Coevolving Market Strategies for CAT

Rahul Iyer and Joseph Reisinger

December 8, 2006

Abstract

We perform a preliminary exploration of the market mechanism strategy space in CAT, focusing specifically on the charging policy. In addition to hand-designing several strategies, we also employ coevolutionary policy search to automatically generate novel charging policies. Coevolution is a powerful method for facilitating open-ended search and has been shown to generate robust solutions to complex problems. Furthermore, it can be extended to provide monotonic progress guarantees, allowing for a natural synthesis of guided and open-ended search. Although we are unable to demonstrate evolved strategies that outperform the best hand-designed strategies for a given parameter setting, coevolution has still been useful as a tool for identifying which hand-designed strategies might perform well in a particular setting. This paper summarizes our approach and points out several areas of future work that should be explored to fully characterize the application of coevolution to CAT. Ultimately we believe that by bootstrapping from simpler hand-designed strategies, coevolution can be leveraged to find strategies that perform well in a large variety of environments.

1 Introduction

The *Trading Agent Competition* (http://www.sics.se/tac) was introduced to promote research at the intersection of Artificial Intelligence (AI) and microeconomics by providing complex benchmark environments for autonomous agents to compete. Past competitions were directed towards the development and analysis of agent strategies that trade amongst themselves within a fixed market. This year a new competition, TAC: Market Design (CAT), will be introduced with the goal of generating novel market mechanisms themselves.

Mechanism design is a branch of economic game theory with the goal of implementing game rules for maximizing some solution concept given a set of agents with private preferences [7]. Mechanism design has received the most attention in auction literature, for example most modern financial and trading firms use a continuous double auction mechanism to facilitate trading, but it has also been applied to several computational problems in distributed computing. One interesting recent direction involves designing *adaptive* mechanisms that

implement solution concepts robustly in many different scenarios [9]. Indeed the CAT competition is designed explicitly to further research into adaptive mechanism design.

Market competition is generally believed to remove unwanted firms that are not profit maximizers, resulting in equilibrium. For example, firms employ a variety of pricing rules, but only those rules that provide a good approximation to profit maximization survive. Likewise, natural selection, occurring in the process of evolution results in animal behavior that is well adapted to the environment [15]. In the simplest case this environment is fixed, while in other cases the environment is itself composed of other individuals who are subject to the same forces of selection. What is optimal for any firm/animal in this setting is to make its decision depending on the distribution of the behaviors in the population with which it interacts. Coevolutionary theory allows us analyze this process of evolutionary selection in such an interactive environment.

Coevolution has been successfully applied to both the development of bidding agent strategies and the design of auction markets themselves [10]. In this work we examine the application of coevolution to the design of adaptive market mechanisms for facilitating commodity trading in CAT. Specifically, we attempt to address several important questions:

- Does coevolution help elucidate which strategies may be optimal given the market dynamics?
- Can coevolution be combined with simpler strategies in order to bootstrap learning?
- Can evolved strategies beat simple dominating strategies in the long run?
- Can the application of artificial coevolution lead to *novel* strategies that are both complex and robust?

To address these questions, we combine NeuroEvolution of Augmenting Topologies (NEAT), a powerful policy search reinforcement learning algorithm, with several robust hand-designed market strategies in order to coevolve effective market charging rules. NEAT represents solutions as neural networks: hierarchical combinations of sigmoid functions which are capable of approximating any function. Using coevolution in this principled manner, we posit that the space of market strategies can be explored more thoroughly than is possible when designing them by hand.

This paper is organized as follows. Section 2 provides a brief overview of the NEAT framework, how monotonic progress can be ensured in coevolution and how such an algorithm can be applied to CAT. Section 3 provides results comparing the evolved market with hand-designed strategies and gives learning curves from the evolutionary process. Section 4 discusses some implications of the work, section 5 summarizes areas for future work and section 6 concludes.

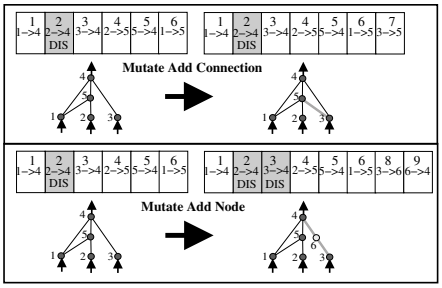

Figure 1: **NEAT genetic encoding and mutation operators.** Neural network topologies are directly encoded using a variable-length representation and undergo topological complexification through the two structural mutation operators presented here.

2 Coevolutionary Policy Search in CAT

Neural network strategies for setting fees are coevolved using the NEAT algorithm and tested against a variety of hand-designed strategies. This section is divided into 4 parts: Section 2.1 describes the NEAT framework and section 2.2 describes the MaxSolve algorithm for ensuring monotonic progress in coevolution. Section 2.3 describes the specific setup of the CAT market and also explains how coevolution is performed in the system. Finally, section 2.4 describes the hand-designed fixed strategies tested.

2.1 NeuroEvolution of Augmenting Topologies

NeuroEvolution of Augmenting Topologies (NEAT) [12] is a policy-search reinforcement learning method that uses a genetic algorithm to find optimal neural network policies. NEAT automatically evolves network topology to fit the complexity of the problem while simultaneously optimizing network weights. NEAT employs three key ideas: 1) incremental complexification using a variable-length genome, 2) protecting innovation through speciation, and 3) keeping dimensionality small by starting with minimally connected networks. By starting with simple networks and expanding the search space only when beneficial, NEAT is able to find significantly more complex controllers than other fixed-topology learning algorithms. This approach is highly effective: NEAT outperforms other NE methods on control tasks like double pole balancing [12] and robotic strategy-learning [13]. These properties make NEAT an attractive method for evolving neural networks in complex tasks.

