# Generating reward structures on a parameterized distribution of dynamics tasks

Abe Leite[1] and Eduardo J. Izquierdo[1,2]

[1]Cognitive Science Program, Indiana University Bloomington
[2]Luddy School of Informatics, Computing, and Engineering, Indiana University Bloomington
Corresponding email: abrahamjleite@gmail.com

**Abstract**

In order to make lifelike, versatile learning adaptive in the artificial domain, one needs a very diverse set of behaviors to learn. We propose a parameterized distribution of classic control-style tasks with minimal information shared between tasks. We discuss what makes a task trivial and offer a basic metric, time in convergence, that measures triviality. We then investigate analytic and empirical approaches to generating reward structures for tasks based on their dynamics in order to minimize triviality. Contrary to our expectations, populations evolved on reward structures that incentivized the most stable locations in state space spend the *least* time in convergence as we have defined it, because of the outsized importance our metric assigns to behavior fine-tuning in these contexts. This work paves the way towards an understanding of which task distributions enable the development of learning.

## Introduction

Natural learning is extraordinary for its versatility. Living organisms can learn to perform a seemingly limitless variety of behaviors; for example, a human can learn to swim or ride a bicycle and a three-legged dog can learn to walk again. In the field of artificial intelligence, there has long been interest in modeling the evolution and development of versatile learning behaviors and this is now seen as a pinnacle of the field of AI (Clune, 2019). Yet this sort of versatility is only adaptive when learning is faced with a similarly limitless repertoire of behaviors to confront, meaning that choosing a good distribution of tasks is an essential step in modeling the evolution of learning.

### Related work

Computational models of "versatile behavior" include fixed policies that are adaptive for multiple tasks, learning strategies tuned for finite sets or narrow infinite distributions of tasks, and general learning strategies that give rise to adaptive behavior on wholly unrelated tasks. We consider this last type of versatility particularly interesting and pay special attention in our review to the types of task distributions that have given rise to these general learning strategies.

An ample body of work has approached the problem of behavioral versatility by designing several different tasks and training neural circuits to solve all of them. Initial attempts to train neural networks to solve multiple tasks involved training subnetworks on each of the tasks and then combining the circuit (Kodjabachian and Meyer, 1998). Izquierdo and Buhrmann (2008) demonstrated the use of evolutionary algorithms to train the same circuit to perform two entirely different control tasks without the need to modularize the circuit or train it sequentially. More recently, a similar approach has been used to train recurrent neural networks (RNNs) to solve up to 20 different but related cognitively-inspired input-output tasks (Yang et al., 2019). In these cases, a fixed network gives rise to behavioral versatility through its dynamics coupled with the environment.

Combining development of learning with a discrete selection of tasks, Kirsch et al. (2019) have trained their MetaGenRL algorithm on three complex robotics tasks, and Miller (2020) has evolved his developmental neural networks on a selection of well-understood benchmark tasks in supervised and reinforcement learning. One challenge with using a finite set of tasks like this is that it takes a lot of work to design each new task by hand.

In one of the earliest attempts at generating infinite task distributions as a way to help improve the generality of solutions to a problem, Hillis (1990) used parasites as the inspiration to co-evolve the tasks as well as the solutions to a sorting optimization problem. More recently, similar approaches have taken on the label of adversarial learning. For example, Wang et al. (2019) used a relatively similar co-evolutionary strategy to vary a bipedal walking task alongside the training of the circuit. Although both co-evolutionary and adversarial strategies generate tasks to help improve the training of the solution, the way that they have been used so far involves a distribution of tasks that is restricted to a single, relatively narrow, domain.

An ideal approach would be to develop a task generator that is sufficiently general to contain all instances of a problem of interest. A seminal example is the NK tunably rugged fitness landscape generator developed by Kauffman and Weinberger (1989). One important step in this direction was taken recently by Oh et al. (2020), who used a broad dis-

tribution of discrete-action grid world and Markov decision process (MDP) tasks to train their Learned Policy Gradient (LPG) meta-learning algorithm, which achieves exceptionally strong generalization (from these toy problems to Atari games).

We hypothesize that the lean nature of these toy problems, where performing well is far more about learning capabilities than about environment information, is one of the greatest reasons behind this performance. An additional powerful benefit of LPG's approach is that grid world and MDP tasks are relatively well-understood, like NK landscapes but unlike the high-dimensional benchmark tasks (including Atari games themselves) often used in reinforcement learning.

