The Multi-objective Evolution of Mobile Robot Behavior

Praveen Koduru
Electrical and Computer Engineering
Kansas State University
Manhattan
KS 66502

Ashish Ahuja
Electrical and Computer Engineering
Kansas State University
Manhattan
KS 66502

Kyle McDowell David
Electrical and Computer Engineering
Kansas State University
Manhattan
KS 66502

Lukas Lansky
Electrical and Computer Engineering
Kansas State University
Manhattan
KS 66502

Sanjoy Das
Electrical and Computer Engineering
Kansas State University
Manhattan,
KS 66502
sdas
@ksu.edu

Stephen Welch
Division of Biology Kansas State University
Manhattan,
KS 66502
welchsm
@ksu.edu

ABSTRACT

Autonomous mobile robot navigation is a complex problem and evolutionary principles applied to such problems provide good solutions with relatively less computational effort. This also allows an automatic evolution of such systems. We describe how to evolve a neural network control system for a mobile robot using a simulator applying concepts of multi-objective optimization. Sometimes a single objective may not be adequate to describe the desired performance of the robot. In such cases, typecasting the problem as a multi-objective problem becomes necessary. In this paper we investigate the possibility of using evolutionary algorithms to evolve a controller for a mobile robot with multiple objectives to be satisfied simultaneously. Such behavior includes obstacle avoidance, smooth motion and target acquisition. The novelty of this method lies in the evolution of different navigational behaviors simultaneously using concepts of Pareto-optimality and evolutionary algorithms. A neural network is utilized to provide the control structure for the navigation of the mobile robot. A multi-objective evolutionary algorithm (FSGA) is utilized to identify the optimal neural network weights. The simulation results show that the proposed methodology is efficient and robust for evolving different behaviors simultaneously. Simulation results are provided and discussed.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods*

General Terms

Algorithms, Performance, Design.

Keywords

Evolutionary robotics, Genetic Algorithms, Multi-objective optimization, Pareto-optimal and Neural Networks.

1. INTRODUCTION

Autonomous mobile robot navigation is typically formulated as follows: given a robot and a given a set of sensors to measure environmental conditions, we need to find a control system for the robot that can move the robot in the environment such that the robot motion is collision-free and satisfies certain optimization criteria. In this paper we address the problem of evolving autonomous mobile robot navigation in an environment where multiple costs are to be optimized using evolutionary techniques. Evolutionary robotics aims to develop a suitable control system of the robot through artificial evolution [1]. The execution of the motion is mediated by a controller, which is often implemented as a neural network of a specific topology [2, 3]. Evolutionary robotics uses techniques, such as genetic algorithms, genetic programming, and evolutionary strategie to evolve the controllers for the robot. An initial population of different genotypes, each encoding the robots architecture, is created at random. Each robot is allowed to interact with the environment and fitness is assigned to each of them. The designer defines a fitness function, which measures the ability of a robot to perform a desired task. The designer plays a passive role and the behaviors emerge automatically though evolution due to interactions between the robot and its environment. The robot and the environment form a highly dynamical system, in which the robot’s decision at a particular time depends on the sensory information and its previous actions. The initial controllers may not behave optimally, but the controller improves its performance and produces better results gradually with incremental generations of evolution. Robots with higher fitness are allowed to reproduce to hopefully create better solutions. The population of solutions is modified using operators as crossover and mutation and also hybrid local improvement search techniques as Nelder-Mead Simplex [4] and ultimately good solutions are obtained through generations.

Most evolutionary robotics problems utilize single objective functions [2]. But however in practice we often find the need to evolve behaviors with multiple objectives [5, 6, 7, and 8]. These multiple objectives could be dependent on various parameters as length of the path, time taken to travel to the destination, etc. A simple example would be to consider the case of target acquisition behavior in a dynamic environment, with the objectives to be minimized are the total distance traveled and total time taken to reach the target. In such cases, it is not always the path associated with minimal distance coincides with minimal time path. One possible case is that the robot could be stationary at one place, and wait for the obstacle to move away from its path and then move straight towards the target. This would be the shortest path length solution, but not necessarily the shortest time solution. The other extreme solution could be the robot navigating around the obstacles and moving towards the target. This would be the minimal time solution but at the same time it may not be the minimal distance traveled solution. In between these two extreme solutions there could be more solutions that are also possible, but none of them could be minimal time or minimal distance traveled paths. Generally to solve such multi-objective problems, a weighted objectives approach is implemented where the *n*-objective cost functions f_1, f_2, to f_n, are aggregated in a linear fashion to get a single objective, as $F = \sum_i w_i f_i$. Selecting suitable weights a solution with certain desired characteristics is obtained.