Each genome in NEAT includes a list of *connection genes*, each of which refers to two *node genes* being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an *innovation number*, which allows finding corresponding genes during crossover (figure 1). Innovation numbers are inherited and allow NEAT to perform crossover without the need for expensive

topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of matching different topologies [11] is essentially avoided. NEAT speciates the population so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population. The reproduction mechanism for NEAT is *explicit fitness sharing* [6], where organisms in the same species must share the fitness of their niche, preventing any one species from taking over the population.

The principled complexification exhibited by NEAT is a desirable property in competitive coevolution: As the antagonistic populations refine their strategies and counter-strategies, complexification becomes necessary in order to generate novel strategies without "forgetting" past strategies [13]. The next section describes how monotonic progress can be guaranteed in coevolutionary NEAT.

2.2 Monotonic Progress in Coevolution

To ensure that the evolved specialists work well against a range of opponents, the opponent strategies themselves are evolved simultaneously through coevolution. In coevolution, an individual's fitness is evaluated against some combination of opponents drawn from the evolving populations, rather than against a fixed fitness metric. This approach yields several major benefits over traditional evolution: 1) Coevolution allows the opponent strategies to be learned by the algorithm, reducing the amount of information the algorithm designer must provide *a priori*, 2) Under certain conditions, coevolution may facilitate *arms races*, where individuals in both populations strategically complexify in order to learn more robust behaviors [14], 3) Coevolution may reduce the total number of evaluations necessary to learn such robust strategies, leading to more efficient search [2].

In order to facilitate arms races and make coevolution efficient, the algorithm needs to ensure monotonic progress. Without such a guarantee, as evolution progresses populations can "forget" past strategies, resulting in cycling behavior [2, 4]. Before monotonic progress guarantees can be achieved, however, it is first necessary to define the desired *solution concept*. In game theory, a solution concept is defined as "any rule for specifying predictions as to how players might be expected to behave in any given game" [8]. In artificial coevolution, since the dynamics of interactions between organisms can be controlled using the fitness function, any desired solution concept can be implemented by simply manipulating the structure of the fitness payoffs. Algorithms implementing monotonic progress towards several such solution concepts have been proposed: The Pareto-Optimal Equivalence Set (IPCA) [3], Nash Equilibria [5], and Maximization of Expected Utility (MaxSolve) [2].

For the CAT domain, we employ a simplified variant of MaxSolve, a solution concept for maximizing the expected utility of each individual. Such a solution concept is useful in games where the space of opponent strategies cannot be fully enumerated and thus generalizations regarding the utility of a strategy must be

drawn from a limited set of experiences. Formally, for a set of candidate solution strategies $\mathbf{C}$, a set of test strategies $\mathbf{T}$ and a game $\Gamma = (A_i, u_i)_{i \in I}$, the set of strategies satisfying the maximization of expected utility solution concept can be defined as

$$S1 = \{C \in \mathbf{C} | \forall_{C' \in \mathbf{C}} : E(u_C(C, T)) \geq E(u_{C'}(C', T))\}$$

for some $T \in \mathbf{T}$. Algorithmically, this solution concept can be implemented simply by maximizing the sum of an individual's utilities across all tests. Although this formulation assumes that all tests are weighted equally, it has been shown to perform well in practice [2].

2.3 Coevolving Adaptive Charging Policies in CAT

In addition to the actual auction mechanism, CAT specialist markets must implement rules governing four basic policies, charging, shout accepting, clearing and pricing.

- *Charging*: Each market charges the traders for executing various actions in the market. The fees charged are for registration, information, shout placement, transaction execution and a profit share. These fees are announced at the beginning of each game day and determine the profit made by a market during the game.
- *Shout Accepting*: A shout is a bid placed by a bidder or an ask placed by a seller. A market has to decide on a rule that determines if a shout placed by a trader can be placed in the order books of the market.
- *Clearing*: The market has to determine a period in the course of the day to clear the order books. Clearing is the process of matching as many possible unmatched shouts and executing the transactions for the matched shouts. In the standard implementation the market clears whenever a bid is higher than ask. In general, the market must clear at least once in a day since the order books are reset at the end of the day.
- *Pricing*: When a bid and an ask are matched, the market has to determine the price for the commodity that is being traded. This price must be between the bid value and the ask value.

For this paper, we focus only on finding charging policies for our market and assume the following fixed strategies for the other policies:

- *Pricing*: A k-pricing policy is used with $k = 0.5$. Thus the price will always be halfway between a matched bid and ask
- *Shout Accepting*: By default the CAT game assumes NYSE rules that state that a shout can be accepted only if it improves the state of the order books. New bids must be higher than all uncleared bids and for new asks must be less than the current asks.

- *Clearing*: Whenever a new shout is placed the market is always cleared.

For the charging policy, neural network controllers are evolved for setting the fees each day. Each network takes as input the average and standard deviation of each fee value (information, shout, profit, transaction and registration) set by the opponents in the previous day. Thus the architecture is fundamentally reactive, although networks are capable of evolving recurrent connections, which allow for a form of memory. Networks have five outputs corresponding to each fee type; the final setting is scaled between zero and the maximum for that fee. We decided to focus solely on the charging policy because it has the most significant impact on the specialist profit. However, once robust charging policies are found, the next logical step would be to implement more intelligent pricing, clearing and accepting policies.

