A Model of Evolution and Learning

Vladimir G. Red'ko
Institute of Optical Neural Technologies, Russian Academy of Science
Vavilova Str., 44/2, Moscow, 119333, Russia.
redko@iont.ru

Oleg P. Mosalov
Moscow Institute of Physics and Technologies
Institutsky per., 9, Dolgoprudny, Moscow region, 141700, Russia.
olegmos_@mail.ru

Danil V. Prokhorov
Research and Advanced Engineering, Ford Motor Company
2101 Village Rd., MD 2036, Dearborn, MI 48124, U.S.A.
dprokhor@ford.com

Abstract – **We study a model of evolving populations of self-learning agents and analyze the interaction between learning and evolution. We consider an agent-broker that predicts stock price changes and uses its predictions for selecting actions. Each agent is equipped with a neural network adaptive critic design for behavioral adaptation. We discuss three cases in which either evolution or learning, or both, are active in our model. We show that the Baldwin effect can be observed in our model, viz., originally acquired adaptive policy of best agent-brokers becomes inherited over the course of the evolution. We also compare the behavioral tactics of our agents to the searching behavior of simple animals.**

I. Introduction

In (Red'ko et al, 2005) we proposed a simple model of evolving population of autonomous adaptive agents*. We demonstrated that policies learned by successful agent-brokers become inherited over the course of Darwinian evolution. This phenomenon is called the Baldwin effect (Baldwin, 1896, Weber and Depew, 2003). In this paper we carry out a more detailed analysis of the interaction between learning and evolution in populations of adaptive agents.

The model of behavior used in this paper and (Red'ko et al, 2005) is based on a neural network adaptive critic design (ACD) (Werbos, 1992, Prokhorov and Wunsch, 1997), which includes two neural networks (NNs): a model NN and a critic NN. The model NN predicts the state of the environment for the next time step, and the critic NN provides action selection on the basis of the model predictions. These NNs can be optimized by both learning and evolution. In other words, each agent is autonomous, and it behaves according to its own learned/evolved ACD control algorithm (distributed, rather than centralized, control of all agents).

In comparison with other investigations of the interaction between learning and evolution (Hinton and Nowlan, 1987, Ackley and Littman, 1992, Belew and Mitchell, 1996, Suzuki and Arita, 2004), our study concentrates on self-learning agents. Though our work is similar to that of (Ackley and Littman, 1992), the control system of our agents is more advanced. First, it utilizes the ACD architecture, with the well-investigated temporal difference algorithm (Sutton and A. Barto, 1998) as its learning method. Second, our agent control system includes the model NN, thereby allowing the agent to predict the future state of the environment and to use its predictions for action selection. In general, having a model for explicit predictions of environment states can add new capabilities to control systems of adaptive agents. As the ACD architecture could be functionally similar to animal control system (see, e.g., Werbos, 1992), it is interesting to compare the behavior of our agents with that of simple animals.

This paper consists of 10 sections. In Section II we define the agent-broker and its task. The ACD learning algorithm is described in Section III. This is followed in Section IV by a description of evolution via mutations the offsprings of the best agent's genome. We describe our simulations and results in Sections V-IX. Section V describes general features of agent adaptation. Section VI characterizes specifics of pure learning (learning without evolution). Section VII discusses interaction between learning and evolution. Peculiarities of model NN predictions are discussed in Section VIII. Section IX compares the agent policy with the behavior of simple animals, and Section X concludes the paper.

* An abbreviated version of some portions of this article appeared in Red'ko et al. (2005), as part of the IJCNN 2005 conference proceedings, published under the IEEE copyright.

II. Agent Task

Inspired by (Prokhorov et al, 2001), we implement the adaptive agent-broker which predicts future changes of the stock price and tries to increase its wealth by buying and selling stocks. The agent has its resource distributed into cash and stocks. The sum of these is the net capital of the agent $C(t)$. The agent decision is the variable $u(t)$, which is the fraction of the agent's capital that is currently invested in stocks. The environment is determined by the time series $X(t)$, $t = 1,2,\ldots$, where $X(t)$ is the stock price at the moment t. The goal of the agent is to increase its capital $C(t)$ by changing the value $u(t)$. The capital dynamics is described by

$$C(t+1) = C(t)\ \{1 + u(t+1)\ \Delta X(t+1) / X(t)\} \times$$
$$[1 - J\,|u(t+1) - u(t)|], \qquad (1)$$

where $\Delta X(t+1) = X(t+1) - X(t)$ is the current change of the stock price, and J is a parameter that takes into account expenses of the agent when buying or selling stocks. The factor in the braces corresponds to the change of the capital as the result of stock price changes. The factor in the square brackets reflects the transaction cost. Following (Moody et al, 1998), we use the logarithmic scale for the agent resource, i.e., $R(t) = \log C(t)$. The current agent reward $r(t)$ is defined by the expression: $r(t) = R(t+1) - R(t)$:

$$r(t) = \log\ \{1 + u(t+1)\ \Delta X(t+1) / X(t)\} +$$
$$\log\,[1 - J\,|u(t+1) - u(t)|]. \qquad (2)$$

For simplicity, and unlike (Prokhorov et al, 2001), we assume that the variable $u(t)$ takes only two values, $u(t) = 0$ (all in cash) or $u(t) = 1$ (all in stock).

