Tackling the Simple Supply Chain Model

Timothy Gosling and Edward Tsang, *University of Essex, UK*

***Abstract*— In the future a need will exist, if it does not already, to automate supply chains as trading electronically becomes increasingly important. Using the Simple Supply Chain Model (SSCM) allows a supply chain situation to be captured for experimentation. This paper describes efforts to evolve strategies for tackling SSCM specified problems through the use of a Strategy Framework (SSF) and Market Simulation System (SMSS). While the SSF provides a basic strategy representation system, the SMSS evolves strategies over multiple supply chain simulations using Population Based Incremental Learning with Guided Mutation. The paper further discuss some of the techniques being used to analyse the resultant data.**

I. Introduction

In the future a need will exist, if it does not already, to automate supply chains as trading electronically becomes increasingly important ([1], [2], [3]).

In order to automate a supply chain the strategies to be used by each participant must be developed in some way. While game theory and theories of equilibria are effective tools for the analysis of different problems they are unable to deal with the increased complexity and uncertainty inherent in many real-life situations, including supply chains, and as such solutions must be hand crafted using the available domain knowledge. While this approach can prove effective the resultant strategies may still need tuning, a process complicated by uncertainty over parameter interaction and the effects of unexpected situations. The use of Evolutionary Computation (EC) provides a potential solution to this problem. If the supply chain environment can be captured and a reasonable strategy representation provided, EC has the potential to evolve robust strategies and/or parameter sets.

The Simple Supply Chain Model, outlined below, provides one way of capturing a supply chain environment in such a way that EC may be applied. This paper is concerned with how a strategy framework and market simulation system have been created to tackle SSCM defined problems and how the resultant data may be analysed.

A. The Simple Supple Chain Model (SSCM)

The Simple Supply Chain Model (SSCM, [4]; [5]) provides a way to describe simple but non-trivial supply chain situations in their entirety.

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 was partially inspired by the Trading Agent Competition ([6]).

The SSCM description takes the form of the information known by each participant in the chain at the start of a set time period, the types of goods available and the communication mechanism being used. Participants may be one of three types, customers, suppliers or middlemen. Customers are aware of their own requirements to be fulfilled, when this needs to be achieved, for how much and which middleman they know capable of assisting them. Suppliers know what products they can supply, what their value is, how much they are able to supply and what middlemen may be interested. Middlemen have knowledge of potential customers and suppliers. All participants are restricted in the number of outbound communications they can make. See Figure 1.

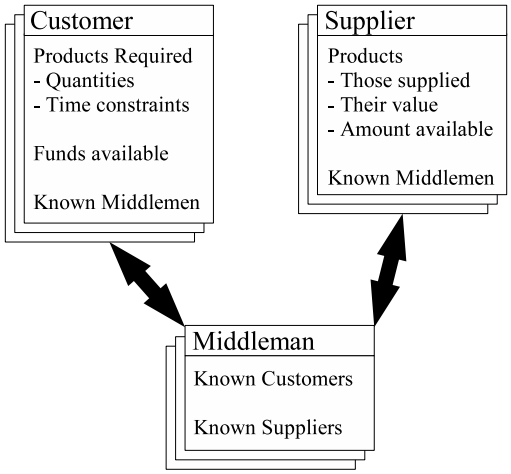

Fig. 1. SSCM Overview

While the SSCM constrains the initial information known by participants and how they communicate, it does not restrict what they can do and how they should act.

B. SSCM Scenarios

Since the range of problems that can be represented by the SSCM is large, in order to begin tackling it the first step is to further restrict how the SSCM will be used, how participants will act and what will be studied. We initially decide that the strategy of the middleman is of most interest and so the restrictions placed on the SSCM will direct attention this way. We also decide to use a simple alternating offers negotiation mechanism for communication.

Two restricting scenarios are detailed here.