Evolved charging strategies are evaluated in matches with six specialists run for 50 days. Only the last 15 days are counted towards the actual score. 100 trading agents are used, one half using the GD strategy and one half using random constrained. Five of the specialists in each match are evolved and one is a fixed strategy to ensure a minimum level of performance.

Specialist fitness within a single bout is calculated as

$$f(o) = p_o / \max(1, (p_{high} - p_o)),$$

where p_o is the final profit of the organism and p_{high} is the final profit of the maximum scoring organism. Fitness is calculated as the sum of the scores obtained during the MaxSolve evaluations and during N normal evaluations,

$$F(o) = \sum_{d \in \mathbf{T}} f_d(o) + \sum_{i=0}^{N} f_{X_i}(o),$$

where o is the organism being evaluated and X_i is a random variable mapping i to some combination of opponent strategies from the current population. In all reported experiments, each organism plays 10 matches against randomly chosen opponents from the current population ($N = 10$) and two matches against each champion in the archive. The test archive is initially empty and new tests are added each generation if the current generation champion's fitness exceeds the previous best fitness. This evaluation strategy focuses coevolutionary search on individuals capable of performing well against all previous champion strategies.

2.4 Fixed Charging Strategies

In order to ensure a high safety level for the coevolved strategies, six different basic strategies were implemented as fixed opponents:

- *Fixed High*: A fixed charging policy that sets fees at the maximum level
- *Linear*: A simple linear strategy that increases the fees linearly with respect to the number of days elapsed.

- *Exponential*: A dynamic charging policy which ramps fees up exponentially each day to a fixed maximum. A single parameter is used to control the rate of increase.

- *Logistic Growth*: A dynamic charging policy where a logistic function is used to model the growth of fees. Logistic functions can be defined as

$$N(t) = K/(1 + e^{-\alpha t - \beta}),$$

where K is the carrying capacity, the permissible value for the function, α is the growth rate of the function, and β is the placement of the growth along the x axis. For our experiments we use

$$\alpha = \frac{ln81}{\Delta t},\ \beta = -t_m \alpha.$$

These settings convert the general α-parameter to one that determines the period in which the fee increases from the 10% level to the 90% level and the β parameter to one that determines the mid-point of the curve i.e. the point at which the function reaches half of the maximum value. Hence the Logistic function used can be expressed as

$$N(t) = K/(1 + e^{-\frac{ln(81)}{\Delta t}(t - t_m)}),$$

where K is the maximum fee level, Δt is the time taken to grow from the 10% fee value to the 90% fee value and t_m is the midpoint value, the point at which the fee level is half of K.

- *Trader Sensitive*: Fees are increased when the number of trader registration increases. The percentage of traders registered to the market is divided in to two discrete regions: low and high. On the basis of the level of traders registered in the current day, the market fee level for the next day is chosen from low, medium and high.

See figure 2 for a qualitative comparison of the growth of the various non-adaptive strategies.

3 Results

Two separate sets of results were obtained, one set for preliminary competition and one set for the final competition. Although these two competitions only differed in terms of the maximum fee values for registration and information, the two environments admit very different sets of optimal strategies. In particular, a fixed high charging strategy outperforms the exponential strategy under the initial settings, but the exponential and other ramping strategies beat fixed high in the final settings. In both cases, the distribution of traders participating in the markets had little impact on the market dynamics.

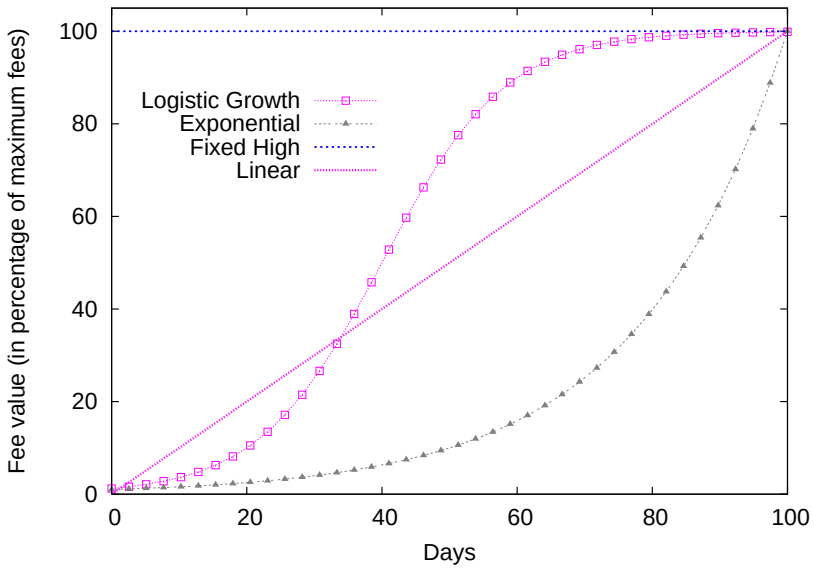

Figure 2: Fee growth schedules for the hand-coded strategies.

3.1 Basic Strategies

To evaluate the logistic growth strategy, we compared it with the exponential function which was the dominant strategy in the new evolutionary runs. The results indicate that the logistic function with the right parameters performs significantly better than the exponential strategy and the fixed high strategy.

Three different parameter settings of the logistic growth strategy were played against the Fixed High and the Exponential strategy.

