In J-P. Rennard (Eds.), Handbook of research on nature inspired computing for economics and management, Chapter 18, 2006 (to appear)

Games, supply chains and automatic strategy discovery using evolutionary computation

Timothy Gosling, Nanlin Jin and Edward Tsang

University of Essex
Department of Computer Science,
University of Essex,
Wivenhoe Park,
Colchester,
CO4 3SQ
United Kingdom

Telephone: +44 1206 872770

Fax: +44 1206 872788

Email: {tdbgos, njin, edward}@essex.ac.uk

In J-P. Rennard (Eds.), Handbook of research on nature inspired computing for economics and management, Chapter 18, 2006 (to appear)

Games, supply chains and automatic strategy discovery using evolutionary computation

Abstract- The use of Evolutionary Computation is significant for the development and optimisation of strategies for dynamic and uncertain situations. Evolutionary Computation has already been used successfully for strategy generation and this chapter introduces three such cases in the form of work on the Iterated Prisoners Dilemma, Rubinstein's Alternating Offers Bargaining Model and the Simple Supply Chain Model. The last of these demonstrates how recent statistical approaches to Evolutionary Computation have been applied to complex supply chain situations that traditional game-theoretical analysis has been unable to tackle.

INTRODUCTION

The use of Evolutionary Computation is important in the development of strategies for dynamic, uncertain situations or for any situation where a simple strategy has many parameters to tune. While game theory and theories of equilibrium are highly effective tools for the analysis of various problems they suffer from being unable to deal with the increased complexity and uncertainty inherent in many real-life situations. Strategies requiring a large number of parameters to be tuned can not effectively be optimised by hand both because those numbers may be so large but primarily because interactions between the parameters are often difficult to understand.

One such problem is that of supply chains and what strategies should be used by participants' to operate effectively within them. Tackling this problem is important because trading electronically will become increasingly important in the future and a need will exist, if it does not already, for many of the transactions to be handled fully automatically (0; 0; 0). Even relatively simple supply chain scenarios prove difficult to analyse and it is usually necessary to resort to domain knowledge in order to develop strategies. While this approach to strategy creation is capable of producing good solutions it is difficult to foresee how they will respond in unexpected situations, guarantee robustness and ensure maximum effectiveness in the face of change. Furthermore even an effective hand crafted solution is likely to require a large number of parameters to be tuned and doing this manually could well prove impossible either because the number of parameters is so large or because they interact in a way that is difficult to understand.

Evolutionary Computation (EC) gives us the potential to address these issues. By defining the supply chain environment, or indeed any other environment, in terms of a reasonable strategy representation scheme and practical strategy evaluation mechanism, EC is able to evolve strategies and/or good parameter sets to tackle the problem.

In this chapter we will be looking at three different strategy generation problems and how EC can be used to tackle them. The first of these, Iterated Prisoners Dilemma (IPD), introduces strategy generation using EC and shows how different algorithms have been used to tackle the same problem. The second problem, Rubinstein's Alternating Offers Bargaining Model (RAOBM), is used to demonstrate that EC can find a known optimal strategy. The final

problem is defined by the Simple Supply Chain Model (SSCM). For the SSCM we show how EC can be used to tackle a far more complex strategy evolution problem by using a supporting strategy framework. In each case we examine why a particular EC algorithm is most appropriate while discussing past efforts and presenting recent work.

GAMES THEORY

Game theory has been highly successful in its application to situations such as the prisoner's dilemma (PD) and Rubinstein's bargaining game along with many others. By starting from a notion of rationality and often, complete information it has proven invaluable and provided a good indication of how to behave in different situations. Since its initial formulation various theories of equilibrium have been posited to help explain how and why certain outcomes do (or should) occur within a game. Some of these, along with other terms, will be referred to during the course of this chapter and we briefly recap on these now.

Dominant strategy – A strategy that yields superior results regardless of the opponent's move.

Dominant Strategy Equilibrium – The outcome of a game reached when all players have a dominant strategy and play it.

Nash Equilibrium – The set of possible results reached by player's playing the best possible strategy in response to their opponents move.

Sub Perfect Game Equilibrium (SPE) – The result of a game if each player moves such that a Nash Equilibrium strategy is played at each sub game, avoiding the worst possible out comes at each stage of a game.

Evolutionary Stable Strategy (ESS) – A strategy that dominates the population and can not suffer from invasion by other (mutant) strategies.

Game theory, as stated in the introduction, while highly successful in analyzing different situations, has difficulty dealing with problems containing a considerable degree of uncertainty or of a highly dynamic nature (essentially the same thing). If a problem cannot effectively be captured then its subsequent analysis by game theory is not possible. While ESS can help explain why, and under what conditions, a particular strategy may become dominant within a population, it cannot tell us what that strategy may be without the associated prior game analysis. Evolutionary Computation, by comparison, offers a way to develop strategies from scratch and discover which (if any) of these are dominant; provided that a good strategy representation scheme and evaluation method are used.

EVOLUTIONARY COMPUTATION AND PBIL

As is described in earlier chapters, Evolutionary Computation covers a wide range of powerful problem solving tools, or Evolutionary Algorithms (EAs), that have been inspired by nature and make use of the concept of natural selection to improve a population of solutions. Three of these algorithms are considered in this chapter, Genetic Algorithms (Mitchell, 1998),

Genetic Programming (Banzhaf, 1998) and Population Based Incremental Learning (Baluja, 1994; Sebag, 1998).

