Multi-Behaviour Robot Control using Genetic Network Programming with Fuzzy Reinforcement Learning

W. Wang[1], N.H. Reyes[1], A.L.C. Barczak[1], T. Susnjak[1], and Peter Sincak[2]

[1] Massey University, Albany, New Zealand
N.H.Reyes@massey.ac.nz
[2] Technical University of Kosiče, Slovakia
peter.sincak@tuke.sk

Abstract. This research explores a new hybrid evolutionary learning methodology for multi-behaviour robot control. The new approach is an extension of the Fuzzy Genetic Network Programming algorithm with Reinforcement learning presented in [1]. The new learning system allows for the utilisation of any pre-trained intelligent systems as processing nodes comprising the phenotypes. We envisage that compounding the GNP with more powerful processing nodes would extend its computing prowess. As proof of concept, we demonstrate that the extended evolutionary system can learn multi-behaviours for robots by testing it on the simulated Mirosot robot soccer domain to learn both target pursuit and wall avoidance behaviours simultaneously. A discussion of the development of the new evolutionary system is presented following an incremental order of complexity. The experiments show that the proposed algorithm converges to the desired multi-behaviour, and that the obtained system accuracy is better than a system that does not utilise pre-trained intelligent processing nodes.

1 Introduction

Fuzzy logic, reinforcement learning and evolutionary algorithms constitute a handful of complementary family of algorithms that found enormous successes in building cores of intelligence for robots. Firstly, fuzzy logic is a computational paradigm that provides a mathematical facility for transforming vague linguistic rules into precise real-time control systems. To mention a few, it was used effectively for smoothly navigating robots and vehicles [2,3], autofocusing cameras, colour correction for object recognition tasks [4] and even automated space docking of satellites. As the control requirements grow, however, more fuzzy systems needs to be constructed and calibrated. As demonstrated in [2,5], steering a robot for target pursuit with obstacle avoidance behaviours and automatic speed adjustment can be achieved using a cascade of four fuzzy systems with the guidance of an informed optimal search algorithm, called A*. Building fuzzy systems necessitate expert knowledge in terms of control rules, and efficient calibration techniques tailored particularly for the control problem at hand. For calibrating fuzzy systems for robot navigation, an example can be found in [6].

Secondly, reinforcement learning is an inherently on-line learning paradigm that allows robots to learn how to behave well through repeated interaction with the environment, as conditioned by a reward function. In the course of its training, a robot is able to learn dynamically a policy for maximising its rewards in the long run. By combining fuzzy logic and reinforcement learning together into one hybrid system, the fuzzy rules can be determined automatically on-line using the mechanics of the RL, without any prior knowledge of the environments dynamics, while paving the way for attaining optimal behaviour.

Lastly, Genetic Network Programming is a recent advancement over Genetic Programming [7], allowing for a population of individuals (genotypes) to be represented using directed graph structures. A phenotype is comprised of three node types, namely, a start node, processing nodes and judgement nodes. GNP was introduced in [7] and was previously fused with Fuzzy Logic and RL algorithms for learning a wall-following behaviour of a Khepera robot [1]. In [1], the fuzzy logic component was used merely for implementing the fuzzy judgement nodes of the Fuzzy GNP-RL algorithm. The degree of firing of each membership function in the judgement node is used as a measure of probability for selecting the next node during node transitions. Furthermore, the processing nodes tested in [1] did not make use of more sophisticated trained intelligent systems. Studies in the literature also indicate that a GNP-RL integration was investigated, but with findings showing less efficient exploration ability.

This work is an extension of the work presented in [1], and the main challenge addressed is that of building a hybrid evolutionary learning methodology for generating intelligence with multi-objective behaviours that allows for the utilisation of any pre-trained intelligent sub-systems as processing nodes inside a phenotype. We envisage that compounding the GNP with more powerful processing nodes would extend its computing prowess. As proof of concept, we demonstrate that the extended evolutionary system can learn multi-behaviours by testing it on the robot soccer domain to learn both target pursuit and wall avoidance simultaneously during training.

2 Methods

A schematic diagram of the new hybrid evolutionary learning algorithm is depicted in Fig.1.

