Simulation of Bipedal Robot by Computational Intelligence Algorithm

Haoming Zhang *zanhoming@gmail.com*
Yu Cai *caiyu0227@g.ucla.edu*
Youyang Luo *luo2016@g.ucla.edu*
Dui Lin *dui@g.ucla.edu*
Sheng Zhang *shengzhang@ucla.edu*

***Abstract*—In this paper, we evolve a bipedal robot using Open AI gym and Python. The bipedal robot can keep itself in a balance state and walk far distance without falling down. The environment is set by Box2D open source physics engine and the platform is Open AI gym. The locomotion of the robot is controlled and learned by reinforcement learning which implemented by the neural network and evolved by the genetic algorithm. The proposed approach results in an effective training with relatively less generations. It produces good results and ensures the optimized performance of the robot.**

***Index Terms*—Bipedal Walker, OpenAI gym, Reinforcement Learning, Neural Network, Genetic Algorithm**

1. Introduction

Since recent years, the designation and construction of bipedal robot is a promising field which attracted lots of attention; there are several good reasons to develop it despite the fact that the implementation of the algorithm to control its locomotion is more difficult comparing with controlling traditional robots such as wheeled robot. First, bipedal robots are able to move in areas that are normally inaccessible to wheeled robots. Second, comparing with wheeled robot, walking robots cause less damage on the ground. Third, bipedal robots are easier for people to interact with[1]. However, considering the expense of constructing a real bipedal robot, simulating it using computational intelligence method is a good way to optimize its performance before transplanting it to a real one.

The bipedal robots are dynamic subactuated systems[2] with high complexity. Generally, there are two parts we should control: the locomotion of the bipedal robot and the evolution and parameter selection of the control algorithm to let the robot walking and keep its stability. Thus, a hybrid model should be introduced, which uses mechanical modeling for position and speed control and the computational intelligence methods for the selection of parameters of the control algorithm.

There are several methods which had been conducted to attain a satisfactory solution, including neural network(NN)[3-5], fuzz systems[6] reinforcement learning(RL)[7] and genetic algorithm(GA)[8]. In this paper, our model is simulated based on the framework of RL, making the robot towards true autonomy. Within this framework, we will use different algorithms for different parts of the control system.

Considering the first part, the bipedal locomotion control, which is a challenging issue that had intrigued researchers for a number of years. The first successful solutions emphasized walking with static balance, passing through a succession of states of static equilibrium. This resulted in undesirable constraints on both the biped structure and walking efficiency. Generally, static bipedal walkers had large feet and moved slowly[9, 10]. However, if walking with dynamic balance, the projected center of mass is allowed outside of the area inscribed by the feet, and the walker may falling during parts of the gait cycle. To solve this problem, in this paper, for the locomotion control part, we implement the evolving NN algorithm. The advantages are obvious[11]: First, NN is robust to a noise present in interactions between robots and environment. Second, synaptic weights and thresholds make the primitive components manipulated by the evolutionary process set to be at the lowest level possible. Thus, undecidable choices made by the human designer can be avoided. Third NN's generalizing abilities can complement the gap between the virtual environment and the real world; if we want to transplant the virtual robot on a real one, using NN is a good choice.

The second part is the evolution and parameter selection of the control algorithm; for any control method, some completely or partially specified joint trajectory or gait is needed. The nominal trajectory may mimic human walking[12, 13] and the gait may be specified in some intuitive way[14]. Selection of the control parameters and nominal trajectory determines the quality of control. In the last two decades, lots of work focusing on the selection of control parameters had been done[8, 14-16]. Many earlier works just select some or all of the parameters intuitively, making this work time consuming and the result is often not optimum. If some special goal (such as to walk as fast as possible or walking on all kinds of terrains) is desirable, the design may become even harder. To avoid this problem and optimize the performance of the robot, we use the GA to select parameter and evolve the locomotion control system which based on NN. Using GA, the design of the biped controller and gaits can be formulated as parameter search problem[16]; the goal of our designation is walking as far as possible without falling down, as the target the NN algorithm will evolve to.

While the interaction between the robot and the environment is essential for the designation of a virtual robot,

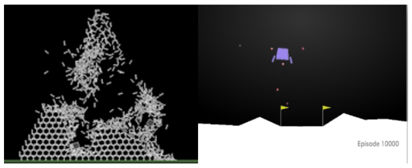

Figure 1: 2D Plots Simulated by Box2D

selecting a suitable platform which can offer satisfactory environment is very important. In this paper, we implement our algorithm based on the platform Open AI gym, a toolkit designed for reinforcement learning research[17]. This platform combine the best elements of previous benchmark collections, including Arcade Learning Environment (ALE)[18], RLLab benchmark for continuous control[19] etc. The environment is implemented by the Box2D physical engine, an open source C++ 2D rigid body simulation library for procedural animation[20]. This engine is versatile, as shown in Fig.1, it can simulate very sophisticate dynamic 2D plots. It has some features which suitable for our simulation work, including the continuous collision detection, continuous physics with time of impact solver and persistent body-joint-contact graph. Although there are some limitations like stability degrades or joint stretching in some special cases, it can totally satisfy the need of our work.

