U.P.B. Sci. Bull., Series C, Vol. 72, Iss. 3, 2010 ISSN 1454-234x

HYBRID GENO-NEURO SYSTEM FOR OPTIMIZATION OF CONTROL SOLUTION SELECTION IN MULTI MOBILE ROBOTS

Fadi ISSA[1], Ioan DUMITRACHE[2]

În această lucrare vom prezenta un nou sistem hibrid care combină reţelele neurale artificiale şi algoritmele genetice pentru a găsi strategia de conducere în sistemele multi-robot, deoarece a alege o soluţie specifică de control executată de un robot pentru a-şi îndeplini sarcina cerută poate afecta performanţa celorlalţi roboţi şi astfel, a întregului grup. Soluţia propusă de noi foloseşte noul sistem hibrid cu scopul de a căuta cea mai bună strategie de conducere pentru fiecare robot din grup, pentru optimizarea performanţei globale.

In this paper we present a new hybrid system that combines artificial neural networks and genetic algorithms for control solution selection in multi mobile robot systems, since choosing a specific control solution for a robot to execute its required task can affect the performance of other robots, thus, the performance of the robot group as a whole. Our developed solution uses the new hybrid system in order to search for the best control solution selection for each robot in the group, in a way that achieves the optimum performance of the entire group.

Keywords: genetic algorithm, artificial neural networks, multi-robot groups, performance optimization

1. Introduction

In recent years, interest towards investigating multiple rather than single mobile robots has increased [1], accompanied by technological evolutions of computers, sensors and actuators [2], leading to increased complexity of multi-robot systems. This complexity is reflected in larger team sizes, and greater heterogeneity of robots and tasks. This allowed development of mobile robots to move from specific purpose robots to multi purpose with several tasks abilities robots that can also be used in different environments. A mobile robot can do different tasks (Fig.1); each task can have several possible control solutions or strategies that implement it, and each one of these control strategies has advantages and disadvantages, making these control strategies suitable for some

[1] PhD Student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, fadiissa82@yahoo.com
[2] Prof., Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania

environments or conditions while not suitable for others. This presents the issue of what is the best combination of control strategies that we can assign to the set of robots to give optimal overall performance.

The importance of control strategies assignment grows with the complexity (in size and capability) of the system under study, since there will be too many variables at play. When a task can have several approaches of implementations (several control strategies), the best control strategy must be chosen for each robot to execute its own task. Deciding which control strategy is the best does not depend only on the individual performance of a robot in its specific task, but it also depends on how this control strategy affects the other robots performances, and thus affecting the performance of the whole group.

Since choosing the best strategy for each robot will not always guarantee the best overall performance, a selection mechanism must be implemented to choose what control strategy each robot can follow to fulfil its goal. Usually, the different implementations are made to suit different conditions and environments, or they are provided by different researchers (each one addressing the same task from a different point of view, and leading to advantages and disadvantages for each approach over the other).

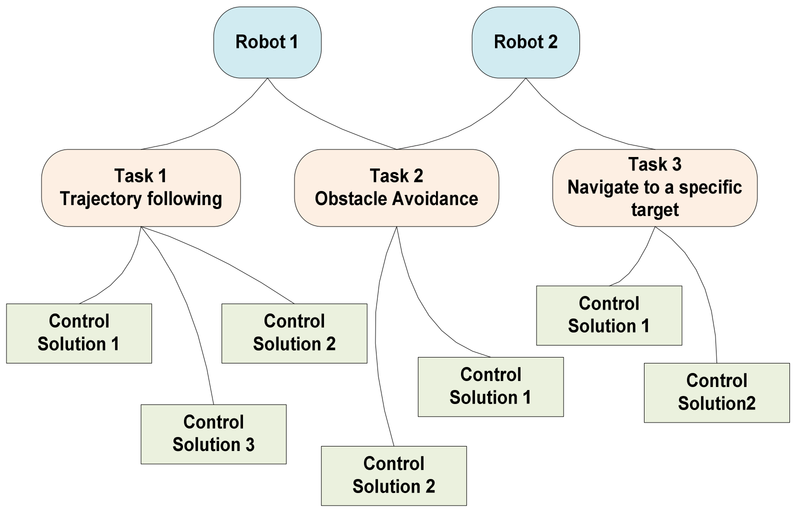

Fig. 1. An example of two mobile robots, each of them capable of doing two tasks, and each task having several control solutions

