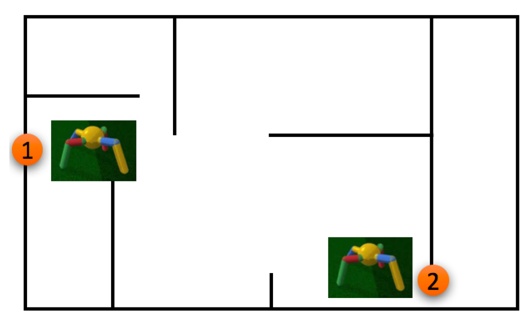
Policy Optimization by Genetic Distillation

Tanmay Gangwani, Jian Peng

Computer Science Department, University of Illinois, Urbana-Champaign

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

A. Motivation and Goals



- Distill knowledge from *locally* well-behaved agents into a single *globally* well-behaved agent.
- Start with a population of agents and gradually merge policies over rounds of a genetically-inspired iterative algorithm.

B. GPO Algorithm

population $\leftarrow \pi_1, \ldots, \pi_m$
repeat
  population $\leftarrow$ MUTATE(*population*)
  parents_set $\leftarrow$ SELECT(*population*, fitness)
  children $\leftarrow$ empty set
  for tuple$(\pi_x, \pi_y) \in$ *parents_set* **do**
    $\pi_c \leftarrow$ CROSSOVER(π_x, π_y)
    add π_c to *children*
  end for
  population $\leftarrow$ *children*
until k steps of genetic optimization

C. Crossover Operator

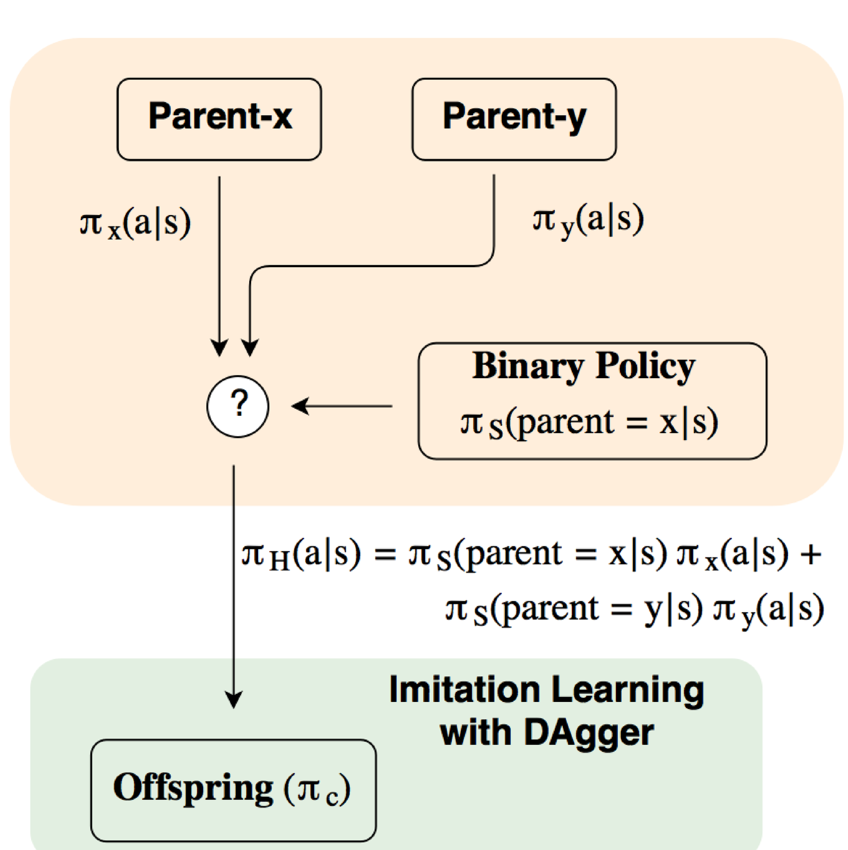


Figure: Schema for combining parent policies to produce an offspring policy. The two-level policy (orange box) is used as the expert for imitation learning, wherein the KL-divergence between the expert and the offspring is minimized.

