Modifying Neuro Evolution For Mobile Robotic Behavior Development

Sekou Remy and Ashraf Saad

School of Electrical and Computer Engineering,Georgia Institute of Technology,
Savannah, GA 31407, USA

Abstract. This work examines the effect of two modifications of typical approaches to evolving neuro controllers for robotic behaviors. First evolutionary methods were constrained to modify only one element of the population, the only element to be evaluated in the robot. Secondly the algorithm was allowed to incorporate genotypes provided by external sources. These modifications were evaluated through the use of a mobile robot simulator. Each was allowed to evolve in an arena that allowed it to interact with other robots. Experiments were conducted to investigate the effect of sharing genotypes and their corresponding fitness among homogeneous robots - the robots differed only in the initial random phenotype. The experiments showed that the ability to incorporate successful genotypes from others increased the rate at which evolution progressed. Communication of good genotypes allowed behaviors to get fitter faster, and made small initial population sizes feasible.

1 Introduction

Autonomous mobile robots have long been desired by the research community. The benefits of interaction between a robot and other external, possibly human, agents has been displayed for instance, see [5]. The ability to communicate has been shown to provide great advantages for teams of robots. [3] and [19], among others have highlighted cases where communication assisted teams of robots in accomplishing given tasks. It is not too far of a stretch to fathom that the sharing of knowledge or information can also assist individual robots in a team to learn to follow a wall or execute some other individually oriented task.

Communication, while a useful component, does not provide autonomy, in any degree by itself. A necessary and sufficient skill is the ability to decide what to do. The ability of a mobile robot to make decisions is implicitly based on the premise that there are choices, two or more options that can be selected, each with different merit.

Most work in robotics assumes that there is a preexisting set of choices to choose from and a preexisting set of algorithms to make these choices. These assumptions are by no means oversimplifications of the challenges of robotics tasks; e.g., path planning, but in they do however produce brittle robots [12] that do not adapt to or take into consideration changes in their environments or changes within themselves. If the robot's creators are able to identify every possible situation that the robot will encounter and

Remy S. and Saad A. (2005).
Modifying Neuro Evolution For Mobile Robotic Behavior Development.
In *Proceedings of the 1st International Workshop on Multi-Agent Robotic Systems*, pages 155-164
DOI: 10.5220/0001192901550164

devise solutions for each of these, then brittleness would not be an issue worth mentioning. But if the robot exists in a dynamic, unknown, or unpredictable environment, has noisy actuators, or sensors that produce noisy measurements, then adaptation and learning are qualities that are necessary to allow the robot to perform despite the presence of challenging limitations.

It must be stated that autonomous decision making is closely linked to the desired outcome or goal of the robot's operation. Not only is it important to know the choices, the effects of choices, and how to select between choices; it is also important to know the desired goal or outcome as well. When dealing with mobile robotics it is helpful to borrow a page from [1]; the choices that are to be made are choices between different *behaviors* that the robot can manifest.

The ability to learn two-dimensionally, that is to learn how to execute a particular behavior, as well as to learn when to choose to execute that behavior, allows a robot to act autonomously in the face of dynamic environments, sensors, and actuators. This work investigates enhancements to one approach to learning behaviors, the neuro controller.

2 On Evolving a Neuro Controller

2.1 What is a Neuro Controller?

This research seeks to evolve a neuro controller that allows the robot to display a desired behavior, in this case the wall following behavior. A neuro controller is an Artificial Neural Network (ANN) structure similar to that shown in figure 3, that connects neurons in such a manner that when the correct weights and biases are selected the controller produces the desired output from its inputs.

2.2 What is a Genetic Algorithm?

Genetic algorithms (GAs) are often viewed as function optimizers [20]. They are also heralded as tools that can provide robust and powerful adaptive search mechanisms [16]. These two seemingly disparate definitions are two sides of the same coin. GAs are a family of computational models that are loosely based on Darwinian evolution. GA implementations begin with a typically random population of potential solutions, and through a process of mutation and crossover, create new potential solutions. Each solution is evaluated and given a fitness value, which is a measure of how well that solution scored. As the algorithm is executed, the population as a whole evolves becoming fitter; the search for the fittest element becomes the selection of the fittest element in the final population. The fittest element is also the solution that optimizes the fitness function. GAs have been applied to many types of problems including some attempts for the evolution of neuro controllers [2, 4, 7].