In a dynamic environment, when the conditions and requirements from a mobile robot can change without prior informing, the robot will find itself puzzled on what control approach should it use to execute its given task. In this paper we present an adaptable system that deals with this situation by using both learning and evolution in order to assign the robots the optimum control solution that best suits the new conditions in order to obtain the optimum overall performance from the whole mobile robots group. The learning part will rely on artificial neural networks, which will learn from current and previous test data. The evolution will be carried by genetic algorithms, which will be used in combination with the ANN in order to evolve a set of inputs (inputs for ANN, they represent control strategies for robots) that, if assigned to robots, will yield a better performance than the set of tested solutions till that moment.

The organization of the paper is as follows: in section 2 we will have a look at the control strategies assignment issue, while section 3 will review currently used approaches that combine ANN and GA, then we describe the structure of our hybrid system that we will use to find the optimum solution. The experimental results are given in section 4 which is followed by an analysis and a conclusion in section 5.

2. Problem statement and current approaches

There are some tasks that would benefit from having multiple mobile robots; therefore the effective use of multiple robots for these tasks is an important goal of robotics research. It is difficult to study the behaviour of a group of mobile robots accomplishing different tasks, since some control strategies will work well in small groups, but will not scale to larger groups [3]. Some strategies will suffer with interference from other robots, while other strategies may be affected by obstacles. We are facing a case in which there is a trade-off between the various approaches to a specific robotic task in a specific environment and specific conditions.

Each mobile robot has several control strategies to choose from for its required task, and when we have several robots working at the same time, choosing the right control strategy for each robot becomes a difficult task. This is because the choice of control strategies for some robots can affect the performance of other robots. An example of such a task with multiple control strategies is obstacle avoidance; it has tens of control approaches, and new ones keep coming. A robot has to select the proper control strategy that implements its required task depending on the conditions and constrains it encounters, and on targeted performance.

Therefore, when a task can be done in several ways, the best way has to be chosen intelligently. In order to do this, there should be provided a metric for each task implementation approach which can be used to compare two candidate approaches for a task to select one of them, thus, control strategies that implement the same task can differentiate themselves from other strategies in several areas (some are suitable for certain environments, some are faster to accomplish the same goal, some use less resources, some are more accurate). Each one of these areas has a performance measurement that allows comparing different strategies according to the same basis.

The areas from which we can obtain performance measurements can be either hardware related (such as the memory needed by each method, or the processing power needed, or the communication overhead and latency), or non-hardware related (such as the accuracy of the method, the time it takes to accomplish a predefined task, or any other measurement that is of interest to the user). In some measurements, the performance of one control strategy can be superior to the others, while for other measurements, this may not be true. The explanation is that algorithms differ among themselves in their ability to scale up in terms of number of robots, number of targets, number of obstacles, and environment fitness.

We think it will be good if each robot chooses the most suitable control strategy according to the varying conditions. When environment changes, there is a need to change the control strategy, if a robot is working alone. This will not be a problem, since it can choose what best suits it to do its required task in a way that gives the best performance (according to the preferred performance measurement). But once several robots act together, things start to become challenging. There is the possibility that one control approach for one robot can affect the performance of another robot, or can prevent it from doing its task at all.