III. Agent Learning

The agent control system is a simplified ACD. Our adaptive critic scheme consists of two NNs: a model and a critic (see Figure 1). The goal of the adaptive critic is to stochastically maximize the utility function $U(t)$ (Sutton and Barto, 1998):

$$U(t) = \sum_{j=0}^{\infty} \gamma^j r(t+j), \qquad t = 1,2,\ldots, \qquad (3)$$

where $r(t)$ is an instantaneous reward obtained by the agent and γ is the discount factor ($0 < \gamma < 1$). Making the realistic assumption $|\Delta X(t+1)| << X(t)$, we specify that the ACD state $\mathbf{S}(t)$ at moment t depends on two values, $\Delta X(t)$ and $u(t)$: $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$. (It is also possible to use $\mathbf{S}(t) = \{X(t), \Delta X(t), u(t)\}$.)

The role of the model is to predict changes of the stock time series. The model output $\Delta X^{pr}(t+1)$ is based on m previous values of ΔX: $\Delta X(t-m+1), \ldots, \Delta X(t)$, which are used as the model inputs. The model is implemented as a multilayer perceptron (MLP) with one hidden layer of tanh nodes and linear output, and it is trained by the usual method of backpropagation with the gradient descent.

The critic is intended to estimate the state value function $V(\mathbf{S})$ (estimate of U in (3)) for the current state $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$, the next state $\mathbf{S}(t+1) = \{\Delta X(t+1), u(t+1)\}$, and its predictions $\mathbf{S}^{\mathbf{pr}}{}_u(t+1) = \{\Delta X^{pr}(t+1), u\}$ for two possible actions, $u = 0$ or $u = 1$. The critic is also an MLP of the same structure as the model, but it is trained by the temporal difference method (Sutton and Barto, 1998).

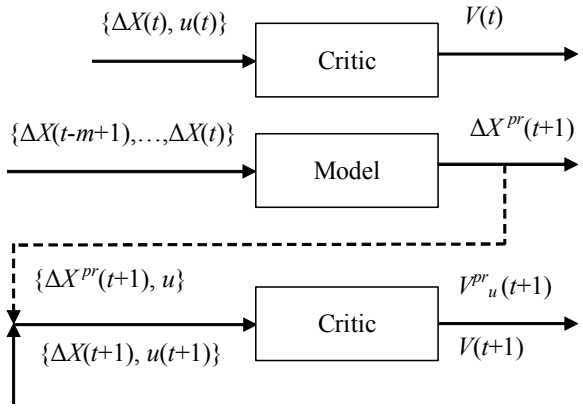

Fig. 1. Our ACD. The model predicts changes of the time series. The critic (the same NN is shown in two consecutive moments) forms the state value function for the current state $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$, the next state $\mathbf{S}(t+1) = \{\Delta X(t+1), u(t+1)\}$, and its predictions $\mathbf{S}^{\mathbf{pr}}{}_u(t+1) = \{\Delta X^{pr}(t+1), u\}$ for two possible actions, $u = 0$ or $u = 1$.

At any moment t, the following operations are performed:

1) The model predicts the next change of the time series $\Delta X(t+1)$.

2) The critic estimates the state value function for the current state $V(t) = V(\mathbf{S}(t))$ and the predicted states for both possible actions $V^{pr}{}_u(t+1) = V(\mathbf{S}^{\mathbf{pr}}{}_u(t+1))$, where $\mathbf{S}^{\mathbf{pr}}{}_u(t+1) = \{\Delta X^{pr}(t+1), u\}$, and $u = 0$ or $u = 1$.

3) The ε-greedy rule is applied (Sutton and Barto, 1998): the action corresponding to the maximum value $V^{pr}{}_u(t+1)$ is selected with probability $1 - \varepsilon$, and an alternative action is selected with probability ε ($0 < \varepsilon << 1$). (For example, if the action "buy" corresponds to the maximum of $V^{pr}{}_u(t+1)$, then it is selected with probability $1 - \varepsilon$; alternatively, the action "sell" is selected with probability ε.)

4) The selected action is carried out. The transition to the next time moment $t+1$ occurs. The current reward $r(t)$ is

calculated in accordance with (2) and received by ACD. The value $\Delta X(t+1)$ is observed and compared with its prediction $\Delta X^{pr}(t+1)$. The NN weights of the model are adjusted to minimize the prediction error using the error backpropagation and the gradient descent with $\alpha_M > 0$ as the model learning rate.

5) The critic computes $V(t+1)$. The temporal-difference error is calculated (in its simplest form):

$$\delta(t) = r(t) + \gamma V(t+1) - V(t) . \quad (4)$$

Alternatively, critic predictions $V^{pr}(t)$ and $V^{pr}(t+1)$ can be used in (4).

6) The weights of the critic neural network are adjusted to minimize the temporal-difference error (4) using its backpropagation and the gradient descent with $\alpha_C > 0$ as the critic learning rate.

