Can Chaos be utilized as exploration noise for locomotion learning?

Worasuchad Haomachai[1] and Poramate Manoonpong[1,2*]
[1]Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2]Vidyasirimedhi Institute of Science & Technology, Rayong, Thailand
*haomachai@gmail.com, *poramate.m@vistec.ac.th*

1 Introduction

There is compelling evidence to support that chaotic patterns of behavior exist in many biological systems [1,2]. For example, Maye et al. [2] observed behavioral indeterminacy (comparable to a chaotic pattern) during spontaneous flight maneuvers (searching behavior without any external cues) in Drosophila fruit flies. This suggests that chaotic dynamics may be involved in the biological neural control underlying spontaneous behavior. It also raises the question, "Can chaos be utilized in artificial neural control for robot locomotion learning?" To address the question, this study investigates and compares the use of chaotic exploration noise and standard Gaussian noise for robot locomotion learning. Although chaos has been used to tackle machine learning problems (such as classification) [3], until now, it is yet to be thoroughly explored for locomotion learning.

2 Materials and Methods

For our investigation, we construct a locomotion controller as a reinforcement learning framework, so that our robot (here, a gecko-like robot) has to learn to walk. The controller, configured as a neural central pattern generator (CPG) with a radial basis function (RBF)-based premotor neuron network (Fig. 1), is based on our previous work [4]. In this controller, the robot joint trajectories are encoded in the output weights connecting the CPG-RBF network to the motor neurons (dashed lines in Fig. 1). The output weights are learned using a probability-based black-box optimization (BBO) approach to optimize joint trajectories with respect to robot walking performance.

Our simulated gecko-like robot consists of a bendable body with three active joints for lateral body undulation and four identical legs, each with four active joints for leg forward/backward motion ($j1$, Fig. 1), leg elevation/depression ($j2$, Fig. 1), foot attachment/detachment ($j3$, Fig. 1), and leg flexion/extension ($j4$, Fig. 1). The body joints and leg joints are controlled separately. For simplicity, we employ a CPG signal with predefined connections to drive the three body joints ($b1, b2, b3$) to achieve a C-shaped standing wave pattern (see Fig. **??**, left), while the trajectories or patterns of all 16 leg joints ($j1, j2, .., j16$) are optimized by BBO.

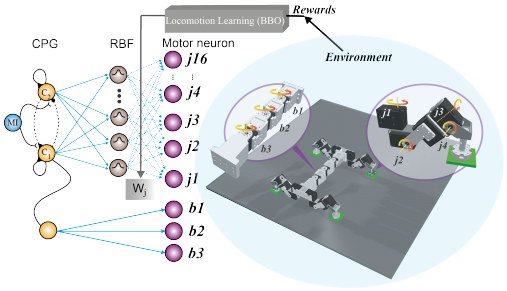

Figure 1: An overview of the CPG-RBF control network with BBO applied to the gecko-like robot. The robot simulator CoppeliaSim and physics Vortex are used for robot simulation.

2.1 Locomotion Control Network

The whole network consists of three layers. For the input layer, a CPG based on a two-neuron recurrent network is used to produce two sinusoidal-like signals. These CPG output signals are fed into the hidden layer which consists of 20 RBF neurons. The RBF neurons are equally distributed along one period of the CPG output signals. Subsequently, RBF outputs are projected to all motor neurons for controlling robot motor joint positions. According to the setup, the network's output weights can be optimized via BBO to shape and translate the original CPG sinusoidal-like signals into final complex joint trajectories that yield great robot walking performance.

2.2 Locomotion Learning

Algorithm 1 BBO

while *cost not converged* **do**
  for $k \in K$ **do**
    $R_k = execCPGRBFN(w_{k,j} + \varepsilon_k)$
  end
  for $k \in K$ **do**
    $S_k = e^{\lambda \cdot \frac{R_k - \min(R)}{\max(R) - \min(R)}}$
    $P_k = \frac{S_k}{\sum_{k=1}^{K} S_k}$
  end
  $\delta w_{k,j} = \sum_{k=1}^{K} P_k \cdot \varepsilon_k$
  $w_{k,j} \leftarrow w_{k,j} + \delta w_{k,j}$
  $\varepsilon_k \leftarrow \gamma \cdot \varepsilon_k$
end

The BBO method, used here called PI^{BB}, is a parameter perturbation approach and a variant of an evolutionary algorithm. It is employed to optimize the output weights in order to generate optimal joint trajectories that maximize a reward, which basically describes how well the robot performs (Fig. 2A). The pseudocode of BBO can be seen in Algorithm 1. BBO is a probability-based ap-