## Our approach

We have devised a parameterized distribution of tasks in the continuous domain inspired by classic control tasks such as the inverted pendulum task. Our aim is that one could sample a number of tasks from this distribution in order to assess an agent's single- and multi-task learning capabilities. In doing so, one can use the distribution as a platform for modeling the evolution or development of learning. Due to its simplicity, this distribution may enable greater versatility within the continuous domain, like simple and tractable grid-world tasks have in the discrete-action domain.

However, we have found that for many of the tasks that result when one samples from this distribution indiscriminately, random performance and optimal performance are nearly indistinguishable: we consider such tasks trivial and not useful for understanding or developing an agent's learning capabilities. Moreover, given the wide variability of generated tasks, we could find no obvious correlation between non-triviality and the dynamical features of sampled tasks.

In order to address this issue, we take two further steps: first, we begin to describe non-trivial learning as a measurable property of a set of optimization curves, using a metric of triviality we call time in convergence (TIC). Second, we explore ways of manipulating the generated tasks to minimize the time in convergence, fixing a set of generated dynamics and exploring different strategies for deterministically generating reward structures based on these dynamics.

To summarize our contributions: (1) We offer a parameterized distribution of classic control-style tasks in the continuous domain. It can be sampled from or even optimized on, given an appropriate objective function for tasks. (2) We propose an empirical metric for how non-trivial it is to solve a task and discuss the benefits and drawbacks of this metric. (3) We provide an early data-driven analysis of which methods of imposing a reward structure on a set of generated task dynamics make the task as non-trivial as possible.

# Reinforcement learning framework

In models of reinforcement learning, one simulates an *agent* and its *environment*. The agent must manipulate its environment in some desired way to receive a *reward*, while provided with a stream of observations of the environment.

This means that an agent can be modeled by an equation mapping its current internal state $S$ and any reward $R$ and observation $O$ from the environment to its future internal state and the action $A$ it provides to its environment: $S_{agent}, O, R \rightarrow S'_{agent}, A$. This mapping encompasses both learning and behavior. The behavioral portion of the map, $O \rightarrow A$, is often called the agent's *policy*.

Similarly, the environment can be modeled by an equation mapping its own current state and the action from the agent to its future state, a new observation for the agent, and any reward given to the agent: $S_{env}, A \rightarrow S'_{env}, O, R$.

Although these tasks are traditionally called reinforcement learning tasks, an agent's policy can equally be developed via population search. A good policy for a reinforcement learning task is one that results in high accumulated reward over a number of discrete *episodes*, each of which starts from its own initial conditions.

## Example

Figure 1 illustrates these concepts on the classic inverted pendulum task, in which an agent is tasked with exerting a torque on the fulcrum of a rod pendulum in order to balance it upright. This task has one cyclic variable, which is the angle $\theta$ of the pendulum with respect to the horizontal axis, and one bounded variable, which is the angular velocity $\omega$ of the pendulum. (These types of variables will be defined in the following section.) The angular velocity is clipped to not exceed 8 radians/s, and normalized to the range $[-1, 1]$.

Figure 1A illustrates the way that present states map to future states (without input from the agent) in the pendulum task. Because this is an intuitive system, we will use it to illustrate some important types of dynamics that will be used in the following section. Although the system has two equilibrium points, at which its rate of change is zero, its dynamics around these locations are quite different. In a neighborhood of the top equilibrium point, increasing or decreasing both $\theta$ and $\omega$ results in the state evolving away from the point, but perturbing them in perfectly opposite directions will result in the state evolving back towards the point. The fact that this class of equilibrium point is stable along one axis and unstable along another is why it is called a "saddle point". On the other hand, if one perturbs the state away from the bottom equilibrium point, it neither returns nor goes further away, rather orbiting the bottom in a stable limit cycle. Because this point has an axis along which it is not strictly stable or unstable according to its local properties, it is called a "non-hyperbolic equilibrium point".

