Co-evolutionary Emergence of Traffic Navigation

Learning Through Evolved Behavior Layers

Matt Fair
University of Minnesota Morris
600 E. 4th Street
Morris, MN 56267
fair0065@mrs.umn.edu

Eugene Michtchenko
University of Minnesota Morris
600 E. 4th Street
Morris, MN 56267
mich0272@mrs.umn.edu

ABSTRACT

In this paper we present the *traffic circle problem* which involves evolution of traffic navigation among autonomous robots. This problem can be broken down into smaller sub problems which we call behavior layers: *path planning*, *object avoidance*, and *unidirectional navigation.* For the purpose of this paper we focused on *object avoidance* and *unidirectional navigation.*

Because this problem is very complex and dynamic, it presented a large number of influential factors that needed to be broken up and analyzed individually. To fulfill the first behavior layer, we replicated the object avoidance and simple navigation experiments developed by Nolfi and Floreano [4]. The results obtained from multiple trial runs were analyzed in order to gain better understanding of the genetic algorithm (GA) properties. The two main GA parameters that we focused on were population size and degree of mutation. This analysis helps determine the degrees of freedom and effective sizes of the two variables.

In our experiments we introduced a concept called *zones*, which are used to promote unidirectional navigation through evaluation criteria, the fitness function. The first experiment, called *directional zones*, demonstrated that the robots learned how to minimize the risk of going the wrong direction which maximized their fitness value, but did not achieve the desired behavior of unidirectional navigation. Second experiment, *sequential zones* exhibited an elementary level of unidirectional flow, but the flow would break when faster moving individuals ran into slower moving ones, thus causing congestion.

The results yielded from the the analysis and experiments help to explain and eliminate a number of complex elements introduced by the *traffic circle problem*, which in turn could help improve future work on co-evolutionary strategies.

Keywords

Neural Networks, Evolutionary Robotics, Genetic Algorithm, Traffic Circle Problem, Parameter Optimization

1. INTRODUCTION

Evolutionary learning is a new revelation in the field of autonomous robotics. Evolutionary robotics simulates Darwin's natural selection to evolve behavior that closely fits the evaluation criteria. This is done by selecting well-performing individuals and replicating them with a certain level of mutation to create the next generation where the process will be repeated. This technique offers versatile and robust solutions for unpredictable environments as well as bypasses the difficulties that may have not been originally anticipated by the designer. Artificial evolution can thrive on complexities of dynamic environment and present rich solutions. Challenges arise when designing a control system for simultaneously co-evolving multiple robots that often tend to function independently and to take actions in their own best interest, thereby increasing the entropy of the system. However, the main motive for research in co-evolutionary robotics is that more complex tasks can be solved collectively then individually.

Traffic navigation is an integral part of every day life. Along with this arise complexities such as coordination of traffic flow and safety. A common problem is when too many cars do not work together traffic jam or gridlock can occur. Traffic circles were developed to offer a solution for cars to safely navigate intersections of major roads. It does this by forcing cars to slow down, thus eliminating high speed accidents, which are major causes of fatal crashes. However, this does not solve the first problem of traffic congestion. This is due to navigation rules that are applied in *traffic circle problem*, which we explain in later sections of the paper.

The goal of our research presented here is to evolve neural network controllers for robots using genetic algorithms (GA), to navigate a traffic circle. From this, we hope to observe rules that emerge from the system that maintain efficient flow of traffic while still maintaining road safety. This navigation problem can be broken down into several sub-problems which we call behavior layers: *object avoidance*, *unidirectional navigation*, and *path planning.* In this paper we focus on the first two layers.

Prior to explaining the problem more in depth, we cover some related work that has been done in this field and background. Some of the major concepts covered are neural networks, their architecture, and previous experiments such as object avoidance. We replicated some of the object avoidance experiments and analyzed the graphical representations the obtained results, to gain a better understanding of the GA parameters that offer variability in results. The two parameters we concentrated on were population size and mutation rate.

The conclusions drawn from the analysis helped us to conduct our experiments more efficiently. The two conducted experiments *directional zones* and *sequential zones* used a new concept of zones to help navigate through the built maps.

In our conclusion, we discuss our results and the future outlook of collective evolution for designing robot control systems, as well as the differences between central and decentralized embodied systems.