Current efforts to address the control strategy selection issue are small, ad hoc in nature, and relatively little has been said regarding it as a general issue in multi-task multi-mobile-robot systems. In order to know the best combination of control strategies that will give the best overall performance, we cannot rely on the testing of all possible combinations of control strategies. This will not be practical and it will be time consuming. The goal of this approach (which is to find the best combinations of control strategy) becomes more difficult as we add more robots to the group or we add more control strategies that implement the tasks required for the robots.

We propose a hybrid system that tries to address this issue, it is based on Artificial Neural Networks and Genetic Algorithms. Our system will take the whole group performance into consideration to balance the control strategy selection.

Neural networks can be used for prediction with various levels of success. Their advantage can be the automatic learning of dependencies from sample data only, without any need to add further information, and after they finish learning they are able to catch hidden and strongly non linear dependences even when there is a significant noise in the training data set.

3. The Hybrid System

Learning and evolution are two fundamental forms of adaptation. There has been a great interest in combining learning and evolution with artificial neural networks (ANN's) in recent years, leading to development of increasingly powerful neuroevolution techniques [4].

For the control strategy assignment issue, we developed an adaptable intelligent system that consists of two parts: a learning part and an evolving part.

1. The learning part will use artificial neural networks, trying to approximate (as a function) the usually non linear relationship among the different variables of the system (which include the control solutions for each robot).

2. The evolving part will use genetic algorithms to search for better solutions than what have been already tested.

The artificial neural network is trained with the hope to discover hidden dependences among the input variables, so it becomes able to use these dependences (synapse weights) for prediction (estimation) of the mobile robots performance.

Since both genetic algorithms and neural networks are inspired by computation in biological systems, genetic algorithms have been used in conjunction with neural networks in three major ways:

a) to optimize the network architecture [5]. The GA evolves the neural networks topologies for function approximation (including the problem of specifying how many hidden units a neural network should have and how the nodes are connected).

b) to train (set) the weights of a fixed architecture. While most work related to using GA and ANN focuses on only one of the previous two options, some researchers as in [6] investigated an evolutionary approach in which the architecture and the weights are optimized simultaneously.

c) genetic algorithm is used to set the learning rates which in turn are used by other types of learning algorithms [7]. The evolution of learning rules can be regarded as an adaptive process of automatic discovery of novel learning rules [8].

For our research, we investigate a new way of using GA and ANN. Our purpose of using ANN is to help reaching an acceptable solution sooner than waiting for real testing (whether it is by simulation or real world tests). To achieve this, the artificial neural networks must be trained; so it should get an approximation of a function that represents the performance of the system according to the input factors (in our case they will be the control strategies used by the robots, besides some other variables that are not related directly to the robots, like the environment constraints). And by relying on this proximity function, we can get immediately the expected performance of a group of mobile robots if we put the problem conditions as an input to this ANN.

The steps of how this hybrid system works are briefly shown in Fig.2:

1. A random solution is generated (a set of control strategies, one for each robot). Then, this solution is tested to know how it performs.