Figure 1B shows the task's reward structure and an evolved policy's trajectory on the task. In this version of the task, a positive reward is given when the pendulum is stationary and pointing upright, and a negative reward is given when the pendulum is stationary and pointing down. The

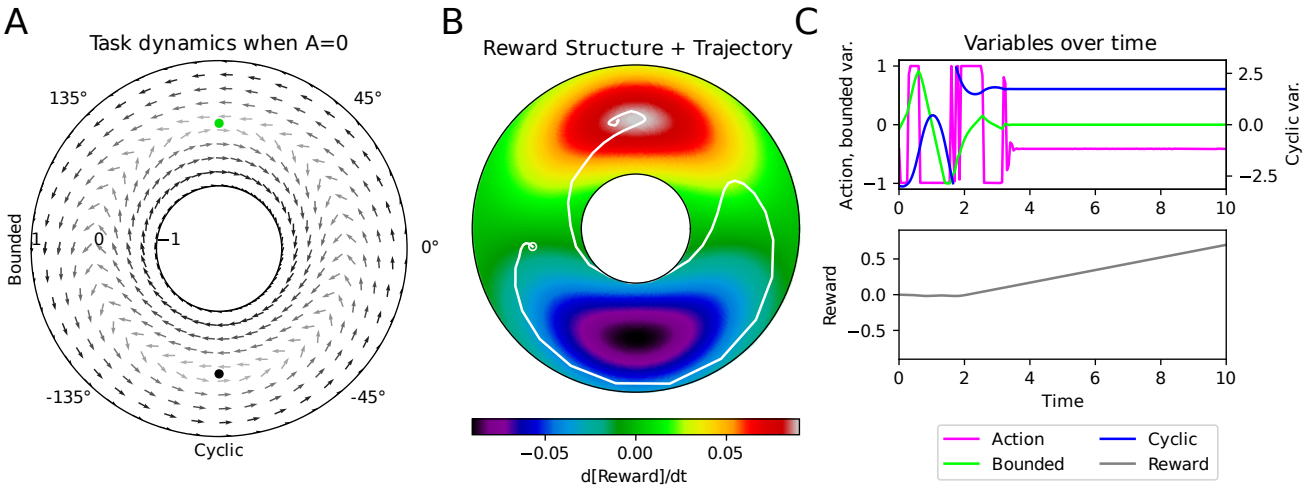


Figure 1: (A) The inverted pendulum task's dynamics with no torque exerted. The cyclic variable $\theta$ is shown by angle with $\theta = 0$ at the right-hand side, and the bounded variable $\omega$ is shown by radius with $\omega = -1$ on the inside of the ring and $\omega = 1$ on the outside. The direction in which the system evolves is shown by the arrows; a greater rate of change is denoted by a darker arrow. The saddle equilibrium point at $(\frac{\pi}{2}, 0)$ is shown in green and the non-hyperbolic equilibrium point at $(-\frac{\pi}{2}, 0)$ is shown in black. (B) The reward structure of the task is shown in color with the greatest reward at the red and white areas and the lowest reward at the purple and black areas. An evolved agent's state trajectory is shown in white, beginning at the white circle. (C) The state variables and the evolved agent's action over the 10 second episode are shown in the top panel and the reward over the episode is shown in the bottom panel.

agent (Figure 1C) briefly exerts a positive torque, increasing the pendulum's counterclockwise angular velocity, and then exerts a strong negative torque to build enough clockwise speed to lift the pendulum to its peak. It exerts another positive torque to slow down the pendulum when it reaches the 9 o'clock position, then finally balances it upright with a moderate clockwise torque. The reward increases continually once the pendulum is in the incentivized area.

## Proposed task distribution

Given a set of state and action variables, we generate a distribution of dynamics tasks where each state variable's rate of change is a polynomial function of the state and action variables. This is easy to compute, and polynomials up to a fixed degree of complexity are also a relatively easy parameter space to explore. We do not impose any particular reward structure on each task, although one can generate a polynomial reward function with the same structure as the rate-of-change functions. This parameter space contains the inverted pendulum task, and we use that task to illustrate the relation between a task's parameters and its dynamics.

### State variables

We base our environmental state variables on the types of variables encountered in nature. Two very common nonlinearities in nature are (1) cyclic processes, such as turning in a circle, where one will "wrap around" to one's original location if one moves consistently in a single direction, and (2) bounded processes, such as running into a wall, where one will actually stop moving if one moves consistently in a single direction (even if one keeps walking into the wall).

We take inspiration from these natural non-linearities and define two classes of state variables: (1) cyclic variables, which internally range $[0, 2\pi)$ but interact with other state variables and with the agent through their sine and cosine, and (2) bounded variables, which we define as 1-dimensional variables clipped to the range $[-1, 1]$, even if their dynamics as written would move them outside of it.

One task may have an arbitrary number of cyclic and bounded variables, but we consider the number of variables to be a hyperparameter rather than a parameter of each particular task. Any task within a particular parameter space will have the same number of cyclic, bounded, and action variables. In this paper, we will focus on tasks with one cyclic and one bounded variable.