2. RELATED WORK

Artificial evolution can produce a robot control systems that are competitive with hand design systems [4]. An example of this can clearly be seen in object avoidance experiments where evolved neural controller can successfully navigate in any environment. Regardless of the dynamics of the problem, navigation relies on the proper mapping of external sensors to motor actions. This creates a feedback loop where the next motor actions depend on the current sensor reading, and similarly the sensor readings depend on the previous motor actions. This feedback loop can be sliced into steps, at which time the control system can be evaluated to check if it meets the criteria of the desired behavior through the use of a fitness function. At the end a single compound fitness value can be derived from the multiple fitness evaluations, to represent the score of each of the evaluated control systems. This provides the framework for, most frequently three dimensional, graphical structure called *Fitness Landscape.*

The concept of a fitness landscape is used to present a graphical relationship between genotypes (genes that encode the desired qualities) and success in replication. A substantial amount of research has been put into a search for the highest landmarks that represent local and global maximums [5]. The search for maximums is done by changing various system parameter values until optimal or accurate approximations can be found, which yield the best possible results in the given task space.

Swarm robotics is a co-evolutionary approach for emergence of global behavior. It was inspired by the social insect metaphor where there are elements of collective behavior in a decentralized system. In a swarm system, each autonomous robot is able to make its own decisions towards solving a simple task, while the swarm as a whole can solve more complex tasks that an individual can not [6]. This is much like the behavior that is desired in the *traffic circle problem*, where each robot or vehicle work together to optimize the flow of traffic through a traffic circle.

3. BACKGROUND

3.1 Neural Networks

An artificial neural network (ANN) is a system that consists of many nodes (neurons), which are connected together by synapses. Every synapse has an assigned threshold or a weight value that the input must overcome in order for that neuron to fire. Each neuron is connected to one or more other neurons that create a network. The main idea of neural networks is to simulate parallel distributed processing, very similar to biological neurons in the human brain.

In this research ANNs were used for as robot control systems. The network contained six inputs neurons, which correlated with six infrared sensors located on the robot, and two output neurons that controlled speed and direction of two motors. The input values range between [0,+1], where the higher values indicate a closing in obstacle within the proximity. Motor values range between [-1,+1], where 0 indicates that the motor is idle, 1 when the motor is running full force forward, -1 when the motor is spinning full force in reverse. By evolving the weights between these neurons, one can achieve the desired behavior.

Neural networks are particularly useful for designing robot control systems because they are flexible enough to allow adaptability to different environments without the need to further change the weights. There are several architectures one can use when designing a neural network to obtain different results. In our research, we primarily focused on *feed forward* neural networks.

3.2 Genetic Algorithm

A Genetic Algorithm is a computer simulated search procedure that behaves similarly to Darwin's natural selection – survival of the fittest. A certain number of individuals is chosen to represent a population, where each one represents a potential problem solution. Every agent is placed into an environment where evaluation of the agent's behavior generally takes place every time step, based on the fitness function. At the end of the run, an average fitness value is calculated that will represent the agent's ability to perform the given task. The highest ranking individuals are selected to create a sequence of new populations by mutating and recombining (mating) the gene pool. Variability in selected parameter values such as population size and mutation rate, can offer unique solutions.

3.3 Object Avoidance

Object avoidance is the ability to navigate around in an environment without causing collisions with obstacles. This technique is a foundation for many other experiments. Our replicated and newly-developed experiments use the object avoidance fitness function developed by Nolfi and Floreano, which breaks down into three parts [4].

$$\begin{aligned} \Phi = V(1 - \sqrt{\Delta v})(1 - i) \\ 0 \leq V \leq 1 \\ 0 \leq \Delta v \leq 1 \\ 0 \leq i \leq 1 \end{aligned} \tag{1}$$

The first part V is a sum of the absolute values of both motors. This encourages motor use, regardless of its direction. As long as the value is above zero, it contributes to the overall fitness value. If motor readings are negative and positive, this indicates that the robot is spinning around. To prevent this, straight movement needs to be encouraged and rewarded. The second part $(\sqrt{1 - \Delta v})$ accomplishes this by giving higher fitness values to robots with small differences between motor speeds. This induces the wheels to spin in the same direction. The third part, (1-i), is the actual object avoidance. The variable i gets its value from the highest