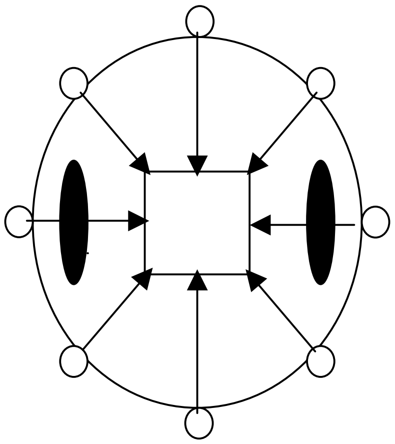
Figure 1. Schematic of the Mobile Robot used.

The major reasons why multi-objective search techniques would be better than single objective search techniques can be explained as follows. 1. By using multi-objective optimization routines, we can get all the possible solution set possible with single run and can use the best possible solution from the set of pareto-front optimal solutions set. In a single objective routine, multiple runs have to be done with different weights to get all the possible solutions. 2. The solutions obtained in a multi-objective search could be queried to get the required specifications the user demands from the algorithm. For example, if the user wants a robot that would go from Point A to Point B in within a certain time T, and using up only a certain amount of energy E, we can look up at the solutions of the pareto-front and clearly demark the solutions that satisfy such criteria. This may not be always possible with single objective runs, where additional training or rerun of the algorithm with a modified fitness function would be necessary. 3. Finally, it may be the case that the objectives we are trying to evolve are conflicting in their cost functions. In such cases, the single objective function may not be able to provide solutions which are good in both the costs with just a single run, while a multi-objective approach would provide a set of solutions which are good in either costs or intermediate solutions which are non-dominated.

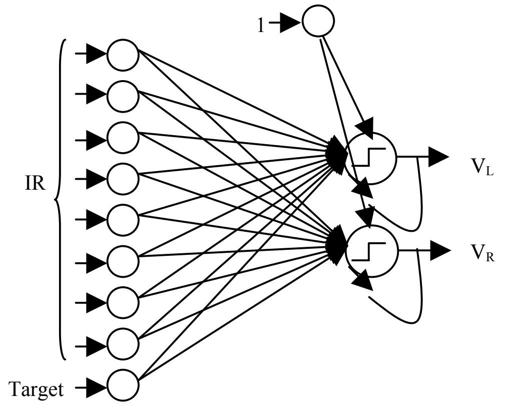

Figure 2. Schematic of the Neural Network used

2. ROBOT DESCRIPTION

The robot model that has been used in the simulations is a miniature mobile robot with differential drive steering. The robot has a circular shape with 55mm diameter. A schematic of the robot is shown below in Figure 1. The robot has two motors that are capable of moving in both directions. The output of the motors is limited such that the maximum velocity of the robot is restricted to 2.5cm/sec in either direction. The robot is provided with eight infrared proximity sensors placed symmetrically on the robot. These sensors can only detect obstacles in the range of 0-5cm in our simulations. The values of infrared sensors were normalized to the range of [0, 1], where 0 denotes no obstacle detected and 1 denotes robot sensor is touching the obstacle. For safety reasons the robot is assumed to have a collision if the sensor reading is greater than 0.98. This avoids the robot to move further and actually collide into the obstacle. For target acquisition we assume the targets to be bright lights placed at a certain height. These light sources are assumed to be sources of energy for the robot.