- LOG-HIGH: Midpoint(t_m) = 0.7, Growth period(Δt) = 0.1
- LOG-MID: Midpoint(t_m) = 0.5, Growth period(Δt) =0.4
- LOG-LOW: Midpoint(t_m) =0.3, Growth period(Δt) =0.7

Figures 3 and 4 show the average trader and profit distribution over 30 games for the initial and final settings respectively . Each game has a game length of 100 days with the profit recorded for a random sample of days (30 to 50 days) chosen from the second half of the game.

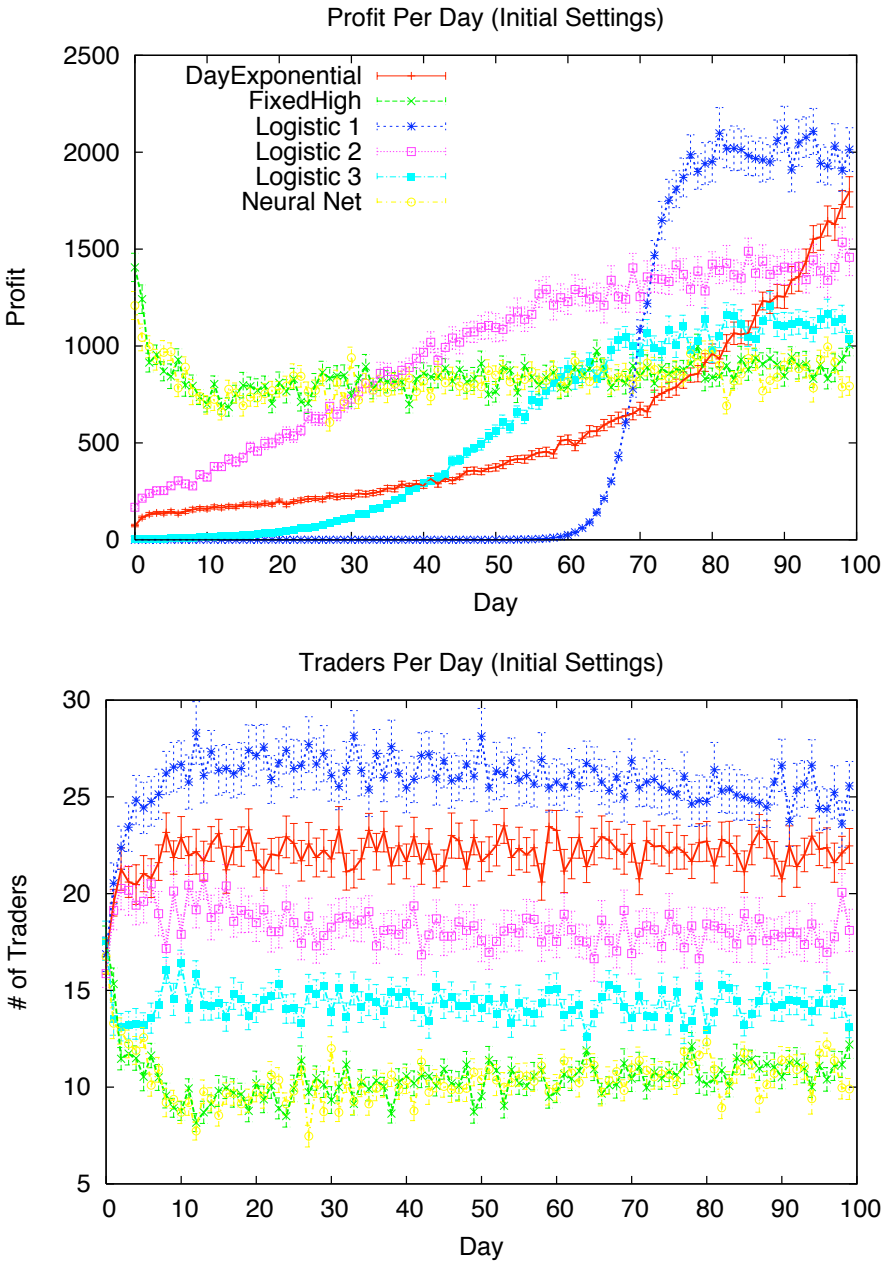

Figure 3: Average profit and trader distribution for basic strategies for the initial settings. Logistic 1 corresponds to LOG-HIGH, Logistic 2 corresponds to LOG-LOW and Logistic 3 corresponds to LOG-MID.