The cyclic variables in their aggregate are denoted $C$; the individual cyclic variables are denoted $C_1, C_2, \cdots, C_n$. The aggregated bounded variables are similarly denoted $B$, with individual variables denoted $B_1, B_2, \cdots, B_m$.

### Action variables

The agent provides real-valued actions within the range $[-1, 1]$ to the environment. The action variables are denoted $A$ as an aggregate and $A_1, A_2, \cdots, A_k$ individually.

## Dynamics model

In our model, each state variable's rate of change is defined as a polynomial function of $A$, $sin(C)$, $cos(C)$, and $B$; these are collectively called $X$. Each individual input variable $X_i$ refers to the $i$th variable when $A$, $sin(C)$, $cos(C)$, and $B$ are concatenated (in this order). Following these variables, the final element of $X$ is defined to be 1 for computational simplicity. This means that the number of elements of $X$ ($|X|$) is $|A| + 2|C| + |B| + 1$.

We will consider sparsely encoded polynomials. We define hyperparameters $t$ and $d$ to limit the complexity of the polynomials that can be represented. Each rate of change polynomial has up to $t$ terms of up to degree $d$. This means that the parameters for a single variable's rate of change are (1) the real coefficients of the polynomial $k_1, k_2, \cdots, k_t$ and (2) the integer indices of the factors of each term $a_{1,1}, \cdots, a_{1,d}, \cdots, a_{t,1}, \cdots, a_{t,d}$. The rate of change of state variable $y$ is then written

$$\frac{dy}{dt} = \sum_{i=1}^{t} k_i \prod_{j=1}^{d} X_{a_{i,j}} \tag{1}$$

Terms of less than degree $d$ include indices pointing to the final element of $X$, which is by definition 1. Polynomials with fewer than $t$ terms include terms with 0 coefficients, which will not affect the rate of change.

Each state variable $C_i$ or $B_i$ has a rate of change of this form, meaning that the entire system's dynamics have $(|C| + |B|)t$ real parameters and $(|C| + |B|)td$ integer parameters.

The equations are integrated by Euler integration. In this paper, we use an integration timestep of 0.05s.

## Reward model

This way of generating dynamics does not constrain one to any particular way of defining reward. One may give the reward variable a parameterized rate-of-change polynomial like those of the system dynamics. In this case, the step reward provided to the agent will be the product $\frac{dR}{dt}(\Delta t)$. While this has advantages should one want to co-evolve tasks and agents, we have found this approach to produce a concerning number of trivial tasks. Alternative approaches to generating reward structures based on information about the task dynamics will be discussed in the following section.

We generally expect reward to increase or decrease over time as a function of state alone rather than state and action. (Action is sometimes directly included in reward functions to represent energy costs.) This lets one interpret particular areas as low- or high-reward, and we consider it a useful constraint, though it is not required by this framework.

Many real-world and simulated tasks, including homeostasis and the cart-pole task, require agents to maintain some viability conditions at all times, with survival more important than optimality. One may use a bounded state variable initialized to 1 with a non-positive rate of change to represent the agent's viability in these tasks; only the initialization to 1 requires any modifications to our model.

## Example: inverted pendulum task dynamics

The following are the equations for the inverted pendulum task when expressed as a system of this form. There is one cyclic variable $C_1$, one bounded variable $B_1$, and one action variable $A_1$. In this case, our polynomials include up to 2 terms of up to degree 1 each.

$$\frac{dC_1}{dt} = 8B_1 \tag{2}$$

$$\frac{dB_1}{dt} = \frac{6}{8}A_1 - \frac{15}{8}cos(C_1) \tag{3}$$

These dynamics are shown with $A = 0$ in Figure 1A.

## Example: generated task dynamics set 15

After we generated 25 sets of task dynamics for the experiments described below, we selected task 15 as representative of the range of dynamics found across the 25 tasks.

The following are the equations for task 15's dynamics, which use one cyclic variable, one bounded variable, and one action variable. In this case, the polynomials include up to 4 terms of up to degree 3 each.

$$\begin{aligned}\frac{dC_1}{dt} = \; & 1.46A_1B_1cosC_1 - 0.46A_1^2cosC_1 \\ & + 0.90B_1^2 + 1.43B_1^2cosC_1\end{aligned} \tag{4}$$

$$\frac{dB_1}{dt} = 0.57A_1B_1 - 0.85A_1B_1 - 0.91sinC_1 - 0.05A_1cosC_1 \tag{5}$$

The resulting dynamics are shown in full in Figure 2A. These figures are available for other tasks on request.