D. Contrast with Parameter Crossover

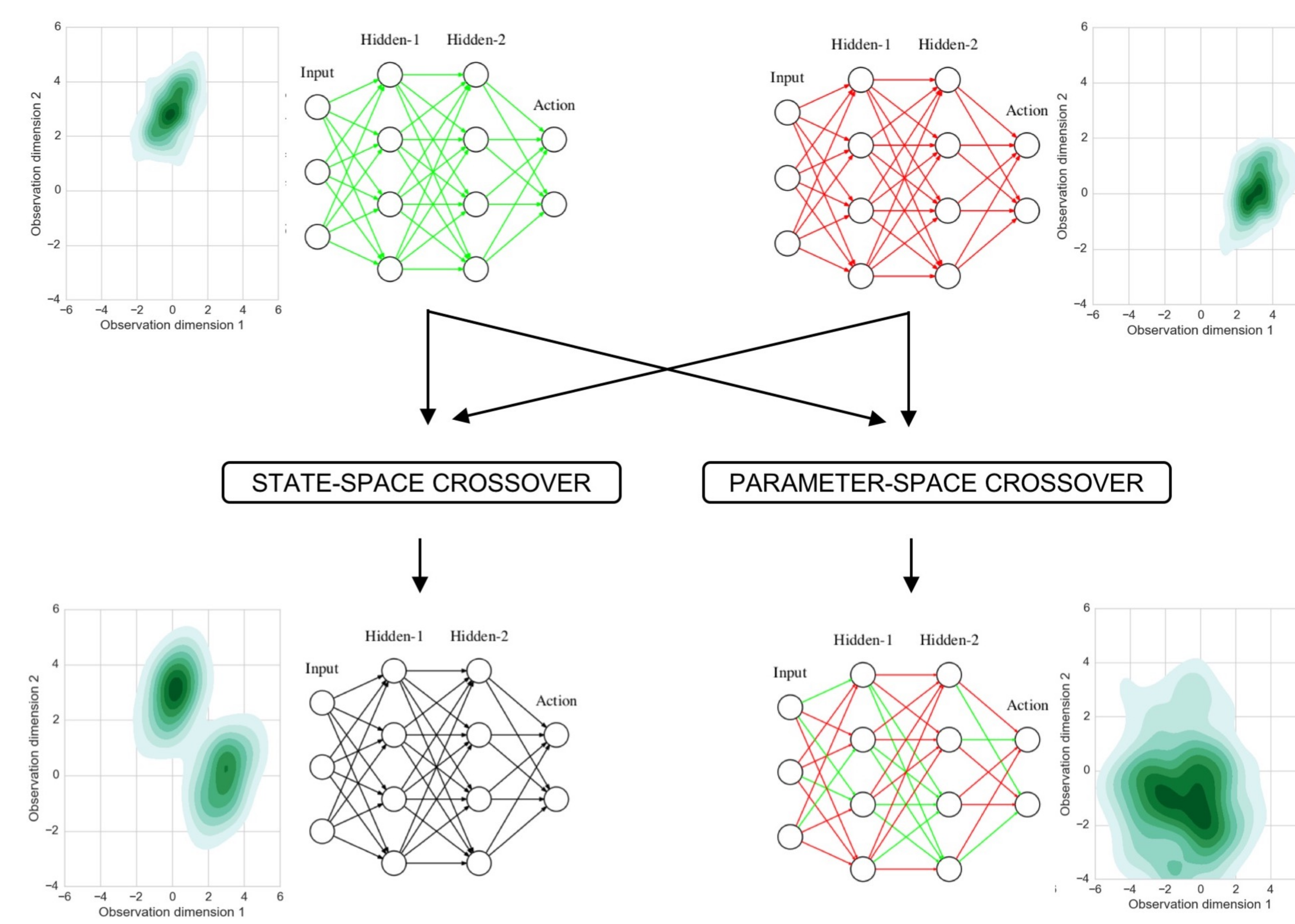


Figure: Different crossover strategies for neural network policies. State-visitation distribution plot next to each policy depicts the slice of state-space where that policy gives high returns. In a naïve approach like parameter-space crossover (shown in bottom-right), edge weights are copied from the parent network to create the offspring. Our proposed state-space crossover operator, instead, aims to achieve the behavior shown in bottom-left.