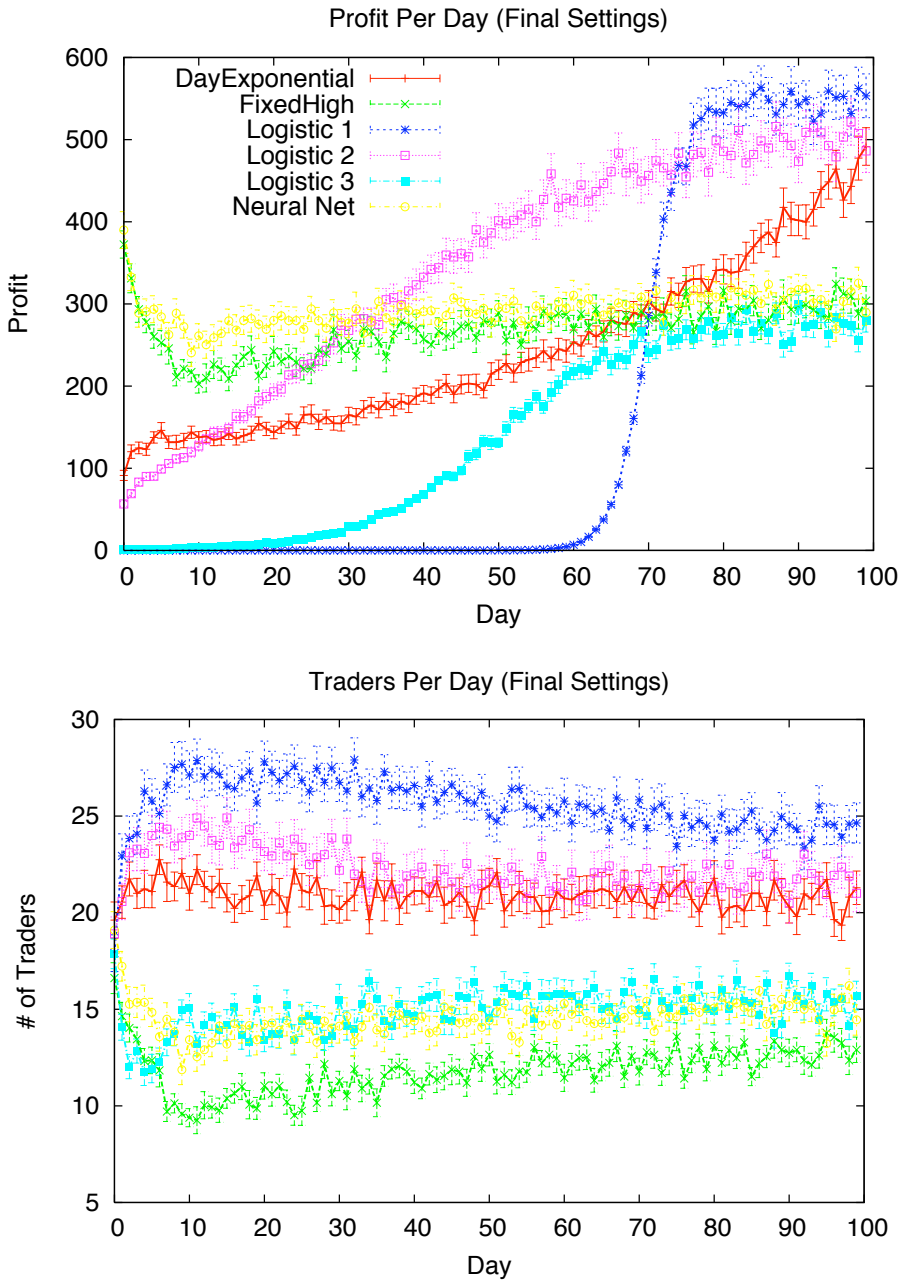

Figure 4: Average profit and trader distribution for basic strategies for the final settings. Logistic 1 corresponds to LOG-HIGH, Logistic 2 corresponds to LOG-LOW and Logistic 3 corresponds to LOG-MID

Strategy	Average Profit (Initial)	Average Profit (Final)
LOG-LOW	1393.68	343.02
LOG-MID	1031.70	254.830
LOG-HIGH	1358.12	388.12
Exponential	930.40	333.74
Fixed High	883.71	263.46

Table 1: Average profit for basic strategies over 30 games. The profit was recorded for a random period of 30 to 50 days

Figure 3 demonstrates how the Neural Net developed using Fixed High as the dominant strategy follows the fixed high curve very closely in terms of traders and profit. However the logistic markets maintain higher profit by ensuring a high number of traders. Similarly in Figure 4 we can see that LOG-HIGH maintains a high number of traders and hence ends up with a sizeable profit.

It should be noted that in both the cases, the logistic function markets are the winners only because the second half of the game is used for profit calculation. If the entire game is used for profit comparison then the fixed high strategy performs better. The average profits made per day by each strategy are given in Table 1.

Pairwise t-tests were used to test the significance of these results. A summary of the results is shown below. Here the probability p is the probability of the significance result being incorrect due to noise.

Initial Settings:

1. All Logistic functions significantly outperform Fixed High ($p < 4.6 \times 10^{-5}$)
2. LOG-HIGH and LOG-LOW significantly outperforms Exponential ($p < 4 \times 10^{-6}$)
3. LOG-HIGH significantly outperforms LOG-MID ($p < 2 \times 10^{-5}$)
4. LOG-MID does not show significant improvement over Exponential ($p > 0.1$)
5. Exponential and Fixed High do not have any significant difference in performance ($p > 0.3$)

Final Settings:

1. LOG-LOW and LOG-HIGH significantly outperforms Fixed High ($p < 4.6 \times 10^{-5}$)
2. LOG-HIGH significantly outperforms Exponential ($p < 0.018$)
3. LOG-HIGH significantly outperforms LOG-MID ($p < 1 \times 10^{-6}$)
4. LOG-MID and LOG-LOW do not show significant improvement over Exponential ($p > 0.8$)
5. Exponential significantly outperforms Fixed High ($p < 0.00065$)

Strategy	Average Profit (Initial)	Average Profit (Final)
LOG-HIGH	x	449.06
LOG-LOW	x	358.62
Exponential	967.40	324.87
Fixed High	1263.27	297.23
NN 14	x	181.87
NN 42	x	267.55
NN 104	1279.66	x

Table 2: Performance of evolved strategies vs. fixed strategies. An 'x' indicates that the strategy was not run under that set of conditions. NN104 is the champion of the initial coevolutionary run and NN42 is the champion from the final coevolutionary run.

3.2 Evolved Strategies

Learning curves for the two evolutionary runs (initial and final settings) are given in figure 5. Both runs exhibit a high amount of variance in champion scores, indicating that more trials should be run to improve confidence. Also, both runs exhibit a positive slope in average fitness, indicating that neither run converged.