# Reward structures

Life exists at the boundary of order and chaos, and many biological behaviors are aimed at maintaining an improbable state of order. A single neuron actively transporting ions against their gradients and a person cleaning their room are both making use of the fact that agency (in its many forms) can make otherwise vanishingly improbable outcomes happen. Because biological behaviors have so much to do with maintaining improbable states of the world, we propose building reward structures on tasks based on the probability of different states being encountered in the task dynamics.

One can take many approaches to understand the probability of states in an agent-environment system; we focus on two. First, we consider analytic approaches that use the local properties of the system dynamics to predict which points are attractive or repulsive. Second, we consider empirical approaches that simulate the system to generate hitmaps showing which points are most frequently encountered.

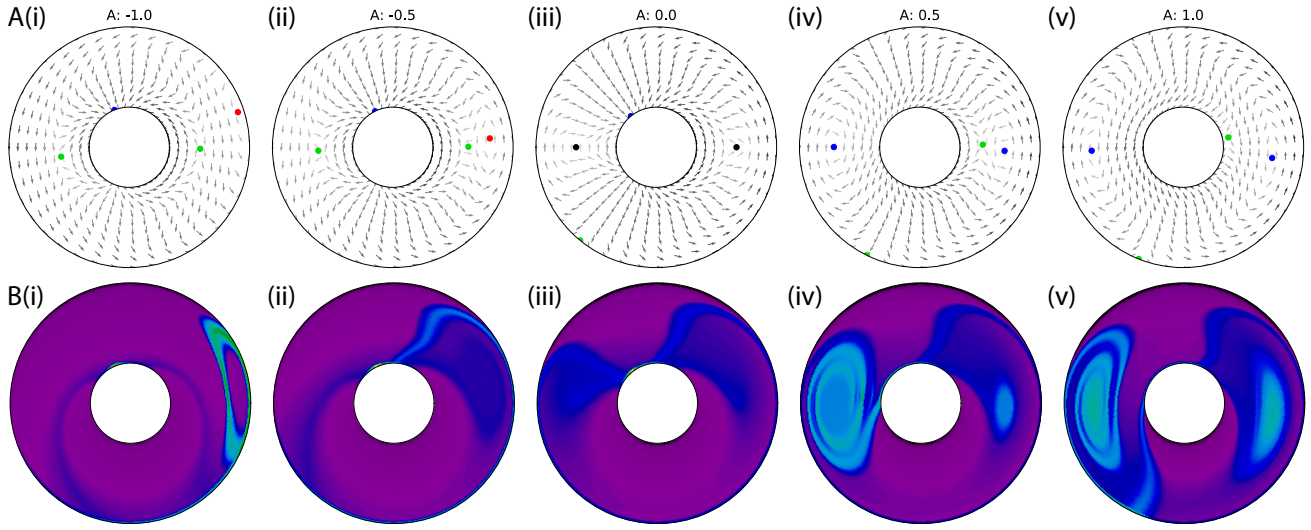


Figure 2: Task 15 dynamics (A) and hitmaps (B) across different actions (i-v). State space is shown as in Figure 1. (A) Stable, unstable, saddle, and non-hyperbolic equilibrium points are shown in blue, red, green, and black respectively. (B) The most frequently encountered points are shown in red and white and the least in purple and black. Note that the most frequently hit area, by orders of magnitude, is the hotspot near $(\frac{\pi}{2}, -1)$, but this is hard to see because the histogram is very fine.

We discuss the broad benefits and drawbacks of these approaches and propose specific treatments to generate reward structures that fall under the approaches. Examples of each treatment on task dynamics set 15 are shown in Figure 5.

## Analytic approaches

In analytic approaches, one analyzes equilibrium points in state space to find whether they are locally stable, unstable, saddle points (stable and unstable along different axes), or non-hyperbolic (neither stable nor unstable along some axis, often since other points on the axis are also in limit sets).

This requires locating equilibrium points in state space, where the rate of change for all state variables is zero, and analyzing the stability of these points by understanding how the rate of change changes in a neighborhood of the points.