IV. Evolutionary Algorithm

An evolving population consists of n agents. Each agent has a resource $R(t)$ that changes in accordance with values of agent rewards: $R(t+1) = R(t) + r(t)$, where $r(t)$ is calculated in (2).

Evolution passes through a number of generations, $n_g = 1, 2, \ldots, N_g$. The duration of each generation n_g is T time steps (agent lifetime). At the beginning of any generation, initial resource of each agent is zero, i.e., $R(T(n_g-1)+1) = 0$.

The initial synaptic weights of both NNs (model and critic) form the agent genome $\mathbf{G}=\{\mathbf{W}_{M0},\mathbf{W}_{C0}\}$. The genome $\mathbf{G}$ does not change during the agent life, and it is fixed when the agent is born. However, synaptic weights of the NNs $\mathbf{W}_M$ and $\mathbf{W}_C$ are changed during agent life via learning described in Section III.

At the end of each generation, the agent having the maximum resource $R_{max}(n_g)$ is determined (the best agent of the generation n_g). This best agent gives birth to n children that constitute a new (n_g+1)-th generation. The children genomes $\mathbf{G}$ differ from their parent genome by small mutations taken from a normal distribution. Cross-over is not used in this model.

At the beginning of every new (n_g+1)-th generation, we set for each agent $G_i(n_g+1) = G_{best,\,i}(n_g) + \text{rand}_i$, $\mathbf{W}_0(n_g+1) = \mathbf{G}(n_g+1)$, where $\mathbf{G}_{\mathbf{best}}(n_g)$ is selected from the best agent of the previous n_g-th generation, and rand_i is $N(0, P^2_{mut})$, i.e., a normally distributed random number with zero mean and standard deviation P_{mut} (mutation intensity), which is added to each synaptic weight.

Thus, the genome $\mathbf{G}$ changes only via evolution, whereas the synaptic weights $\mathbf{W}$ are adjusted only via learning.

V. Main Results of Simulations

In our computer simulations, we deal with two examples of the time series:

1) sinusoid:

$$X(t) = 0.5(1 + \sin(2\pi t/20)) + 1 , \quad (5)$$

2) stochastic time series from (Prokhorov et al, 2001, Example 2):

$$X(t) = \exp(p(t)/1200),$$
$$p(t) = p(t-1) + \beta(t-1) + k\,\lambda(t),$$
$$\beta(t) = \alpha\beta(t-1) + \mu(t), \quad (6)$$

where $\lambda(t)$ and $\mu(t)$ are two random normal processes with the zero mean and the unit variance ($N(0,1)$), and where $\alpha = 0.9$, $k = 0.3$.

Several parameters are set to the same values for all simulations: discount factor $\gamma = 0.9$, number of inputs of the model NN $m = 10$, number of hidden neurons of the model and critic $N_{hM} = N_{hC} = 10$, learning rate of the model and critic $\alpha_M = \alpha_C = 0.01$, parameter of the ε-greedy rule $\varepsilon = 0.05$, mutation intensity $P_{mut} = 0.1$. Other parameters (generation duration T and population size n) are set to different values, depending on the simulation, as specified below.

We analyze the following cases:

- Case L (pure learning); in this case we consider a single agent that learns by means of temporal difference method, as detailed in Section III;
- Case E (pure evolution), i.e., evolving population without learning;
- Case LE, i.e., evolution combined with learning, as described in Section IV.

We compare the agent resource values attained during 200 time steps for these three cases of adaptation. For the cases E and LE, we set $T = 200$ (T is generation duration) and record the maximal value of agent resource in a population $R_{max}(n_g)$ at the end of each generation. For the case L (pure learning), we have just one agent whose resource is reset $R(T(n_g-1)+1) = 0$ every $T = 200$ time steps for consistency with the cases E and LE. In case L the index n_g gets incremented by one after the passing of every T time steps, and $R_{max}(n_g) = R(T\,n_g)$.

The plots $R_{max}(n_g)$ vs. n_g for the sinusoid (5) are shown in Figure 2. The plot of $R_{max}(n_g)$ for the case L beyond $n_g = 500$ is shown in Figure 2a.

In order to exclude the decrease of the value $R_{max}(n_g)$ due to the random choice of actions when applying the ε-greedy rule for the cases LE and L, we set $\varepsilon = 0$ after $n_g =$

100 for the case LE and after $n_g = 2000$ for the case L (note the increase of $R_{max}(n_g)$ after $n_g = 100$ and $n_g = 2000$ for the corresponding cases in Figures 2 and 2a; setting $\varepsilon = 0$ earlier is also possible with the same effect.). The results are averaged over 100 simulations with $n = 10$, $T = 200$.

Figure 2 demonstrates that both learning combined with evolution and pure evolution ensure approximately the same value of final resource $R_{max}(500) = 6.5$. However, the combination of evolution with learning results in agents which attain larger values of R_{max} faster, especially for larger T; this is due to the synergy of evolution and learning in searching for the best agent policy. Interestingly, Ackley and Littman (1992) did not observe as clear an advantage of the case LE over the case E as demonstrated here.