2. Bipedal Walker Model Structure

Our bipedal robot is simulated via computed torque control, the structure is shown in Fig.2, there are five parts of the robot: the body, two upper legs and two lower legs, each part is conjuncted by joints circled in red. The locomotion of the robot origin from those four joints; by changing the torque of each joint, the robot can move forward and keep balance. The robot can get the information from environment via the sensor, the red arrow shown in Fig.2. The sensor mainly collect information of the terrace. By interacting with the environment, the locomotion of the joints will be modified, letting the robot to perform better.

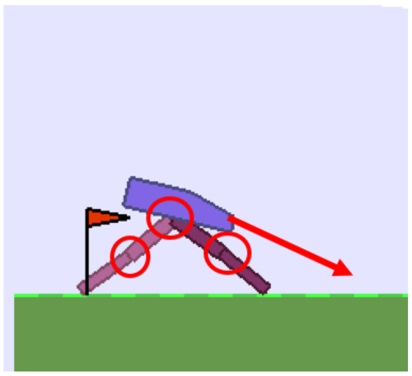

Figure 2: Structure of the Simulated Bipedal Robot

3. Reinforcement Learning

RL is a branch of Artificial Intelligence. It is a technique useful in solving control optimization problems. It allows agents to make the optimal decisions within a specific context to maximize its performance. The procedure is shown in Fig.3. A reward feedback is required for the agent to

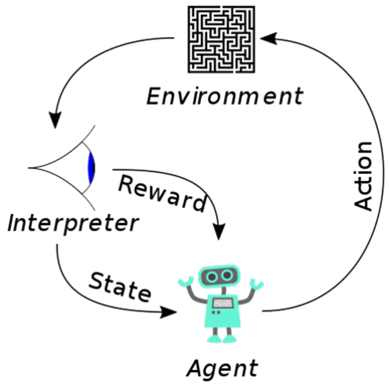

Figure 3: Procedure of Reinforcement Learning

learn its behavior, which is known as the reinforcement signal. A typical flow of RL is: an agent takes actions in an environment, and an observer will interpret into a reward and a representation of the state which is fed back into the agent. In our project, the learning process of the bipedal walker is implemented using combination of a feed forward NN and a GA.

3.1. Artificial Neural Network

Artificial Neural Network (ANN) is a type of computational model that emulates the structure and function of biological neural network. ANN is a non-linear model that can explores the complex correlation between inputs and outputs. An ANN can consist of thousands of neurons. The first layer has input neurons which send data via synapses to second layer of neurons. Complex network systems will have more layers of neurons. Typically, an ANN is determined by three types of hyper-parameter:

1) The interactions pattern between different layers of neurons.
2) The weights of the interconnections, which will be optimized in the learning process.
3) The activation function that convert a weighted sum of previous layer of neurons to an

Normally, a widely used and simple interaction of neurons in a feed forward neural network can be expressed as follows:

$$h_{W,b}(x) = f(W^T x) = f(\sum_i^n W_i x_i + b)$$

where $f : R \rightarrow R$ is an activation function, is the weight vector and is the input vector of one layer, as shown in Figure 4.

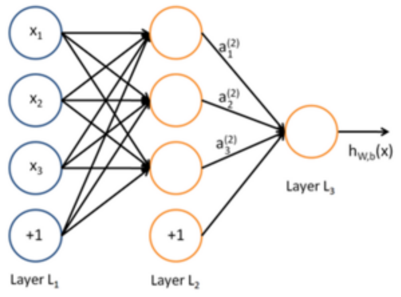

Figure 4: A Simple Neural Network

In general, the learning process can be divided into three paradigms: supervised learning, unsupervised learning and reinforcement learning. In this project, the neural networks will be rated with a fitness score and the hyper-parameters are optimized though Genetic Algorithm, which in the scope of reinforcement learning.

3.2. Genetic Algorithm

Genetic Algorithm (GA) is a search algorithm used to solve both constrained and unconstrained optimization problem, which belongs to evolution algorithms (EA). The evolution starts with a population with randomly generated individuals, it's an iterative procedure where each individual in a generation will be evaluated with a fitness score. To mimic the real evolution process, the individuals with greater fitness have higher probability of survival and the new generation will be generated through crossover and mutation. With these artificial process (selection, crossover and mutation), the population will evolve towards best traits. Such process will be repeated over and over again, until certain termination conditions, such as a solution that meets criteria will be found or a maximum number of generation will be reached.