The last of these, PBIL, is a relatively new statistically approach to EC that combines the concept of a GA with that of reinforcement learning techniques (such as neural networks).

A PBIL algorithm may make use of the same solution representation as a GA however, instead of a population of solutions, PBIL makes use of a probability distribution. The probability distribution represents the likelihood of a solution string's elements (or alleles) of taking on a particular value. Test solutions are generated from this distribution, evaluated and used to reinforce the distribution; good solutions increase the likelihood of their element's values recurring in future and the reverse for bad solutions. Like GA, PBIL may make use of mutation to help increase solution diversity and forms of elitism to focus the search (Gosling 2005). While quite new, PBIL has already proven useful for various types of problem solving (Sukthankar, 1998; Inza, 1999). The basic operation of a PBIL algorithm is shown below:

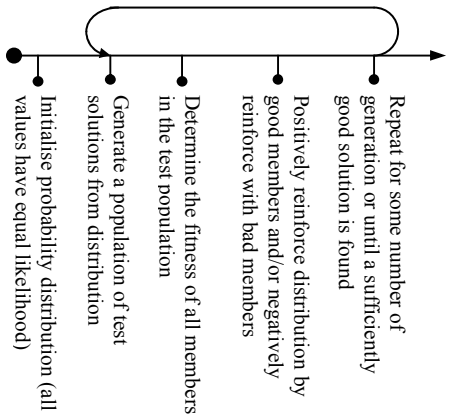


Figure 1: Basic operation of PBIL

While discussing EA's throughout this chapter two key features will recur. First, the solution representation used by the algorithm is critical to the success of the algorithm in tackling the problem; a good representation should reduce the search space as far as possible and limit or remove the possibility of invalid solutions being generated to avoid a combinatorial explosion. Secondly, the evaluation mechanism must successfully distinguish the quality of different solutions but at the same time be computational efficient.

ITERATED PRISONERS DILEMMA

Introduction

Iterated Prisoner's Dilemma is an extension of the well known Prisoner's Dilemma (PD) game. In PD two players, A and B, play two possible moves, cooperate or defect, simultaneously. The combined choices determine each player's score. IPD is PD played over some number of rounds, the scores accumulating. The pay-off table for PD is shown below:

Pay Off (A / B)		Player A: C	Player A: D
Player B	C	3 / 3	5 / 0
Player B	D	0 / 5	1 / 1

Figure 2: Prisoner's Dilemma Pay Off Table

The sole Nash Equilibrium for PD is the play (defect, defect). If the number of rounds is known this is also the best play for IPD. If, however, the number of rounds is unknown an incentive exists for each player to cooperate in order to avoid uncertain future punishments from the other for defecting. How to play under these circumstances is open to debate,

cooperate is desirable as the long term pay-off for the players would be better but the strategy should not be open to exploitation.

EC has previously been used to study IPD strategies. In 1987 Axelrod first did so, running a competition in which strategies were learnt using a GA based system. Axelrod concluded that Tit-for-Tat (TFT) was the dominant strategy for IPD (0). This result sparked some debate with various other experiments and analysis by other researchers, some of whom considered this conclusion incorrect (0; 0; 0; 0). Further EC based work is discussed below in relation to IPD strategy representation.

We now introduce recent experiments conducted into IPD using a similar setup to Axelrod. This work aimed to discover how effective Population Based Incremental Learning (PBIL) was for strategy generation using the more established Genetic Algorithms (GA) for comparison (0; Gosling, 2005).

Representing IPD strategies

When developing a strategy it is important to determine what the players know about the situation, history of play, each others' behavior and how, generally, they should respond to that information (probabilistically or deterministically). The answer to these questions has a profound effect on the nature of the strategies that can be produced, how they can be represented and subsequently, on the type of EAs that may be used to evolve them.

In answering these questions for IPD a considerable degree of variation is possible. For instance Nowak and Sigmund (0; 0) dealt with players that responded probabilistically to a

memory of only the last round of play. The strategy representation scheme consisted of four variables, the chance of defection given each possible pay-off. Evolving strategies using this representation lead to a dominant Pavlov strategy (0) occurring within the population.

Crowley (Crowley, 1996) dealt with varying player memories and IPD strategies based on sets of hierarchical rules. The rules essentially pattern matched different situations within the player memory with more explicit rules having greater precedence. For small rule sets and low memory Crowley found that rules similar to TFT would evolve but in the case of longer memory and large possible rule sets he noted that far more complex strategies emerged. Crowley argued that these more complex strategy hierarchies might provide some indication of how cooperation evolves within nature.

Further examples of the diversity of representations possible when using EAs are Fogel (1993), who evolved finite state machines in an attempt to probe the necessary conditions for cooperation to emerge, and Jang (2004), who determined the effect on behavior of players with no memory that used fixed sequences of moves with varying lengths.

In all of the above cases a variation on GA was used to evolve strategies. This was possible since the types of strategies selected and their representations could be thought of as fixed length strings - an ordered set of variables that required optimisation. Since GAs are designed for the optimisation of such strings they could be applied successfully to these situations. GP, by comparison, makes use of variable size tree representations of a solution and so is unsuitable in this instance.

PBIL, however, can make use of the same representation schemes as a GA which should make it equally applicable.