We analyzed the agent policy and found that the optimal policy corresponds to the following agent behavior. The agent buys stocks when it predicts that the stock price will rise, and it sells stocks when it predicts a declining stock price. Although this strategy sounds obvious, it is made feasible by the relative dependability of the agent's predictions, and by the zero transaction costs J (Simulations with non-zero transaction costs J demonstrate the consistent tendency to less frequent switching between two possible actions, as compared with the case of $J = 0$.) With this policy, the agent attains $R_{max}(500) = 6.5$ which appears to be the asymptotic value, as shown in Figure 2 for the specified parameters of our simulation. Analysis of agent behavior demonstrates that both pure evolution and evolution combined with learning are able to find the optimal policy.

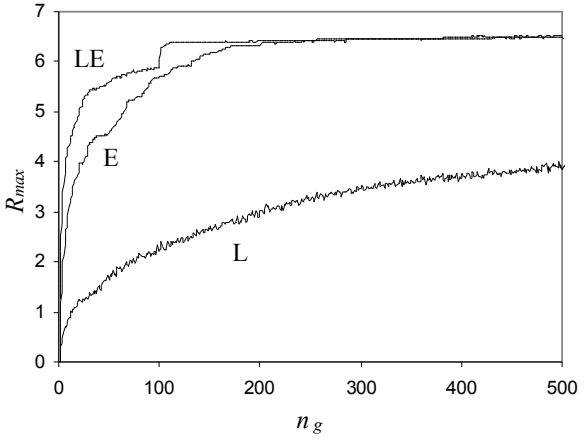

Fig. 2. The plots of $R_{max}(n_g)$ for the sinusoid (5). The curve LE corresponds to the case of evolution combined with learning, as detailed in Section IV. The curve E corresponds to the case of pure evolution (without learning). The curve L corresponds to pure learning. Each point of the plots represents the average over 100 simulations, each starting with a different random seed; $n = 10$, $T = 200$. At $t = 100$ we set $\varepsilon = 0$ in the ε-greedy rule (the case LE).

Figure 3 shows the plot $R_{max}(n_g)$, along with the standard deviation $\sigma(n_g)$ of $R_{max}(n_g)$, for the case LE. The values $\sigma(n_g)$ characterize a distribution of $R_{max}(n_g)$ over different random seeds. Figure 3 demonstrates that the increase of $R_{max}(n_g)$ is accompanied by the increased variance. The curve $\sigma(n_g)$ peaks in the region of rapid increase of $R_{max}(n_g)$. This feature is quite general, as curves $\sigma(n_g)$ have similar maximum locations for all three cases L, E and LE.

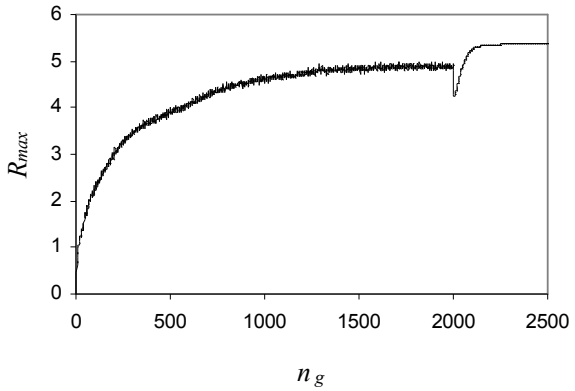

Fig. 2a. The plot of $R_{max}(n_g)$ for pure learning. The same plot (curve L) as in Figure 2, but for longer n_g. At $t = 2000$ we set $\varepsilon = 0$ in the ε-greedy rule.

It is noteworthy that this phenomenon is similar to the stage of generalization in classical conditioning experiments on animals. There is an active random search at moments of maximal increase of conditioned reaction (Kotlyar & Shulgovsky, 1979). In classical conditioning, the dependence of conditioned reaction on the number of experiments is similar to the curve $R_{max}(n_g)$ in Figure 3. The active random search for an adequate reaction is similar to the value $\sigma(n_g)$ maximization in the region of rapidly increasing $R_{max}(n_g)$.

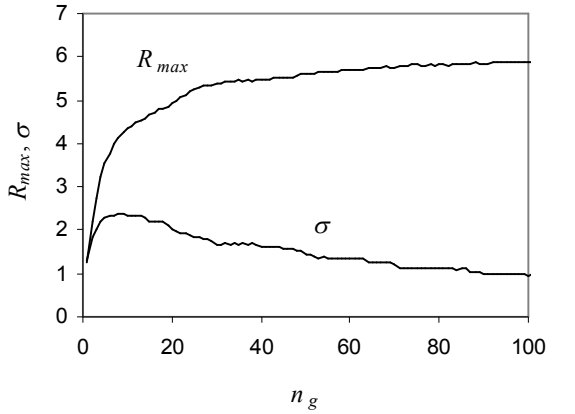

Fig. 3. The maximum resource in the population $R_{max}(n_g)$ attained by the end of each generation and the standard deviation $\sigma(n_g)$ of $R_{max}(n_g)$ vs. the generation number n_g for the evolution combined with learning on the sinusoid (5). Each point of the plots of $R_{max}(n_g)$ represents the average over 100 simulations, each starting with a different random seed. The values $\sigma(n_g)$ characterize a distribution of $R_{max}(n_g)$ over different random seeds; $n = 10$, $T = 200$.

