langdon and R. Poli, *Foundations of genetic programming*, Springer, 2001.

[60] P.L. Lanzi and A. Strada, *A statistical analysis of the trading agent competition 2001*, ACM SIGecom Exchanges **3** (2002), no. 2, 1–8.

[61] P. Larranaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2001.

[62] B. Linster, *Evolutionary stability in the repeated prisoners' dilemma played by two-state moore machines*, Southern Economic Journal (1992), 880–903.

[63] J.A. Lozano, P. Larrañaga, Inza I., and E. Bengoetxea, *Towards a New Evolutionary Computation: advances in the estimation of distribution algorithms*, Springer, 2006.

[64] SM Lucas and G. Kendall, *Evolutionary computation and games*, Computational Intelligence Magazine, IEEE **1** (2006), no. 1, 10–18.

[65] S. Luke et al., *Genetic programming produced competitive soccer softbot teams for robocup97*, Genetic Programming (1998), 214–222.

[66] Toru Ishida Makoto Yokoo, Edmund H. Durfee and Kazuhiro Kuwabara, *The distributed constraint satisfaction problem: Formalization and algorithms*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING **10** (1998), no. 5, 673–685.

[67] N. Matos, C. Sierra, and N. Jennings, *Determining Successful Negotiation Strategies: An Evolutionary Approach*, Proceedings of the 3rd International Conference on Multi Agent Systems (1998), 182–189.

[68] G.F. Miller, P.M. Todd, and S.U. Hegde, *Designing neural networks using genetic algorithms*, Proceedings of the third international conference on Genetic algorithms table of contents (1989), 379–384.

[69] M. Mitchell, *An introduction to genetic algorithms*, MIT Press, 1998.

[70] H. Muehlenbein and T. Mahnig, *FDA - A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions*, Evolutionary Computation **7** (1999), no. 4, 353–376.

[71] R. Myerson, *Game theory: Analysis of conflict*, Cambridge, MA: Havard University Press, 1991.

[72] J. Oliver, *A machine learning approach to automated negotiation and prospects for electronic commerce*, Journal of Management Information Systems **13** (1997), no. 3, 83–112.

[73] M.J. Osborne and A. Rubinstein, *A Course in Game Theory*, Mit Pr, 1994.

[74] M. Pelikan and H. Muehlenbein, *Marginal Distributions in Evolutionary Algorithms*, 4 thInternational Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June (1998), 24–26.

[75] Martin Pelikan and Heinz Mühlenbein, *The bivariate marginal distribution algorithm*, Advances in Soft Computing - Engineering Design and Manufacturing (London) (R. Roy, T. Furuhashi, and P. K. Chawdhry, eds.), Springer-Verlag, 1999, pp. 521–535.

[76] A.E. Roth, *Game-Theoretic Models of Bargaining*, Cambridge Univ Pr, 1985.

[77] A. Rubinstein, *Perfect equilibrium in a bargaining game*, Econometric **50** (1982), no. 1, 97–110.

[78] ———, *A bargaining model with incomplete information about time preferences*, Econometric **53** (1985), no. 5, 1151–1172.

[79] T. Sandholm, *Distributed rational decision making*, Multiagent Systems A Modern Approach to Distributed Artificial Intelligence (G. Weiss, ed.), 201-258, 1999, p. MIT press.

[80] ———, *Expressive commerce and its application to sourcing*, Proceedings of the Eighteenth Conference on Innovative Applications of Artificial Intelligence, 2006.

[81] Carles Sierra Sarvapali D. Ramchurn, Nicholas R. Jennings, *Persuasive negotiation for autonomous agents: A rhetorical approach*, IJCAI Workshop on Computational Models of Natural Argument, 2003, pp. 9–17.

[82] R.E. Schapire, P. Stone, D. McAllester, M.L. Littman, and J.A. Csirik, *Modeling auction price uncertainty using boosting-based conditional density estimation*, Proceedings of the Nineteenth International Conference on Machine Learning (2002), 546–553.