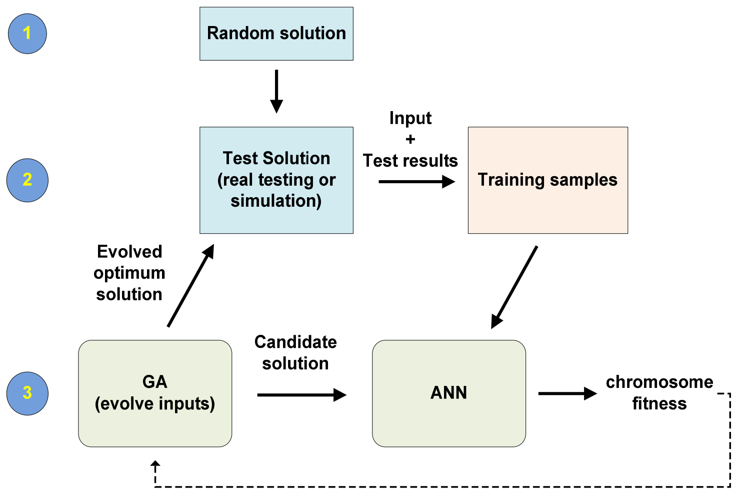

Fig. 2. Steps explaining how the hybrid system works

2. The results from the random solution are transformed to training sample data; this sample data will be used to train the ANN, and it will grow by time (at this stage there is only one training sample).
3. The Genetic algorithm will start to evolve chromosomes of variables as inputs for the ANN, until it reaches a chromosome that gives the best fitness in a population after evolving for a sufficient number of generations. Then, it will send it for real testing (It can also be simulation), and after the test is done, the results are added to the sample training data, so the ANN modifies its weight as a result of the newly modified samples. This process will repeat until the ANN reaches a good approximation of the performance of the mobile robot groups. It is worth noting that the training data are only those that were really tested and got performance results via simulation or real world testing. This is done to ensure that the performance approximation of the ANN is as close as possible to the real values. The chromosomes that are not really tested, and just checked using the ANN, are not added to the training samples.

This training and evolving process can continue until it reaches an acceptable solution, or it meets a stop condition (like reaching or overcoming a specific performance or giving the same result or solution for several consecutive generations).

In spite of that our goal is to optimize the performance in one performance area, the training data can still contain the results of all available performance measurements, this will help to save a lot of time in the future when we need to optimize the same set of mobile robots for a different performance measurement.

Since we cannot know exactly after how many training samples it will be sufficient for the ANN to learn to approximate the function, then the results that we will get from using the hybrid system are not going to be very reliable at first (but their reliability increases as the training samples increase over time). This is because at the beginning of the learning process, and when the number of the training samples is relatively small, the difference between the real test value and the ANN output will be large since the ANN did not have enough training samples, while when more samples are added by time, the difference between the real test results and the estimating results will get smaller and smaller, showing us that the ANN is really learning the approximation function.

There are two types of inputs that can affect the performance of the mobile robot group (Fig.3): controllable variables (such as the control strategy for each robot, and other variables that can be modified by the user) and uncontrollable variables (or perturbation, such as environment variables and other constraints that the user cannot modify), the GA evolves only what is controllable or adjustable (variables that can be set by the user).

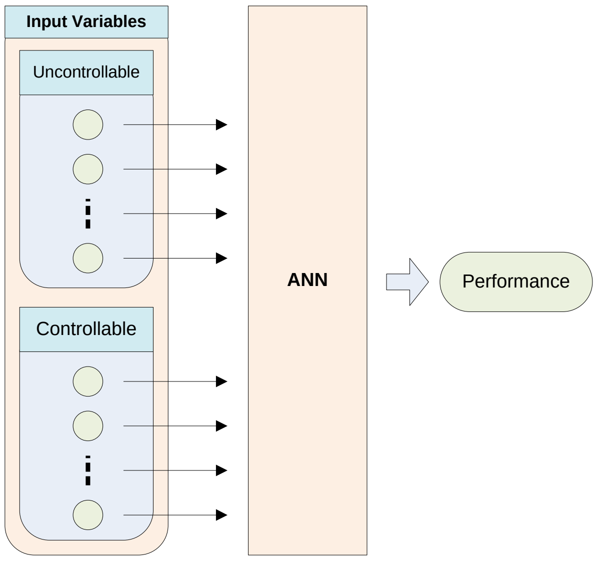

Fig. 3. Types of inputs for the ANN

Optimizing a neural network architecture and neuroevolution [9] (using artificial intelligence to evolve a neural network or optimize its topology) is important because the speed and accuracy of learning and performance are dependent on the network complexity. The performance will depend on several factors which include: the choice of the neurons, the ANN topology and the learning methods [10]. For our approach, each ANN will be composed of 3 layers: one input layer, one hidden layer and one output layer.

The number of neurons in the input layer will be equal to the number of controllable and uncontrollable variables, and they will use a linear transfer function.