The robot has to reach within a radius of 1cm of the light source's actual position to recharge itself. The light sensor on the robot measures the intensity of light and provides it a feedback in which direction is the nearest light source. The robot initially has a minimal amount of battery life, and has to reach the target and get recharged so that it can move for another few steps. After it has recharged at the target, the light switches off and the robot has to search for a new source. The objective of the robot would be to navigate in the environment autonomously and try to find light sources where it can recharge itself, while also trying to do obstacle avoidance and maintain a smooth motion.

3. EVOLUTIONARY PROCESS

3.1 Structure of Neural Network

The control system for the autonomous robot was chosen to be a neural network. Neural networks are appropriate because of their adaptive nature that is an important issue for autonomous robots. This allows the robot to work even under different conditions in different environments. An efficient way of training neural networks for mobile robot is by using evolutionary algorithms to train the synaptic weights of the neural network. The evolutionary procedure employed in the simulations is a multi-objective genetic algorithm FSGA [9], to evolve the weights of the neural network. Figure 2 shows a schematic of the neural network architecture that was used. The architecture of the neural network was fixed for all different simulations. The neural network inputs are the eight infrared sensor readings, the direction in which the target is located and the recurrent inputs at the output layer. There is no hidden layer in the network. The output of the neural network is directly used to control the motor activation. The total number of weights that were to be evolved for the network was 22. The weights were individually coded as floating point numbers on the chromosome. The length of the chromosome was 22 and a population size of 50 was used for all simulations. Initially weights are randomly generated in the range of ±0.5 for each individual. The multi-objective genetic algorithm then evaluates each member of the population according to some fitness measure decided by the user, depending on the task to be solved. Details of the multi-objective genetic algorithm are explained in the next section.

3.2 Multi-Objective Evolutionary Algorithm

Evolutionary algorithms have emerged as one of the most popular approaches for the complex optimization problems. They draw upon Darwinian paradigms of evolution to search through the solution space (the set of all possible solutions). Starting with a set (or population) of solutions, in each generation of the algorithm, new solutions are created from older ones by means of two operations, mutation and crossover. Mutation is accomplished by imparting a small, usually random perturbation to the solution. In a manner similar to the Darwinian paradigm of survival of the fittest, only the better solutions are allowed to remain in a population, the degree of optimality of the solution being assessed through a measure called fitness.

3.2.1 Multi-objective Optimization

When dealing with optimization problems with multiple objectives, the conventional concept of optimality does not hold good [10, 11]. Hence, the concepts of dominance and Pareto-optimality are applied. Without a loss of generality, if we assume that the optimization problem involves minimizing each objective $e_i(.)$, $i = 1...M$, a solution u is said to dominate over another solution v iff $\forall i \in \{1,2,\ldots,M\}$, $e_i(u) \le e_i(v)$ with at least one of the inequalities being strict, i.e. for each objective, u is better than or equal to v and better in at least one objective. This relationship is represented as $u \succ v$. In a population of solution vectors, the set of all non-dominating solutions is called the Pareto front. In other words, if S is the population, the Pareto Front Γ is given by,

$$\Gamma = \{u \in S \mid \forall v \in S, \neg(v \succ u)\} \tag{1}$$

The simplistic approach of aggregating multiple objectives into a single one often fails to produce good results. It produces only a single solution. Multi-objective optimization on the other hand involves extracting the entire Pareto front from the solution space. In recent years, many evolutionary algorithms for multi-objective optimization have been proposed [10, 11].

3.2.2 Fuzzy Dominance

Assume an overall minimization problem involving M objective functions $e_i(.)$, $i = 1 \ldots M$. The solution space is denoted as $\Psi \subset \Re^n$. Given a monotonically non-decreasing function $\mu_i^{dom}(\cdot)$, whose range is in [0, 1], $i \in \{1,2,\ldots,n\}$ such that $\mu_i^{dom}(0) = 0$, solution $u \in \Psi$ is said to i-dominate solution $v \in \Psi$, if and only if $e_i(u) < e_i(v)$. This relationship can be denoted as $u \succ_i^F v$. If $u \succ_i^F v$, the degree of fuzzy i-dominance is equal to $\mu_i^{dom}(e_i(v) - e_i(u)) \equiv \mu_i^{dom}(u \succ_i^F v)$. Fuzzy dominance can be regarded as a fuzzy relationship $u \succ_i^F v$ between u and v. Solution $u \in \Psi$ is said to fuzzy dominate solution $v \in \Psi$ if and only if $\forall i \in \{1,2,\ldots,M\}$, $u \succ_i^F v$. This relationship can be denoted as $u \succ^F v$. The degree of fuzzy dominance can be defined by invoking the concept of fuzzy intersection and using a t-norm,