1) Scenario One: Scenario one is comparatively simple and places many restrictions on the SSCM. The strategy of customers and suppliers is largely determined by the scenario although there is some scope for when the customer may communicate with the middleman initially and how the

supplier will go about bargaining for products. The idea is to place the middleman between a satisfaction problem on one side and an optimisation problem on the other.

Customers have a fixed requirement, know only of one middleman and are required to initiate communication. They should not expect any negotiation from the middleman and will simply wait to see if the middleman can fulfill their requirements.

Suppliers are only able to supply one product and only one supplier exists for each. Suppliers know of no middlemen initially and should not initiate negotiations. When dealing with middleman negotiations a supplier will try to match the required products as closely as possible but will not offer alternatives. A supplier will continue to negotiate as long as it is able to provide a subset of the requested products.

Middlemen know of all suppliers and no customers initially. Having received customer requirements the middleman will negotiate with suppliers for those products before informing the customer if it is able to fulfill their requirements.

2) Scenario Two: Scenario two relaxes restrictions on use of the SSCM and the behavior of agents in order to make the problem more interesting. The middleman is now able to negotiate for alternative product sets with the customer and may have to choose between multiple suppliers on the other side.

Customers are happy to accept variation in requirements within certain bounds, to this end they will provide an initial requirement but may be willing to agree to a different but similar allocation. They still only know of one middleman and are responsible for initiating communication.

Suppliers act as in Scenario One but multiple suppliers may exist for each product.

Middlemen are largely as in Scenario One but now have the option of trying to find alternative product sets for Customers. Middlemen may have the option to select between different suppliers for one product or to make use of all suppliers as required.

II. The SSCM Strategy Framework (SSF)

The SSCM Strategy Framework (SSF) provides a conceptual framework through which middleman strategies may be developed to tackle different SSCM Scenarios. The SSF is based on two ideas. Firstly customer requirements provide the motivation for all middleman actions. Secondly, that communication should be minimised.

The SSF consists of a customer requirement grouping mechanism, a control logic for processing the groups and a series of specific behaviour adjusting decision points within that logic. The decision points allow for the variation and evolution of middleman behaviour while the basic framework and its control logic should result in reasonably sensible middleman operation. Grouping together customer requirements allows them to be dealt with as an amalgamation and so reduce the communication burden on all parties.

The SSF handles communication between market participants via a simple alternating offers negotiation mechanism. In this way messages are restricted to either a product set to be negotiated over and a price, or an accept/reject of a previous offer. This common mechanism is simple but flexible enough to allow a range of operations including customer and supplier side negotiations and product supplier discovery. In this context a customer requirement is the latest product set suggested by the customer.

The customer requirement group mechanism makes use of three group types: Basic, Pre-Negotiation and Failure. These are shown in context in Figure 2. The following sections describe these groups in more detail.

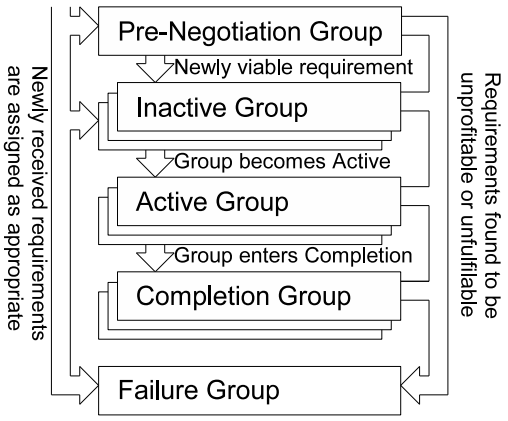

Fig. 2. SSF Groups

A. SSF Basic Groups

The SSF specifies that customer requirements are handled in groups. This mechanism is intended to minimise the communication burden on all participants by removing the need to fulfill each requirement separately.

A Basic Group is the primary manifestation of this idea and consists of its collected customer requirements, associated supplier negotiations and the set of products obtained so far.

A Basic Group goes through up to seven phases in attempting to fulfill customer requirements.

In the first, Inactive, phase the group collects new customer requirements according to some requirement similarity criteria.