In recent work we examined the relative effectiveness of PBIL and GA in the context of IPD. For our experiments we selected the classic Axelrod representation. This representation is based upon players responding deterministically to a memory of only the last three rounds of play. Since both players make one move per round and there are only two possible moves (cooperate or defect) a complete player memory comprises 6 elements each of two possible values. This memory can be translated into 6 binary bits with 1 representing defect and 0 cooperate. 6 bits can be arranged in only 64 possible ways so, with consistent play, a player has only 64 possible responses to its memory. We therefore use a 64-bit string to represent the player's strategy, 1 bit for each possible memory configuration. Since at the beginning of the game a player would have no past memory, an additional 6 bits is provided to represent a player's starting memory or pre-disposition. Thus the total strategy representation is 70 bits in length. This is shown below:

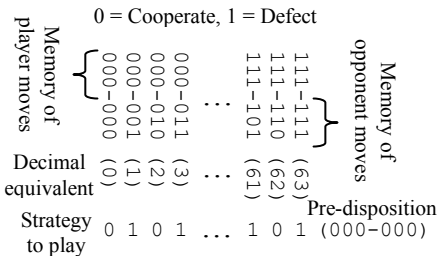


Figure 3: Axelrod IPD representation

GA v PBIL comparison

In order to compare GAs and PBIL two systems were set up and run independently. Both algorithms were used to produce a set of strategies that were then compared in a tournament.

Comparing the GA and PBIL approaches in this way was not straightforward. While the two approaches used the same representation and both used tournaments to evaluate their strategies each operate in very different ways. The PBIL test population can not be directly compared to the GA's population for instance. For the GA, the entire population is evaluated and essentially all (or at least much of) that information used for the ongoing evolutionary process. With a PBIL implementation, only one or two members of the test population are used to update the probability distribution so, for a large population, much of the evaluation information would be lost. Comparing GA and PBIL algorithms based on similar population sizes and number of generation would therefore be unfair.

To achieve a fairer comparison the PBIL system here made use of a relatively large pool of generated test strategies. Each of these test strategies was played against members of a smaller evaluation population. The best scoring member from the evaluation population was then used to reinforce the probability distribution. In this way members of the evaluation population receive high quality evaluations and fewer evaluations are performed per generation. Comparing the GA and PBIL can now be done on the basis of the number of evaluations (IPD games played) used by each. The formula below shows how the populations and generations of each system may be balanced to allow the comparison:

GAevals & PBILevals - Total number of solution evaluations
GApop - The GA population size
GAgens - The number of generations the GA runs for
PBILpop - The population size of the PBIL system
PBILtestpop - The testing pool size of the PBILsystem
PBILgens - The number of generation the PBILsystem runs

$$GA_{evals} = GApop \times (GApop - 1) \times GAgens$$
$$PBIL_{evals} = PBILpop \times PBILtestpop \times PBILgens$$
$$GA_{evals} = PBIL_{evals}$$

In the graphs that follow, the GA and PBIL strategies were compared at time steps equivalent in terms of number of evaluations. The interval between time steps (in PBIL generations) can be found by the following:

$$ComparisonOutputInterval = PBILgens / GAgens$$

Results and Conclusions

While an extensive set of comparisons was made the best PBIL and GA parameter sets used are listed below:

Type	**GA**	**PBIL**
Population Size	100	5
Learning Pool	NA	99
Mutation Rate	0.007	Not Used
Learning Rate	NA	0.025
Generations	300	6000
Data Points	300	300

Table 1: Best GA and PBIL configurations

In order to provide a comparison of relative effectiveness over time, the experiments were repeated 100 times for each set of parameters and the resulting strategies played against one another.

PBIL consistently performed slightly worse against the best GA configuration, although in the early stages it does a little better than the GA.

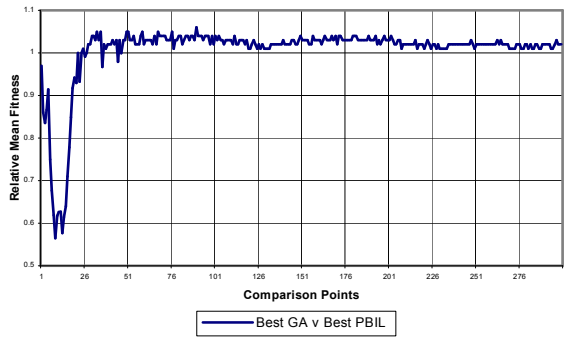


Figure 4: Mean GA strategy fitness / Mean PBIL strategy fitness over time. PBIL starts off well but ultimately does slightly worse than GA. Value greater than one indicates superior GA performance.

Under low population conditions however, a PBIL with an identical number of evaluations does better than GA. Reproducing similar GA conditions to those used by Axelrod, for example, shows PBIL to have superior performance to GA.