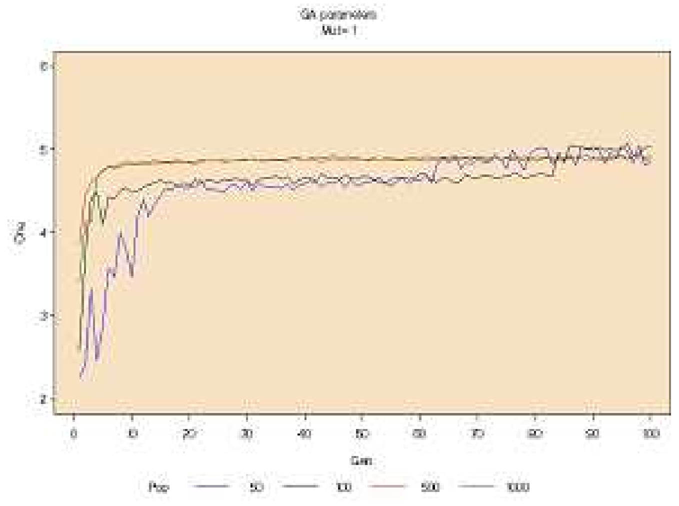

Figure 1: Mutation rate at 1

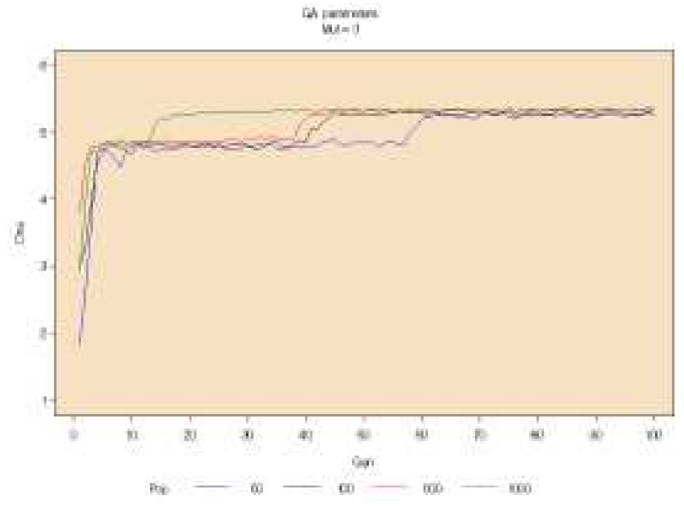

Figure 2: Mutation rate at 3

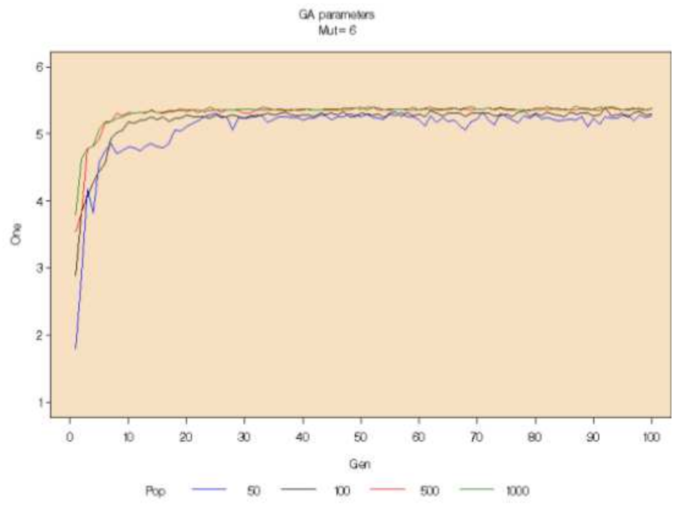

Figure 3: Mutation rate at 6

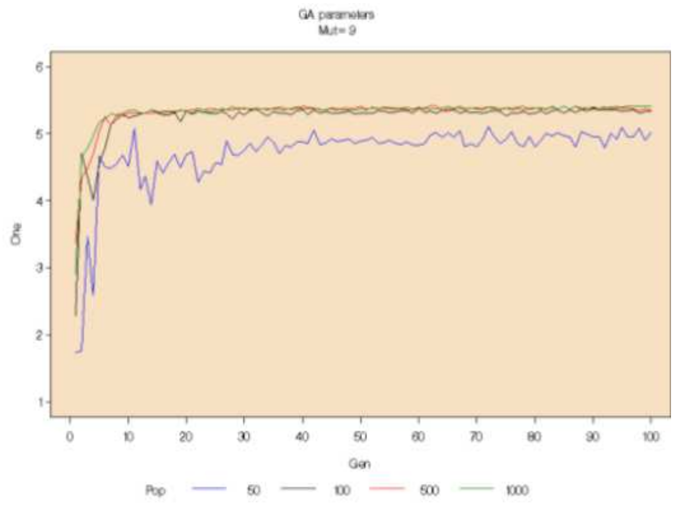

Figure 4: Mutation rate at 9

sensor readings. By default the input values are 1; the closer a robot comes to an object the closer sensor values approach 0. This encourages robots to stay in the middle of the road, rather then scrape or ride along the side of a wall.

4. ANALYSIS OF GA PARAMETERS

The main objective of our analysis of GA parameters is to obtain a qualitative and quantitative understanding the genetic algorithm. The perpetuating parameters are population size (δ) and mutation rate (β). There are other GA parameters such as generation size, steps (frequency of evaluations), epochs (trial runs per individual), etc., analysis of which is outside the scope of this paper. With a wide range of variability in parameter values, it is essential to look at most effective settings that approximate as close to an optimal solution as possible.

The problem with the results derived from a statistical analysis is that they differ depending on the complexity of a given task. In this case we chose to analyze the fitness values obtained from obstacle avoidance experiments, results of which will help us make a prediction about the effective setting for more complex tasks such as the traffic circle problem.

In all of our trial runs we kept the following variables constant based on Nolfi and Floreano's experiments: epoch was set to 3 trials per individual, number of steps set to 500, and generation size was 100. With these fixed parameters we varied population size at 50, 100, 500, and 1000. Similarly the mutation rate was varied 1, 3, 6, and 9 percent. All possible combinations of the two parameters were produced and analyzed. Due to time constrain, we were able to run each combination only once. In the future, more trial runs will have to be conducted to ensure the results. At the moment this provides us with a preliminary overview.

4.1 Effective population size