[83] A. Schwartz, *Enron's Missed Opportunity: Enron's Refusal to Build a Collaborative Market Turned Bandwidth Trading Into a Disaster*, Berkeley Roundtable on the International Economy, 2003.

[84] M. Sebag and A. Ducoulombier, *Extending population-based incremental learning to continuous search spaces*, 5th Conference on Parallel Problems Solving from Nature, 1998, pp. 418–427.

[85] Nicholas R. Jennings Shaheen S. Fatima, Michael Wooldridge, *An agenda-based framework for multi-issue negotiation*, Artificial Intelligence **152** (2004), 1–45.

[86] P. Stone and A. Greenwald, *The First International Trading Agent Competition: Autonomous Bidding Agents*, Electronic Commerce Research **5** (2005), no. 2, 229–265.

[87] Rahul Sukthankar, Shumeet Baluja, and John Hancock, *Multiple adaptive agents for tactical driving*, Applied Intelligence **9** (1998), 7–23.

[88] TAC Team, M.P. Wellman, P.R. Wurman, K. O'Malley, R. Bangera, D. Reeves, and W.E. Walsh, *Designing the Market Game for a Trading Agent Competition*, IEEE Internet Computing, **5** (2001), no. 2, 43–51.

[89] P. Toulis, D. Kehagias, and P.A. Mitkas, *Mertacor: a successful autonomous trading agent*, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (2006), 1191–1198.

[90] Edward Tsang, *Foundations of constraint satisfaction*, Academic Press San Diego, 1993.

[91] Marksim Tsvetovatyy, Maria Gini, Bamshad Mobasher, and Zbigiew Wieckowski, *Magma: An agent-based virtual market for electronic commerce*, Journal of Applied Artificial Intelligence **11** (1997), no. 6, 501–523.

[92] R. Unger and J. Moult, *Genetic algorithms for protein folding simulations.*, J Mol Biol **231** (1993), no. 1, 75–81.

[93] P. Vas, *Artificial-Intelligence-Based Electrical Machines and Drives: Application of Fuzzy, Neural, Fuzzy-Neural, and Genetic-Algorithm-Based Techniques*, Oxford University Press, 1999.

[94] I.A. Vetsikas and B. Selman, *A principled study of the design tradeoffs for autonomous trading agents*, Proceedings of the second international joint conference on Autonomous agents and multiagent systems (2003), 473–480.

[95] W.E. Walsh, *Market protocols for decentralized supply chain formation*, Ph.D. thesis, University Of Michigan, 2001.

[96] M.P. Wellman, J. Estelle, S. Singh, Y. Vorobeychik, C. Kiekintveld, and V. Soni, *STRATEGIC INTERACTIONS IN A SUPPLY CHAIN GAME*, Computational Intelligence **21** (2005), no. 1, 1–26.

[97] M.P. Wellman, A. Greenwald, P. Stone, and P.R. Wurman, *The 2001 trading agent competition*, Fourteenth Conference on Innovative Applications of Artificial Intelligence, 2000, pp. 935–941.

[98] M.P. WELLMAN, A.M.Y. GREENWALD, P. STONE, and P.R. WURMAN, *The 2001 Trading Agent Competition*, Electronic Markets **13** (2003), no. 1, 4–12.

[99] M.P. Wellman, P.R. Jordan, C. Kiekintveld, J. Miller, and D.M. Reeves, *Empirical game-theoretic analysis of the TAC market games*, AAMAS-06 Workshop on Game-Theoretic and Decision-Theoretic Agents (2006).

[100] P. Wurman, M. Wellmand, and W. Walsh, *The michigan internet auctionbot: A configurable auction server for software agents*, In Second International Conference on Autonomous Agents, 1998, pp. 301–308.

[101] F. Zhang, YF Zhang, and AYC Nee, *Using genetic algorithms in process planning for job shop machining*, Evolutionary Computation, IEEE Transactions on **1** (1997), no. 4, 278–289.

[102] G. Zlotkin and J.S. Rosenschein, *Mechanisms for Automated Negotiation in State Oriented Domains*, Journal of Artificial Intelligence Research **5** (1996), 163–238.