4. Implementation

In our project, a feed forward neural network is used to control the bipedal walker. The agent includes 4 action parameters(each parameter corresponding to a joint) and will return 24 observation parameters so that the input level and the output level of neural network has 24 and 3 neurons respectively. The input observation features will be learned iteratively through hidden levels and after a forward propagation among hidden neurons, the action-control parameters will be given out from the 4 output neurons to guide the locomotion regime of the agent. A fitness (reward) will be evaluated based on the distance and the speed of the locomotion. The numbers of hidden levels and the neurons in each level are chosen artificially based on empirical rules and they are fixed throughout the project. The main factors that we tune are the values of weights and biases and the activation function. The learning process of the weights and biases will be discussed in this section and the results of different activation function will be compared in the experiment section. At the beginning, the weights and biases of the first population are generated randomly, ranging from -1 to 1. Each neural network will then be evaluated under the physical model. Thus, the probability of survival of individual linearly decreases as the fitness decreases and in the crossover step, the more fit individuals will be more dominating. Mutation will occur at mutation rate. The flowchart of the training process is shown as Fig. 5:

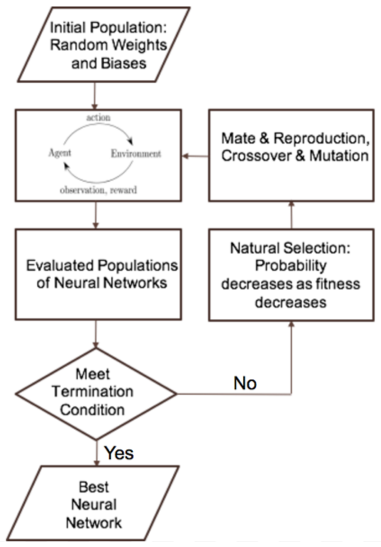
Initial Population: Random Weights and Biases
action
Agent
Environment
observation, reward
Mate & Reproduction, Crossover & Mutation
Evaluated Populations of Neural Networks
Natural Selection: Probability decreases as fitness decreases
Meet Termination Condition
No
Yes
Best Neural Network

Figure 5: Flowchart of Training Process

5. Experiment

5.1. Video Demonstration

A video demonstration is available at the authors website for the project, index.html.

The video starts by the first generation of the agent. The agent cannot walk a single step and even cannot stand stability. Next, the agent starts to gain information of the environment by getting rewards from the previous episode, and according to the rewards or fitness to adjust the mutation of the neural network inside the agent, making the agent mutate toward a state that is fitting the environment. Though for a long time, the agent still cannot stand and just falls down, it finally learns to maintain its balance by moving one leg forward and the other leg kneeing down. Then it struggles walking forward due of the stability of the mentioned pose which turns out to be a triangle. The environment finally resets due to the elapse of the valid time. During the valid time, the agent tries to both maintain balance and moving forward, yet it's difficult to do so. Therefore the agent is stuck this state for a long period of time, i.e. uses lots of episodes to overcome this state. Finally, the agent can moving forward by one leg stretching forward and the other leg kneeing down and following the former leg. The agent eventually learns to move forward steadily.

5.2. Performance

Aside from the visual perception of the agent walking in the output demo, we can evaluate the model by the performance of fitness, which is defined as the performance of both the locomotion distance and the walking speed. Fig.6 shows the maximum fitness among one generations neural networks. The first generations maximum fitness is -75, while the 50th generations maximum fitness is increasing to 109. As the result, it shows on the video demo that the agent start from being unstable to being able to move forward steadily. In the meantime, though the maximum fitness fluctuates, it grows up by and fluctuate around a large value. The fluctuation is due to the algorithm picking random values in the mutation process. We also compare the effect of using different activation functions, hyperbolic tangent function, linear function and sigmoid function.

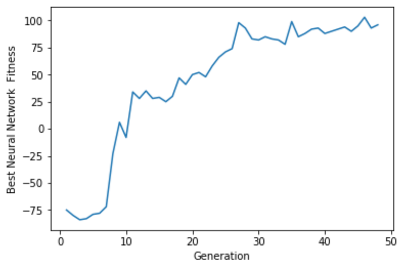

Figure 6: Maximum Fitness of Each Generation

From the corresponding video demos, the one using sigmoid function is the worst. It cannot moving forward steadily. The other two perform similarly, both moving forward steadily eventually. Fig.7 shows the comparison of the maximum fitness of the different activation functions.