$$\mu^{dom}\left(u \succ^F v\right) = \bigcap_{i=1}^{M} \mu_i^{dom}(u \succ_i^F v) \tag{2}$$

Given a population of solutions $S \subset \Psi$, a solution $v \in S$ is said to be fuzzy dominated in S iff it is fuzzy dominated by any other solution $u \in S$. In this case, the degree of fuzzy dominance can be computed by performing a union operation over every possible $\mu^{dom}\left(u \succ^F v\right)$, carried out using *t*-co norms as,

$$\mu^{dom}(S \succ^F v) = \bigcup_{u \in S} \mu^{dom}(u \succ^F v) \tag{3}$$

In this manner, each solution can be assigned a single measure to reflect the amount it dominates others in a population. Non-dominated individuals within a solution will be assigned zero fuzzy dominance, as for any non-dominated individual $\mu_i^{dom}(u \succ_i^F v)$

Further details of this concept can be found in [9]. In the present work, the membership $\mu_i^{dom}(u \succ_i^F v)$ is piecewise linear, and given by,

$$\mu_i^{dom}\left(\Delta e_i\right) = \begin{cases} 0, & \Delta e_i \le 0 \\ (\Delta e_i)/\Delta_i & 0 < \Delta e_i < \Delta_i \\ 1, & \Delta e_i \ge \Delta_i \end{cases} \tag{4}$$

where, $\Delta e_i = e_i(v) - e_i(u)$. The union and intersection operators follow the standard min and max definitions [12].

3.2.3 The Fuzzy Simplex Genetic Algorithm

Fuzzy dominance time makes it possible to assign a single measure of fitness to multiple individuals. Computing the mutual fuzzy dominance in a population of individuals enables local gradient descent based techniques to be applied in a multi-objective framework. In [9] a strategy was proposed that applied a local search procedure, the Nelder-Mead algorithm [4], in conjunction with a genetic algorithm.

A simplex in n -dimensions consists of $(n+1)$ solutions u_k, $k = \{1,2,\ldots n+1\}$ which are its vertices. In a plane, this corresponds to a triangle. The solutions are evaluated in each step and the worst solution w is identified. The centroid of the simplex is then evaluated, excluding the worst solution and the worst point is reflected along the centroid. If $c = \sum_k u_k - w$ is the centroid, the reflected solution is

$$r = c + (c - w) \tag{5}$$

Usually, the worst point w is replaced with the reflected point r in the simplex, but if the r is better than any solution in the simplex, the simplex is further expanded as,

$$r_e = c + \eta(c - w) \tag{6}$$

Where η is called the expansion coefficient. However, if the reflected solution r is worse than w, the simplex is contracted and the reflected solution is placed on the same side of the centroid. When solution r is not worse than w, but worse than any other solution in the simplex, the simplex is still contracted, but the reflection is allowed to remain on the other side of the simplex. Reflection is carried out as follows,

$$r_c = c \pm \kappa(c - w) \tag{7}$$

In the above equation, κ is called the contraction coefficient. Solution w is replaced with the new one, r, r_e, or r_c in the next step. The simplex algorithm is allowed to run for multiple steps before it converges.

Fuzzy dominance is the objective function to which Nelder-Mead is applied. This allows the Nelder-Mead algorithm to push solutions towards regions of lower dominance, i.e. the Pareto front.

A binary tournament selection is implemented in the genetic algorithm that selected two individuals at random from the population with replacement, and picks the one with the least fuzzy dominance. An offspring t, was computed from two parents u and v in the following manner,

$$t = \zeta u + (1 - \zeta)v \tag{8}$$

where ζ is a uniformly distributed random number in [0,1].

