Neural network based control of multi-agent system

B.V.Kryzhanovsky[1], M.V.Kryzhanovsky[1], A.B.Fonarev[2]

[1]Institute of Optical Neural Technologies RAS, Vavilova st. 44/2, Moscow 199333
kryzhanov@mail.ru, http://www.iont.ru
[2]Department of Chemistry and Physics, Kean University, Union, New Jersey 07803

Abstract. A neural network approach to the problem of discrete optimization of multi-agent system's functional is discussed. The approach ability is illustrated by solving the problem of assigning targets among a number of agents. The proposed neural network algorithm makes it considerably easier to solve the problem of decentralized control and decreases the amount of computations in m^2 times, where m is the number of targets. The algorithm is based on the fact that the behavior of each agent results in an optimization of the state of the system as a whole. The modification of the functional proposed here allows one to avoid usual problems in selecting the Lagrangian coefficient.

1. Introduction

The point of decentralized collective control of a group of agents is as follows [1]. At each moment an agent has to plan his behavior for himself. The agent chooses the way of control depending on current environment and probable behavior of other agents within a short period of time. The optimal control suggests that each agent takes the behavior that adds most to the objective functional.
The method was used [2] to tackle a model problem of target assignment, the algorithm involving partial enumeration. The paper offers another approach which simplifies the solution. Moreover, this approach implies that behavior of each agent tends to improve the state of the whole system rather than his own state.

2. Statement of the centralized control problem

Let there be m targets and N agents R_k ($k = 1,\ldots,N$). Agents have to strike the targets under conditions when there is no knowing about the adversary's counteractions. This means that some agents may be killed and the number of agents varies. Let the damage from the i-th target hit ($i = 1,\ldots,m$) be

$$\Phi_i = \Phi_i(n_i) \qquad (1)$$

where n_i is the number of the agents that hit the i-th target, and $\Phi_i(n_i)$ are nonlinear monotone increasing functions. The possible behavior of these functions is shown in Fig.1, where $n_{\max}$ is the number of the agents whose hitting a target secures its destruction, and the maximum damage caused by destruction of the *i*-th target is normalized to unit $\Phi(n_{\max}) = 1$.

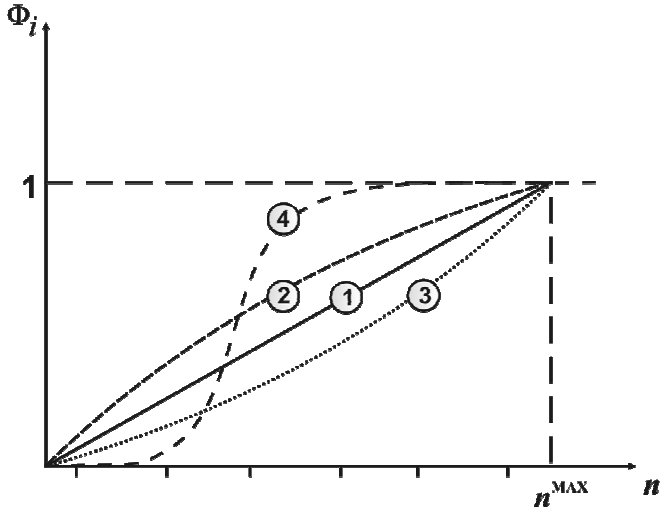

Fig. 1. Possible types of functions $\Phi_i(n)$

The total damage should be chosen as a functional to optimize. In other words, the optimization of behavior of the multiple-agent system comes down to finding the maximum of function

$$\Phi = \sum_{i=1}^{m} \Phi_i(n_i) \qquad (2)$$

given

$$\sum_{i=1}^{m} n_i = N \qquad (3)$$

Finding the maximum of function Φ on the assumption of (3) is reduced to defining the maximum of functional

$$W = \Phi - \mu\left(N - \sum n_i\right)^2 \qquad (4)$$

where μ is the Lagrangian coefficient. The problem is usually solved by the well-know method of gradient descent [3], that is, by computing the gradient of the functional and finding the increment of the variable $\delta \boldsymbol{n} = \alpha\, dW / d\boldsymbol{n}$ that ensures optimization. However, in our case the approach can't be applied because extremum of W is searched in the configurational space of variables. Because the components