Population size, δ, is a fundamental parameter in genetic algorithms and knowing the approximate optimal value settings for δ will give us the most effective gene pool from which selections can be made. We were able to draw an analysis of the population factor by examining the predictor variable - generations - that in this case is [0,100], versus the response variable - fitness value. (Figures 1-4) show the four mutation rates, β. Each of the β graphs contains data plots of population 50, 100, 500, and 1000.

With β fixed at 1 percent (Figure 1), there does not appear to be any variation in result that can be made conclusive without further examination of additional trial runs. However, β set to 3 percent gives us more clear insight (Figure 2). It is very noticeable that δ at 1000 is the first to make a sudden genetic jump to a higher level in the fitness value, Φ, between generations 12 and 18. Similar genetic jumps can be found in all of the δ. δ at 500 makes a rapid jump in the Φ at generations 39 through 43, δ at 100 makes a jump at 42 to 48. Finally, δ at 50 trails significantly further behind, making its jump from generation 58 to 62.

With β at 9% (Figure 4), we found very interesting occurrences of Φ reaching 5.4 when δ was 1000, the highest value acquired among all the trial runs. It became apparent to us that with a larger pool of selectees to choose from and with β at a local extreme results the exploration of fitness landscape to its maximum potential.

The effective setting for δ depends on the complexity of the task. For the given task such as simple navigation with obstacle avoidance, the most effective δ is 100. However, as more dynamics are introduced this value would vary.

4.2 Degree of mutation

Small β would appear to be effective in small populations with the GA gradually selecting the neural network controllers with the desired behaviors that produce the best fitness evaluations. If a large value for β is selected, the results of which can be seen in (Figure 4), then the fitness value becomes too unstable when δ is 50. This is due to the fact that the desired behavior can be quickly achieved and just as quickly lost because there is a small number of individuals that are selected for recombination of the next generation. Our hypothesis on this is that the higher evaluated controllers are carried over more accurately to the next generation in smaller population when mutation is very little because it is easier to preserve desired qualities. Also the benefit of this is that this approach requires less computation and resources.