Solutions were mutated with a probability of β, by adding a random number with zero mean, that followed a Gaussian distribution with a spread σ, according to,

$$u = u + N(0, \sigma) \tag{9}$$

3.3 Fitness Functions

The desired behavior of the mobile robot is based on the following abilities: (a) moving forward as fast as possible, (b) moving in as straight a line as possible, (c) keeping as far away from obstacles as possible, (d) getting close to specified targets as possible and (e) maintaining its heading towards the targets as much as possible. In order to evaluate the individual's fitness in the evolutionary algorithm, we used the following fitness functions. We have used two different fitness functions given in equations (10) and (11).

$$F_1 = \sum_{i=1}^{steps} V_i(1 - \sqrt{DV_i})(1 - I_i) \tag{10}$$

$$F_2 = \sum_{i=1}^{steps} \frac{1}{(1 + d_i)(1 + \Delta_i)} \tag{11}$$

where V_i is the average rotation speeds of the two wheels, DV_i is the algebraic difference between signed speed values of the wheels, I_i is the activation value of the proximity sensor with the highest activation at time i, d_i is the distance from the nearest target and Δ_i is the scaled difference of the heading direction of robot and the direction where the nearest target is present at time i. The values are summed up over number of steps the robot runs without collision. The function F_1, is a common fitness function used for Khepera robot evolutionary algorithms [1, 2]. This fitness function has three components: the first part is maximized by speed of the robot, the second one by straight-line motion and the third by obstacle avoidance. Since the robot has a circular shape and the wheels can rotate in both directions, this function has a symmetric surface with two equal maximum, each corresponding to one direction of motion. The second fitness function F_2 has two components: the first part is maximized by getting close to the nearest target and the second part is maximized by keeping the heading of the robot towards the nearest target. The first fitness tries to make the robot go as fast and straight as possible at the same time avoid the obstacles in the path. The second fitness makes sure that the robot tries to reach the targets while it is trying to satisfy the first objective function. These two objectives can be conflicting in nature. The robot might come into a situation where it would want to reach a target but at the same time would like to avoid hitting into the obstacle in front of the target. So using a single objective function for both these behaviors may not provide optimal solution. The only way to solve such objectives simultaneously is to use a multi-objective optimization algorithm.

4. SIMULATION RESULTS

In this section we show some results of the simulations done on the mobile robot using the multi-objective search algorithm described in the previous section. Three different test cases were considered for the evaluation of the approach. In case (a) a very simple environment is used with no obstacles in the path of the robot. Only the enclosure wall for the environment is considered as the obstacle to avoid the robot from going out of bounds. The evolutionary algorithm was run for 50 generations. Figure 3 shows the simulation of the one of the robots in the Pareto front. The robot starts from an initial position of [10,10] and an initial heading of $\pi/2$. The robot is always trying to reach the nearest target from its current position. It has successfully reached all the targets in the environment. It can be noted that the path is nearly smooth, in a straight line as much as possible, and as far away from the walls as possible. In case (b), we have used a room like scenario, where the environment in case (a) was modified to have walls within the path of the motion of the robot. The robot was supposed to reach the targets while avoiding the obstacles in the path and at the same time try to do it at the maximum possible velocity and moving in a straight line as much as possible. Figure 4 shows the navigation of one of the robot in the Pareto front obtained in this case. From the figure we can observe that the robot is always tying to reach the target, while at the same time its controller is avoiding obstacles and trying to maintain a straight line of motion.

To avoid the notion of robot learning the behavior for only a specific situation, the same evolved robot was run in a different scenario with the targets placed at a different position from what the robot was actually trained on. The robot had to complete the task without any additional training. Figure 5 shows the navigation of the robot evolved in Case (b) run in a different scenario from what it was actually trained on.

In case (c), the environment used is very much similar to the one used in evolution of Khepera robot navigation behavior. Figure 6 shows the navigation of one the evolved robot in the Pareto-front. The evolved robot was tested in an environment with changes to both the obstacle positions and targets positions. The navigation of the robot is shown in Figure 7. From the figures we can clearly note that the robot has been able to learn a multiple objective behavior successfully and the learned robot is able to move in the environment with modifications to both the obstacles and targets.