2.3 Why Evolve the Solution?

The weights and biases are typically derived through some training method. Most training algorithms are based on a gradient descent method. Because of this they can get

trapped in a local minimum and sometimes, as in the case of multi modal or non differentiable error functions, these methods are incapable of finding the global minimum [21]. Since the GA is not crippled by such, it can be applied to determine the weights.

3 Modifications

3.1 Change 1

This work proposes the modification of the family of GAs that allows a single genotype to be actively evolved. This genotype will always be selected as one of the parents for the crossover operation and one of its offspring selected to replace it. This genotype will also be the subject of mutation and any other single parent evolutionary technique. This modification was developed for application to neuro controller evolution but can be applied to domains with similar needs such as ANN classifiers [13, 10].

3.2 Motivation

Traditional GA approaches employ random selection of the solutions used in the processes that evolve the population. This creates a problem since the GA changes a population in a random manner as generations evolve. There is no way to monitor a solution's progress over generations. This is relevant because each solution is directly mapped to a phenotype that creates a neuro controller. Traditional approaches essentially evaluate random attempts at creating the best controller. While this may work well for applications that are abstract in nature, when applied to a real robot it is problematic since the changes can have catastrophic consequences especially when the robot is interacting with external, possibly human agents. We believe that a better approach would be to continuously modify an existing behavior in the search of better performance. This approach does allow less variation but it reduces the possibly negative effects of unevaluated genotypes.

3.3 Change 2

This work also proposes a method that allows genotypes to be added to a population from an external source.

3.4 Motivation

If the GA is able to incorporate other genotypes that are very fit, the population will get fitter faster as the search is no longer completely random. Moreover, if multiple GAs use the same fitness function to evaluate genotypes and are operating over the same solution space then the search should be faster since they are operating in parallel and sharing the successes, in the form of high fitness genotypes that they encounter.

4 Experimental Setup

4.1 The Behavior

In our experiments, robots are expected to learn the wall following behavior. This means that upon completion of a learning period that the robot will be able to traverse the outline of the arena without bumping into avoidable objects or getting trapped in corners or other features of the arena. The expected path of the robot neither has a preference for clockwise nor counter clockwise direction since both are acceptable in this case.

4.2 The Robot

This work was based on the capabilities of the Khepera robot. This robot was selected because it was easy to interface with and did not possess any highly specialized sensors or actuators. The devices found on this robot are common to many other mobile robots. For this work eight proximity sensors, two actuator motors, and the communication turret were used. The orientation of these sensors is shown in figure 1.

To learn to follow a wall, the robot will have to develop a mapping between the proximity sensors readings and the actuator values. The mapping should have the effect of allowing the robot to perform actions such as turn away from obstacles or drive forward if there is no obstacle. The robot will have no cognitive layer so there is no stored representation of the world or walls for example, but the behavior will be displayed in a reactive manner based on the mapping from proximity sensors to left and right actuator motors.

4.3 The Implementation Environment

The experiment was conducted completely using MatLab. The neural network was provided using the built in MatLab toolbox and the evolutionary algorithm was a modified version of the GAOT toolbox [9]. Each robot operated in an arena that was populated by other robots with which it could communicated, sharing genotype and fitness information if permitted. A snapshot of one such arena is shown in figure 2. The simulator used was the KIKS simulator [18]. The arena and each robot in it were generated by individual instances of a KIKS simulator. In this work, each of these instances was initiated on a single computer, Pentium II class or newer, that had at least 128MB of RAM.