Table 3 compares the champions of the two coevolutionary runs to the best performing basic strategies. Under the initial settings, the best neural network found (NN104) does not make a profit significantly different from the fixed high strategy ($p = 0.81$). However, the neural network is able to significantly outperform the exponential ramping strategy ($p < 10^{-3}$). Under the final parameter settings, the exponential strategy performs significantly better than the best neural network found (NN42; $p < 0.012$). Furthermore, the LOG-HIGH strategy significantly outperforms the exponential ($p < 10^{-5}$). Finally, it is important to note that in the second run, the neural network champion from generation 42 (NN42) significantly outperforms the champion from generation 14 (NN14; $p < 10^{-6}$), indicating that evolution is indeed improving upon past solutions. Plots depicting specialist profits and number of traders per day under the final settings are given in figure 6.

4 Discussion

Comparing the basic strategies it can be seen that the Exponential strategy outperforms the Fixed High strategy for the final settings but is unable to beat it significantly in the initial settings. However the logistic strategy outperforms the exponential and the fixed high strategies for both the settings of fee level. Logistic maintains a low fee value initially and then ramps up its fee level after attracting a high number of traders. This strategy generates high profit because traders do the majority of their exploration in the first 20 days and are thus mostly exploiting in the latter half of the game and cannot learn quickly enough

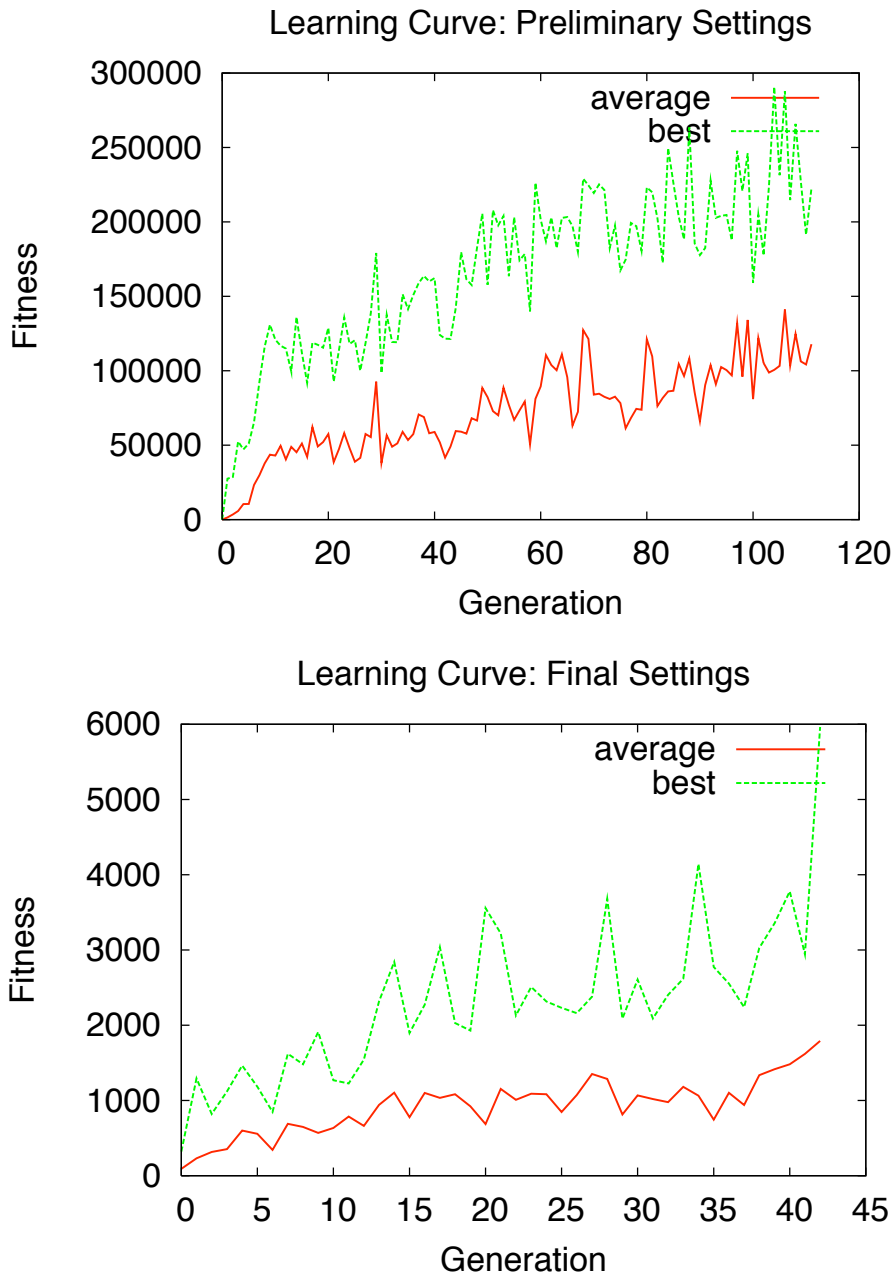

Figure 5: Learning curve for coevolution with the initial and final settings. In the initial case, evolved agents quickly learn to mimic the fixed high strategy, but cannot outperform it. In the final case, evolved markets are unable to beat the exponential charging strategy.