E. MUTATE and SELECT Operators

MUTATE perturbs the parameters of the neural network policy. Instead of random perturbations, we use standard policy-gradient algorithms (PPO, A2C) to move the parameters in the direction of the noisy gradients approximated from sampled trajectories.

Data Sharing: When mutating multiple policies in parallel, a policy π_i can also use data samples from other *similar* policies for off-policy learning. For example, with the PPO objective, the modified gradient for π_i is

$$\nabla_{\theta_i} L^{PPO}(\theta_i) = \left(\sum_{j \in \mathbb{S}_i} \hat{\mathbb{E}}_{j,t} \left[\frac{\nabla_{\theta_i} \pi_{\theta_i}(a_t|s_t)}{\pi_{\theta_j^{(old)}}(a_t|s_t)} \hat{A}_t \right] \right) - \nabla_{\theta_i} \hat{\mathbb{E}}_{i,t} \left[\beta KL[\pi_{\theta_i^{(old)}}(.|s_t), \pi_{\theta_i}(.|s_t)] \right]$$

where $\mathbb{S}_i \equiv \{ j \mid KL[\pi_i, \pi_j] < \epsilon$ before the start of current round of mutation$\}$ contains similar policies to π_i (including π_i).

SELECT chooses policies-pairs $\{\pi_x, \pi_y\}$ with high fitness for the crossover step. Different fitness functions are possible:

- Performance fitness as sum of expected returns of both policies, i.e.

$$f(\pi_x, \pi_y) \stackrel{\text{def}}{=} E_{\tau \sim \pi_x}[R(\tau)] + E_{\tau \sim \pi_y}[R(\tau)]$$

- Diversity fitness as KL-divergence between policies, i.e.

$$f(\pi_x, \pi_y) \stackrel{\text{def}}{=} KL[\pi_x, \pi_y]$$

Link to our paper - `https://arxiv.org/abs/1711.01012`
Contact details - gangwan2@illinois.edu, jianpeng@illinois.edu

F. Crossover Performance

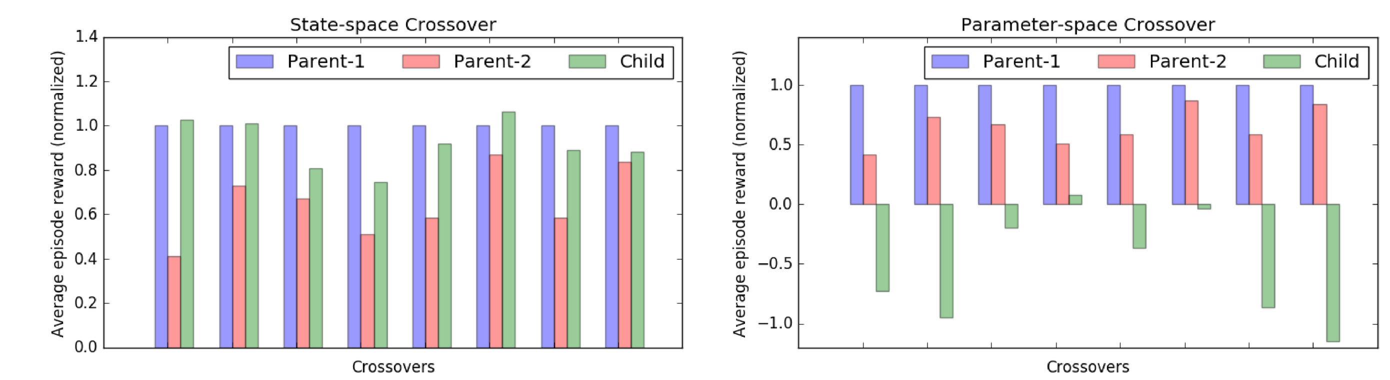


Figure: Average episode reward for the child policies after state-space crossover (left) and parameter-space crossover (right), compared to the performance of the parents. All bars are normalized to the first parent in each crossover. Policies are trained on HalfCheetah.

