Red'ko V.G., Prokhorov D.V. Learning and Evolution of Autonomous Adaptive Agents // Advances in Machine Learning I. Dedicated to the memory of Professor Ryszard S. Michalski. Koronacki, J., Ras, Z.W., Wierzchon, S.T. (et al.) (Eds.), Series "Studies in Computational Intelligence". Springer. ISSN: 1860-949X Vol. 262, 2010. PP. 491-500.

Learning and Evolution of Autonomous Adaptive Agents

Vladimir G. Red'ko[1] and Danil V. Prokhorov[2]

[1] Center of Optical Neural Technologies, Scientific Research Institute for System Analysis, Russian Academy of Science, Vavilova Str., 44/2, Moscow, 119333, Russia, vgredko@gmail.com

[2] Toyota Technical Center, Ann Arbor, MI, USA, dvprokhorov@gmail.com

Abstract – We study a model of evolving populations of self-learning agents and analyze the interaction between learning and evolution. We consider agent-brokers that predict stock price changes and use these 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 learning, or evolution, 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. Additionally, we analyze influence of neural network structure of adaptive critic design on learning processes.

Key words: autonomous agents, reinforcement learning, neural networks, adaptive critic designs, evolutionary algorithm

1. Introduction

The current paper overviews and generalizes our previous work on modeling learning and evolution of autonomous adaptive agents [1, 2]. The control system of the agent ensures autonomous, self-learning agent behavior. The agent control system is based on a neural network adaptive critic design (ACD). The ACD includes two neural networks (NNs), model and critic. The model predicts the state of the environment for the next time step, and the critic is used to select actions on the basis of model predictions. These NNs can be optimized by both learning and evolution. Learning uses reinforcement learning method [3]. Darwinian type of evolution includes mutations of NN synaptic weights and selection of agents that have the best control systems.

The model of autonomous adaptive agents is developed in the framework of the animat approach to artificial intelligence that was proposed in the early 1990s [4, 5]. This research methodology implies understanding intelligence through simulation of artificial animals (*animats*) in progressively more challenging environments. We design the model for simple example of autonomous adaptive agent-brokers that can buy and sell stocks in order to increase their capital. The model is investigated by means of computer simulations.

We analyze specifically the interaction between learning and evolution. It should be noted that the rather non-trivial Baldwin effect [6, 7] can be observed in our model, viz., originally acquired adaptive policy of best agent-brokers becomes inherited over the course of the evolution. In particular, in some simulations acquired features that at initial generations are obtained by means of learning become inherited at fifth generation. This means that NN synaptic weights that are initially adjusted by reinforcement learning are then rediscovered during mutations and selections. Though the evolutionary processes are of Darwinian type, these processes have certain Lamarckian properties.

The paper is organized as follows. Section 2 includes the description of the model. The simulation results are described in Section 3; this Section includes analysis of the interaction between learning and evolution and comparison of two variants of neural network structure of adaptive critic design: with one hidden layer and with two hidden layers. Section 4 concludes the paper.

2 Description of the Model

2.1 Agent Task

Inspired by [8], 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 to modify 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 $(1 > J \geq 0)$ 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 [9], 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 we assume that the variable $u(t)$ takes only two values, $u(t) = 0$ (all in cash) or $u(t) = 1$ (all in stock).

2.2 Agent Learning

The agent control system is a simplified ACD. Our adaptive critic scheme consists of two NNs: a model and a critic (see Fig. 1).

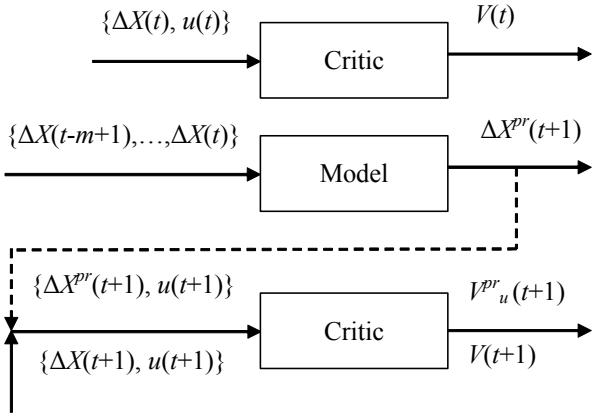

Fig. 1. The scheme of the considered adaptive critic design (ACD). The ACD consists of two neural networks (NNs): a model and a critic. The model predicts changes of the time series. The critic (the same NN is shown in two consecutive moments) forms the state value function for the current state $\mathbf{S}(t)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

The goal of the adaptive critic is to stochastically maximize the utility function $U(t)$ [3]:

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

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 $u(t)$: $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$.