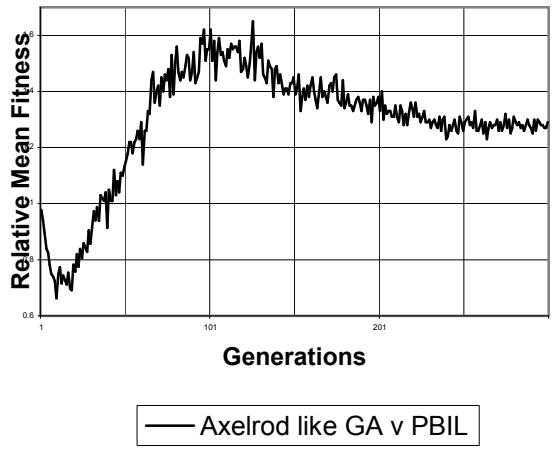


Figure 5: PBIL v GA under Axelrod like conditions. PBIL rapidly performs better than GA. Values greater than one indicate superior PBIL performance.

The reason for this appears to be that the theoretically infinite population of PBIL is able to overcome the shortcomings of a lack of solution diversity within a small GA population.

The conclusion of this work was that GA in general is slightly superior to standard PBIL for strategy generation in this context. PBIL, however, is more effective when only a small population size or a small number of evaluations are possible.

Efforts to improve PBIL further (Gosling, 2005), using a novel mutation operator, have allowed it to compete favorably with GA in this context under all conditions.

The use of EAs in this and other work supports the idea that EC is effective at the generation of game strategies.

In J-P. Rennard (Eds.), Handbook of research on nature inspired computing for economics and management, Chapter 18, 2006 (to appear)

RUBINSTEINS BARGAINING GAME

Introduction

The Rubinstein's Alternating Offer Bargaining Model (RAOBM; 0), is a simple economic complete information game that involves the division of some quantity, a unit pie for instance, between two players. The aim of the game is for the two players (A and B) to come to a mutual agreement about how to divide the pie. The game proceeds in rounds, in each round one player is able to make an offer for how much they should receive and their opponent is able to accept or reject that offer. If the offer is rejected another round occurs with the rejecting player making the next offer. This continues until agreement is reached. To add incentive for the players to agree, each is subject to a discount factor, thus how much they receive is reduced by the effects of them waiting to obtain it. Since this is a complete information game the players each know their and their opponents discount factors. Game theoretic analysis of this game provides a Sub Game Perfect Equilibrium as follows (see 0; 0 for proofs):

$DisA$ = Player A discount factor (makes first offer)
$DisB$ = Player B discount factor (responds first)

$$GetsA = \frac{1 - DisB}{1 - DisA \times DisB}$$

$$GetsB = 1 - GetsA$$

If PlayerA, the first player to make a move, uses the above formula to calculate their offer (*GetsA*) there is no rational reason (from a game theory point of view) for the opponent, PlayerB, to reject it (and obtain *GetsB*).

A considerable body of work exists studying the RAOBM both in its traditional form and under various alternative conditions. When the model is altered such that the players have incomplete information (0; 0; 0; 0) or are boundedly rational or irrational (0; 0) the scope for individual strategies increases dramatically. EC has also been applied to the RAOBM, comparisons being made to the Sub Game Perfect Equilibrium under various conditions (0; 0; 0).

While games such as the RAOBM have been extensively studied and solutions are known for various conditions, other games can prove too complex for traditional analysis. EC is of use in studying such games both to obtain an idea of equilibriums that may exist for the game and to provide a reasonable playing strategy. To have confidence in this idea we now introduce recent work that compared the game theory results for RAOBM with game playing strategies evolved using Genetic Programming (0). The aim of this work was to establish if GP could be used to effectively approximate the games SPE and so provide a case for its use in tackling problems that are too difficult to analyse with traditional game theoretic approaches.

Representing RAOBM strategies

The first step in tackling RAOBM with EC was the same as with the IPD problem above, that is, one of representation. While RAOBM is a complete information game, the aim of approximating the SPE (and so finishing in the first round) leaves players with little knowledge. Assuming a unit pie, the players only know one another's discount factors and who starts first. With this in mind the objective becomes one of finding a representation that can make use of this information to come up with a good first offer from the starting player (PlayerA) that would be accepted (hopefully) by the second player (PlayerB). Essentially

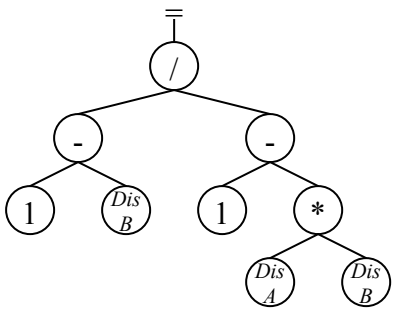


Figure 6: Example of a GP structure. This example shows how the games SPE would be represented.

what we are looking for is a mechanism that allows the evolution of formulae that can tie the discount factors together to generate and offer (and accept/reject threshold).

GA and PBIL are not easily able to do this as they deal with fixed length string representations of problems. Representing solutions is possible, for instance by allowing evolution of the offer/threshold directly, but these may lack generality (as in this case).

Genetic Programming (GP) by contrast is able to evolve variable size tree structures that may represent formulae directly. Instead of defining the meaning of a string the use of GP requires the selection of an appropriate symbol set to use within the tree structures. In general keeping the set of symbols as simple as possible is the best strategy, allowing evolution do the rest. In the case of the RAOBM we used the set of non-terminal symbols [+,-,/,*] and terminals symbols [*ADis*, *BDis*, 1, -1] (*ADis* and *BDis* being the PlayerA and PlayerB discount factors respectively). This symbol set is simple but provides sufficient flexibility for evolution of formulae, like that of the games SPE, to occur, see below:

Because the roles of PlayerA and PlayerB within the RAOBM game are slightly different two separate populations are maintained for PlayerA and PlayerB strategies. To determine the fitness of a tree structure within a given population it is evaluated and used to play games

against all of the structures in the opposing population. The resulting accumulated pay off is used as the structures fitness.