4.4 Communication

The simulated robots communicated with each other using a simulated wireless communications channel. This capability mirrors those afforded to the real Khepera robot by its communication turret, complete with bandwidth limitations, interference and message segmentation and reassembly.

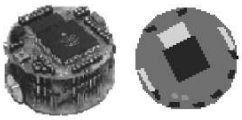

Fig. 1. On the left an image of the Khepera robot base, on the right a sketch of the same showing wheels sensors and turrets

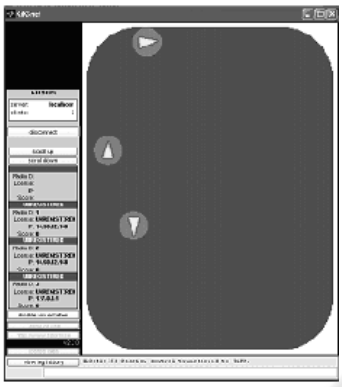

Fig. 2. An arena with three robots

5 Implementation

5.1 The infrastructure

The genetic operations of selection, mutation and combination operated on the genotype. When the genotype's fitness was to be determined, it was decoded into a phenotype - an instantiation of a neurocontroller - that was evaluated in the robot. During the process of evaluation, the robot interacted with its environment. The robot's motion was controlled only by the phenotype provided to it. If the robot was permitted to do so it collected genotypes that were transmitted to it. When evaluation was complete it returned these along with the current genotype which then sported an updated fitness value. The following sections discuss attributes of these components in closer detail.

5.2 The Neuro Controller

The input layer of the neuro controller contained eight input nodes, one node per sensor, while the output layer contained two such nodes, one for each motor. There were five nodes in the hidden layer. These nodes are connected first by forty synapses from input to hidden layer, and then ten from hidden to output layers. These numbers result from a design choice that fixed the number of hidden nodes, hence constrained the structure of the neural network that served as the neuro controller. The network was fully connected and did not utilize a bias value. The neuro controller being evaluated at any given time is the current phenotype. Each of the input nodes was connected to a sonar sensor and the output nodes to the actuators, one to the left and the other to the right motor.

5.3 The GA Parameters

The genotype was comprised of a vector of signed real numbered elements, one for each synapse weight and, one extra for the fitness of that genotype. The initial populations were comprised of randomly generated genotypes. The GA used a value of 1e-6 for epsilon, the resolution and a normal geometric selection function with parameter p = 0.08. All other parameters are included in tables 1 and 2

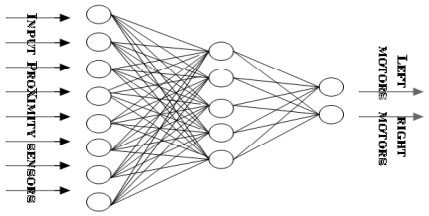

Fig. 3. The neurocontroller

Table 1. Mutation parameters

Mut. Types	Boundary	Multi non-uniform	Non-uniform	Uniform
Iterations	4	6	4	4
Delta maxgen.	0	100	100	0
shape param.	0	3	3	0

Table 2. Crossover parameters

Crossover Types	Arithmetic	Heuristic	Simple
Iterations	2	2	2
Retries	0	3	0

5.4 The Fitness Function

The fitness function for this application was devised such that the robot would be rewarded for having one of its sides close to the wall without touching the wall, and for having no obstacle in front of it. Close was defined as a sonar sensor reading between seven hundred and nine hundred and ninety nine units. The reward was applied when the side closest to an obstacle was within these prescribed bounds. There was also a bonus reward applied for robots that displayed more forward motion than backward motion. If the sonar abutted an object, the sensor would produce a reading very near to 1024. The aim was not to collide with objects, so sonar values larger than 999 were not given a reward. We do not claim that this is the perfect fitness function. However, this one has been demonstrated via trial and error, to perform well for this application, and hence was selected. There is much that can be said about the “art of selecting a fitness function” please refer to the work of [6, 15, 11, 8] for further discussion of this issue.