An Inactive group will become Activated at which point it attempts to obtain any existing unneeded products held by the middleman that would help fulfill its customer requirements. The group then becomes Active.

An Active group attempts to fulfill its requirements through negotiation with the suppliers. If a problem arises with a requirement the group may enter a Waiting state. Once negotiations succeed or fail the group enter Successful or Unsuccessful Completion.

A Waiting group attempts to bring all supplier negotiations into a safe state to drop a customer requirement without obtaining then unneeded goods. Successful or not a group will then become Active.

A group that enters Successful completion informs each customer of its success in fulfilling its requirement before returning any spare resources for general use. An Unsuccessful group simply tries to close any outstanding negotiations with minimum loss. After completion all groups are considered to be Finished and may be removed.

B. SSF Pre-Negotiation Group

While the Basic Groups handle supplier side negotiation the Pre-Negotiation group (or groups, depending on implementation) handle initial customer side negotiation.

Before a customer requirement is assigned to a Basic Group it must be both likely to be profitable and probably feasible. The Pre-Negotiation group accepts customer requirements that are likely to be profitable but are unlikely to be feasible. The group then attempts to negotiate with the customer to find an acceptable feasible requirement that can then be passed on to a Basic Group for fulfillment.

To negotiate with a customer the Pre-Negotiation group presents a series of profitable, feasible alternatives. If the customer responds with the same set or a similar profitable and feasible set, attempts to fulfill the requirement can begin.

C. SSF Failure Group

The SSF Failure group is a simple catch all group for all negotiations (customer or supplier) that should be failed. As such unprofitable customer requirements end up here along with any requirements later dropped by a Basic or Pre-Negotiation group. The groups sole function is to, as rapidly as possible, inform each negotiation partner that the middleman rejects the most recent offer.

D. SSF Groups and Communication Priority

Each group in the middleman is essentially operated in parallel, each generating any required outbound messages. On each iteration however only one of these messages will be sent, the result of which may affect subsequent message generation. To this end a communication priority system must be put in place to select the sent message. This mechanism constitutes one of the decision elements of the control logic discussed below.

E. SSF Control Logic

While the SSF Groups provide the primary framework for handling customer requirements, they also require a control logic to determine their operation. The SSF control logic includes a number of decision points that affect agent behaviour. These decision points require further specification but allow a middleman's strategy to be evolved or otherwise adjusted. The following algorithms are used by an SSF based middleman to control its use of groups.

An SSF strategy works over a number of iterations, each iteration sending at most one outbound communication to another market participant. This process looks as shown in Algorithm 1.

Algorithm 1 SSF Top Level Iteration

```
1: while NOT Market Stop Condition Occured do
2:     Process All Inbound Messages
3:     Process Failure Group, record possible message
4:     Process Pre-Negotiation Group, record possible message
5:     Process All Basic Groups, record possible message
6:     Select And Send One Message (if any)
7: end while
```

The length of the market is specified by the SSCM. Many SSF iterations may occur per SSCM time unit, the timing being applied externally.

The Failure Group process simply selects the highest priority negotiation to report a reject message to and generates this message for consideration.

If a message is selected and sent this must be reported back to the source group. For the Failure Group this means a negotiation can be safely discarded. For the Pre-Negotiation group this means the attempted alternative needs to be removed from the set of possible alternatives for a customers requirement. For Basic Groups various effects are possible including removal of successful customer requirements and attaching newly initiated negotiations.

Since all SSF Groups consist of negotiations it is important to process all inbound messages before proceeding (Algorithm 2). Part of this process is to assign new customer negotiations (requirements) to a relevant group for handling.

Algorithm 2 Process All Inbound Messages

```
1: for all New Messages, msg do
2:     if New Customer Negotiation (msg) then
3:         if Profitable (msg) then
4:             if Feasible (msg) then
5:                 Assign To Or Create Basic Group (msg)
6:             else
7:                 Assign To Pre-Negotiation Group (msg)
8:             end if
9:         else
10:            Assign To Failure Group (msg)
11:        end if
12:    else
13:        Process Message In Relation To Associated Group (msg)
14:    end if
15: end for
```

Assignment of customer requirements to a basic group must be dealt with via some similarity mechanism to group

requirements together. If no suitable inactive group is found a new basic group is formed.