The use of two distinct populations is known as a co-evolutionary approach and is appropriate when competing strategies have to operate under different conditions from one another. In this case PlayerA and PlayerB strategies would evolve such that PlayerA represents the SPE shown above while PlayerB would evolve a correspondingly different structure that would accept the value generated by PlayerA.

Experiments and Results

Using the system of representation and co-evolution described above experiments were run using the following parameters GP:

Parameter	**Value**
Nodes	Non-Terminal (+,-,*,/) Non-Terminal (1,-1,DisA,DisB)
Population Size	100 * 2 (200)
Generations	300
Initial Tree Depth	5
Maximum Nodes	50
Mutation Rate	0.01-0.5
Crossover rate	0-0.1

Table 2: GP Configuration parameters

The experiments were run 100 times each on 10 sets of discount factors; the average score of the best performing individuals from the final generation were used to determine how closely

the population had converged towards the SPE (shown below). As can be seen the SPE was approximated reasonably well in most cases. It may be observe that this approximation began to break down where the discount factors tended towards the extreme.

Sets of discount factors (*ADis*,*BDis*)	**SPE (*GotA*)**	**GP – Experimental Average of Player A**	**Standard Deviation of Player Average**
(0.1, 0.4)	0.625	0.9101	0.0117
(0.4,0.1)	0.9375	0.9991	0.0054
(0.4,0.4)	0.7143	0.8973	0.0247
(0.4,0.6)	0.5263	0.509	0.0096
(0.4,0.9)	0.1563	0.1469	0.1467
(0.5,0.5)	0.6667	0.6745	0.0271
(0.9,0.4)	0.9375	0.9107	0.0106
(0.9,0.6)	0.8696	0.8	0.1419
(0.9,0.9)	0.5263	0.5065	0.1097
(0.9,0.99)	0.0917	0.1474	0.1023

Table 3: ROABM Results, GP approximates the game SPE well

Conclusions

GPs more complex representation scheme can be used effectively to approximate game theoretically derived equilibriums.

The GP derived approximation tends to break down when conditions are extreme.

While GP does not provide an exact match for the theoretically derived SPE for this game, it does provide a reasonable approximation in most cases. This tends to suggest that GP would be useful in studying other problems that require a more complex representation.

THE SIMPLE SUPPLY CHAIN MODEL

Introduction

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 (0; 0; 0).

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. As has been stated, while traditional economic approaches are effective in analysing simple games they fail to tackle the more dynamic problems faced in supply chain situations and as such cannot be made full use of. The application of knowledge and experience to develop strategies is possible but suffers from uncertainty about how robust these strategies would be, especially in unusual circumstances, and how to optimise them for maximum effect.

Making use of evolutionary computation within this domain is reasonable given its application to other economics problems. As we have shown, provided we can define a reasonable strategy representation it should be possible for an EA to evolve an effective and robust solution.

To begin tackling the supply chain problem it is first necessary to model the supply chains we are interested in more precisely. A system such as the Simple Supply Chain Model (SSCM) provides one such way and we will introduce this shortly.

Having accomplished this, the next task, as discussed earlier, is to develop a system of representation for possible solutions and a framework within which that representation may be used and evaluated. We also need to consider what sort of evolutionary algorithm would be appropriate for the learning process and how it should be applied.

It should be noted that considerable effort has gone into using EC and other techniques for negotiation and bargaining with computers. The Trading Agent Competition (Wellman, 2000) for example partially inspired the SSCM. Some examples of work in negotiation are Sandholm (2002), Fatima (2000) and Bartolini (2005), while Fatima (2005b) provides a comparison of evolutionary and game-theoretic approaches to bargaining.

The SSCM

The Simple Supply Chain Model (0; Gosling, 2003b) has been developed to allow the specification of a simple supply chain starting state. To this end it models three different types of participant in the supply chain, Customers, Suppliers and Middlemen.

Customers have requirements that they wish to be fulfilled. These requirements are for a set of goods at some maximum price within a certain time frame. Customers require the use of a Middleman to obtain these sets of goods and so have knowledge of some set of Middlemen and a maximum outbound communication capability.

Suppliers are able to supply goods at some minimum price and some maximum quantity over the course of the scenario being modeled. They sell via the Middlemen and so have a known set of Middlemen along with a maximum outbound communications allocation.

Middlemen are responsible for matching up sets of requirements to available products in an attempt to make a profit. They are defined purely in terms of their known Suppliers, Customers and a maximum outbound communications capacity. These are the focus of study here.

The SSCM defines supply chains in terms of these different participants, the set of products, the amount of time available for deals to be struck and the communication scheme used by the participants to interact. The SSCM does not impose restrictions on the way in which participants may attempt to resolve the chain only the way in which the chain is initially set up and the means by which communication can occur.

Representation and Evaluation

Determining a representation scheme and evaluation mechanism for SSCM strategies is challenging simply because the number of possibilities are large. To reduce the scope

somewhat, first define different SSCM scenarios that restrict further the conditions the participants may face under a given SSCM instantiation and secondly concern ourselves primarily with the Middleman strategy and assume simple strategies on the part of the Customers and Suppliers. These restrictions may be relaxed later.