5.5 The Experiment

Twelve arenas of three robots each were set up. The robots in each arena exercised the same communication strategy and had the same maximum generation size for the GAs. These experiments were run for generation sizes increasing from ten to thirty, and also with and without the aid of communicated data. When communication was not allowed, the data was transmitted and then destroyed by the reciever upon reciept, so each arena and robot had the same communications load.

6 Results

A cursory glance at figure 4 may lead one to surmise that the modifications did not produce good results. Closer inspection shows that this is not the case. Table 3 shows

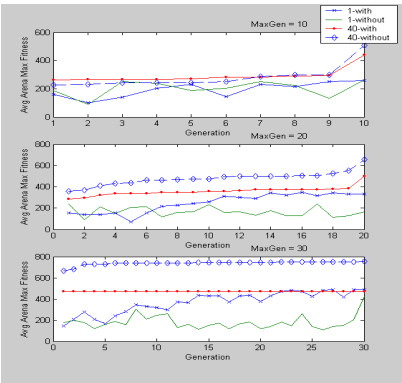

Fig. 4. Avg. arena fitness vs. generation size of robots w/ & w/o the benefit of communication

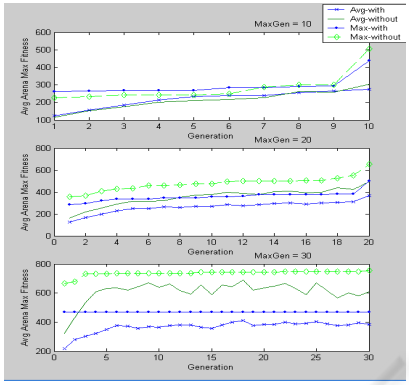

Fig. 5. Avg. arena fitness & max. arena fitnesses vs. generation size using basic GA

the slope, intercept and residue of the trend lines of the graphs. The emboldened numbers indicate the use of data communicated from other robots. The apparently excellent performance of the unchanged evolved neuro controller seems to be a result of good initial populations, which are randomly generated. The rate of improvement, which is captured by the slope value, is consistent in all cases that change #1 was not made or both modifications were made.

Table 3. Slope, intercept and residue data for trend lines of graphs in Figures 4 and 5

	Modified GA			Basic GA						
				Avg of arena average			Max of arena average			
	Slope	Intercept	Residue	Slope	Intercept	Residue	Slope	Intercept	Residue	
10	**13.897**	**117.2**	**97.826**	**15.507**	**133.15**	**44.583**	**11.899**	**226.56**	**115.91**	
20	**13.305**	**110.08**	**155.59**	**8.1603**	**178.32**	**99.574**	**6.186**	**293.78**	**101.7**	With Comm
30	**10.48**	**207.57**	**236.97**	**3.1925**	**317.49**	**153.19**	**9.30E-15**	**469.33**	**4.90E-13**	
10	5.5333	171.8	162.5	17.817	114.2	34.055	20.745	169.07	162.74	
20	-1.8739	185.56	184.2	12.383	224.87	130.56	9.899	373.98	117.76	No comm
30	0.51131	171.57	352.82	2.8923	564.15	368.73	1.4446	714.73	352.82	

There was definitely a benefit to communication in the cases where the initial population size was one. This benefit disappeared when the population size was increased to forty. Figure 4 shows data for the average maximum arena fitness of the populations as a function of generation. Figure 5 shows this data and the average of the average arena fitnesses, giving a better view of the increasing fitness of all the members of the population over generations.

Figure 6 shows the paths of two different robots. The panel on top shows a robot with a high fitness, and the one below it, a robot with a low fitness value. These images give a sense of how the fitness function can relate to demonstrated behavior.

