Motor learning in the game QWOP

Matt Sheen* and Andy Ruina**
Cornell University, Ithaca, NY, USA
* *mws262@cornell.edu*
** *ruina@cornell.edu*

1 Motivation

To the general public, the problem of walking coordination seems like it must be intuitive. We know that locomotion control is difficult, but this can be hard to demonstrate in a hands-on way. Several years ago, a "running simulator" game called QWOP exploded on the internet (30 million hits), becoming notorious for its difficulty despite seemingly simple controls. It received considerable media attention and spawned Guinness World Record competitions.

We attempt to beat the world record time by learning a policy for controlling QWOP. This game gives rise to quite a few interesting challenges including: locomotion controller design with black box dynamics, control optimization of a system which may only be forward simulated from a fixed initial condition, very low-dimensional control of a high-dimensional system, and dealing with unusual types of noise (e.g the game's internal clock cycles).

2 About QWOP

QWOP is a web browser-based (Adobe Flash) ragdoll physics game. The goal is to make a 2D cartoon athlete run 100m on a track (although most players won't get more than a few meters). The player uses the Q, W, O, and P keys on the keyboard to apply coupled torques on the legs. Q and W apply opposite torques to the thighs; O and P apply opposite torques to the calves. Arm and ankle torques are coupled to these actions, and the neck is free to move.

3 Approach

Several factors limit the approach for finding a good controller.

1. The game always starts with the runner at rest at the start line. We cannot optimize certain parts of a gait or search for initial conditions of a periodic motion.
2. We can see the state of the runner, but the dynamics are unknown.
3. The game's discrete physics steps make adjustments constantly necessary and the existence of purely periodic steps unlikely. We ran optimizations to find an open loop running gait. After an initial transient, the controls become nearly periodic. However, the discrete timesteps means that the commands are always too long or short.
4. Simulations run in real time only. We must be efficient in our function evaluations.

For these reasons, we are trying variants of Q-learning. Reinforcement learning lends itself well to this problem since we can do little besides observe successes and failures based on states and actions. Much like dynamic programming, this approach attempts to find a global value or utility of each state/action pair. Thus the optimal control is to then follow the trail of highest utility points. Dynamic programming usually propagates the utilities back from the solution which we cannot do in this problem. For Q-learning, we explore from the start point and gradually improve our estimate of the utility by observing good and bad results (forward motion and falling down). By adjusting the learning rates, we can begin to latch onto good solutions and avoid exploring the entire state space. Q-learning has recently been used to learn and beat records in classic Atari arcade games [1].

4 Implementation Details and Preliminary Work

Although source code is not officially available, we have decompiled the game in order to add code which can pass the runner's state from Flash to an applet running in the browser. This information is transmitted over a fast local connection (UDP) to MATLAB. Commands equivalent to the Q, W, O, and P keys are directly injected into the game. Due to large irregularities in the game's internal clock, all commands are timed based on the number of physics engine steps taken (also corresponds to the frame rate). For open loop optimization, the sequence of keystrokes mimics that of experienced players, but the precise timings are left as decision variables. In this case, the game becomes an integer programming problem which we perform using a version of CMA-ES (Covariance Matrix Adaptation - Evolutionary Algorithm [2]). We are able to get long open loop runs (roughly 40m so far) under these ideal circumstances, but since the game's internal clock varies constantly in the unhacked version of the game, we will need feedback to actually break the record.

For reinforcement learning of a policy, we are currently using the state of the torso + a phase of the legs as a proxy for the full state. This is based on the intuition that torso angle and height seems to be the primary attributes good players look

at to adjust controls. Rewards for individual moves are based on horizontal distance traveled since the last action. We are using a discretized state to look up controls. This approach has preliminarily yielded runs 15m. The discretization needs much adjustment, and we may find it necessary to use a function approximator like an artificial neural network.

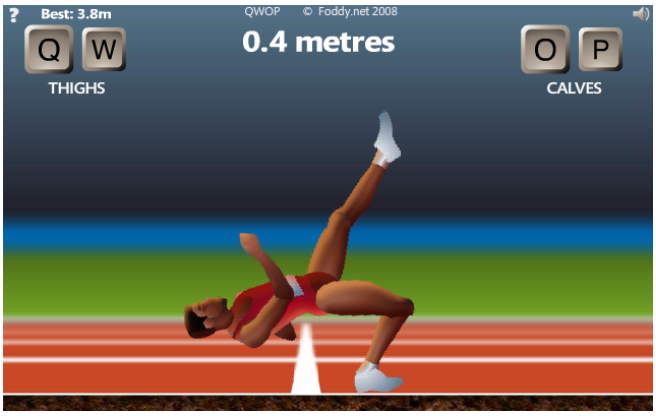

Figure 1: A screenshot from the game QWOP.

References

[1] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[2] Nikolaus Hansen. A CMA-ES for Mixed-Integer Nonlinear Optimization. [Research Report] RR-7751, 2011