In the simplest scenario we assert that there is one Supplier per product and that Suppliers are passive and have no knowledge of the Middlemen to begin with and that they will not initiate contact with a Middleman once known. Middlemen have no prior knowledge of Customers but know of each of the Suppliers they may need to fulfill a Customer's requirements. Customers know only of one Middleman each and initiate contact sometime prior to their earliest cut off point for obtaining goods. These restrictions simplify the Middleman strategy both by removing the need to mitigate the effects of Customers attempting to find deals elsewhere and reducing the choice of Suppliers. In more complex scenarios these restrictions have been relaxed.

With this first scenario as a starting point it is possible to begin defining a strategy representation.

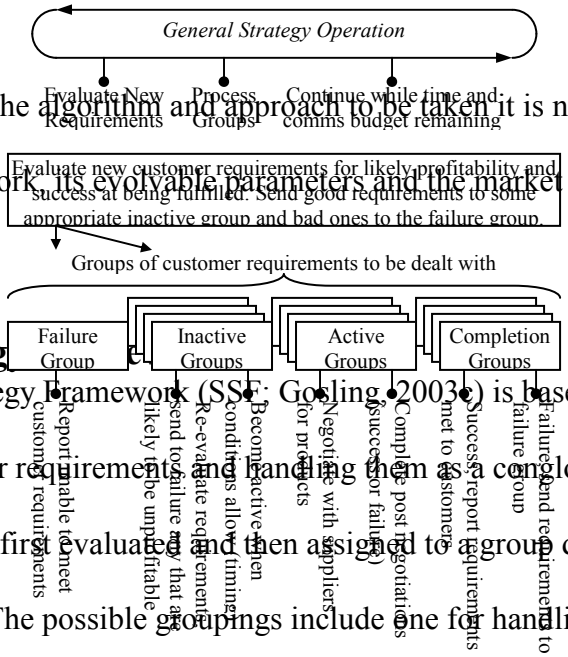
Initially we consider how the evaluation of any resultant strategy should be undertaken. The problem maps well into a multi-agent environment and so it makes sense to build a market simulation system within which the participants can be configured in line with the SSCM and use their strategies to attempt to resolve the chain. When the chains' run time is up the effectiveness of each strategy can be assessed. Since this process is likely to be

computationally expensive and/or time consuming it is reasonable to assume the total number of participants within the system will be limited. For this reason it would not be possible to evaluate many strategies simultaneously. From the discussion of IPD strategy generation above this would suggest that PBIL would be superior to GA under these conditions since it is effective at leveraging small test populations for learning.

A second consideration is the complexity of the strategies to be used. The initial reaction is that GP would provide the flexibility required to define a complex SSCM strategy and this would certainly be the case. The problem with this approach however comes in two parts. Firstly defining a symbol set of sufficient subtlety and complexity to capture the various aspects of a participant's role is difficult. Secondly having defined such a set, ensuring that viable strategies result is problematic; while evolution is powerful the representation must provide some guidance for it to stand a good chance of success. In each case it seems reasonable to provide a basic strategy framework within which the algorithm can evolve the control aspects of the strategy. This removes the problem of wholly invalid strategies being developed and helps reduce the complexity of the symbol set. The downside of this is that multiple elements within the framework would need to be evolved simultaneously and the complexity of how to combine these multiple elements would additionally complicate the use of a GP algorithm. To simplify the strategy problem further the framework can be extended with reasonable control elements the parameter of which may then be evolved by an algorithm. If this approach is taken far enough it is possible to remove the need for GP altogether and evolved the parameters directly. This is what we have done here building on Matos's (1998) bargaining work in particular for the agent negotiation elements. With the

strategy representation reduced to a fixed length string of parameters it is possible to use PBIL or a GA. As stated, a GA would have difficulties under the limited population size available (as indeed would have GP) so we elect to use PBIL in this instance.

Having selected the algorithm and approach to be taken it is necessary to outline the SSCM Strategy Framework, its evolvable parameters and the market simulation system with which it will be used.



SSCM Strategy Framework

The SSCM Strategy Framework (SSF; Gosling, 2003e) is based around the idea of grouping together customer requirements and handling them as a conglomerate. Incoming customer requirements are first evaluated and then assigned to a group depending on some set of characteristics. The possible groupings include one for handling requirements the system has deemed impossible to fulfill or unprofitable. Requirement groups start in an inactive state (in which requirements are continually re-evaluated), progress to becoming active (during which supplier negotiations are undertaken) and finally move to a completion state for reporting back to the customers. The basic outline of this process is shown below (Figure 7).

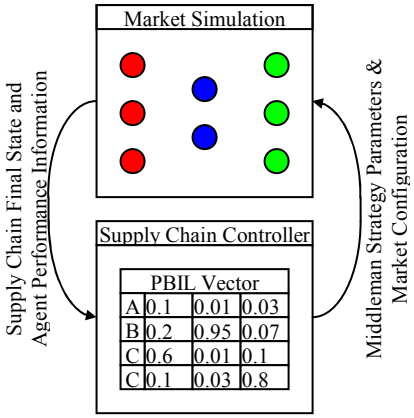
Primary parameters within the SSF are those relating to the evaluation of customer requirements, the dispersal of requirements to groups and the negotiation mechanism used with the supplier. For example, the negotiation process, based on work by Matos (0), requires a set of fourteen parameters for each product type under consideration. These parameters control estimates for likely values of products, tactics used for negotiation and importance weighting for those tactics. Other parameters include control for how quickly groups should become active and what requirements should be accepted.