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. The operation of this neural network is described by the following expressions:

$$\mathbf{x}^{\mathbf{M}} = \{\Delta X(t-m+1),\ldots,\Delta X(t)\},\ y^{M}_{j} = \tanh\left(\sum_i w^{M}_{ij} x^{M}_{i}\right),\ \Delta X^{pr}(t+1) = \sum_j v^{M}_{j} y^{M}_{j}, \quad (4)$$

where $\mathbf{x}^{\mathbf{M}}$ is the neural network input vector, $\mathbf{y}^{\mathbf{M}}$ is the vector of outputs of hidden layer neurons, w^{M}_{ij} and v^{M}_{j} are neuron synaptic weights.

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 $\mathbf{S}^{\mathbf{pr}}{}_{u}(t+1) = \{\Delta X^{pr}(t+1), u(t+1)\}$ for two possible actions, $u(t+1) = 0$ or $u(t+1) = 1$. The critic is also an MLP of the same structure as the model. The operation of the critic neural network is described by the following expressions:

$$\mathbf{x}^{\mathbf{C}} = \mathbf{S}(t) = \{\Delta X(t), u(t)\},\quad y^{C}_{j} = \tanh\left(\sum_i w^{C}_{ij} x^{C}_{i}\right),\quad V(t) = V(\mathbf{S}(t)) = \sum_j v^{C}_{j} y^{C}_{j}, \quad (5)$$

where $\mathbf{x}^{\mathbf{C}}$ is the neural network input vector, $\mathbf{y}^{\mathbf{C}}$ is the vector of outputs of hidden layer neurons, w^{C}_{ij} and v^{C}_{j} are neuron synaptic weights.

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 $\mathbf{S}^{\mathbf{pr}}{}_{u}(t+1) = \{\Delta X^{pr}(t+1), u(t+1)\}$, and $u(t+1) = 0$ or $u(t+1) = 1$.

3) The ε-greedy rule [3] is applied: 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 by means of the method of error backpropagation [10] 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:

$$\delta(t) = r(t) + \gamma V(t+1) - V(t)\ . \quad (6)$$

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

Note that the error of the model neural network is known, it is equal to the value $E = \Delta X(t+1) - \Delta X^{pr}(t+1)$, so this network is trained to minimize the value E by means of usual error backpropagation method [10]. On the other hand, the critic neural network is learned using rewards and punishments, so this network is trained by means of the temporal difference method that is specially developed in reinforcement learning research [3].

2.3 Evolutionary Algorithm

We consider Darwinian evolution of agent populations. We use a simple version of the genetic algorithm [11,12] without crossovers. Our evolutionary algorithm is described below.

The evolving population consists of n agents. Each agent has a resource 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, $n_g = 1,2,\ldots$ The duration of each generation n_g is T time steps. At the end of each n_g-th generation, the best agent that maximally increases its resource $R(t)$ during generation time is selected, this best agent gives birth all n agents of the next (n_g+1)-th generation. Any agent has its genome $\mathbf{G}$; the children genomes $\mathbf{G}$ differ from their parent genome by small mutations.

The agent genome $\mathbf{G}$ is equal to the initial synaptic weights of both NNs (model and critic), $\mathbf{G} = \{\mathbf{W}_{\mathrm{M0}}, \mathbf{W}_{\mathrm{C0}}\}$, these weights are obtained by the agent at the moment of its birth. The evolutionary process is of Darwinian type, so the agent genome $\mathbf{G}$ does not change during the agent life. However, synaptic weights of the NNs $\mathbf{W}_{\mathrm{M}}$ and $\mathbf{W}_{\mathrm{C}}$ are changed during agent life via learning described above.

At the beginning of any generation, the initial resource of each agent is zero, i.e., $R(T(n_g-1)+1) = 0$. The best agent of the generation n_g has the maximum resource $R_{max}(n_g)$ at the end of the generation.