One may also find "equilibrium points" on the outer lattice where all bounded variables are either -1 or +1, when the rates of change of the bounded variables are effectively zero because they are running into the clipping condition. These points are located by finding all points on the outer lattice where $\frac{dC_i}{dt}$ is zero for all $C_i$ and $\frac{dB_i}{dt}$ is effectively zero for all $B_i$. These points are considered stable along the $B$ axes and are classified as stable, saddle, or non-hyperbolic according to their local behavior on the $C$ axes.

Importantly, equilibrium points and their stability depend on action; indeed, this is part of how agents are able to manipulate the task dynamics. In order to get a sense for what points are attractive or repulsive overall, one can find equilibrium points across a range of different actions and create a combined set of equilibrium points. See Figure 2A for examples of how equilibrium points can depend on action.

One advantage of analytic approaches is that they produce compact, understandable representations of attractive and repulsive points in the state space. On the other hand, local dynamics do not always tell the full story of which areas of state space are the most attractive. For example, in Figure 2A(i), there is an unstable equilibrium point that in fact falls within a stable limit cycle visible in Figure 2B(i).

**Stable** In the analytic stable treatment, a reward structure is generated by finding the equilibrium points at each $A \in \{-1, -0.5, 0, 0.5, 1\}$, and assigning each equilibrium a score according to its class: +1 for stable equilibrium points, +0.25 for non-hyperbolic equilibrium points, -0.25 for saddle equilibrium points, and -1 for unstable equilibrium points. (Non-hyperbolic points are considered relatively stable because they often appear when a point is adjacent to another limit set. A saddle point is relatively unstable because unless one is on its stable axis, one will eventually be repelled away from it along its unstable axis.)

Every point $x$ in the space is then assigned a reward depending on each equilibrium point's score and distance from the point $x$ (where cyclic variables are treated as their embedding on the circle). In particular, if the equilibrium points are denoted $\{p_1, p_2, \ldots, p_n\}$ and their scores are denoted $\{s_1, s_2, \ldots, s_n\}$, then $\frac{dR}{dt}$ at an arbitrary point $x$ is determined by the following equation:

$$\frac{dR}{dt}(x) = \sum_{i=1}^{n} (s_i) e^{-d(x,p_i)^2} \tag{6}$$

**Unstable** In the analytic unstable treatment, a reward structure is generated in precisely the same way with the opposite scores assigned to equilibrium point classes: -1 for stable equilibrium points, -0.25 for non-hyperbolic equilibrium points, +0.25 for saddle equilibrium points, and +1 for unstable equilibrium points. Note that the unstable reward function is precisely -1 times the stable reward function.

**Saddle** In the analytic saddle treatment, a reward structure is generated in the same way as the prior two treatments except that the following scores are assigned to equilibrium point classes: +1 for saddle equilibrium points, +0.25 for unstable equilibrium points, -0.25 for non-hyperbolic equilibrium points, and -1 for stable equilibrium points.

### Empirical approaches

In empirical approaches, the global behavior of the system is observed in systematically controlled conditions in order to empirically infer which points in state space are the most attractive (probable) or repulsive (improbable). This is done by systematically sampling all combinations of initial conditions and fixed actions to a certain resolution and then simulating the system forward or backward in time. In this work, a resolution of 300, 151, and 41 values is used for each cyclic, bounded, and action variable respectively, and each simulation is run for a period of 20s, double the length of standard episodes, in order to better understand where the most ultimately stable or unstable points are located.

A histogram of states hit over time is obtained and slightly blurred to remove artifacts introduced by the sample of initial conditions. The results of this process, restricted to actions -1, -0.5, 0, 0.5, and 1, are shown in Figure 2B.

Finally, the histogram is blurred further, averaged across actions, and processed to incentivize states in the most or least frequently hit locations.

The advantages of these empirical approaches include that they provide good global coverage of the space and that they do not rely on direct mathematical knowledge of the task dynamics. Unlike the analytic approaches, they can recognize the vicinity of stable limit cycles as attractive even when they enclose unstable equilibrium points. The primary downside of these approaches is that they are wholly intractable in higher dimensions.

**Stable** In the empirical stable treatment, the simulation is run forwards in time. The most frequently hit locations receive the highest reward.

**Antistable** In the empirical antistable treatment, the simulation is run forwards in time. The most frequently hit locations receive a negative reward. This reward function is equal to -1 times the empirical stable reward function.

**Additional treatments** The following treatments were tested in our experimental setup so we are including them to avoid selective reporting. We are not focusing our analysis on them at this time:

1. In the empirical unstable treatment, the simulation is run backwards in time – a classic method to find unstable limit sets. The most-hit locations receive the highest reward.