The number of neurons in the hidden layer is set according to each case or problem type, and these neurons use sigmoid transfer functions [5] (it is the most common type used with backpropagation learning).

For the learning algorithm, we used backpropagation [11], which is one of the most popular training algorithms for feed forward Neural Networks [12].

Our hybrid approach will use the GA to try to evolve the best possible inputs for the ANN (thus the best possible control strategies for individual robots).

As for the choice of using either online learning algorithm [13] or offline one, we chose the offline one, since there are always new sample data added continuously to existing training data, but we do not want to risk spoiling the existing ANN in case some added samples are corrupted, or cause the ANN not to learn properly, and thus loose what have been learnt till now. This is why we use offline training. So, when we start training the neural network, we use the available training data and make them the training samples for the ANN. And while the neural network is learning, even if new samples are added or existed, we do not add them to the sample data that the neural network is training on. We keep them until the ANN finishes learning on the previous collection of training samples, then, we save this neural network and make it the one used as the representative of the approximation of the system performance function. So, we can use it with the GA system to evolve an optimum solution. And while this ANN is used with the GA system, a new ANN will be trained on a sample or a training data that will include all the sample tests done till now (even the ones we kept from adding to the previous ANN). This way, we can be sure that the ANN used with the GA system is a finished one and not an ANN that is still training.

A neural network can be only representative of one performance measurement. Although it is possible to make one neural network that approximates several performance measurements, this will come with the cost of more complex topology or architecture of the neural network, since it will have more outputs because of the increased number of performance measurements which are, by the way, the outputs of the ANN. It is a good practice when we have more than one possible performance measurement to include also the result of these performance measurements in a separate training sample for a different ANN. By following this practice it will save us a lot of testing time when we want to change the type of the performance measurement that we want to optimize. So, instead of starting to collect sample data from the beginning, we will rely on the sample data taken during the tests for other neural networks that had the goal of optimizing different performance measurements. By choosing several neural networks, one for each performance measurement, we will simply use the relevant ANN when we need to optimize the performance of the group of robots for a different performance measurement.

The GA component will assign a specific gene to each possible variable, and when it will evolve the inputs of the ANN, it will use one of these variables for each input of the ANN, even if its value is not yet presented in the training samples (Fig 4).

The GA will be used to evolve the inputs of the ANN. And since these inputs can be either controllable or uncontrollable variables, and the controllable ones are the only ones that we can change, then the GA will only evolve the controllable variable which are, in our case, the control strategies for each robot. These control strategies will be represented as chromosomes. Then a genetic search is applied on the encoded representation to find a set of control strategies that gives the best overall performance according to the ANN approximation function.

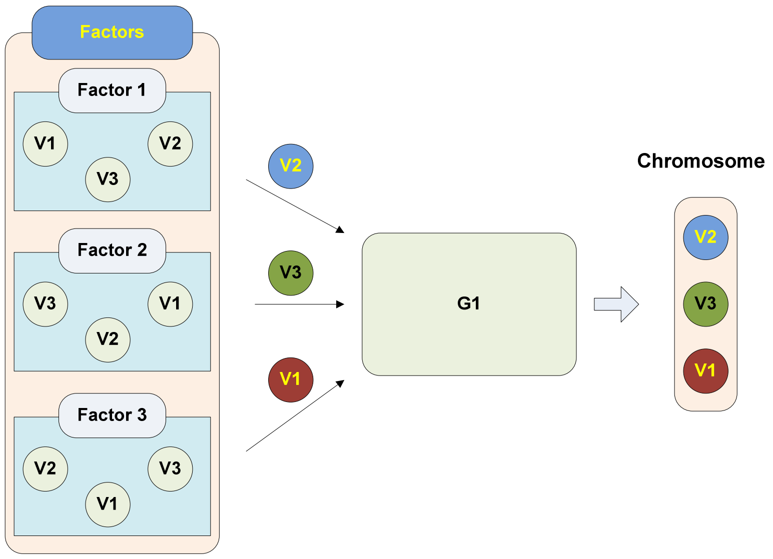

Fig. 4. GA generating a chromosome from available variables that affect the performance of robots

For the topology of the ANN, and more specifically, the number of neurons in the output layer, we noticed that for smaller and simpler problems, one neuron in this layer was sufficient. This is for when the performances of control strategies of the group's robots are not very dependent on each other. In this case, this neuron represents the performance of the whole group of robots. (Fig.5).