of vector $\boldsymbol{n} = (n_1, n_2, ..., n_m)$ are integers, their increments are also integers and should be described by a vector $\boldsymbol{u}$ with components $u_i = 0, \pm 1$. For this reason, we employ the following iteration process to find the optimum of functional *W*. Let the initial number of agents be $\boldsymbol{N}_0$ and the initial target distribution be $\boldsymbol{n} = \boldsymbol{n}_0$. At the first stage of the iteration process we expand functional $W = W(\boldsymbol{n})$ around $\boldsymbol{n}_0$ by setting $\boldsymbol{n} = \boldsymbol{n}_0 + \boldsymbol{u}$, where $\boldsymbol{u}$ is the sought-for increment vector. Then, by expanding (4) around $\boldsymbol{n}_0$ and replacing small increments $\delta\boldsymbol{n}$ by components of vector $\boldsymbol{u}$, we can write the sought-for expression to inessential constant as

$$W(\boldsymbol{u}) = \sum_{i=1}^{m} p_i u_i - \frac{\mu}{2}\left(N_0 - N + \sum_{i=1}^{m} u_i \right)^2 \qquad (5)$$

or in a similar way

$$W(\boldsymbol{u}) = -\tfrac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} T_{ij} u_i u_j + \sum_{i=1}^{m} h_i u_i \qquad (6)$$

where

$$\begin{aligned} h_i &= p_i + \mu(N - N_0) \\ T_{ij} &= \mu \\ p_i &= d\Phi_i / dn_i \big|_{\boldsymbol{n}=\boldsymbol{n}_0} \end{aligned} \qquad (7)$$

Expression (6) agrees with what is defined as energy in the Hopfield model [4] given T_{ij} is the interconnection matrix, and h_i are the neuron thresholds. Accordingly, we can search for the maximum of quadratic functional (5) by using the conventional neural-net optimization algorithm [5-9]. At first we set the neural net to the initial state $u_i = 0$ and then launch the procedure of ascent up the landscape[1]. After having determined vector $\boldsymbol{u}$, which brings functional $W = W(\boldsymbol{n})$ to the maximum at around point $\boldsymbol{n}_0$, we go to the vicinity of the next point $\boldsymbol{n}_1 = \boldsymbol{n}_0 + \boldsymbol{u}$. Here we find new values of p_i (derivatives of functions Φ_i at $\boldsymbol{n}_1$) and optimize functional (6) over again. The iteration process continues until the necessary accuracy of the solution.

However, some difficulty in selecting the Lagrangian coefficient for the penalty function and inconvenience of its use in the case of decentralized control make such straight application of the iteration process to finding vector $\boldsymbol{u}$ somewhat awkward. It is necessary to modify selection of vector $\boldsymbol{u}$ so that penalty is smallest after an iteration step. To get necessary expressions, let's consider the use of the gradient method for this problem in the real-number domain n_i. In this case, known formulae are used to calculate the increments. In our notification the formulae take the form:

$$u_i = \alpha \partial W / \partial u_i = \alpha (p_i - r_0) \qquad (8)$$

[1] Usually a functional of the opposite sign undergoes the surface-descent optimization.

$$r_0 = \mu\left(N_0 - N + \sum_{i=1}^{m} u_i \right) \tag{9}$$

Let's modify the finding of vector $\boldsymbol{u}$ to provide the smallest value of penalty. We replace expression (8) with a new one

$$u_i = \alpha(p_i - r_0 + \bar{p}) \tag{10}$$

and try to optimize the process by choosing suitable parameter $\bar{p}$. For this purpose we determine the penalty after one iteration step:

$$u_i^{(1)} = u_i^{(0)} + \delta u_i \tag{11}$$

$$\delta u_i = \alpha(p_i - r_0 + \bar{p}) \tag{12}$$

$$r_0 = \mu\left(N_0 - N + \sum_{i=1}^{m} u_i^{(0)} \right) \tag{13}$$

$$r_1 = r_0(1 - m\mu\alpha) + \mu\alpha\left(m\bar{p} - \sum_{i=1}^{m} p_i \right) \tag{14}$$

The last expression means that given

$$\bar{p} = m^{-1}\sum_{i=1}^{m} p_i \tag{15}$$

and certain value of $\mu\alpha$, the penalty decreases and $r_1 = 0$ for $r_0 = 0$.