Message processing in relation to an ongoing negotiation in a group may mean handling new product sets, prices or accept/reject messages. A Group's response to these messages may vary and some effects may be delayed until the group processing stage.

For the Pre-Negotiation group a new message will be a customer response to a suggested alternative. If the customer has agreed the new requirement is then assigned to a Basic Group.

For Basic Groups negotiation messages will be from suppliers. This may include alternative product sets being offered, new prices or accept/reject responses. A Basic Group's response may be passive, simply waiting for the next active group phase to supply a response or more active, removing requirements that are no longer viable because a product is unavailable.

Pre-Negotiation Group Processing (Algorithm 3)is important for the successful fulfilment of otherwise unviable customer requirements.

Algorithm 3 Process Pre-Negotiation Group

```
for all For Each Customer Requirement (req) do
    if NOT Waiting For Response (req) then
        if Profitable (req) then
            Prune Available Alternative (req)
            if NOT Has Alternative (req) then
                Assign To Failure Group (req)
            end if
        else
            Assign To Failure Group (req)
        end if
    end if
end for
if Any Non-Waiting Requirements then
    req = Highest Priority Non-Waiting Requirement
    alt = Best Requirement Alternative (req)
    Generate Message For Alternative (alt, req)
end if
```

Basic Group processing is complicated both by the number of groups but also the number of different states the groups may be in.

While there is a chance of dropping unprofitable requirements the Waiting state attempts to get all relevant negotiation into a safe state (one in which a response is not being waited for), it does this by preventing the group from sending any messages. If negotiations urgently require a response the state will exit and allow negotiations to continue.

F. SSF Decision Points

While the SSF grouping mechanism and control logic provide a framework for SSCM Middleman Strategies, it

Algorithm 4 Process All Basic Groups

```
for all Basic Groups (bg) do
    if Inactive (bg) then
        Process Inactive Group (bg)
    else if Activated (bg) then
        Process Activated Group (bg)
    else if Active (bg) then
        Process Active Group (bg)
    else if Waiting (bg) then
        Process Waiting Group (bg)
    else if Completion (bg) then
        Process Completion Group (bg)
    else if Finished (bg) then
        Remove Group (bg)
    end if
end for
```

Algorithm 5 Process Inactive Group (bg)

```
for all Requirements (req) do
    if NOT Profitable (req) OR NOT Feasible (req) then
        Assign To Failure Group (req)
    end if
    if Should Become Activated then
        Group State Set To Activated
    else if Should Become Finished (if empty) then
        Group State Set To Finished
    end if
end for
```

Algorithm 6 Process Activated Group (bg)

```
Determine Amalgamated Requirement
Obtain Free Resources
Group State Set To Active
```

Algorithm 7 Process Active Group (bg)

```
for all Requirements (req) do
    if NOT Profitable (req) AND NOT Obtained Items
    For (req) then
        if Negotiations Safe (req) then
            Assign To Failure Group (req)
        else if Safe To Wait then
            Set Group To Waiting State
        end if
    end if
end for
if NOT Waiting then
    if Should Enter Completion then
        Group State Set To Completion
    else
        if Waiting Supplier Negotiation then
            supneg = Highest Priority Supplier Negotiation
            Generate Message For Supplier Negotiation
            (supneg)
        end if
    end if
end if
```

Algorithm 8 Process Waiting Group (bg)

```
for all Requirements (req) do
    if NOT Profitable (req) AND NOT Obtained Items For (req) AND Negotiations Safe (req) then
        Assign To Failure Group (req)
    end if
end for
if NOT Safe To Wait then
    Group State Set To Active
end if
```