G. Comparison with Baselines

- Implementation is based on OpenAI rllab framework. We benchmark continuous control locomotion tasks based on MuJoCo.
- GPO is run for 12 rounds with a population of 8 policies, and simulates 8 million timesteps in total for each environment.
- The first baseline algorithm, **Single**, trains 8 independent policies with policy gradient using 1 million timesteps each, and selects the policy with the maximum performance at the end of training
- The second baseline algorithm, **Joint**, trains a single policy with policy gradient using 8 million timesteps.

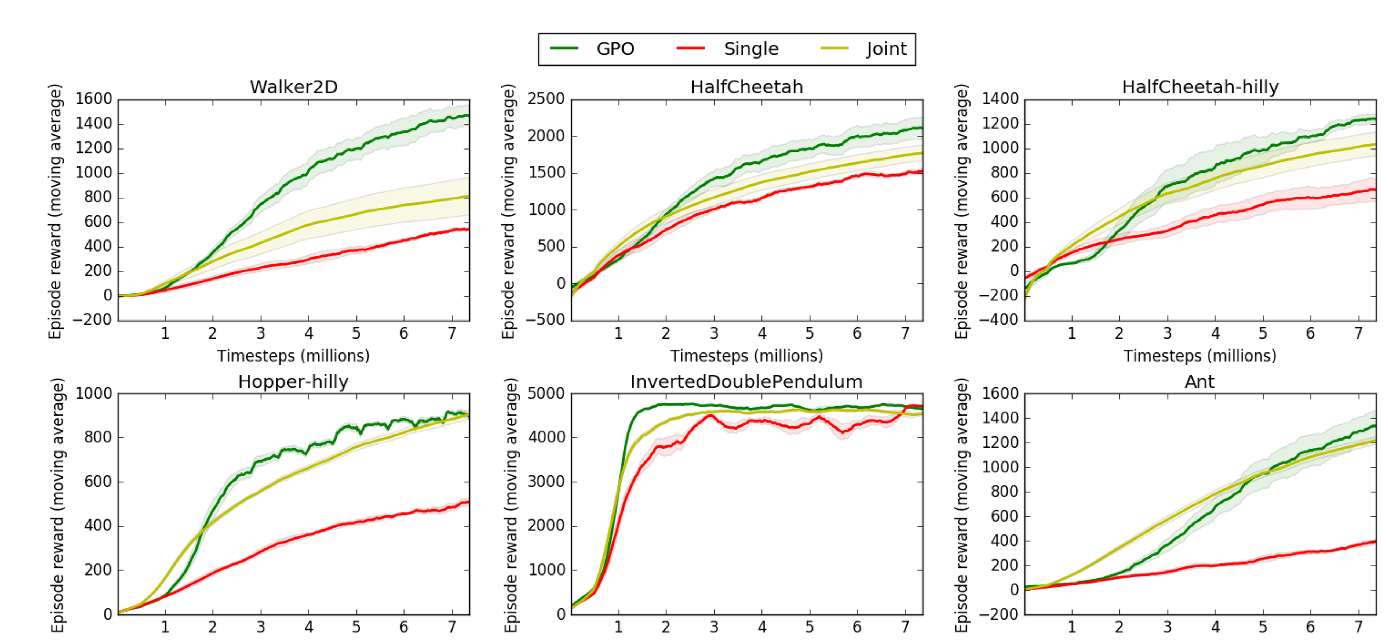


Figure: Performance of GPO and baselines on MuJoCo environments using PPO.

H. Summary

- An algorithm for population-based policy optimization for deep reinforcement learning, inspired by ideas from neuroevolution.
- Policy crossover in state-space using imitation learning to distill information from parent policies into an offspring policy.
- Exploit noisy policy-gradients estimates for mutation; and fitness-based selection operator for quality control.
- Also in the paper:
  - Ablation studies
  - Results with more environments; and A2C algorithm
  - Scalability and wall-clock time analysis