Let's apply the same approach to the work in the configurational space. By analogy with the modified rule (10) for setting the increment vector $\boldsymbol{u}$, we alter the original functional $W(\boldsymbol{n})$ so that it is written as:

$$W(\boldsymbol{u}) = \sum_{i=1}^{m}(p_i - \bar{p})u_i - \frac{\mu}{2}\left(N_0 - N + \sum_{i=1}^{m} u_i \right)^2 \tag{16}$$

Clear that the modification comes down to simple substitution of p_i by their deviations from the mean value $\bar{p}$. Besides, it is necessary to modify the initial state[2] of the Hopfield network by setting $u_i = \mathbf{sgn}(p_i - \bar{p})$. A lot of experiments show that modification of functional $W(\boldsymbol{n})$ to expression (16) allows the optimization problem to be solved without usual difficulties caused by the necessity to choose the

[2] To optimize functional (7), we assume that initially u_i=0.

Lagrangian coefficient. The results of the experiments are given in the next paragraph.

3. Experimental results

Computational experiments showed that the maximum of functional $W(\boldsymbol{n})$ can be found successfully and we obtain the global extremum when using function Φ of type 1-3 (see Fig.1).

Figures 2 and 3 give the results of a particular experiment. The experiment had the following conditions: number of targets $m = 5$, number of agents $N = 15$, the damage function for the *i*-th target is taken to be

$$\Phi_i = 1 - \exp(-n_i / \tau_i) \qquad (17)$$

where $\tau_i = \tau_0 / i$, and the exponential constant is taken to be $\tau_0 = 10$. Figure 2 presents the resulting target distribution given initial distribution $n_1(0) = N$, $n_i(0) = 0$ for $i \neq 1$. Figure 3 shows time dependence of total damage $\Phi = \Phi(t)$ corresponding to reassignment of targets among the agents in the course of optimization.

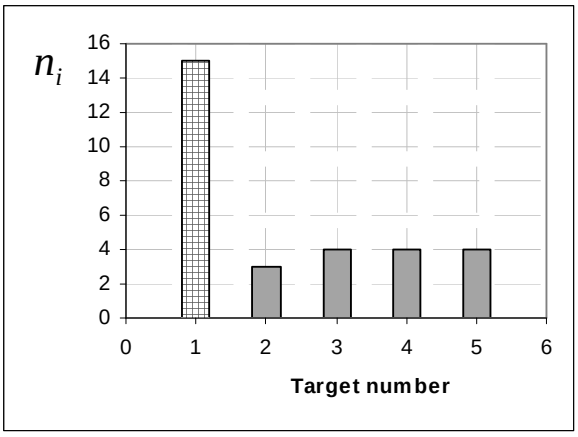

Fig. 2. Distribution over the targets after optimization (the hatched area is the initial distribution ($n_1 = N$) and the filled up area is the final distribution).

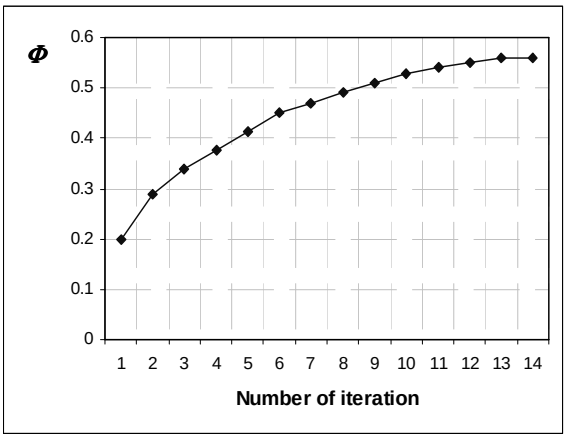

Fig. 3. Increasing of target functional Φ in optimization process.