2.1 Modified GNP Individual (Phenotype)

The first major change made to the original Fuzzy GNP-RL[1] is that the new proposed architecture can now accommodate any pre-trained complete intelligent systems as processing nodes within a GNP individual. A sample GNP individual is given in Fig.2, detailing its composition: start node, simple action-generator processing nodes, intelligent system processing nodes and judgement nodes. In the experiments, as a proof of concept, we have set some of the processing nodes to be a Fuzzy-RL system that was pre-trained to learn the ball pursuit behaviour. This intelligent processing node feeds on an input coming from the environment (i.e. angle of the robot from the ball) and outputs a precise steering angle for controlling the robot. The rest of the processing nodes were set to be simple action-generator nodes. During the GNP evolutionary learning phase, the Fuzzy-RL processing node is set to run only the Greedy policy for action selection. The rationale behind this approach is that if you have previously developed a well-trained intelligent system that can efficiently implement a particular behaviour, it can be readily integrated into the hybrid architecture as a processing node within a GNP individual without requiring any further training for that particular intelligent processing node. It is the role of the evolutionary mechanisms of the GNP to alter the connections between the nodes, and allow for mutations. On the other hand, during the testing phase, the Fuzzy-RL processing node employs only the e-greedy policy for action selection; thus, allowing for immediate adaptation to a dynamically changing environment.

The design of the composition of the processing nodes was suited to the problem domain. As can be viewed from an example in Fig.3, six different processing node types were defined corresponding to six different behaviours. The first four processing nodes are all related to wall avoidance actions, one per wall. On the other hand, the fifth processing node is for adjusting the speed accordingly, having the ball distance as input. Finally, the last processing node type corresponds to the the Fuzzy-RL algorithm trained for ball pursuit.

There are two types of judgement nodes used within the GNP individual (see the hexagonal nodes in Fig.3). The first one is used for judging the angle of the robot in relation to the ball, while the second type is used for finding which of the four walls is closest to the robot. An execution time of zero is set for the judgement nodes as they do not execute any robot actions - based on the decision made, the next node can be executed soon after the judgement node finishes. The problem specific settings can be viewed from Fig.3.

2.2 Node Execution Time Component

The second major change done to the system is the new implementation of the execution time (referred to as "'time delay"' in the original GNP [8]) of each node comprising a GNP individual. To illustrate how the execution time of nodes works, referring to the example in Fig.2, according to the time allotment, the Fuzzy-RL node will execute 5 complete times of ball pursuit behaviour before transitioning to the next node (which is the judgement node, in the example). It is important to note that the actual nodes executed within an individual GNP relies solely on what the robot is experiencing while interacting with the environment. The decisions selected by the judgement nodes dictate the flow of control. As for evaluating the performance of a GNP individual, there is a pre-defined maximum training time used.