With a large β, the results are outstanding with large δ. With large mutation rate the local maximums are found a lot faster and more accurately. This is illustrated in (Figure 4) where δ at 1000 achieves and maintains the highest Φ value of 5.4. In small population the Φ results become completely unstable and unpredictable. For object avoidance, the most effective β setting is 3%.

5. THE TRAFFIC CIRCLE PROBLEM

A traffic circle is a road junction at which traffic streams circularly around a central island. It was originally designed so cars could merge into gaps in rotary traffic and remain circling until the opportunity for the desired exit would come up. This system forces a constant motion in the same direction creating a smoother flow than a traffic light would. The direction differs from country to country but for the purpose of our research we chose counter clockwise. This eliminates the need for left turns flowing in and out of the circle, therefore avoiding cross traffic turning which reduces a number of accidents [1]. Even though the traffic circle may prove to be safer than regular road intersections, during times of heavy traffic a gridlock can occur.

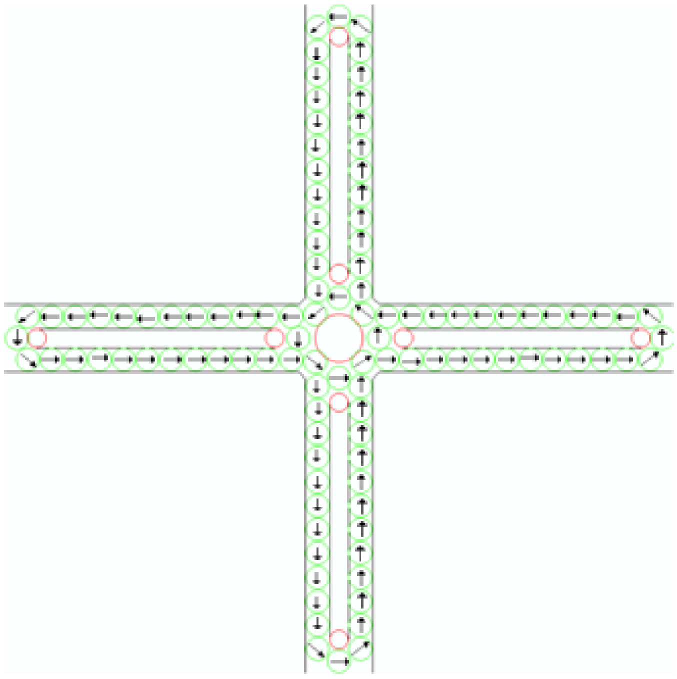
Figure 5: Traffic Circle Map

Our goal is to design a system that will derive the behavior of navigating a traffic circle among autonomous robots while still maintaining safety. The *traffic circle problem* is the base structure for many smaller problems. It can be broken down into three major behavior layers such as *path planning*, *object avoidance*, and *unidirectional navigation*. By solving these smaller sub-problems, one should be able to arrive with the solution for larger *traffic circle problem*. *Object avoidance*, navigation in an environment with out colliding into obstacles, has been solved by previous researchers. In our following experiments we addressed the *unidirectional navigation*, where the encouraged behavior was traffic flow in the same direction.

6. STEP BEHAVIOR EXPERIMENT SETUP

6.1 Simulator

The experiments here were carried out using a freely distributed simulator, YAKS (Yet Another Khepera Simulator), that evolves the weights of the neural networks for the robot controller. YAKS supports the use of multiple neural network configurations on different robots at a time. This allows the evaluation of different types of network architectures and the ability to compare the resulting behaviors. This feature was used in the *Directional Zones* experiment to compare feed forward and modular networks.

YAKS simulates Khepera robots, which are circular robots weighing 70 g each, with a diameter of 55 mm, and a height of 30 mm. Robots have two wheels controlled by independent DC motors that can move in two directions, forward and reverse. Kheperas have 6 proximity sensors on one side and 2 on the other. Individual unit price of physical robots ranges from $2000 to $1000 USD.