However, in the case when we using function Φ of type 4 (see Fig.1) the method was found to give usually a local maximum rather than a global one. To illustrate this fact, the conditions of the experiment were changed. With the same *m* and *N*, expression (17) defined damage functions for targets $i = 1,...,4$ only, and the fifth target function Φ_5 was replased by sigmoidal one $\Phi_5 = \Phi(0) + \mathbf{th}(n/\theta)$. The initial distribution ($n_1(0) = N$, $n_i(0) = 0$, $i \neq 1$) gave $n_1 = 5$, $n_2 = 4$, $n_3 = n_4 = 3$, $n_5 = 0$ and the maximum damage function $\Phi = 0.24$. The other initial conditions ($n_5(0) = N$ and $n_i(0) = 0$ for $i \neq 5$) produced $n_1 = n_2 = n_3 = n_4 = 0$, $n_5 = N$ and greater maximum $\Phi = 0.32$.

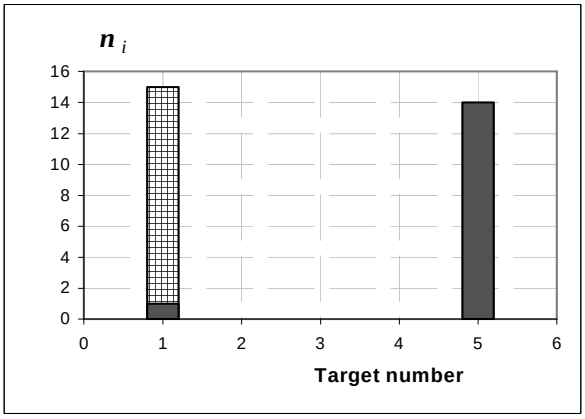

Fig. 4. The outcome of optimization. Target distribution: the hatched area is the initial distribution (n_1 = N), the filled up area is the resulting distribution (n_1 = 1, n_5 = 14).

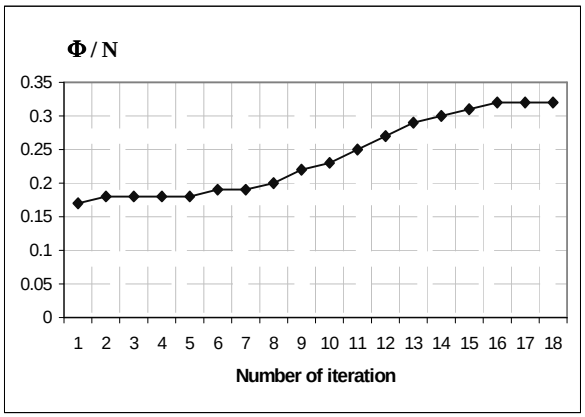

Fig. 5. Dependence of Φ on time (number of iteration) during the optimization process.

Final improvement of algorithm. In this connection, we changed the way of computing p_i, in particular, the form of function $\Phi_i(n)$ that enters in derivative calculations. Function $\Phi_i(n)$ was used to find derivative $p_i = d\Phi_i / dn$ for the whole range $0 \le n \le N$ and then its maximum $p_i^{\max}$ and corresponding point $n_i^{\max}$ were found. In the range $0 \le n \le n_i^{\max}$ we set $p_i = p_i^{\max}$, for other n derivative p_i is determined as before. This change in the computation procedure allowed us to find the global optimum of function Φ. Figures 4 and 5 illustrate this situation in particular case $n_1(0) = N$.

So we determined how many agents shoot at each target when there is centralized control. What is more, the way targets are distributed among agents is simple. The agents keep on shooting at the target as long as the number of such agents remain the same or rises. The other agents change their aims according to their order number and distance to the nearest target.