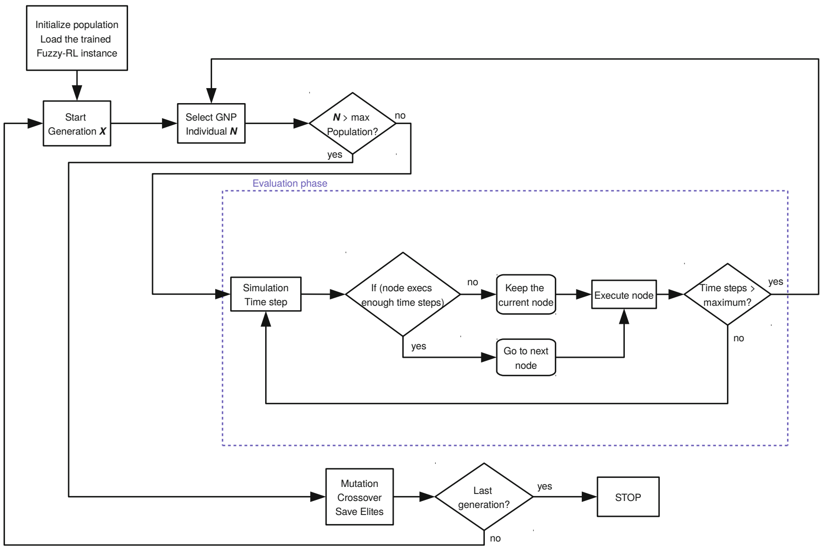

Fig. 1. Schematic diagram of the GNP with Fuzzy-RL node

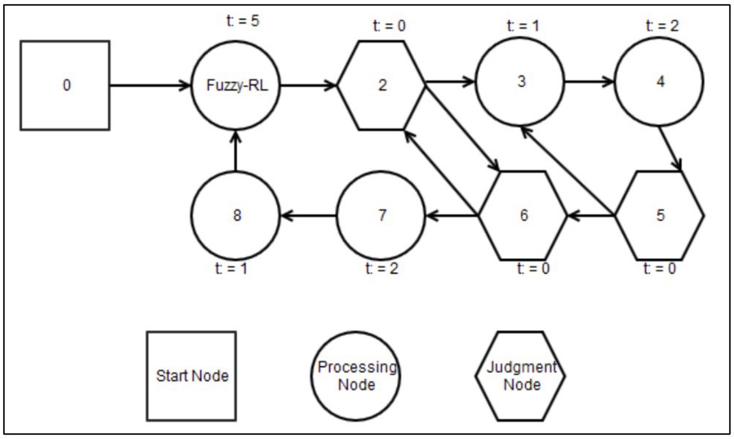

Fig. 2. A sample of the modified GNP individual used in this research

2.3 Fitness of a GNP Individual

The fitness function (Algorithm 1) was tailored specifically to the desired multi-behaviour, as it is the basis for selecting the best GNP individuals. It consists of three parts: the ball pursuit behaviour fitness, the speed control fitness and the wall avoidance fitness. Note that the parameters, (e.g. thresholds for Distance-FromBall and speed) can all be adjusted in the algorithm.

```
Require: Thresholds for: Speed $S_1, S_2, S_3, S_4$; Distance from ball $D_{b1}, D_{b2}, D_{b3}$;
    Fitness adjustment $F_{a1}, F_{a2}, F_{a3}, F_{a4}, F_{a5}, F_{a6}$; Distance from Wall $D_{w1}, D_{w2}$;
1: individual starts
2: $Fitness = 0$
3: repeat {each time step}
4:   if $DistanceFromWall < D_{w1}$ then
5:     if $AngleFromWall <= -90$ or $AngleFromWall > 90$ then
6:       $Fitness+ = (F_{a1} + BallPursuitReward)$
7:     else
8:       $Fitness+ = BallPursuitReward$
9:     end if
10:  end if
11:  if $(DistanceFromBall < D_{b1}$ and $speed == S_1)$ or
     $(D_{b1} <= DistanceFromBall < D_{b2}$ and $speed == S_2)$ or
     $(D_{b2} <= DistanceFromBall < D_{b3}$ and $speed == S_3)$ or
     $(DistanceFromBall >= D_{b3}$ and $speed == S_4)$ then
12:    $Fitness+ = F_{a2}$
13:  end if
14:  if $DistanceFromWall < D_{w2}$ then
15:    $Fitness- = F_{a3}$
16:  end if
17:  if Robot moves away from the ball then
18:    $Fitness- = F_{a4}$
19:  end if
20: until time steps exceeded the maximum training time for one individual
21: $Fitness+ = F_{a5} - F_{a6} * DistanceFromBall$
22: individual training ends
```

Algorithm 1. Fitness Function. The following thresholds were found empirically: $S_1=0.5, S_2=1.0, S_3=1.5, S_4=2.0, D_{b1}=10, D_{b2}=20, D_{b3}=50, F_{a1}=20, F_{a2}=6, F_{a3}=50, F_{a4}=15, F_{a5}=500, F_{a6}=10, D_{w1}=20, D_{w2}=15$.

Lastly, a hill climbing algorithm is employed in tandem with the evolutionary learning of GNP (see line 15 of Algorithm 2). Its job is to explore the different ways of connecting the nodes using a greedy approach, to further improve an individual's fitness. As this greedy search is very time-consuming, it is employed only to a few top GNP individuals, after every 10 generations, or so. This can also be employed when the top fitness of the population is no longer improving after a number of generations.