At the beginning of every new (n_g+1)-th generation, we set for each agent $G_i(n_g+1) = G_{best,\,i}\,(n_g) + \mathrm{rand}_i$, $\mathbf{W}_0(n_g+1) = \mathbf{G}(n_g+1)$, where $\mathbf{G_{best}}(n_g)$ 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 **G** changes only via evolution, whereas the synaptic weights **W** are adjusted only via learning.

3 Results of Simulations

3.1 General Features of Agent Learning and Evolution

The described model was investigated by means of computer simulations. We used two examples of model time series: 1) sinusoid:

$$X(t) = 0.5\,[1 + \sin(2\pi t/20)] + 1\ , \qquad (7)$$

2) stochastic time series [8]:

$$X(t) = \exp[p(t)/1200]\ ,\ \ p(t) = p(t-1) + \beta(t-1) + k_1\,\lambda(t)\ ,\ \ \beta(t) = k_2\,\beta(t-1) + \mu(t)\ , \qquad (8)$$

where $\lambda(t)$ and $\mu(t)$ are two random normal processes with zero mean and unit variance, $k_1 = 0.3$, $k_2 = 0.9$.

Some parameters were set to the same values for all simulations. Specifically, we set, $\gamma = 0.9$, $m = 10$, $\alpha_M = \alpha_C = 0.01$, $N_{hM} = N_{hC} = 10$, $\varepsilon = 0.05$, $P_{mut} = 0.1$, where N_{hM} and N_{hC} are numbers of hidden neurons of the model and critic. Other parameters (n, T) were set to different values, depending on the simulation, as specified below. Agent expenses are neglected ($J = 0$) in the main part of simulations.

We analyze the following cases:

1) case L (pure learning); in this case we consider a single agent that learns by means of temporal difference method;

2) case E (pure evolution), i.e., evolving population without learning;

3) case LE, i.e., learning combined with evolution, as described above.

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 $R(T(n_g-1)+1) = 0$ after the passing of every $T = 200$ time steps for consistency with the cases E and LE. In case L the index n_g is incremented by one after every T time steps, i.e., $R_{max}(n_g) = R(Tn_g)$. The plots R_{max} vs. n_g for the sinusoid (7) are shown in Fig. 2. 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. The results are averaged over 1000 simulations with $n = 10$, $T = 200$.

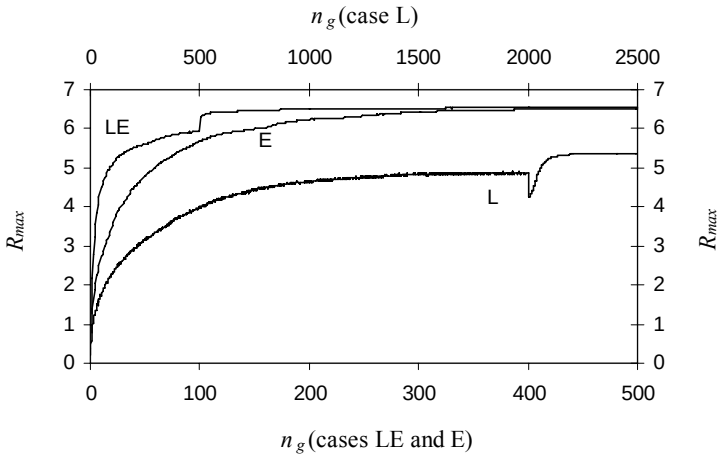

Fig. 2. The plots of $R_{max}(n_g)$ for the sinusoid (7). The curves LE, E and L correspond to the cases of learning combined with evolution, pure evolution and pure learning, respectively. Each point of the plots represents the average over 1000 simulations; $n = 10$, $T = 200$.

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. Analysis of agent behavior demonstrates that both pure evolution and learning combined with evolution are able to find the optimal policy. With this policy, the agent attains asymptotic value $R_{max} = 6.5$ (see Fig. 2). For the case L, the asymptotic value of R_{max} is only 5.4. Analysis reveals that the pure learning is able to find only the following satisfactory policy. The agent buys stocks when stock price rises (or falls by a small amount) and sells stocks when stock price falls significantly. So, the agent prefers to keep the capital in stocks in this case.

We described results for the sinusoidal time series (7). It should be noted that similar learning and evolutionary processes are observed for stochastic time series (8).