VI. Learning without Evolution

Figures 2 and 2a demonstrate that the simple form of learning implemented here is imperfect, as it can find only a suboptimal policy even if the training continues for many generations. The asymptotic value of R_{max} for the sinusoid is only 5.4, which is considerably smaller than the asymptotic value R_{max}= 6.5 corresponding to the optimal policy (in the limit of large n_g). Our examination of action sequences selected by the learning agent reveals that the pure learning is able to find only the following satisfactory policy. The agent buys stock when stock price rises (or falls by a small amount) and sells stocks when stock price falls significantly (Figures 4 and 5).

Thus, in the case of pure learning, the trained agent appears to prefer to keep the capital in stock.

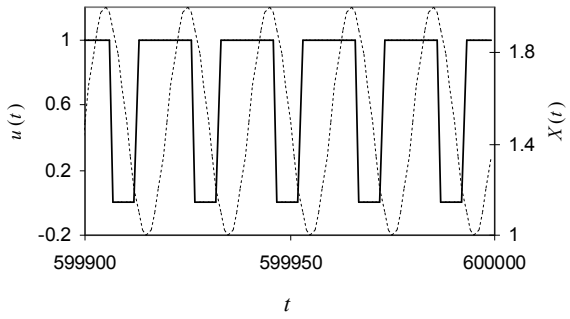

Fig. 4. The agent policy selected by the learning agent (solid line). Actions are characterized by values $u(t)$: $u = 0$ (all in cash) and $u = 1$ (all in stock). This is the case of pure learning on the sinusoid $X(t)$ (dotted line).

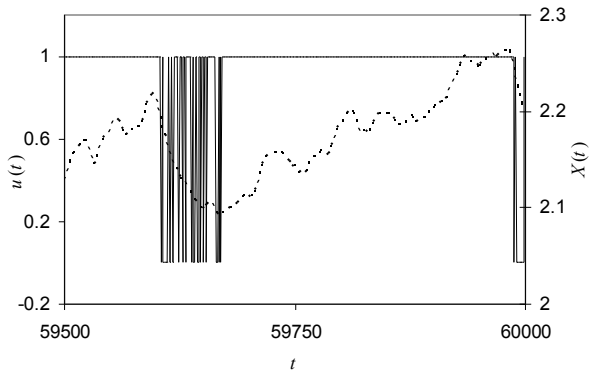

Fig. 5. The agent policy selected by the learning agent (solid line). Actions are characterized by values $u(t)$: $u = 0$ (all in cash) and $u = 1$ (all in stock). This is the case of pure learning on the stochastic time series $X(t)$ (dotted line). Suboptimality of the agent policy is evident as the agent either prefers to keep its capital in stock or is very uncertain in its decisions ($t \approx 59625$).

VII. Interaction between Learning and Evolution: the Baldwin Effect

As shown in Figure 2 for the case of sinusoid, the pure evolution is able to find the optimal policy in all experiments. For the case of stochastic time series, the optimal policy can also be found by the pure evolution, but only in some experiments. For example, for the case N_g = 300 generations and generation duration T =200, evolution was able to find optimal policy for eight out of 10 simulations. The typical example of essentially optimal policy is illustrated in Figure 6.

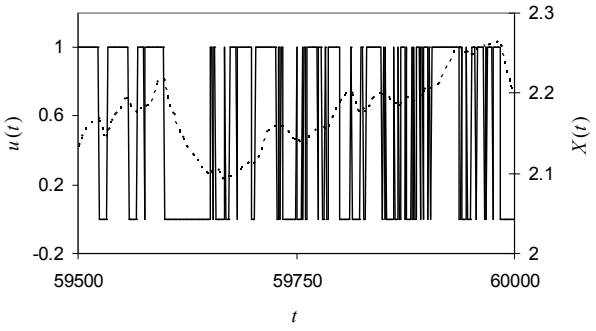

Fig. 6. The agent policy selected by the best agent in the population (solid line). Actions are characterized by values $u(t)$: $u = 0$ (all in cash) and $u = 1$ (all in stock). This is the case of pure evolution on the stochastic time series $X(t)$ (dotted line); $n = 10$, $T = 200$. The agent policy is practically optimal, as it buys/sells stocks anticipating the stock price rises/falls (to be compared with Figure 5).

However, searching for the optimal policy by means of pure evolution is slower than when combining evolution with learning, as becomes apparent when examining the curves E and LE in Figure 2. While learning in our model is not optimal by itself, it helps evolution to find better policies.

The role of learning in evolving agent populations can be observed as the Baldwin effect, or the genetic assimilation of initially learned features as a consequence of Darwinian evolution. This effect is found in several experiments, one of which is shown in Figure 7.

We examine how the best agent resource $R_{max}(t)$ changes during the first five generations for the sinusoid time series (5). By the end of each generation, the resource trend is clearly upwards. Figure 7 shows that during the early generations (generations 1 and 2), any significant increase of the agent resource begins only after a lag of 100 to 300 time steps. The best agent optimizes its policy by learning. Subsequently, the best agents find an advantageous policy faster and faster. By the fifth generation, a newborn agent "knows" a decent policy because it is encoded in its genome **G**, and the learning does not improve the policy

significantly. Thus, we can see that the initially learned policy becomes inherited (the Baldwin effect).