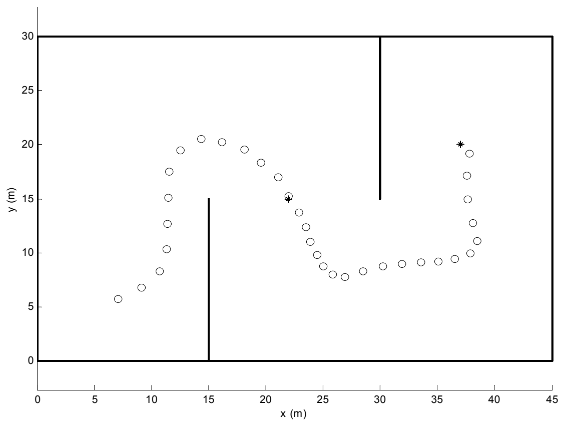

Figure 4. Navigation of the evolved robot in Case (b)

5. CONCLUSIONS

Although applying evolutionary algorithms to optimize the behavior of mobile robots has been attempted they have been mostly concerned with single objective optimization. Unfortunately, in more complex environments, a single objective approach may fall short of accomplishing the task

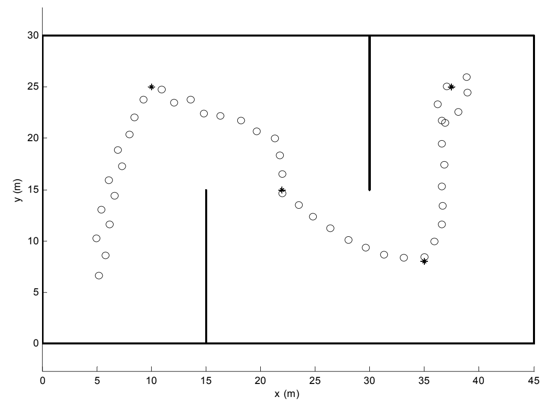

Figure 5. Navigation of the evolved robot in Case (b) in a different environment

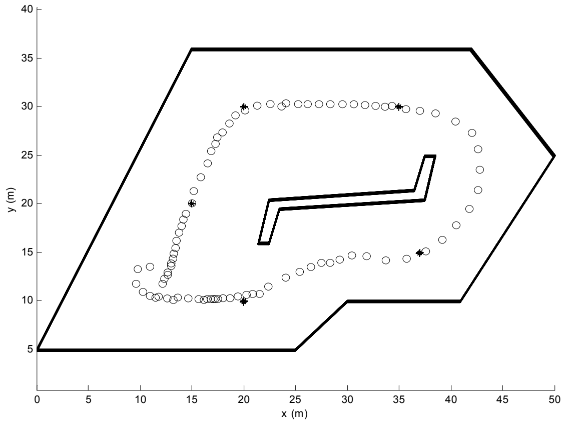

Figure 6. Navigation of the evolved robot in Case (c)

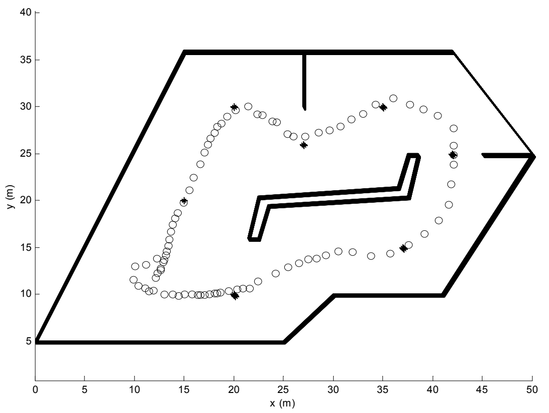

Figure 7. Navigation of evolved robot in Case (c) in a different environment

of evolving the desired behavior. Under these circumstances, a multi-objective optimization approach becomes useful. Furthermore, applying a multi-objective technique produces a Pareto front of possible behaviors. It offers the designer a wide spectrum of possible behaviors to incorporate into a mobile robot. It also allows the user to incorporate a more complex set of behaviors into the robot that normally would not be possible using a simple evolutionary approach. With a Pareto front of optimal solutions, the designer can consider the possibility of switching behaviors in a robot en route. For instance, a robot that is too cautious in avoiding obstacles, may switch to a more aggressive behavior. The ability to change behavioral patterns is handy in dynamic environments where a suitable objective cannot be devised a priori. A multi-objective approach would also be particularly useful in real world applications, where it becomes essential to deploy robots in environments where they have not been trained to navigate before.