2.4 Experiment Results and Discussion

The experiments were carried out using our own simulation of the robot soccer platform, using simplified physics, which includes both collision and friction. Utilising the proposed fitness function (Algorithm 1), and the following GNP parameters: population size: 200, number of mutations: 77, number of crossovers: 120, tournament size: 5, probability of mutation: 0.1 and probability of crossover: 0.5.

```
1: Load trained Fuzzy-RL instance into the GNP individual
2: Initialise the population
3: **repeat** {for each generation}
4:   **repeat** {for each individual}
5:     **repeat** {for each time step}
6:       Execute current node
7:       Update environment
8:       Update fitness of individual
9:       **if** there is enough time steps for this individual **then**
10:        go to next node
11:      **end if**
12:    **until** time steps exceeded the max value of individual
13:    Calculate final fitness of individual
14:  **until** all individuals have been evaluated
15:  Apply hill-climbing algorithm for elites {e.g., top 3 individuals}
16:  Keep elites, select more individuals using tournament selection
17:  Apply Mutation Operation
18:  Apply Crossover Operation
19: **until** maximum generation is reached
```

Algorithm 2. GNP with Trained Fuzzy-RL

The hybrid evolutionary algorithm was run for at least 50 generations onwards, per experiment.

Our first initial training results did not readily give us a good solution. Even after 200 generations, the best individual ran in circles. This compelled us to increase the number of instances for each type of nodes within a GNP individual, to allow the GNP to generate more variations in the phenotypes.

Consequently, after many experiments, it was observed that identifying the final best GNP individual necessitates picking at least the top 5 elites and evaluating them visually. The best performing individual that completely succeeded in meeting the desired behaviour; that is, ball pursuit with wall avoidance behaviour and automatic speed adjustment, garnered a fitness of at least 4,000, using Algorithm 1. Interestingly, as depicted in Fig. 4, the best trained robot never makes a mistake of colliding against any of the four walls, while following the ball. The robot is the red square (with yellow lines) and the ball is the little red circle.The dotted red line indicates the path of the ball, while the dotted white lines indicate the path of the robot. Although it is difficult to see in the static pictures, the robot smoothly follows the ball and adjusts speed very rapidly. The space between the dots allows for tracing the variations in speed. In general, the robot behaves in such a way that when the ball gets too close to one of the walls, the robot avoids it; nevertheless, it kept trying to hit the ball, in a somewhat Brownian movement in the region where the ball is. A recorded video of the experiment testing how well the trained robot behaves can be viewed from the following link: http://youtu.be/woqMnbO-CKg

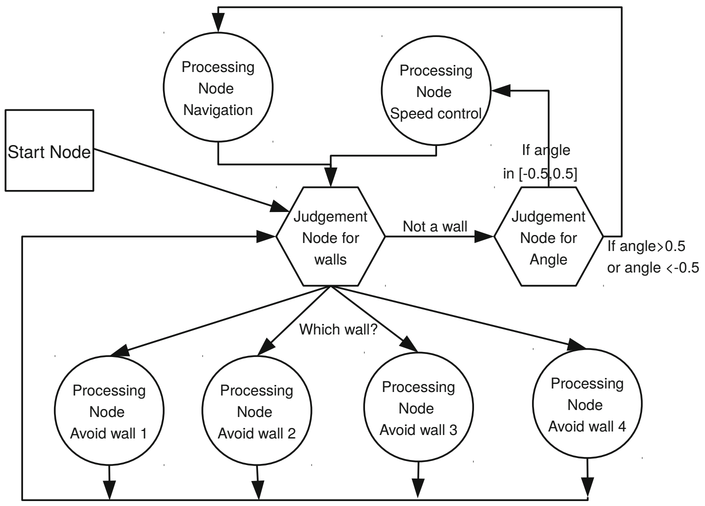
Start Node
Processing Node Navigation
Processing Node Speed control
Judgement Node for walls
Not a wall
Judgement Node for Angle
If angle in [-0.5,0.5]
If angle>0.5 or angle <-0.5
Which wall?
Processing Node Avoid wall 1
Processing Node Avoid wall 2
Processing Node Avoid wall 3
Processing Node Avoid wall 4