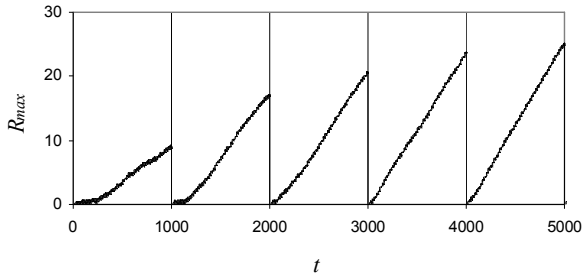

Fig. 7. The plots of the resource $R_{max}(t)$ of the best agent in the population for the first five generations. This is the case of evolution combined with learning on the sinusoid time series; population size n = 10, generation duration T = 1000. The ends of generations are shown by vertical lines. During early generations (generations 1 and 2), there is an obvious delay in the increase of the agent resource. An advantageous policy is found only after some learning period during first 100 to 300 time steps. By the fifth generation, the rapid increase of the resource begins at the start of the generation, demonstrating that the advantageous policy has become inherited.

We analyzed different sets of parameters and revealed that the Baldwin effect is observed reliably if generation duration T is greater than 1000. This means that generation duration T should be sufficiently large to ensure significant learning during each generation. The reader is referred to (Red'ko et al 2005) for additional details of the Baldwin effect as observed with our model.

VIII. Peculiarities of Model Prediction

As described in Section III and illustrated in Figure 1, each agent's ACD control system includes a model NN for predicting the next change $\Delta X(t+1)$ of the time series. We analyzed the operation of the model NN and revealed a very interesting feature. The model NN can produce incorrect predictions, yet the agent can still use these predictions to select appropriate actions. For example, Figure 8 shows the predicted changes $\Delta X^{pr}(t+1)$ and the real changes $\Delta X(t+1)$ of stochastic time series for the case of pure evolution (case E). The model NN predictions optimized by evolution match the shape of the curve ΔX quite well. However, the predicted values $\Delta X^{pr}(t+1)$ differ by the factor of about 25 from their targets $\Delta X(t+1)$.

In Figure 9 we provide another example of the model NN peculiar predictions for the case LE. At first glance, the model NN predictions appear "contrarian", except that their scale is exaggerated again. This example demonstrates that the model NN predictions can differ from the real data not only by scale but also by sign.

Though the model NN predictions may be incorrect quantitatively, we believe that their correctness in shape or correctness to the extent of a simple linear transformation (e.g., sign inversion) is what makes the model NN useful for effective adaptive behavior. These predictions are used effectively by the ACDs of the agents to ensure the optimal policy (as in Figure 6).

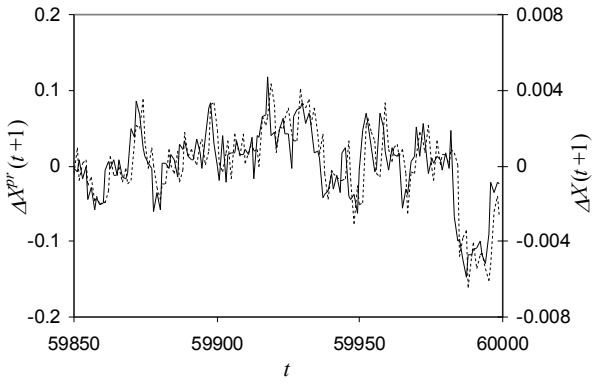

Fig. 8. The plots of predicted $\Delta X^{pr}(t+1)$ (dotted line) and real values $\Delta X(t+1)$ (solid line). This is the case of pure evolution on the stochastic time series; n = 10, T = 200. The shape of both curves are practically the same, but values $\Delta X^{pr}(t+1)$ and $\Delta X(t+1)$ differ drastically.

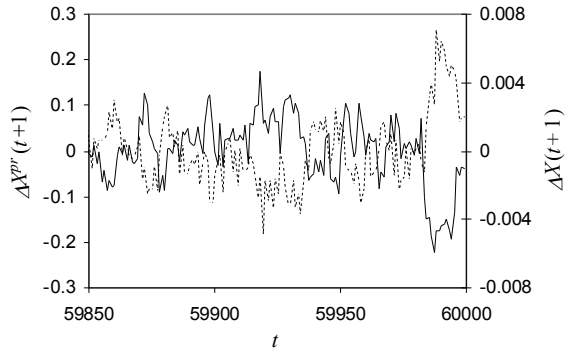

Fig.9. The plots of predicted $\Delta X^{pr}(t+1)$ (dotted line) and real values $\Delta X(t+1)$ (solid line). This is the case of evolution combined with learning on the stochastic time series; n = 10, T = 200. The curves $\Delta X^{pr}(t+1)$ and $\Delta X(t+1)$ differ in both scale and sign.

It is also possible that the observed spontaneous amplification of ΔX^{pr} by the model NN is helpful to achieve stable training and operation of the critic NN because the real values $\Delta X(t+1)$ are too small (on the order of 0.001). Thus, the model NN can not only predict values $\Delta X^{pr}(t+1)$ but also perform useful scale transformations of these values.