Algorithm 9 Process Completion Group (bg)

```
if Supplier Negotiations Outstanding then
    if Can Accept Supplier Negotiations then
        supneg = Highest Priority Supplier Negotiation
        Generate Accept Message For Supplier Negotiation (supneg)
    else
        for all Outstanding Supplier Negotiation (supneg) do
            Assign To Failure Group (supneg)
        end for
    end if
else
    if Any Customer Requirements then
        for all Customer Requirements (req) do
            if NOT Can Fulfill Requirement (req) then
                Assign To Failure Group (req)
            end if
        end for
        if Any Customer Requirements then
            req = Select Highest Priority Fulfillable Requirement
            Generate Accept Message (req)
        else
            Group State Set To Finished
        end if
    else
        Group State Set To Finished
    end if
end if
```

also leaves several decision mechanisms to be defined by any implementation. By parameterising the mechanisms chosen, evolution of Middleman strategies can be performed. The set of implementation decisions is shown below:

The SSF implementation specific mechanisms

- Customer requirement likely profitability
- Customer requirement viability (does the middleman believe the requirements can be met)
- Generating viable alternatives to a customer requirement
- Determining similarity of customer requirements so that they may be grouped together
- Viable alternatives prioritisation
- Customer requirement prioritisation
- When a basic group should become active
- When a basic group should enter completion
- Supplier selection
- Supplier negotiation offer and counter-offer generation
- Outbound messages prioritisation

We ultimately aim to evolve SSF base middleman strategies. With the SSF now defined we next discuss the system used to accomplish this.

III. The SSCM Market Simulation System (SMSS)

The SSCM Market Simulation System (SMSS) allows for the evolution of SSF based middleman strategies through the use of Population Based Incremental Learning with guided mutation (PBIL+GM). The core of the SMSS is a feedback loop that instantiates middleman strategies within a supply chain simulation, evaluates them and then provides positive and negative reinforcement for the next round of middleman instantiations. The supply chains being tackled are defined in terms of the SSCM with the appropriate information being relayed to the various participants at the beginning of the market. The PBIL distribution and market results are recorded at each stage for further analysis. See Figure 3.

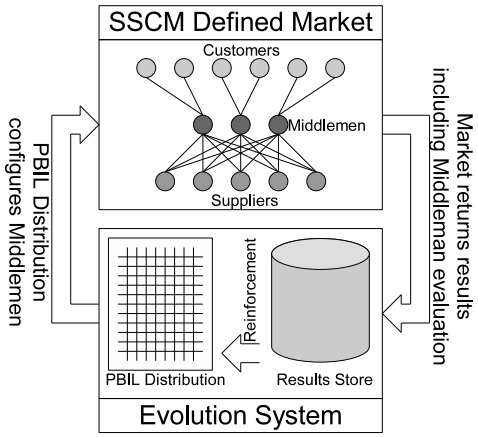

Fig. 3. SMSS feedback loop

The complete SMSS comprises a number of components to provide a high degree of flexibility to the market simulation, the process of evolution and the experiments being run. A basic break down of these components is as follows:

- Agent loader, to load and run all agents in the system.
- Experiment controller (Experimenter), to configure and run experiments.
- Evolution controller (Evolver), houses the PBIL distribution and runs specific experiments.
- Generator, to create SSCM representations solved in each market.
- Timer, to provide timing services to the markets being run.
- Customers, Suppliers and Middlemen, represent that market participants.
- Optionally, an independent Java Messaging Service (JMS) server to allow distribution operation.

The SMSS is implemented in Java and designed to work both on a single local machine or across multiple machines via the JMS Server. This is illustrated in Figure 4.

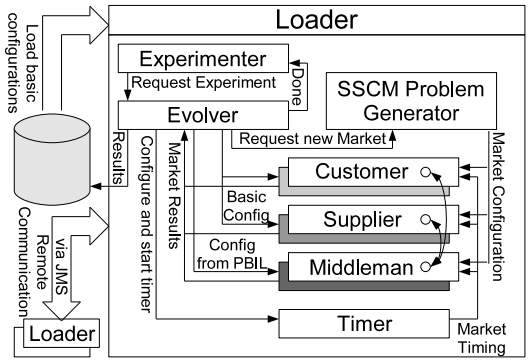

Fig. 4. SMSS component interaction

While the Loader initially loads and configures the SMSS components the Experimenter is responsible for the remainder of SMSS operation.