Fig. 3. Sample GNP individual with minimum number of nodes

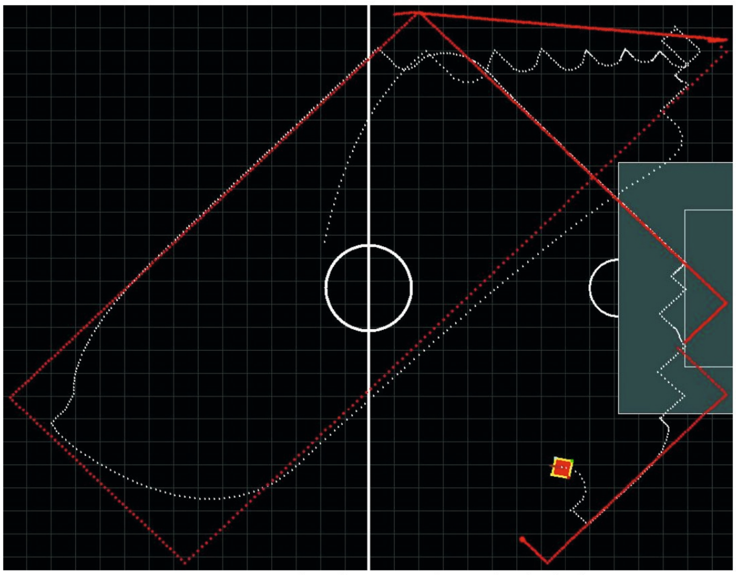

Fig. 4. An example of the general performance of a good individual

3 Conclusion

This work proposes a new hybrid evolutionary learning methodology that allows for the integration of any pre-trained intelligent system into the previous Fuzzy GNP-RL architecture presented in [1]. The proposed modifications strengthens the computing capabilities of the hybrid architecture as it caters for more complex processing nodes in the composition of the GNP individual. There is virtually no limitations to the different types of pre-trained intelligent systems that can be employed as processing nodes in the new architecture. The experiments performed show strong evidences that the new hybrid system can learn multiple robot behaviours with exceptional results. We envisage that using the new architecture, more sophisticated robot learning problems can be solved.

References

1. Sendari, S., Mabu, S., Hirasawa, K.: Fuzzy genetic network programming with reinforcement learning for mobile robot navigation. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Anchorage, Alaska, USA, pp. 2243–2248. IEEE (2011)
2. Reyes, N.H., Barczak, A.L., Susnjak, T., Sincák, P., Vaščák, J.: Real-Time Fuzzy Logic-based Hybrid Robot Path-Planning Strategies for a Dynamic Environment. In: Efficiency and Scalability Methods for Computational Intellect, pp. 115–141. IGI Global (2013)
3. Vascák, J., Reyes, N.H.: Use and perspectives of fuzzy cognitive maps in robotics. In: Papageorgiou, E.I. (ed.) Fuzzy Cognitive Maps for Applied Sciences and Engineering. ISRL, vol. 54, pp. 253–266. Springer, Heidelberg (2014)
4. Reyes, N.H., Dadios, E.P.: Dynamic color object recognition using fuzzy logic. JACIII 8(1), 29–38 (2004)
5. Gerdelan, A., Reyes, N.: Synthesizing adaptive navigational robot behaviours using a hybrid fuzzy a* approach. In: Reusch, B. (ed.) Computational Intelligence, Theory and Applications, vol. 38, pp. 699–710. Springer, Heidelberg (2006)
6. Reyes, N.H., Barczak, A.L.C., Susnjak, T.: Tuning fuzzy-based hybrid navigation systems using calibration maps. In: Kim, J.-H., Matson, E., Myung, H., Xu, P. (eds.) Robot Intelligence Technology and Applications. AISC, vol. 208, pp. 713–722. Springer, Heidelberg (2013)
7. Hirasawa, K., Okubo, M., Katagiri, H., Hu, J., Murata, J.: Comparison between genetic network programming (gnp) and genetic programming (gp). In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 1276–1282 (2001)
8. Katagiri, H., Hirasama, K., Hu, J.: Genetic network programming - application to intelligent agents. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 5, pp. 3829–3834 (2000)