2. The empirical antiunstable reward function is -1 times the empirical unstable function.

3. In the empirical median-stability treatment, the simulation is run forward in time and locations in the hitmap are ranked by frequency. The most and least frequently hit points are both disincentivized; rather, the points with median frequency receive the greatest reward.

### Uniform (null) approach

As well as these dynamics-based reward structures, we also evaluate an uninformed reward structure that rewards negative values for $B_1$ and positive values for $sinC_1$.

## Defining non-triviality

We propose the following working definition of non-trivial learnability given a task/policy search pair: a task is non-trivially learnable if: (1) there exists a solution that is qualitatively different from and significantly better than random behavior, (2) this solution can be found by search, and (3) this solution requires significant search to find.

It is challenging to operationalize these conditions, particularly given that it is not easy to determine the reward magnitude for the worst and best possible policies on a task or whether these policies are qualitatively different from a random policy. (We hope to explore how these issues play out given different task and policy spaces in a future work.)

For the purposes of this paper, we propose a simple metric called time in convergence (TIC) that aims at capturing conditions (1), (2), and (3) above. It is a property of a set of optimization curves that measures how trivial it is for the optimization strategy to find a solution for the task under consideration, according to what proportion of the time each curve spends above a convergence threshold.

In order to calculate time in convergence, consider $n$ optimization curves. First, define "optimal performance" to be the best performance of a streak of 5 consecutive assessments seen across all $n$ runs, and define "random performance" to be the mean performance (over runs) on the first assessment. Note that in both evolution and reinforcement learning contexts, this first assessment generally reflects some early search rather than being fully "random".

Normalize performance scores so that "optimal performance" has a score of 1 and "random performance" has a score of 0, and define a run to have "converged" when it first surpasses a normalized score of 0.95. We can then calculate the time in convergence metric by asking what proportion of

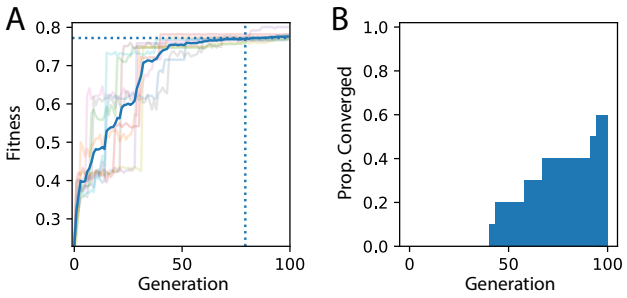


Figure 3: (A) 10 runs of an evolutionary algorithm on this treatment yield a time in convergence of 20.7% – on average, a run spends 20.7 generations above the 95% convergence threshold. (B) The proportion of runs that have converged over time is shown. The area under this cumulative histogram is equal to the time in convergence (TIC).

the full search time the average run spends in convergence. An illustration of this calculation is shown in Figure 3.

Two situations that will result in a very low time in convergence are when a single run far outperforms all other runs and when the best-performing runs improve at a fixed linear rate over time. These situations indeed appear highly non-trivial. On the other hand, situations that would result in a high reported time in convergence include cases where all runs rapidly climb to a peak and cases where performance is very noisy from the start, without any clear trends emerging.

## Experiments

We evaluated the proposed treatments for generating reward structures on 25 sampled sets of task dynamics over 1 cyclic, 1 bounded, and 1 action variable. The dynamics polynomials had up to 4 terms of up to degree 3. Our analytic tools did not find the equilibrium points for 6 of the 25 tasks within an hour; these tasks were excluded from our analysis.

After the tasks and reward structures were generated, 10 evolutionary runs were performed on each condition and the time in convergence metric was calculated on the resulting max fitness curves. The agents were feed-forward neural networks with layers of 3 input, 50 relu, 20 relu, and 1 tanh units. Each of the 100 generations included 25 10-second episodes. Our search used a population size of 200 with an elitist fraction of 20.

In order to understand the resulting policies in the context of the full policy space, we also simulated 25 random policies from the same initial conditions we used for our experiments and recorded the resulting state trajectories.

Based on our intuition that it is more difficult to maintain improbable states, we hypothesized that time in convergence would be lower for the analytic and empirical unstable and antistable treatments than for the other treatments.