Once the Experimenter has confirmed all required components are present it will begin performing a series of specified experiments. Specific experiments are carried out by request to the Evolver. The Evolver carries out an experiment by repeatedly configuring the market participants (Customers, Suppliers and Middlemen), requesting a new SSCM configuration from the Generator and then starting the market Timer. Each participant returns results to the Evolver when the market ends. This information is used to update the Evolvers PBIL distribution and recorded for future use. Multiple markets may be run for each reinforcement on the PBIL to increase certainty about how participants performed. When the experiment is complete the Evolver informs the Experimenter which may then request a new experiment or cause the system to exit.

The range of potential experiment configuration is quite extensive with, for example, each participant parameter specified directly, subject to evolutionary forces or drawn from previous PBIL distributions.

Supplier and Customer implementation is simple and follows the rules laid out by Scenario One and Two. A single Customer agent may, however, represent many Customer market participants. Middleman strategies are constrained by the SSF and (usually) parameterised via the Evolvers PBIL distribution.

In making use of the SSF for the evolution of middleman strategies the various elements outlined in the SSF description must be defined. This is done so here:

SMSS, SSF implementation specifics

1) Requirement Profitability - Based on estimated product cost determined from past transactions or an arbitrary guess price if insufficient information.
2) Requirement Viability - Based on past failure to obtain the required products. A threshold number of failures is used beyond which a product is considered unobtainable.
3) Alternatives Generation - Problem specific. For a TAC like situation all viable start-time, duration pairs of the same duration or less than the original requirement are generated.
4) Requirement Similarity - Each Basic Group handles a certain window of requirement fulfill by time. If groups overlap the least full group is assigned.
5) Alternatives Prioritisation - Based on uclidean distance using duration and start-time. Closest start time wins tie-breaks.
6) Requirement Prioritisation - Based on likely profitability.
7) Basic Group Activation - Specified by a minimum amount of time being needed to negotiation before the earliest requirement in the group must have a response.
8) Basic Group Completion - Specified by a minimum time required to send completion messages before the earliest requirement needs a response.
9) Supplier Selection - Random, even distribution.
10) Supplier Negotiation (offer and counter-offer) - See below.
11) Outbound messages prioritisation - New, Accept, Ongoing Reject. Tie-breaks on soonest to timeout and value.

A. Supplier negotiation, offer and counter-offer generation

In attempting to fulfill their requirements, an SMSS Middleman Basic Group negotiates for each product type separately in order to more easily estimate product values.

Offers and counter-offers are made based on the boulware and conceeder tactic specified by Matos ([7]). This mechanism uses a single parameter to affect how a negotiator adjusts its price over time. Two further parameters multiply the current product value estimate in order to determine the upper and lower price bounds.

Both the middleman and supplier negotiation mechanisms are based on the same principle.

B. Evolution in the SMSS

The SMSS makes use of Population Based Incremental Learning with Guided Mutation (PBIL+GM, [8], [9]) for evolving Middleman strategies.

PBIL combines a traditional evolutionary approach with that of reinforcement learning, replacing a population of solutions with a probability distribution. The probability distribution maintains the chance of each possible value for each variable occurring.

Possible middleman strategies can be generated probabilistically using the distribution. These strategies are then tested in the market. Good strategies are used to increase the probability of their variables values occurring in future while bad strategies may be used to reduce the chance of their variable values reoccurring.

Mutation can be used with PBIL to add additional variability to the solutions generated and so promote greater exploration of the search space. The SMSS makes use of both a traditional mutation operator and guided mutation.

The traditional mutation operator may, with low probability, adjust a strategies variables to some random (evenly distributed) value.