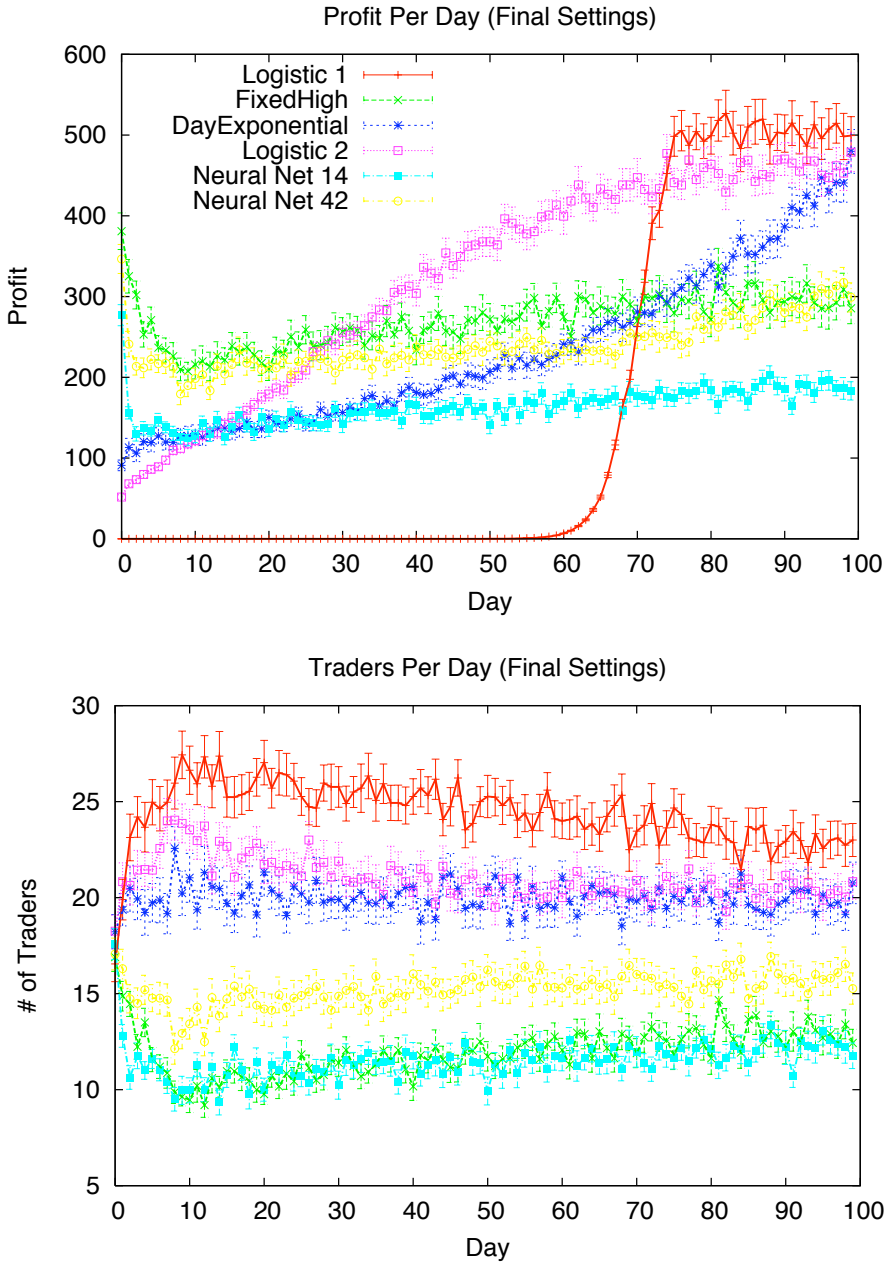

Figure 6: With the final parameter setting, the evolved neural network strategies perform consistently as well as fixed-high, but cannot outperform the exponential or logistic strategies.

about the sudden increase in the fees. The ideal rate of growth and the placement of the growth across the game depends on the game parameters used. The LOG-HIGH strategy has a late growth of fees but with a high rate of growth, a form of trigger strategy that jumps from 0 fees to the maximum fees with-in a span of a day. It is not completely obvious if this is the best strategy for any given setting. We have also not found any significant results that distinguish the performance of the LOG-LOW and LOG-MID function from each other. This demonstrates that the exact parameters that lead to consistent success are difficult to find and indeed may not even exist for a sufficient large subset of environments. Such difficulties indicate that coevolution may indeed be a practical approach to finding the most robust set of parameters.

Coevolution is able to approximate the fixed-high strategy under both parameter settings but is unable (at least in 40 generations) to significantly outperform the exponential ramp strategy. However the second coevolutionary run shows some promise as the champion from generation 42 significantly outperforms the champion from generation 14. Thus, if evolution were run for longer, then it may learn more powerful strategies.

The evolved neural net strategies exhibit few signs of adaptive behavior in the profit graph. In general the strategy employed by the neural network is to charged at a fixed high rate, regardless of the fees of the other competitors. Always charging high is a rather effective strategy that is easy for evolution to find and thus it tends to dominate early evolution. One reason such a strategy might be preferred may be due to the bias generated by the inputs. Other inputs, such as the current day, may help the neural network find more complex adaptive strategies more easily. Without such inputs, in order to find strategies such as the exponential ramp, the neural network must evolve recurrent connections on each output, which is highly improbable at least in the first few hundred generations.

Finally, the general lack of robustness and statistical significance in evaluations (profit is recorded during a fixed portion of the day, networks only get 10 random evaluations) may have washed out the search gradient towards more complex solutions in noise. The fixed high strategy is easy to find regardless of the amount of noise, but finding more complex strategies would require more sensitivity to small changes in fitness, which is not possible with the number of evaluations used. In the future, more evaluations should be performed per network, although doing so would increase the learning time.