In this article we suggest the use of a multi-objective evolutionary approach to model robot behavior. In order to demonstrate the feasibility of such an approach, two distinct components of robot behavior have been considered. Firstly, mobile robots need to navigate along a path that runs free of obstacles and is as smooth as possible. Secondly, the robot also should also perform its task, which in this case is to reach its target. These two tasks are often in conflict, and aggregating them into a single objective will not suffice. Two separate objectives have therefore been formulated and a recent multi-objective approach, the Fuzzy Simples Genetic Algorithm applied to simultaneously optimize both objectives. The effectiveness of this approach has been demonstrated through two separate simulations. Our approach can be easily generalized to handle other objectives also. Thus far, evolving robot behavior has been confined to simple environments. In future, we plan to extend the work to more complex environments. The multi-objective evolutionary approach proposed here offers a versatile alternative to the single-objective approach for determining robot behavior.

6. REFERENCES

[1] Stefano Nolfi and Dario Floreano : Evolutionary Robotics-The biology, Intelligence and Technology of Self-Organizing Machines, MIT Press, Cambridge, MA, 2000

[2] Floreano, D., and Mondada, F.: Evolution of Homing Navigation in a Real Mobile Robot, *IEEE Trans. On Systems, Man and Cybernetics-Part B*, Vol. 26, No. 3, 396-407, 1996.

[3] Floreano, D., and Mondada, F.: Automatic creation of an autonomous agent: Genetic Evolution of a Neural Network driven robot, In Cliff, Husbands, Meyer and Wilson (Eds.), *Proc. Of the Third International Conference on Simulation of Adaptive behavior: From*

Animals to Animats 3, The MIT Press/Bradford Book, 1994.

[4] Nelder, J. A. and R. Mead : A Simplex Method for Function Minimization," *Computer Journal*, Vol. 7, p. 308-313

[5] Paolo Pirjanian**:** Multiple Objective Action Selection in Behavior-Based Control, *The 6th Symposium for Intelligent Robotic Systems*, Edinburgh, Scotland , pp 83-92, July 1998

[6] Paolo Pirjanian:The Notion of Optimality in Behavior-Based Robotics**,** *Journal of Robotics and Autonomous Systems*, Special Issue, 1999

[7] Dong-Oh Kang, Sung-Hun Kim, Heyoung Lee, Zeungnam Bien: Multiobjective Navigation of a Guide Mobile Robot for the Visually Impaired Based on Intention Inference of Obstacles, *Auton Robot*, Vol. 10, No.2, pp 213-230, 2001

[8] Dozier, G., McCullough, S., Homaifar, A., Tunstel, E, and Moore, L., Multiobjective evolutionary path planning via fuzzy tournament selection, *Evolutionary computation proceedings, IEEE World Congress on Computational Intelligence,* The 1998 IEEE International Conference, pp 684-689, 1998

[9] P. Koduru, S. Das, S. M. Welch, J. L. Roe: Fuzzy Dominance Based Multi-objective GA-Simplex Hybrid Algorithms Applied to Gene Network Models, *Lecture Notes in Computer Science: Proceedings of the Genetic and Evolutionary Computing Conference*, Seattle, Washington, 2004.

[10] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review, *IEEE Transactions on Evolutionary Computation*, Vol. 7, no. 2, pp 117-132, April, 2003

[11] Deb, K., Agrawal,S., Pratap, A. and Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6, No.2, pp 182-197, 2002

[12] Mendel, J.M.: Fuzzy logic systems for engineering: A tutorial, *Proceedings of the IEEE*, Vol 83, No. 3, pp 345-377, March, 1995