Guided mutation will, with low probability, adjust a strategies variables towards those of the last best strategies found. This mechanism was found to increase the effectiveness of PBIL when generating strategies for the more simple Iterated Prisoners Dilemma problem ([9]).

The PBIL implementation used by the SMSS allows named variables to be defined in terms of four main properties. The first is whether the variable should be considered to operate over a continuous range or if it is symbolic (used for integer values). The second is the number of blocks or divisions. For a continuous variable this defines the number of divisions used to specify probability across the range. For symbolic variables this is the number of discrete points to be used. Finally the upper and lower range must be specified.

When generating values, symbolic cells simply choose from amongst the set of discrete points probabilistically. For continuous variables a particular division is chosen probabilistically then a value is chosen at random (even distribution) from within the range this represents.

Symbolic and continuous variables respond slightly differently to reinforcement. For symbolic cells a positively reinforced value will cause that values probability to be raised by some amount and all other value probabilities to be lowered. For continuous variables the same effect occurs in relation to the division from which a value derived. In addition the boundaries of the division contract towards the reinforcement causing value, so helping to focus a search in this area.

Negative reinforcement works the same way for both symbolic and continuous cells with the causing discrete point or division losing a certain amount of probability that is distributed back to all other elements.

C. SMSS Parameters

The SMSS is highly configurable, most of this control being available via the Experimenter component. Further control is provided by the Generator configuration that allows variation in the SSCM problems presented to the market participants.

On the Experimenter side the number of experiments and the Generator configurations to use are important parameters, along with how participant variables are instantiated (Static, current evolving PBIL distribution or previous PBIL distribution), control of middleman evaluation and the level of reinforcement and mutation to be used.

The SSCM Generator allows the number and wealth of Customers to be specified along with the availability and cost of products at Suppliers. Further, market length, communication restrictions and control over the range of customer requirements is provided for.

While the above parameters are all specified as part of the experiment being run, Middlemen within a market receive parameters for their SSF based strategies from the PBIL component of the Evovler. These parameters directly affect the behaviour of the Middlemen in the market and are shown in Table I below.

Parameter	Use
Group Activation Time	How soon before the earliest requirement of an inactive group needs a response should the group be activated
Group Active Proportion	What proportion (beyond 50%) of the time after group activation is used for negotiation instead of completion
Supplier Timeout	How long the middleman will wait for a response from the supplier
Unavailable Threshold	The threshold for a product at a given time from a particular supplier being deemed unavailable
Alternative Tries	How many customer requirement alternatives will be tried
For products	
Guess Price	The base price the middleman uses to estimate product value
Product Window	How many of the most recent transactions should be used to estimate
For each Product	
Upper Value Multiplier	Multiple of a products estimated value a middleman is willing to pay maximum
Lower Value Multiplier	Multiple of a products estimated value a middleman is willing to pay minimum
Tactic value	Control variable for the negotiation tactic

TABLE I
SSF MIDDLEMAN PARAMETERS

IV. SMSS RESULTS AND ANALYSIS

As mentioned previously the main output of the SMSS is a series of PBIL distributions and information about the results of each market and any resultant learning.

Analysing this information is itself a challenge. At a minimum level we need to answer the following questions.

- Has anything (useful) been learnt?
- Does the system converge towards similar solutions for a given environment?

Answering the first of these questions is relatively straight forward. The entropy level of the PBIL distribution is recorded over the course of an experiment, as the distribution converges so the entropy drops. This shows learning, of some sort, is taking place. Further the mean and median profits for middlemen over the course of the experiments are recorded, ideally these increase and become consistent over time if the system is learning something useful.

Answering the second question is more tricky. To do this the final PBIL distribution from multiple experiments

maybe examined for their similarity. To do this a number of measures are used.