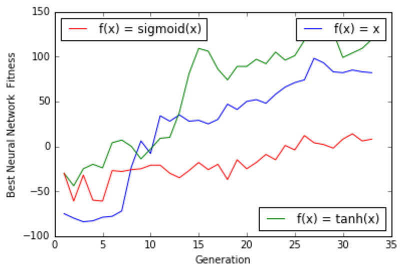

Figure 7: Comparison of Different Activation Functions' Maximum Fitness of Each Generation

The tendency of models with the three functions are all increasing. But with the same generation, the maximum fitness relationship among them is

$$Fitness(tanh) > Fitness(linear) > Fitness(sigmoid)$$

Therefore, the better activation function among them is the hyperbolic tangent function. As for the run time, the time for hyperbolic tangent function, linear function, sigmoid function to converge to a stable fitness are 3 hours, 8 hours and 4 hours. Therefore the best run time belongs to hyperbolic tangent function. In conclusion, according to the visual perception, maximum fitness and runtime, the best activation function selection is hyperbolic tangent function.

6. Conclusion

In this paper, we conducted the simulation of a bipedal walker. The bipedal walker contains 4 action-control parameters and 24 observation parameters and is controlled through Neural Networks. The hyper-parameters of Neural Networks is optimized using Genetic Algorithm so that the bipedal walker can walk without falling down in a steady fashion. Among different activation functions, *tanh* function enables the system to converge in the minimum amount of time.

In the future, effort could be devoted into tuning the structure of hidden levels of the neural network and adapting the bipedal walker to move in a more complex environment.

Acknowledgments

At the point of finishing this project, we would like to express our sincere thanks to all those who have lent me hands in the course, especially to Prof. Demetri Terzopoulos, who has given us so much inspiring ideas on our project.

References

[1] Takemasa Arakawa and Toshio Fukuda. Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization. In *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, volume 2, pages 1495–1500. IEEE.

[2] Marc Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Erin Catto. Box2d: A 2d physics engine for games, 2011.

[5] MY Cheng and CS Lin. Genetic algorithm for control design of biped locomotion. *Journal of Field Robotics*, 14(5):365–373, 1997.

[6] Yunduan Cui, Takamitsu Matsubara, and Kenji Sugimoto. Local update dynamic policy programming in reinforcement learning of pneumatic artificial muscle-driven humanoid hand control. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 1083–1089. IEEE.

[7] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

[8] Hooshang Hemami and B t Wyman. Modeling and control of constrained dynamic systems with application to biped locomotion in the frontal plane. *IEEE Transactions on Automatic Control*, 24(4):526–535, 1979.

[9] Mircea Hulea and Constantin Florin Caruntu. Spiking neural network for controlling the artificial muscles of a humanoid robotic arm. In *System Theory, Control and Computing (ICSTCC), 2014 18th International Conference*, pages 163–168. IEEE.

[10] Emanuil Huluta, Ricardo Freire da Silva, and Thiago Eustaquio Alves de Oliveira. Neural network-based hand posture control of a humanoid robot hand. In *Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2014 IEEE International Conference on*, pages 124–128. IEEE.

[11] Yildirim Hurmuzlu. Dynamics of bipedal galt: Part ii-stability analysis of a planar five-link biped. *TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF APPLIED MECHANICS*, 60:337–337, 1993.

[12] Yang Liang, Liu Zhi, and Zhang Yun. Adaptive fuzzy yaw moment control of humanoid robot based on ankle joint. In *Control Conference (CCC), 2015 34th Chinese*, pages 5999–6004. IEEE.

[13] Zhi Liu, Ci Chen, Yun Zhang, and CL Philip Chen. Adaptive neural control for dual-arm coordination of humanoid robot with unknown nonlinearities in output mechanism. *IEEE transactions on cybernetics*, 45(3):507–518, 2015.

[14] M Masubuchi. Control of a dynamical biped locomotion system for steady walking. *Journal of Dynamic Systems, Measurement, and Control*, 108:111, 1986.

[15] M Masubuchi. A theoretically motivated reduced order model for the control of dynamic biped locomotion. *Journal of Dynamic Systems, Measurement, and Control*, 109:155, 1987.

[16] Kenichiro Nagasaka, Atsushi Konno, Masayuki Inaba, and Hirochika Inoue. Acquisition of visually guided swing motion based on genetic algorithms and neural networks in two-armed bipedal robot. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 2944–2949. IEEE.

[17] Marc H Raibert. Legged robots that balance. 1985.

[18] David J Todd. *Walking machines: an introduction to legged robots*. Springer Science Business Media, 2013.

[19] Carlos Magno CO Valle, Ricardo Tanscheit, and Leonardo A Forero Mendoza. Computed-torque control of a simulated bipedal robot with locomotion by reinforcement learning. In *Computational Intelligence (LA-CCI), 2016 IEEE Latin American Conference on*, pages 1–6. IEEE.

[20] Yujiang Xiang, Jasbir S. Arora, and Karim Abdel-Malek. Physics-based modeling and simulation of human walking: a review of optimization-based and other approaches. *Structural and Multidisciplinary Optimization*, 42(1):1–23, 2010.