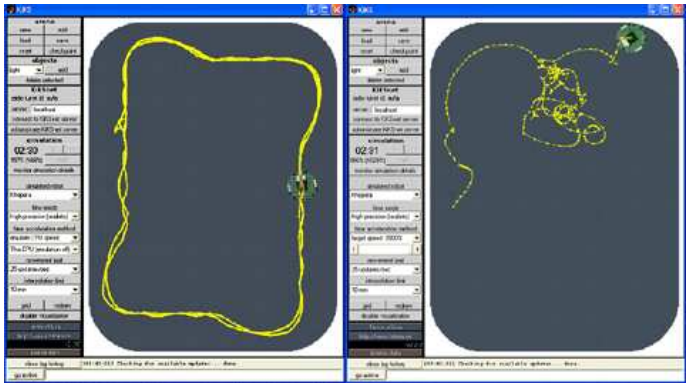

Fig. 6. Paths of two robots with displaying neurocontrollers of different fitnesses

7 Future Work

These experiments while quite promising were very time consuming. There were time gains that are yet to be fully investigated in the data sharing cases, but these experiments were completed on the order tens of hours; the longest taking just over thirty hours. Such long durations seem to be consistent when neuro controllers are evolved, and they make large numbers of experiments prohibitive. Future work will include an implementation of this work with fast genetic algorithms. Still on the challenges of genetic algorithms, the advances made using subpopulations could be incorporated into future work. Neuro Evolution of Augmenting Topologies [17] is also a promising research avenue.

A homogeneous set of robots were used for this research. The robots were identical in their sensor and actuator collection as well as in their phenotype structures. Interesting opportunities lie ahead for investigating non-homogeneous cases. This would hint at the realm of interspecies communication and learning.

Close to the core of this work many potential modifications of interest remain. Only weights of the neuro controller were evolved, the structure could have also been evolved, perhaps even in concert with the weights [14]. The size of the arenas and the number of robots operating in those arenas are also parameters of interest.

8 Conclusion

In robotic applications that evolve neuro controllers for robot motion, evaluation of genotypes mean that the robot's operation either has to be simulated or actually executed. Since in many cases it is not feasible to simulate, actual operation is often necessary. Testing unevaluated geneotypes in a mobile robots is akin to testing experimental planes. There is a great deal of uncertainty, but each test provides great rewards, even in failure. Reducing risk is an important goal in both of these efforts and we believe that making changes to only one genotype allows the mobile robot do so. The risk reduction results because the robot should demonstrate greater consistency at each evaluation since it is likely that most of that genotype remained unchanged.

As those familiar with statistics would expect, reducing the initial population size of a traditional GA to one produced poor results. This work shows that this setback is mitigated by endowing the GA with the ability to share and incorporate some information about genotypes with other independent GAs operating on identical genotype structures and fitness functions. Thus the modifications suggested in this work can be used to address these pitfalls.

Also, since each robot can be initialized with a single genotype, it can be seen that it would be easy to begin the neuro evolution process with a genotype that was the result of prior evolution. This allows a robot to receive a jumpstart. The evolution process does not have to begin randomly, nor does it have to wait until the end of a generation. This is the quality that allows these modifications to display the characteristic of knowledge transfer across lifetimes. This also opens the door for an external agent to interact with a robot's neurocontroller, and to do so at any time.

Learning is a paradigm that is manifested in individuals over their lifetimes. Genetic algorithms function as a proxy for learning since they are manifested by populations over generations; they do this essentially by searching through a solution space. This work modifies the GA so that it more closely imitates learning.

We have shown the benefit of teams of robots sharing information in accomplishing a common task, that of behavior development. The research was performed using a simulated robot. The code used to implement it can be used directly on a real Khepera robot without functional modification. The simulator used was firmly grounded in reality, mimicking many of the challenges that real robots face, such as sensor noise and wheel slippage. This is of import since it can then be expected that these results will hold for a real robot as well. Plans are underway to do so.

Despite the promising results and the potential for future work, there are still some major limitations of this line of research that have not been fully acknowledged by other researchers. Foremost the robot, despite all its evolutionary capability, is still an object of reaction. It has not yet been granted the ability to deliberate or even remember. It acts based on sensor values and were any of those values to be severely flawed the robot would no longer display the desired behavior. Furthermore, while this work places no limitation on the number of behaviors that could be evolved, there is no infrastructure to control when those behaviors are displayed. [12] has done work to use GAs to address this issue with the evolution of behavioral policies. This is of interest and will be investigated to see if this or other methods of deliberation and decision making are appropriate. These are the issues for autonomous robotics; neuro evolution of behaviors is but a tool in a robotician's toolbox.