The first of these measures determines the total, mean and median sum of differences between the probabilities of each value for each variable controlled by the distribution. For integer value variables of a set range this is relatively straight forward matter of direct discrete point, probability comparison. For continuous variables it is more complex. The range is broken down according to the combined divisional boundaries in use by each variable. This combined boundary set has probability assigned to its divisions from each variable proportional to the amount of a covering division (one combined division is at most the same size as an original variable division). Having determined the probabilities for a combined divisions from each variable the same comparison mechanism used for symbolic variables can be applied.
This mechanism is useful for obtaining an idea of the raw difference between PBIL distributions but does not provide a way of determining the nature of this difference.

To better obtain an idea how the differences in a PBIL distribution occur we make use of two further measures. The first of these is the mean point of the probability, or how far through the variables range half the probability occurs. This gives an indication of where in the range the probability is distributed. The second is to measure the deviation from this mean point. The differences in the means and deviations can then be examined to help understand how two variables may deviate.

While both of the above methods help understand how similar variables and whole PBIL distributions are, further work is being undertake to improve techniques to make similarity judgements less subjective.

More complex questions can be asked of the experiments and the resultant PBIL distributions. Amongst these are:

- Do cycles occur within an experiment?
- How do PBIL distributions react to environments other than the ones they formed in?
- How do strategies from different distributions perform in their home environment against invaders?
- What are the limits of adaptation in relation to harsh environments?
- Can these limits be extended by using PBIL distributions from similar but less harsh environments?

The first of these may be tackled by examination of both the market data and the PBIL distributions over time, the aim being to discover cycles of behavior related to the learning taking place.

The remainder can be tackled by the configuration of experiments that define a starting PBIL distribution or allow the definition of a sub-set of middleman agents from a different (fixed or not) PBIL distribution.

Efforts in all these areas are ongoing.

A. Current Experiments

At present experiments are being conducted under Scenario Two settings to test the limits of adaptability and determine how final PBIL distributions cluster for the same, similar and different environments.

V. Conclusions

The SSCM provides a way to capture simple, non-trivial supply chains allowing evolutionary experiments to be carried out.

The SSF provides a way to tackle a subset of SSCM problems by restricting the way in which the SSCM is used and placing some constraints on the behavior of supply chain participants.

The SMSS provides an effective tool for the evolution of SSF based middlemen strategies using PBIL+GM over many SSCM defined problems.

The data generated by the SMSS needs to be analysed and some mechanisms for approaching this have been presented.

Experimentation using the SMSS is ongoing and techniques for further analysis are being developed.

Acknowledgment

This work has been partially sponsored by British Telecommunication PLC as part of an EPSRC studentship.

References

[1] M. He, N. Jennings, and H. Leung, "On agent-mediated electronic commerce," 2003.
[2] T. Sandholm, *Distributed Rational Decision Making*. 201-258, 1999, p. MIT press.
[3] W. Walsh, "Market protocols for decentralized supply chain formation," Ph.D. dissertation, University Of Michigan, 2001.
[4] T. Gosling, "The simple supply chain model and evolutionary computation," in *Congress on Evolutionary Computation 2003 (CEC2003)*, 2003.
[5] T. Gosling and E. Tsang, "Technical report 1: The simple supply chain model (sscm)," Department of Computer Science, University of Essex, Tech. Rep. CSM-392, 2003.
[6] M. Wellman, A. Greenwald, P. Stone, and P. Wurman, "The 2001 trading agent competition," in *Fourteenth Conference on Innovative Applications of Artificial Intelligence*, 2000.
[7] N. Matos, C. Sierra, and N. Jennings, "Determining successful negotiation strategies: An evolutionary approach," in *3rd International Conference on Multi-Agent Systems (ICMAS-98)*, 1998.
[8] S. Baluja, "Population based incremental learning - a method for integrating genetic search based function optimisation and competitive learning," Pittsburgh, PA: Carnegie Mellon University, Tech. Rep. CMU-CS-94-163, 1994.
[9] T. Gosling, N. Jin, and E. Tsang, "Population based incremental learning with guided mutation versus genetic algorithms: Iterated prisoners dilemma," in *Congress on Evolutionary Computation 2005 (CEC2005)*, 2005.