The peculiarities of model NN performance as described in this section are mainly due to the dominant role of evolution over learning for the optimization of agent control systems. Indeed, due to the short generation duration (T = 200) in our simulations, the synaptic weights of ACD NNs are adjusted mainly by evolutionary mutations. Such a process seems to favor agent control systems that are stable in the evolutionary sense.

IX. Comparison with Behavior of Simple Animals

Our agents have two behavioral tactics (buy or sell stocks) and select actions by switching between these tactics. It turns out that this behavior with switching between two tactics is analogous to the searching tactics of simple animals. For example, some species of caddis fly larvae use similar tactics for case building (Nepomnyashchikh, 1998, 2004). The larvae inhabit creek bottoms and build their cases from hard particles of different size. They can use small or large sand particles (Nepomnyashchikh, 1998). Large particles are distributed randomly, but typically occur in groups of several particles. Using large particles, the larva can build cases more quickly and effectively than with small particles, so its preference is evident. The larva uses two tactics: 1) testing particles in its vicinity and building the case from selected particles, 2) searching for a new place with a collection of appropriate particles. Investigations of larva behavior reveal inertia in switching from the first tactic to the second tactic (Nepomnyashchikh, 1998, 2004). If the larva finds a large particle, it continues testing particles until it finds several small particles, and only after repeated failures to find new large particles does the larva switch to the second tactic. During the search for a new place, the larva wanders and sometimes randomly tests particles along its way. It can switch from the second tactic to the first tactic if it finds a large particle. When switching from the second tactic to the first tactic, it may also exhibit inertia. Thus, the switching between tactics resembles a random search with inertial effects. The inertial switching process pays attention only to general large-scale patterns in the environment.

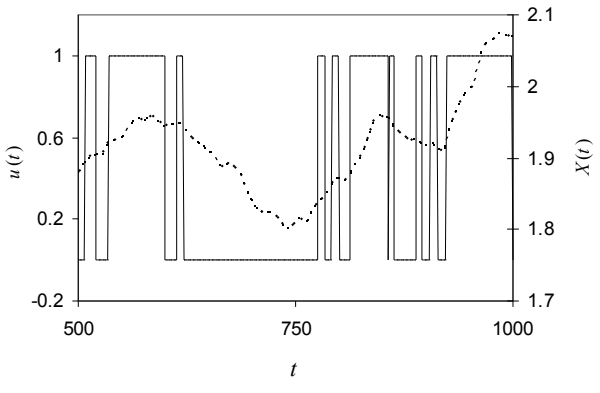

Fig.10. The agent policy selected by the best agent in population (solid line). Actions are characterized by values $u(t)$: $u = 0$ (all in cash) and $u = 1$ (all in stock). This is the case of pure evolution on the stochastic time series $X(t)$ (dotted line); $n = 100$, $T = 200$. The agent policy is similar to behavior of simple animals. The switching appears to be triggered only by general trends in the environment, with inertia when switching between two tactics (buy or sell stocks).

Similar behavior by our agent-broker is observed, when the ACD control system of an agent is optimized by random explorations over a relatively large area, i.e., by means of pure evolution operating in large populations. Figure 10 shows a fragment of the best agent policy found during early stages of pure evolution when the population size is large, $n = 100$. This agent policy is similar to behavior of simple animals. The policy switching between $u = 0$ and $u = 1$ is a reaction only to general patterns of changes in the environment (the agent ignores small fluctuations of the stock price). In addition, the switching is inertial.

X. Discussion and Conclusion

We investigated the interaction between learning and evolution in populations of self-learning agent-brokers. In our simulations we considered three cases in which the synaptic weights of the NNs are adjusted by learning, by evolution, or by a combination of learning and evolution.

We demonstrated that pure learning might be slower than pure evolution, especially when the simple gradient descent rule is used to train the ACD NNs. Yet the combination of learning with evolution seems to result in a useful synergy, as the two methods not only compensate for each other's imperfections, but also yield a superior (often optimal) policy for the agent-broker.

To elucidate the interaction between learning and evolution, we discuss the following thought experiment. Assume that the initial weights of respective generations in the case E (pure evolution) and the case LE (combined learning and evolution) are exactly the same, i.e., mutations add to the weights exactly the same values in both cases. Any additional improvement of performance is achieved only through learning. If learning "fights" evolution, then such additional improvement of performance may not happen (this can also be perceived as a local optimum or suboptimum of the performance). If synaptic weights **W** obtained through mutations are such that learning for improved performance is easy when starting from **W**, then the additional improvement in performance may be tangible. Suppose we employ a very powerful learning algorithm (powerful in the sense that it can guarantee a significant performance improvement by the end of each generation starting from any **W**; this may be quite idealistic in general!). Then gains of performance for the case LE as compared to the case E are guaranteed until we reach a (global) optimum of performance. In its vicinity, such gains would become increasingly small independent of the learning algorithm employed (otherwise it would contradict the assumption of proximity to the optimum). If the case E is allowed to run for a very long time, it should eventually catch up with the case LE in the performance (i.e., its best agent would be as good as the best agent trained in the case LE), but the case LE should on average produce the best

performing agent faster. That is what we observed in our experiments.