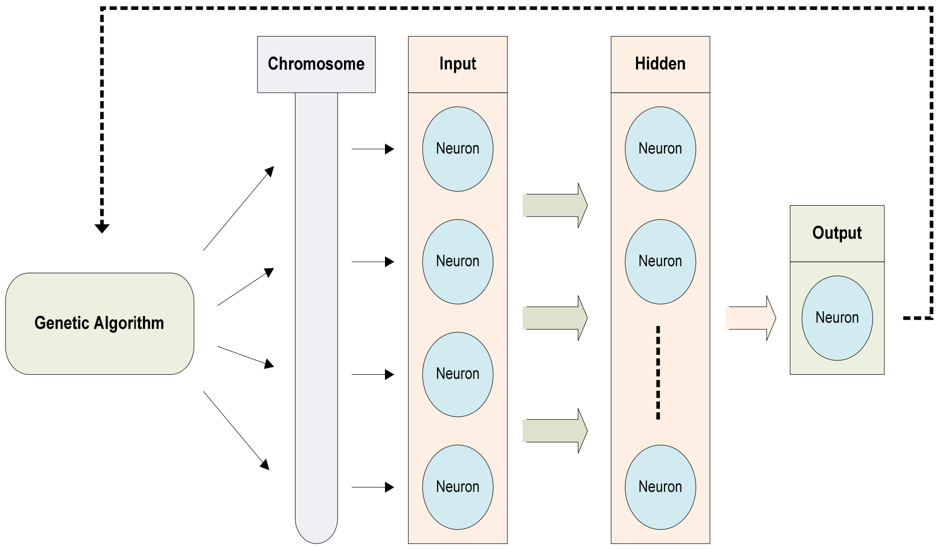

Fig. 5. One neuron in the output layer used to represent the whole group performance

In the case of more complicated problems (when control strategies performances were dependent on each other), we found that it will save us a lot of training time thanks to lower number of neurons if we splitted the neural network into two networks (Fig.6).

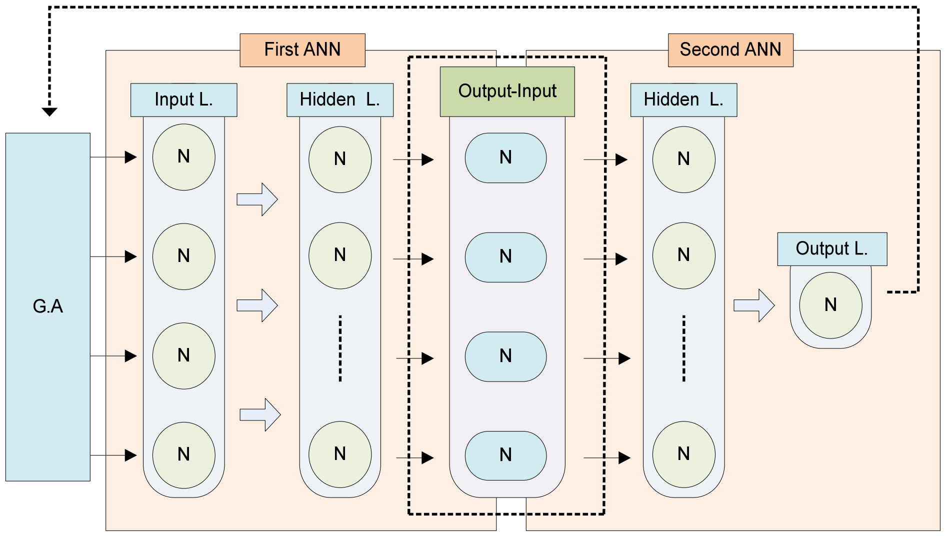

Fig. 6. Two ANNs, one for individual performances and one for entire performance estimation

(We believe that this is because, in the simple case, it was easier to approximate a function that represented the whole performance of the group. While for the more complex problems, this neural network had to represent in its structure several complicated functions like the performance for each robot, and above that, the performance of the whole group, which is calculated based on the individual performance of each robot).