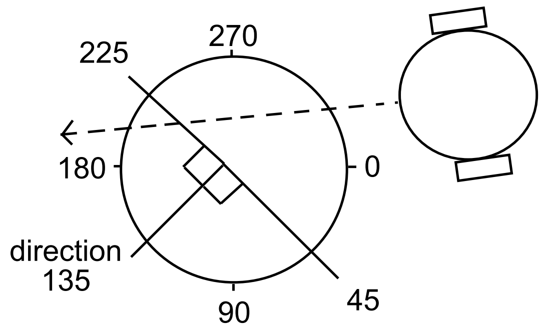

Figure 6: Directional Zone in detail

In designing the YAKS simulator, sensor and motor measurements were recorded from a real Khepera robot. The simulator then cross references these values when it needs to get the values of the sensors or motors. This minimizes calculations the simulator must make which in turn speeds up experimentation. YAKS allows the user to customize and design worlds with walls, various obstacles, and specify start locations. In our experiments we introduced the concept of *zones* (Figure 5), which are circular areas that provide the ability to reference to that specific location on the map. Zones incorporate features such as the number of times a specific robot has visited it and current coordinates of an individual within the zone. The simulator also offers the ability to configure GA parameters used to evolve the neural network controller such as number of generations, population size, epochs, number of steps each robot is evaluated, mutation percentage, number of parents and offspring, and selection method. In the experiments we used a combination of *elite* and *tournament* selection methods.

6.2 Directional Zones

A directional zone experiment works on the principle that the fitness function evaluates the angle under which the robots enter a zone and predicts the degree under which it will exit (Figure 6). All zones have an acceptable 180 degree range specified. Robots that are entering and exiting zones, as specified in the environment through the designated range of the zone, are rewarded with a higher fitness value. Robots that enter from the opposite 180 to 360 degrees and robots that enter within the 180 range with exit angle within the same range, are punished with a lower fitness function value (Equation 2). By using many zones, one can specify a desired traffic flow. The experiment was run with δ set at 100 individuals for 500 generations with β at 3%, each robot was allowed to take 500 steps.

$$\begin{aligned} \Phi' = \Phi + (\kappa\mu) - (\tau\rho) \\ \text{where} \\ \kappa = .02 \\ 0 \leq \mu \leq 108 \\ \tau = .02 \\ 0 \leq \rho \leq 108 \end{aligned} \tag{2}$$

The directional zone fitness function is implemented by combining object avoidance with reward and punishment factors from zone evaluation. κ is a reward constant for going through a zone correctly. μ is the number of zones the robot correctly entered. τ is a punishment constant. ρ is the number of zones incorrectly entered. This function is evaluated at the end of the run. The difference of reward and punishment is added to the average Φ to create a new value Φ'.

6.3 Sequential Zones

The sequential zones experiment, much like the directional zones, attempts to enforces directional control through the use of zones. Each zone has a sequential id number, and the zones are placed on the map in an incremental order, forming the contour of the path that will exhibit the desired direction. For the given experiment, a simple square map, with a pentagon in the center, was constructed. The chosen direction was counter clockwise. The experiment was run with δ set at 100 individuals for 100 generations with β at 3%, each robot was allowed to take 500 steps.

$$\Phi = V(1 - \sqrt{\Delta v})(1 - i)(1 - \zeta) \tag{3}$$

The core of the formula remains the same as in object avoidance, with an added component $(1-\zeta)$, which governs the direction flow. After comparison of current zone to the previous one, ζ is set in the following manner:

$$\begin{array}{ll} \zeta = 0.5 & : \text{if previous zone has a higher value} \\ \zeta = 0.2 & : \text{if the robot is not in a zone} \\ \zeta = 0.0 & : \text{if previous zone has a smaller value} \end{array} \tag{4}$$

The combined effect of all the components of the fitness function promotes straight moving robots to enter zones in incrementing order at maximum speed. This formula is designed to run every evaluation step.

7. EXPERIMENT RESULTS

7.1 Directional Zone

The directional zone experiments did not emerge unidirectional navigation as expected. The exhibited behavior was that robots moved in a straight line from their initial position until a wall was encountered and then just stop. This is due to the fact that they would score a higher fitness value just from a short run that was done correctly and not make any more movement in order to avoid punishment.