We also observed the genetic assimilation of successfully learned features over the course of Darwinian evolution, in several experiments. Known as the Baldwin effect, this is another consequence of the interaction between learning and evolution.

We also found examples in which evolution results in a model NN which does not correctly predict changes of the environment (changes of time series in our simulations). Though it often accurately predicts only the shape of the time series ΔX, the model NN is still useful as its amplified predictions facilitate stable operation of the critic NN.

We also compared the agent-broker's behavior with the searching behavior of simple animals. Animal behavior exhibits inertial effects when switching between different behavioral tactics. Such inertia helps an animal to react adaptively to only general large-scale patterns in environment. We demonstrate that similar inertial behavior can be identified in our model for the case of pure evolutionary search during the early stages of evolution, if the population size is sufficiently large. It is remarkable that our agent-broker, as evolved in a population through a simple stochastic search algorithm, can imitate the behavior of real animals.

We also observed that pure evolution working on the NN synaptic weights can create agent control systems with stable evolutionary behavior. By evolutionary stability we mean the property of phenotypes (and their associated genotypes) to become essentially insensitive to mutations applied in every generation. In fact, evolutionary stability is generally expected, as the agent performance converges to its optimum. However, this does raise the question of how to design best the agent control system for evolutionary stability.

As we discussed only one particular evolutionary model in a simple application (agent-broker), the reader is cautioned about generalizing our results too broadly. Yet it is reasonable to expect that the superiority of evolution combined with learning will hold for other tasks. The well known success of optimization methods employing combinations of genetic algorithms with local gradient based learning algorithms supports this expected outcome, albeit for the Lamarckian rather than Darwinian form of the evolution. When the task is more difficult, pure evolution is expected to be much slower than the combined approach, or equivalently it should require a much larger population size n. Our choice to employ the gradient descent for learning was due to its simplicity. More powerful learning methods can undoubtedly benefit both pure learning and the combination of learning and evolution.

Acknowledgments

The authors thank Dr. Valentin A. Nepomnyashchikh for valuable consultations on caddis fly larvae behavior, as well as William Howell for many helpful comments. This work is supported in part by the Russian Foundation for Basic Research, Grant No 04-07-90038 and the Program of the Presidium of the Russian Academy of Science "Mathematical Modeling and Intelligent Systems", Project No 16.2.45.

References:

Ackley, D. & Littman, M. (1992). Interactions between learning and evolution. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial Life II*. Reading (pp.487-509). MA: Addison-Wesley.

Baldwin, J.M. (1896). A new factor in evolution. *American Naturalist*, 30, 441-451.

Belew, R.K. & Mitchell, M. (Eds.) (1996). *Adaptive individuals in evolving populations: Models and algorithms*. MA: Addison-Wesley.

Hinton, G. E. & Nowlan, S. J. (1987) How learning can guide evolution. *Complex Systems*, 1, 495–502.

Kotlyar, V.I. & Shulgovsky, V.V. (1979). *Physiology of central nervous system*. Moscow: Moscow State University Press (In Russian).

Moody, J., Wu, L., Liao, Y., & Saffel, M. (1998). Performance function and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17, 441-470.

Nepomnyashchikh, V.A. (1998). Selection behaviour in caddis fly larvae. In R. Pfeifer et al (Eds.) *From Animals to Animats 5: Proceedings of the Fifth International Conference of the Society for Adaptive Behavior* (pp.155-160). Cambridge, MA: MIT Press.

Nepomnyashchikh, V.A. (2004). How animals solve poorly-formalized search tasks. In V.I. Arshinov, I.N. Trofimova & V.M. Shendyapin (Eds.) *Synergetics and Psychology. Texts.* Issue 3 (pp. 197-209). Moscow: Cognito-Center (In Russian).

Prokhorov, D., Puskorius, G., & Feldkamp L. (2001). Dynamical neural networks for control. In J. Kolen and S. Kremer (Eds.) *A field guide to dynamical recurrent networks* (pp. 257-289). NY: IEEE Press.

Prokhorov, D.V. & Wunsch, D.C. (1997). Adaptive critic designs. *IEEE Transactions on Neural Networks*, 8, 997-1007.

Red'ko V.G., Mosalov, O.P., & Prokhorov, D.V. (2005). A model of Baldwin effect in populations of self-learning agents. In *International Joint Conference on Neural Networks, IJCNN 2005, Proceedings*, Montreal, Canada.

Sutton, R. & Barto A. (1998). *Reinforcement Learning: An Introduction*. Cambridge: MIT Press.

Suzuki, R. & Arita, T. (2004). Interactions between learning and evolution: outstanding strategy generated by the Baldwin effect. *Biosystems*, 77, 57-71.

Weber, B.H. & Depew, D.J. (Eds.) (2003). *Evolution and learning: The Baldwin effect reconsidered*. MA: MIT Press.

Werbos, P.J. (1992). Approximate dynamic programming for real-time control and neural modeling. In White and Sofge (Eds.) *Handbook of Intelligent Control* (pp. 493 – 525). NY: Van Nostrand Reinhold.