Market Simulation System

The SSCM Market Simulation System (SMSS) provides an environment within which the SSF may be used and under which the parameters are evolved. The SMSS consists of two

Figure 7: SSCM Strategy Basic Framework

core components, an agent based supply chain simulator and a market controller. The market controller maintains a PBIL vector that provides strategy configuration parameters to the supply chain agents. Further, the controller sets-up the supply chain and evaluates the performance of agents once completed. This information can be used to reinforce the PBIL



vector for future generations of supply chain players. This process is briefly outlined below (Figure 8):

The controller is able to configure the supply chain in such a way as to provide different environment in which to evolve strategies. Examples of different environments are ones where the available goods are scarce or customer budgets are very limited.

The result of the SMSS is the controller's final PBIL vector state – this should contain an effective strategy for the environment presented.

Results and Conclusions

The main focus of experimentation with the SMSS was to determine if strategies could be evolved within the environment presented and what the limits of adaptability were.

Figure 8: Market Simulation System Operation

It was found that effective strategies emerged within the SMSS and as expected, that substitution of those strategies in to new environments leads them to adapt to the new conditions suggesting no universal strategy is optimal across all conditions.

Having determined that strategies could evolve within the SMSS we then probed the limits of the system by adjusting the environment in such a way that it became difficult for Middlemen to make a profit. This was accomplished by increasing the stubbornness of Suppliers negotiating over prices. These efforts lead to the determination of an adaptation boundary for this parameter beyond which the system was unable to evolve effective strategies. Further

work suggested that using pre-evolved strategies close to that boundary condition would allow for adaptation under the harsher conditions.

While these results have proven interesting the question of how to analyse them further has proven to be one of considerable importance, visualization has certainly helped but obtaining definitive evidence of why the strategies have adapted to the environment in a certain way has proven more difficult. To this end, analysis of the results ideally requires the development of further analytical tools and this is currently the focus of much effort.

Overall EC has proven effective for evolving strategies in the complex, dynamic environment offered by the SSCM and the SSF and SMSS have proven an effective way of harnessing the power of PBIL to this end.

CONCLUSIONS

This chapter has introduced Evolutionary Computation in the context of two well known games (IPD and RAOBM) and the more complex SSCM. For these games we have shown that EC is able to evolve effective strategies that equate to the known equilibriums. For the SSCM we have shown that with careful consideration it is possible to evolve successful strategies within a strategy framework and supply chain simulation system. Since Game Theory cannot effectively deal with the uncertainties inherent in situations like the SSCM we assert that EC, used appropriately, provides a good alternative for this problem and other complex real-life problems.

On a cautionary note, while EC is effective for strategy generation, care must be taken with the design of a good representation, the selection of an appropriate algorithm and the choice of a reasonable evaluation scheme. A further consideration is that of analysis. As the disagreement over stable IPD strategies demonstrates, results may still be open to interpretation. In the context of the SSCM reaching a full understanding of the results is an issue.

Finally, Evolutionary Computation has many advantages for the generation or optimisation of strategies in challenging environments, this approach has had a successful beginning but its future depends on carefully considered application.

BIBLIOGRAPHY

Ausubel, L.M., Crampton, P., & Deneckere, R.J. (2002). Bargaining with Incomplete Information. Handbook of Game Theory, Vol 3, Amsterdam: Elsevier Science B.V., chapter 50.

Axelrod, R. (1987). The Evolution of Strategies in the Iterated Prisoner's Dilemma. Genetic algorithms and simulated annealing, Research notes in AI. 32-41. Pitman/Morgan Kaufmann.

Banzhaf, W., Nordin, P., Keller R., & Kaufmann M. (1998). Genetic Programming An Introduction. Morgan Kaufmann.

Bartolini, C., Preist, C., & Jennings, N.R. (2005). A software framework for automated negotiation. Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications. Springer Verlag. 213-235.

Binmore, K.. (1994). Game Theory and the Social Contract I, playing fair. MIT Press.

Binmore, K. (1998). Game Theory and the Social Contract II, just playing. MIT Press.

Binmore, K., Piccione, M., & Samuelson, L. (1998). Evolutionary stability in alternating-offer bargaining games. Journal of Economic Theory. 80, 257-291.

Baluja, S. (1994). Population Based Incremental Learning – A Method for Integrating Genetic Search Based Function Optimisation and Competitive Learning. Tech. Rep. No. CMU-CS-94-163. Pittsburgh, PA: Carnegie Mellon University.

Crowley, P.H. (1996). Evolving cooperation: strategies as hierarchies of rules. BioSystems. 37, 67-80.

Faratin, P., Sierra, C., & Jennings, N.R. (2000). Using similarity criteria to make negotiation trade-offs. Proc. 4th Int. Conf. on Multi-Agent Systems (ICMAS-2000), Boston, USA. 119-126.

Fatima, S.S., Wooldridge M., & Jennings, N.R. (2001) Optimal Negotiation Strategies for Agents with Incomplete Information. Proc. 8th Int. Workshop on Agent Theories, Architectures and Languages (ATAL). Seattle USA. 53-68.