5 Future Work

Based on the results obtained in this study, several immediate improvements to the search algorithm suggest themselves:

- One problem with the current coevolutionary setup is that each strategy only plays on average two matches against the champion strategies. Combined with the fixed weighting of champions as fitness objectives, this approach does not give clear indication of which champions are strictly

better than othes. In order to maintain more accurate scores for each champion, but still keep the total number of evaluations low, each champion score should be associated with a confidence level. The confidence level would increase with each subsequent evaluation and could be then used to assign a relative weight to each champion as an objective in the final fitness function.

- Currently the neural networks are given as input only the first and second moments of the distribution of fee values. What other kinds of inputs might be useful? Possible candidates include higher moments, the current game day, the game parameters used and deltas on the fee values. Also for games with a fixed number of opponents it is possible to give each opponent value for each fee as a separate input, instead of computing the average and variance.

- Finally, the homogeneous environments with fixed game lengths are exploitable by simple *trigger* strategies which charge nothing until the final 15 days of the game and then charge maximum. Although no instances of such an exploitative strategy were seen during evolution, in general, running longer games with a random game length and more varied agent environments should help improve the robustness of the evolved strategies.

One possible next step for this research would be to evolve the parameters of a fixed ramping strategy based on the logistic curve. The ramping strategy seems to be powerful for interesting parts of the competition space, but for a given range of CAT settings different parameterizations of the exponential would be optimal. Coevolution would be able to find the optimal parameterizations that are robust across many different opponent settings. Furthermore, once the optimal simple charging strategy is found, it could be used to guide open-ended coevolution for more complex charging functions. Such an approach highlights an important strength of the coevolutionary method: Evolution can be run on top of existing strategies. In other words, if a market strategy is developed that performs well, then that strategy can be added to the mix of fixed opponents that the evolutionary agent must learn to overcome.

In the longer term, it would be interesting to explore other coevolutionary solution concepts to see what impact they have on the market strategies found. In addition to the MaxSolve algorithm, several coevolutionary solution concepts have been defined, including the Nash Memory mechanism [5] and the Pareto-Optimal Equivalence Set [3]. Also in the CAT domain there may be room for collusive strategies that rely on one or more specialists dominating the market to the exclusion of the others. Using a solution concept based on the Shapely Value, it may be possible to evolve such behavior.

Although the charging policy has the most direct impact on a market's profits, there are several other aspects of the mechanism that might be adapted, e.g. the trade clearing policy and the pricing policy. Automated markets catering to automated traders may be able to manipulate these policies to extract more profit from the traders. One possible approach would be to tailor pricing and

clearing on a per-agent basis, extracting more surplus from more successful traders and offering incentives to less successful ones. It would not have been possible to implement these strategies in the beta version of the competition; however, with the latest version providing more flexibility, it would be interesting to see if coevolution can make an impact at these aspects of the game.

It has been shown that Genetic Algorithm can be used to evolve a market mechanism more efficient than human-designed markets [1]. It would be interesting to apply that research in order to evolve market rules for the CAT competition. A hybrid between a one-sided and a two-sided market, more efficient than any human designed market, could be developed through evolution and can be the trading portal for the future.

6 Conclusion

This work serves as an exploratory analysis of the feasibility of applying coevolution to explore simple CAT strategies. Experimental results demonstrated that coevolution is able approximate strong basic strategies in CAT with high maximum fees but is unable to consistently beat more complex strategies in the low maximum fees environment within 40 generations. Poor performance of evolved controllers is due to three factors: 1) high variance in fitness scores, 2) short evolutionary runs with small populations and 3) the lack of inputs indicating the current game day (such an input would allow the neural network to learn strategies that are not purely reactive). The significantly higher earnings of the logistic growth charging policy indicate that one possible research direction is to coevolve more constrained charging functions. These results are a promising start for developing coevolutionary mechanisms for exploration of simple CAT strategies.

References

[1] D. Cliff. Evolution of market mechanism through a continuous space of auction-types. Technical Report HPL-2001-326, HP Labs, 2001.

[2] E. de Jong. The maxsolve algorithm for coevolution. In *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 483–489, New York, NY, USA, 2005. ACM Press.

[3] E. D. de Jong. The incremental pareto-coevolution archive. In *GECCO '04: Proceedings of the 2004 Conference on Genetic and Evolutionary Computation*, pages 525–536. Springer, 2004.

[4] S. G. Ficici. Monotonic solution concepts in coevolution. In *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 499–506, New York, NY, USA, 2005. ACM Press.

[5] S. G. Ficici and J. B. Pollack. A game-theoretic memory mechanism for coevolution. In *GECCO '03: Proceedings of the 2003 Conference on Genetic and Evolutionary Computation*, pages 286–297, 2003.

[6] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154. San Francisco: Kaufmann, 1987.

[7] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, US, 1995.

[8] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, MA, 1991.

[9] D. Pardoe, P. Stone, M. Saar-Tsechansky, and K. Tomak. Adaptive mechanism design: a metalearning approach. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 92–102, New York, NY, USA, 2006. ACM Press.

[10] S. Phelps, P. McBurney, S. Parsons, and E. Sklar. Co-evolutionary auction mechanism design: A preliminary report. In *AAMAS '02: Revised Papers from the Workshop on Agent Mediated Electronic Commerce on Agent-Mediated Electronic Commerce IV, Designing Mechanisms and Systems*, pages 123–142, London, UK, 2002. Springer-Verlag.

[11] N. J. Radcliffe. Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90, 1993.

[12] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[13] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[14] L. Van Valin. A new evolutionary law. *Evolution Theory*, 1:1–30, 1973.

[15] J. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1995.