Therefore, separating the ANN into two neural networks will make it easier for each network to learn. The first ANN will be used to estimate the individual performance for each robot in the group, and the second ANN will be used to estimate the entire group performance depending on individual performances as input. In this case, the first neural network had an output layer with number of neurons equal to the number of robots, which means that each neuron represented the individual performance of the related robot, and the other neural network will represent the function that calculates the performance of the group based on the individual performance of each robot. This means that the second neural network will have an input layer with a number of neurons equal to the number of robots, and its output layer will have only one neuron which will represent the performance of the entire group.

We need the second ANN only in case the entire group performance is calculated from the individual performances in a nonlinear way. If it is a fixed function of the other individual performances (like the sum of them, as a simple example), then the second ANN can be replaced with a well defined function that calculates the group performance based on individual performances. In case it is not linear, or it cannot be represented easily, the second ANN will have the individual performances as inputs, and it will have one output neuron that represents the entire group performance. The output layers of both of these networks will use linear transfer functions.

In both cases, GA evolves for several generations until it reaches a solution (which cannot get better by evolution), and that will be considered the best solution that this hybrid system can give. The solution will be sent to be assigned to robots for real testing or for simulation. After simulating this best solution, its results should be better than all previous training samples, otherwise, this means that the hybrid system cannot evolve more and it reached the best overall solution. But if it was better than all the training samples, then this means that we are advancing towards finding the optimum solution and by adding the results of this specific test to the training data samples, it will help the ANN to approximate the performance function of the group of mobile robots more accurately.

4. Tests and Evaluation

We evaluated the hybrid selection system using simulation. We made a test of a group of ten mobile robots, each one trying to navigate from one point to another, then return to the first point. Each of these robots has three control strategies that it can use to accomplish its task, and each strategy uses different speed for motion. We defined the goal of the whole group to be that all robots should reach the other point and return to the first one in the shortest time possible. Therefore, our performance measurement was time, and we wanted to minimize it. When we started the simulation, we let the system run until there were 4 training samples. Then, the ANN was trained according to these samples. After learning was finished, the GA started to evolve a population of inputs for the ANN; the GA population had ten chromosomes, the crossover method was cross-point, and the mutation rate was set to 30 percent. After evolving for more than 20 generations, it kept reaching the same solution, which was not the optimum one we were searching for, so we let the system to continue running and increased the training samples to 8 samples. After the ANN learned its approximation function based on these 8 samples, the Ga could reach a better solution than the previous one (after 37 generations), then it could not give a better one, and that was still not the optimum we expect. So we waited again till the training samples reached 16 samples, for 16 samples, the GA could reach the optimal solution after 45 generations, and it was indeed the optimal solution we were expecting. Increasing the training samples after that did not affect the obtained result. The optimal solution was not existent in any of the training samples data.

This proves our point that we can use this hybrid system to reach a solution better than the ones available, without necessarily being tested previously. This was done in a relatively short time, after collecting only 16 training samples for the ANN, while for comparison, when we tried to reach the optimum solution using only genetic algorithms without the learning capabilities of the ANN, while keeping the same settings for the GA (the same mutation rate and the same population number), it took it 44 generations to reach the same optimal solution, and considering that each generation has 10 chromosomes, this means there were 440 sample tests, compared to 16 samples needed for the ANN to reach this same result in this particular problem, which is considered a great time saving. This was for a simple case. For more complex cases it could reach the optimum solution after hundreds of generations on a training data of tens of samples. In both cases, it could reach solutions that are not present in the training data, and which delivered better performance than all the previously tested training data.