4. Solving the problem of agents controlled separately

Assignment of targets to autonomously controlled agents is done in the same way as with centralized control. This kind of control regards targets as independent objects that compete with one another for more agents. That is why we represent expression (4) for $W(\boldsymbol{n})$ as sum

$$W(\boldsymbol{u}) = \sum_{i=1}^{m} W_i(\boldsymbol{u}) \qquad (\mathbf{18})$$

where each target has its own optimization functional

$$W_i(\boldsymbol{u}) = (p_i - \bar{p})u_i - \frac{\mu}{2}(N_0 - N + u_i)\left(N_0 - N + \sum_{s=1}^{m} u_s\right) \qquad (19)$$

For an agent to be able to determine his behavior, he should have information about targets, about current sates of all agents, and their intentions to change the behavior. That is why each agent has a table that keeps the state of targets: function Φ_i for the *i*-th target, the number of the agents aiming at it and their order numbers, the derivative of the target function, and the change in the number of agents (the components of vector $\boldsymbol{u}$). The table is filled with data from outside before an agent starts acting.

When moving, agent R_k exchanges information with other agents, receives derivatives p_i, and computes the mean value $\bar{p}$. This allows agents to determine the first approximation for vector $\boldsymbol{u}$. With this in mind, agent R_k directed to the *i*-th target finds $W_i(\boldsymbol{u})$ and, correspondingly, component u_i. After agents determine vector $\boldsymbol{u}$ and put it in the table, the time for decision making comes.

Redistribution of targets among agents is defined by analyzing the components of vectors $\boldsymbol{p}$ and $\boldsymbol{u}$. First a couple of targets whose departure from the mean $|p_i - \bar{p}_i|$ is greatest and components u_i are of opposite sign are processed. The agents aiming at the targets with negative components u_i change targets; specifically, among agents shooting at the *i*-th target the agent with the greatest order number has to change the aim. After deciding to pick a particular target, the agent informs the others about it.

The tables get modified, and the corresponding components of vector $\boldsymbol{u}$ are set to zero. After agents change the aims, the above process repeats.

5. Conclusion

The proposed neural network algorithm makes it considerably easier to solve the problem of decentralized control and decreases the amount of computations in m^2 times, where m is the number of targets. The essence is that the behavior of each agent implies optimization of the whole system. The modification of the functional proposed here allows one to avoid usual problems in selecting the Lagrangian coefficient.

The work is supported by the Russian Foundation for Basic Research (grant No.04-07-90038 and No.05-07-90049)

References

[1] Kaliaev I.A., Gaiduk A.R., Kapustian S.G. Distributed control systems. // M.: Yanus-K, 2002.
[2] Kaliaev I.A. Principles of collective decision making in automatic lift control. // Mechatronics, No.4, 2001.
[3] G.E.Forsythe, C.B.Moler. Computer Solution of Linear Algebraic Systems. Englewood Cliffs, New Jersey, PrenticeHall,1967.
[4] J.J.Hopfield. Neural Networks and physical systems with emergent collective computational abilities. Proc.Nat.Acad.Sci.USA. 79, 2554-2558 (1982).
[5] J.J.Hopfield, D.W.Tank. Neural computation of decisions in optimization problems. Biological Cybernetics 52, 141-152 (1985).
[5] J.J.Hopfield, D.W.Tank. Computing with neural curcuits: A Model. Science 233, 625-633 (1986).
[6] Y.Fu, P.W.Anderson. Application of statistical mechanics to NP-complete problems in combinatorial optimization. Journal of Physics A. 19, 1605-1620 (1986).
[7] T.Poggio, F.Girosi. regularization algorithms for learning that are equivalent to multilayer networks. Science 247, 978-982 (1990).
[8] B.V.Kryzhanovsky, B.M.Magomedov, A.L.Mikaelian. A domaine model of neural network. Doklady Mathematics 71, 310-314 (2005).
[9] B.Kryzhanovsky, B.Magomedov. Application of domain neural network to optimization tasks. Proc. of XVII International Conf. on Artificial Neural Natworks, ICANN 2005. Poland. LNCS 3697, Part II, pp.397-403.