Fatima, S.S., Wooldridge, M., & Jennings, N.R. (2003). Comparing Equilibria for Game-Theoretic and Evolutionary Bargaining Models. AAMAS 2003, Workshop on Agent-Mediated Electronic Commerce V.

Fatima, S.S., Wooldridge M., & Jennings, N.R. (2005). Bargaining with Incomplete Information. Annals of Mathematics and Artificial Intelligence. To appear.

Fatima S.S., Wooldrige M., & Jennings N.R. (2005b). A Comparative Study of Game Theoretic and Evolutionary Model Of Bargaining for Software Agents. Artificial Intelligence Review 23 (2).

Fogel, D.B. (1993). Evolving behaviors in the iterated prisoner's dilemma. Evolutionary Computation, 1(1), 77-97.

Gosling, T. (2003). The Simple Supply Chain Model and Evolutionary Computation. Proceedings of the Congress on Evolutionary Computation 2003 (CEC2003).

Gosling, T & Tsang E. (2003b). Technical Report 1: The Simple Supply Chain Model (SSCM). Technical Report CSM-392, Department of Computer Science, University of Essex.

Gosling, T (2003c). Technical Report 3: The Scenario One Strategies. Technical Report CSM-394, Department of Computer Science, University of Essex.

Gosling, T., Jin, N., & Tsang, E. (2004). Population Based Incremental Learning Versus Genetic Algorithms: Iterated Prisoners Dilemma. Technical Report CSM-401, Department of Computer Science, University of Essex.

Gosling, T., Jin, N., & Tsang, E. (2005). Population Based Incremental Learning with Guided Mutation Versus Genetic Algorithms: Iterated Prisoners Dilemma. Proceedings of the Congress on Evolutionary Computation 2005 (CEC2005).

He, M., Jennings, N.R., & Leung, H. (2003). On Agent-Mediated Electronic Commerce. IEEE Trans on Knowledge and Data Engineering. 15(4), 985-1003.

Inza, I., Merino, M., Larra~naga, P., Quiroga, J., Sierra, B., & Girala, M. (1999). Feature Subset Selection by Population-Based Incremental Learning. A case study in the survival of cirrhotic patients treated with TIPS. Technical Report no. EHU-KZAA-IK-1/99, University of the Basque Country, Spain.

Jang, D., Whigham, P.A., & Dick, G. (2004). On evolving fixed pattern strategies for Iterated Prisoner's Dilemma. CRPIT '26: Proceedings of the 27th conference on Australasian computer science, 241-247.

Jin, N., & Tsang, E. (2005). Co-evolutionary Strategies for an Alternating-Offer Bargaining Problem. CIG2005.

Kraines, D., & Kraines, V. (1993). Learning to Cooperate with Pavlov – an adaptive strategy for the Iterated Prisoner's Dilemma with Noise. Theory and Decision 35, 107-150.

Kreps, D. (1990). Game Theory and Economic Modelling, Oxford University Press. 123-128.

Linster, B. (1992). Evolutionary Stability in the Repeated Prisoners' Dilemma Played by Two-State Moore Machines, Southern Economic Journal. 880-903.

Nowak, M.A., & Sigmund, K. (1992). Tit-For-Tat in a Heterogeneous Population. Nature, vol. 355, no. 6357,pp 250-253.

Nowak, M.A., & Sigmund, K. (1993). A Strategy of Win-Stay, Lose-Shift That Outperforms Tit-For-Tat in the Prisoner's Dilemma Game, Nature, vol. 364, no. 6432, pp 56-58.

Matos, N., Sierra, C., & Jennings, N.R. (1998). Determining Successful Negotiation Strategies: An Evolutionary Approach. Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS-98).

Mitchell, M. (1998). An Introduction To Genetic Algorithms. MIT Press.

Muthoo, A. (1999). Bargaining Theory with Applications. Cambridge University Press.

Myerson, R. (1991). Game Theory: Analysis of Conflict. Cambridge, MA: Havard University Press. 399-403.

Rubinstein, A. (1982). Perfect Equilibrium in a Bargaining Game. Econometric. Vol 50.(1), 97-110.

Rubinstein, A. (1985). A Bargaining Model with Incomplete Information About Time Preferences. Econometric. Vol. 53(5), 1151-1172.

Sandholm, T. (1999). Distributed Rational Decision Making. In: Weiss, G. (ed.): Multiagent Systems A Modern Approach to Distributed Artificial Intelligence. MIT press. 201-

Sandholm, T, & Vulkan, N. (2002). Bargaining with Deadlines. Early version in Proceedings of the National Conference on Artificial Intelligence (AAAI). pp. 44-51. 1999.

Sebag, M., & Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. Proceedings of the 5th Conference on Parallel Problems Solving from Nature. 418-427.

Sukthankar, R., Baluja, S., & Hancock, J. (1998). Multiple Adaptive Agents for Tactical Driving, Applied Intelligence.

Walsh, W.E. (2001). Market Protocols for Decentralized Supply Chain Formation. Doctoral Thesis submitted to the University Of Michigan.

Wellman, M.P., Greenwald, A., Stone, P. & Wurman, P.R. (2000). The 2001 Trading Agent Competition. Fourteenth Conference on Innovative Applications of Artificial Intelligence.