In our tests we chose time as a performance measurement, but there can be other measurements. The concept is still the same, optimizing the control strategy assignment in order to get the optimum group performance.

5. Conclusion

This paper aim was to introduce a new approach for control solution selection in multi mobile robot systems to obtain optimal performance for the whole mobile robots group.

The hybrid system we developed uses genetic algorithms to evolve data sets that are sent after that to an artificial neural network that will learn based on these new training data and then develop to an approximation function that can predict the performance of a robot or a set of robots according to an input set of variables (control solutions).

Our approach proved efficient in finding optimum results based on ANN approximation without simulating them or testing them. This saves a lot of time because testing or simulating take a lot more time (since it is restricted by real time conditions), while the ANN training and GA evolution do not have that restriction, and they are only bound by the computer power.

The experiments we did show that using this hybrid approach of ANN and GA to evolve control assignment solution can lead to a solution that is considered acceptable (compared to other approaches) in a relatively short time.

We plan on improving our hybrid system in the future to also use the GA in optimizing the ANN topology and learning parameters. We think this will help achieving better results with smaller costs of training and evolving.

R E F E R E N C E S

[1] *P.G. Brian, J.M. Maja*, A Framework for Studying Multi-Robot Task Allocation, Conference Paper, Multi-Robot Systems: From Swarms to Intelligent Automata, **Vol. II**, pages 15-26, the Netherlands, 2003

[2] *C. Cote, D. Letourneau, F. Michaud, J.M. Valin, Y. Brosseau, C. Raievsky, M. Lemay, V. Tran,* Code reusability tools for programming mobile robots, Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2004), **Vol. 2**, On page(s): 1820-1825, 2004

[3] *A.D. Gustafan, V.P. Rapaka, S. Deloac*, A comparison of algorithms for teams of robots, IEEE international conference on systems, man & cybernetics: 7 vol, The Hague, Netherlands, (10-13 october 2004)

[4] *D. Floreano, P. Du¨rr, C. Mattiussi*, Neuroevolution: from architectures to learning, Evol. Intel.1:47–62, Springer-Verlag 2008

[5] *S.G. Mendivil, O. Castillo, P. Melin*, Optimization of Artificial Neural Network Architectures for Time Series Prediction Using Parallel Genetic Algorithms, Soft Computing for Hybrid Intelligent Systems, Pages 387-399, Springer Berlin / Heidelberg, 2008

[6] *B. Zhang, H. Mühlenbein*, "Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor", Complex Systems, pages 199-220, 1993

[7] *A. Radi, R. Poli*, Discovering efficient learning rules for feedforward neural networks using genetic programming, Recent advances in intelligent paradigms and applications, Pages: 133-159, Physica-Verlag GmbH Heidelberg, Germany, 2003

[8] *X. Yao,* Evolving artificial neural networks, Proceedings of the IEEE, **Vol. 87**, No. 9, September 1999

[9] *F. Gomez, J. Schmidhuber, R. Miikkulainen*, Accelerated neural evolution through cooperatively coevolved synapses, Journal: Machine Learning Research 9:937–965, 2008

[10] *S. Haykin*, Neural Networks and Learning Machines, Prentice Hall; 3rd edition, 2008

[11] *R. Rojas*, Neural Networks, Springer-Verlag, Berlin, 1996

[12] *Dongsun Kim, Hyunsik Kim, and Duckjin Chung*, A Modified Genetic Algorithm for Fast Training Neural Networks", Advances in Neural Networks, **Vol. 3496**, Pages 660-665, Springer Berlin/ Heidelberg, 2005

[13] *D. Saad,* On-Line Learning in Neural Networks, Cambridge University Press; 1st edition, 1999.