3.2 Interaction between Learning and Evolution. Baldwin Effect

As shown in Fig. 2 for the case of sinusoid, the pure evolution is able to find the optimal policy in all experiments. However, searching for the optimal policy by means of pure evolution is slower than when combining learning with evolution, as becomes apparent when examining the curves E and LE in Fig. 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 Fig. 3. We examine how the best agent resource $R_{max}(t)$ changes during the first five generations for the sinusoid time series (7). By the end of each generation, the resource trend is clearly upwards.

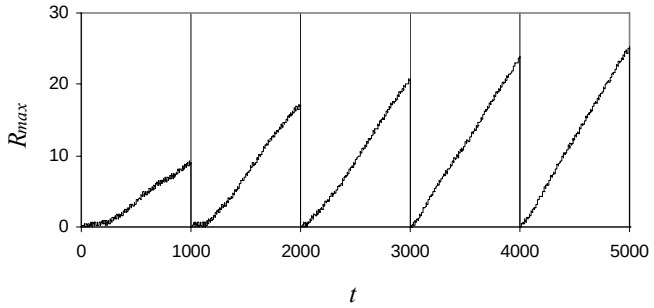

Fig. 3. The plots of the resource $R_{max}(t)$ of the best agent in the population for the first five generations. This is the case of learning combined with evolution on the sinusoid time series; population size $n = 10$, generation duration $T = 1000$. The ends of generations are shown by vertical 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.

Fig. 3 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; this corresponds to the Baldwin effect [6, 7]. Actually, simulations demonstrate that though the evolutionary processes are of Darwinian type, these processes have certain Lamarckian properties.

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.

3.3 Influence of Neural Network Structure on Learning Processes

We demonstrated that evolutionary optimization of simple neural networks (NNs) can be more effective than reinforcement learning. In order to analyze the problem in details, we investigate additionally the influence of neural network structure on learning processes in the control system of the agents. Namely, we compare two variants of NNs structures: with one hidden layer (see above) and with two hidden layers. The operation of NNs with one hidden layer is described by equations (4), (5); slightly modified equations can be used to describe the operation of NNs with two hidden layers.

The control system of agents has the same architecture as described above (see Fig. 1); however, the NNs of the model and the critic can have one (first variant) or two (second variant) hidden layers. For both variants we consider three cases of adaptation: L (pure learning), E (pure evolution), and LE, i.e. learning combined with evolution. The main results for first variant are illustrated in Fig. 2.

We analyze the influence of NNs structure on learning (type L of adaptation). The corresponding results are illustrated by Figs. 4 and 5. These figures show the plots of the mean value $R_{max}(n_g)$ along with standard deviation $\sigma(n_g)$ characterizing the distribution of $R_{max}(n_g)$ with respect to different random seeds.

Fig. 4 demonstrates that for the variant 1 (NNs with one hidden layer) the standard deviation $\sigma(n_g)$ decreases and tends to zero meaning that agents find similar policies in all simulations. In contrasts to this, the standard deviation $\sigma(n_g)$ tends to some large asymptotic value for the NNs with two hidden layers (Fig. 5). Analysis reveals that in this variant, leaning is able to find successfully the optimal policy for some set of initial NNs synaptic weights (approximately for 50% of random seeds), while for other initial synaptic weights learning does not find even a satisfactory policy: agents keep their capital in cash all the time and do not increase their resource.

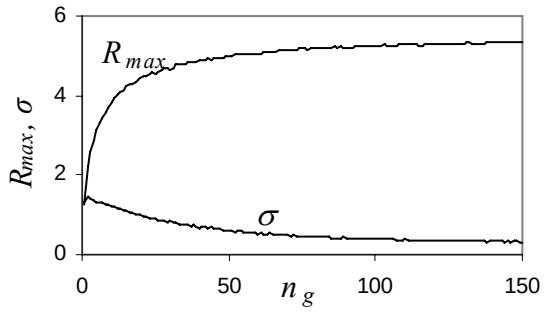

Fig. 4. The plot of $R_{max}(n_g)$ and the standard deviation $\sigma(n_g)$ of $R_{max}(n_g)$ for pure learning (the case L) on the sinusoid time series. Every point of the plot of $R_{max}(n_g)$ represents the average over 1000 simulations; $n = 10$, $T = 200$. The values $\sigma(n_g)$ characterize a distribution of $R_{max}(n_g)$ over different random seeds. NNs have one hidden layer (the variant 1).