Another possible reason why the directional zones did not evolve a unidirectional traffic flow is because there was no direct feedback to the robots. They were placed on the map and given a trial run without any indicators of their current performance, so they had no means of knowing if they are moving in the correct direction until the end of the run. This is equivalent to trying to walk blind fold on a zigzagging line, where even if the subject succeeds, replication of such performance is virtually impossible. In this case the robots were able to learn techniques that would cheat the fitness function but still give them relatively high fitness values.

7.2 Sequential Zones

Due to the environment map being square and the promoted direction being counter clockwise, the neural net controllers learned to circle around the pentagon. They accomplished this by always turning left in order to achieve a high fitness value. For the most part, this seemed to work, and the robots demonstrated unidirectional navigation. However, problems occurred when faster moving robot ran into a slower one, causing oneself to turn around and reverse direction by turning left. The faster moving robot continued its path in the opposite direction until another obstacle would be encountered thus once again causing it to turn left and resume its original course. This approach would seem to work, but only with a very small number of individuals in a large environment where there are less chances for collisions.

8. CONCLUSION AND OUTLOOK

The main objective was to develop an environment that would emerge with a set of rules that would govern traffic flow. This task broke into three behavior layers - *path planning*, *object avoidance*, and *unidirectional navigation*. The emphasis of the paper fell on the last two components. Through replication of object avoidance, we were able to fulfill the first behavior layer towards approaching the global solution. Also by replicating these experiments it became apparent that there is a large number of variables in the genetic algorithm that can vary and if not properly set even skew the results. The two factor that we looked at were population size and mutation rate. Knowing the optimal setting for the two GA parameters, one can efficiently find local maximums. None of the combinations of settings discussed here work best in all circumstances. However, there are effective values for these parameters that work best in certain situations and tasks. For example in very dynamic environment, it would be useful to have a large population size with combination of large percent of mutation. This yields maximum exploration of fitness landscape, resulting unique and more accurate solutions. For obtaining object avoidance the most optimal settings are a combination of δ set at 100 with β at 3%.

Unidirectional navigation, another step behavior, was addressed by two different experiments: *directional* and *sequential zones*. The one that demonstrated partial unidirectional traffic flow was the *sequential zones* experiment. This technique works well in very simple environments with a low number of interacting individuals. In our case, the neural controllers learned to always circle left by only making left turns, a behavior that matched a traffic circle system. However, the rules began to break when faster moving robots ran into slower moving ones.

By performing the above experiments, it became clear to us that all robots were acting in their own self interest, maximizing their fitness values while the ignoring the ultimate goal of the collective. Our setup used a decentralized control system where decisions are made on an individual level. Results suggest future work to move to the alternative of a centralized approach. A central control system governs and evaluates a group's performance rather than individuals. It is possible for a decentralized approach to work in the future, given that all of the step behavior components of traffic circle problem are fulfilled. *Path planning* definitely will play an important role in evolving traffic rules. Up to this point the robots have been demonstrating reactive behavior, but with path planning, this might prove otherwise. Implementation of this technique may eliminate the case where a robot runs into a neighbor causing the faster robot to change direction. Robots also need to learn how to know where they currently are in relation with others. Communication between robots may be necessary. The results in this paper can be used as framework for future experimentation and help improve future work on co-evolutionary techniques.

9. REFERENCES

[1] Insurance Institute for Highway Safety. *Roundabouts*. [online], http://www.hwysafety.org/srpdfs/sr3505.pdf, 2000.

[2] A. Keane and A. Rogers. *Theoretical Aspects of Evolutionary Computing*, chapter An Introduction to Evolutionary Computing in Design Search and Optimisation, pages 1–11. Springer-Verlag, Germany, 2001.

[3] Tom Mitchell. *Machine Learning*, chapter Artificial Neural Networks, pages 81–127. MIT Press and The McGraw-Hill Companies, Inc., Boston, Massachusetts, 1997.

[4] Stefano Nolfi and Dario Floreano. *Evolutionary Robotics*. MIT Press, 2000.

[5] J.L. Shapiro. *Theoretical Aspects of Evolutionary Computing*, chapter Statistical Mechanics Theory of Genetic Algorithms, pages 87–108. Springer-Verlag, Germany, 2001.

[6] Vito Trianni. *Evolution of Coordinated Motion Behaviors in a Group of Self-Assembled Robots*. University of Bruxelles, IRIDIA, Brussels, Belgium, 2003.