References

1. Arkin, R. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge.
2. Balakrishnan, K. and Honavar, V. (1996). Analysis of neurocontrollers designed by simulated evolution. In *Proc. of the IEEE International Conf. on Neural Networks*.
3. Billard, A. and Dautenhahn, K. (2000). Experiments in social robotics: grounding and use of communication in autonomous agents. In *Adaptive Behaviour*, volume 7.
4. Cliff, D., Husbands, P., and Harvey, I. (1993). Analysis of evolved sensory-motor controllers. In *Proc. of the Second European Conf. on Artificial Life*.

5. Connell, J. and Viola, P. (1990). Cooperative control of a semi-autonomous mobile robot. In *Proc. of the 1990 IEEE International Conf. on Robotics and Automation*.
6. Duvivier, D., Preux, P., C.Fonlupt, Robilliard, D., and Talbi, E.-G. (1998). The fitness function and its impact on local search methods. In *Proc. of the IEEE International Conf. on Systems, Man, and Cybernetics*, volume 3.
7. Floreano, D. and Mondada, F. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In *From Animals to Animats 3: Proc. of the Third International Conf. on Simulation of Adaptive Behavior*.
8. Hoque, T., Chetty, M., and Dooley, L. (2004). An efficient algorithm for computing the fitness function of a hydrophobic-hydrophilic model. In *Proc. of the IEEE International Conf. on Evolutionary Computation*, volume 1.
9. Houck, C., Joines, J., and Kay, M. (1995). *A Genetic Algorithm for Function Optimization: A Matlab Implementation*. NCSU-IE TR 95-09.
10. Lippmann, R. P. (1989). Pattern classification using neural networks. In *IEEE Communications Magazine*.
11. Maher, M. and Poon, J. (1995). Co-evolution of the fitness function and design solution for design exploration. In *Proc. of the IEEE International Conf. on Evolutionary Computation*, volume 1.
12. Nehmzow, U. (2002). Physically embedded genetic algorithm learning in multi-robot scenarios: The pega algorithm. In *Proc. of the Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, volume 94, Lund, Sweden. Lund University Cognitive Studies.
13. Ripley, B. D. (1994). Flexible non-linear approaches to classification. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks. Theory and Pattern Recognition Applications*, Berlin. Springer Verlag.
14. Saxena, A. and Saad, A. (2004). Genetic algorithms for ann-based condition monitoring system design for rotating mechanical systems. In *9th Online World Conf. on Soft Computing in Industrial Applications*.
15. Sorokin, S. N., Savelyev, V. V., Ivanchenko, E. V., and Oleynik, M. P. (2002). Fitness function calculation technique in yagi-uda antennas evolutionary design. In *Proc. of the IEEE International Conf. on Mathematical Methods in Electromagnetic Theory*, volume 2.
16. Spears, W. M., DeJong, K. A., Baeck, T., Fogel, D. B., and deGaris, H. (1993). An overview of evolutionary computation. In *Proc. of the 1993 European Conf. on Machine Learning*.
17. Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. In *Evolutionary Computation*, volume 10, Cambridge. MIT Press.
18. Storm, T. (2004). *KIKS is Khepera Simulator 2.2.0. (http://www.tstorm.se/projects/kiks/)*.
19. Wagner, A. and Arkin, R. C. (2003). Internalized plans for communication-sensitive robot team behaviors. In *Proc. of the IEEE/RSJ Conf. on Intelligent Robots and Systems*.
20. Whitley, L. D. (1994). A genetic algorithm tutorial. In *Statistics and Computing*, volume 4.
21. Yao, X. (1999). Evolving artificial neural networks. In *Proc. of the IEEE*, volume 87.