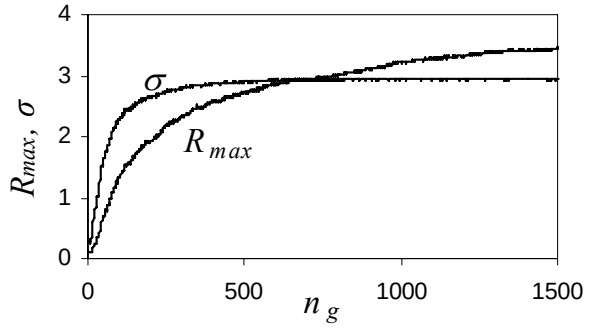

Fig. 5. The plot of $R_{max}(n_g)$ and the standard deviation $\sigma(n_g)$ of $R_{max}(n_g)$ for pure learning (the case L) on the sinusoid time series. Every point of the plot of $R_{max}(n_g)$ represents the average over 1000 simulations; $n = 10$, $T = 200$. The values $\sigma(n_g)$ characterize a distribution of $R_{max}(n_g)$ over different random seeds. NNs have two hidden layers (the variant 2).

Thus, addition of the second hidden layer does result in searching for the optimal policy by means of learning for some random seeds; however, the same type of learning may be very ineffective for other random seeds.

We also analyze evolutionary optimization of agent ACDs. Simulations demonstrate that both evolution (E) and learning combined with evolution (LE) are able to find the optimal policy in control systems with two hidden layer NNs similar to the variant with one hidden layer NNs.

4 Conclusions

We investigated adaptive agents that are able to predict future states, to estimate state value function and to select actions on the basis of predictions and estimations. These adaptive functions can be obtained by means of reinforcement learning in the agent control system. However, our simulations for agents-brokers demonstrate that such control systems are too “intelligent” for the simple agent-broker task. Namely, simple evolution is more efficient in searching for the optimal policy than reinforcement learning. Increase of the number of NNs layers does not improve reinforcement

learning essentially. Our simulations demonstrate also that evolution and learning together are more efficient in searching for the optimal agent policy than evolution or learning alone.

We paid special attention to the interaction between learning and evolution. In particular the Baldwin effect is revealed in our model: originally acquired adaptive policy of best agent-brokers becomes inherited during several generations of evolution. This means that certain Lamarckian properties can be observed during Darwinian evolution.

Acknowledgments

The authors thank Dr. Oleg P. Mosalov for his help with computer simulations. This work is partially supported by the Russian Foundation for Basic Research, Grant No 07-01-00180 and the Program of the Presidium of the Russian Academy of Science "Intelligent informational technologies, mathematical modeling, system analysis, and automatics", Project No 2.45.

References

1. Red'ko, V.G., Mosalov, O.P., Prokhorov, D.V.: A model of Baldwin effect in populations of self-learning agents. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN'2005, Vol. 3, pp. 1355-1360, Montreal, Canada (2005)
2. Red'ko, V.G., Mosalov, O.P., Prokhorov, D.V.: A model of evolution and learning. Neural Networks, 18, 738-745 (2005)
3. Sutton, R. and Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
4. Meyer, J.-A. and Wilson, S.W. (eds.): From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, Cambridge (1991)
5. Wilson, S.W.: The animat path to AI. In: [4], pp. 15-21.
6. Baldwin, J.M.: A new factor in evolution. American Naturalist, 30, 441-451 (1896)
7. Belew, R.K. and Mitchell, M. (eds.): Adaptive Individuals in Evolving Populations: Models and Algorithms. Santa Fe Institute Studies in the Sciences of Complexity, Vol. 26, Addison-Wesley (1996)
8. Prokhorov, D., Puskorius, G., Feldkamp, L.: Dynamical neural networks for control. In: J. Kolen and S. Kremer (eds.): A Field Guide to Dynamical Recurrent Networks, pp. 23-78. IEEE Press (2001)
9. Moody, J., Wu, L., Liao, Y., Saffel, M.: Performance function and reinforcement learning for trading systems and portfolios. Journal of Forecasting, 17, 441-470 (1998)
10. Rumelhart, D.E., Hinton, G.E., Williams, R.G.: Learning representation by back-propagating error. Nature, 323, 533-536 (1986)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. Ann Arbor, MI: The University of Michigan Press (1975). Boston, MA: MIT Press (1992)
12. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)