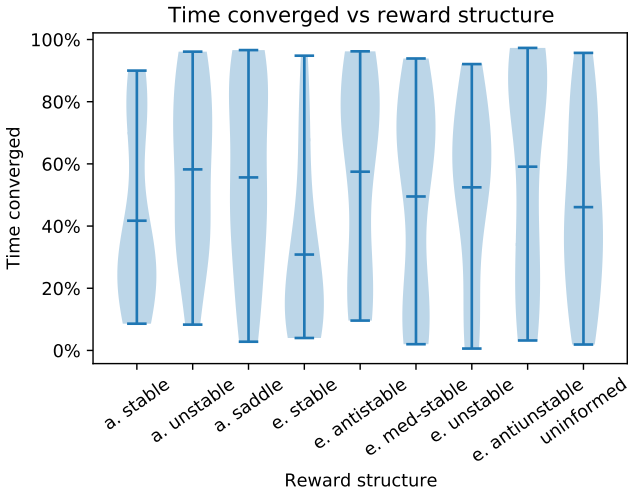


Figure 4: On 19 sampled sets of dynamics, the empirical (31%) and analytic (42%) stable treatments result in the lowest mean time in convergence. Of the discussed treatments, the analytic unstable (58%) treatment results in the greatest time in convergence.

## Results

Contrary to our expectations, the empirical stable treatment produced a lower time in convergence than any other treatment, with a mean time in convergence of 31%. A two-factor ANOVA revealed main effects of both task ($p < 10^{-8}$) and reward structure treatment ($p < 0.05$). Figure 4 illustrates the distribution of time in convergence for each reward structure treatment.

We found that each task dynamics/reward structure pair resulted in different behaviors (see Figure 5 for an example with task dynamics set 15), with the analytic unstable and saddle treatments as well as the empirical antistable yielding behaviors that followed significantly different trajectories than any of the random policies (shown in row A). However, the analytic and empirical *stable* treatments actually had the least time in convergence out of all of the treatments, in spite of following relatively in-distribution trajectories as seen in Figure 5. For task 15, the analytic stable case had a low TIC due to a single particularly high-performance run that outperformed all of the others, meaning that they never converged. The empirical stable case showed continual pronounced improvement over time.

## Discussion

### Innovation and fine-tuning

Why do the stable reward structure treatments result in the lowest time in convergence?

As shown in the evolution curves in Figure 5, evolving agents to solve a reward structure for a given task results in first a sudden leap in performance ("innovation") and then a

"fine-tuning" phase during which the remainder of the optimal fitness is gained. During the "innovation" phase, qualitatively different state trajectories emerge; these state trajectories are then fine-tuned to travel to their destinations along the paths of highest reward and to reach them as quickly as possible. In the analytic unstable and empirical antistable treatment conditions, fitness gains from innovation are large relative to fitness gains from fine-tuning. However, when agents evolve to solve the stable reward structures, they follow more qualitatively similar trajectories to the random trajectories (as seen in Figure 5), and their fitness gains from innovation are smaller relative to their gains from fine-tuning.

Because a greater proportion of the fitness gap between random and optimal performance is occupied by fine-tuning for the stable treatments, the 95% convergence threshold is crossed only after a great deal of fine-tuning has occurred, potentially at a very late generation. Conversely, when a large proportion of the distance between random and optimal performance is occupied by innovation, then once the innovation has occurred, the run is considered to have converged, potentially at a very early generation.

Fine-tuning is of course adaptive, and it is well worth developing learning agents that are capable of fine-tuning. However, innovation is the more impressive capacity, and it is desirable to have a metric that focuses primarily on measuring the development of qualitatively different behaviors rather than fine-tuning, particularly addressing the questions of: (1) how much innovation has taken place between random and optimal performance and (2) how challenging was this innovation from a search perspective?

Our primary growing edge for this work is in exploring metrics that address these questions.

## Future directions

In addition to seeking more targeted metrics of non-triviality, future directions for this work include understanding whether this distribution of low-dimensional dynamics tasks facilitates better generalizability for continuous-action meta-learners such as MetaGenRL, just as LPG's lean distribution of grid-world and MDP tasks facilitated dramatically improved generalizability in the discrete-action domain.

Finally, along the same lines, this work enables co-evolution of tasks and learners. We hope that by either usual meta-learning or such open-ended approaches, this parameterized distribution of simple tasks can enable the evolution of lifelike, versatile learning behaviors in artificial agents.

## Source code

Our source code is available at: github.com/ajleite/dynatad.


## Acknowledgements

We thank the reviewers for their insightful comments. This material is based upon work supported by the National Science Foundation under Grant No. 1845322.


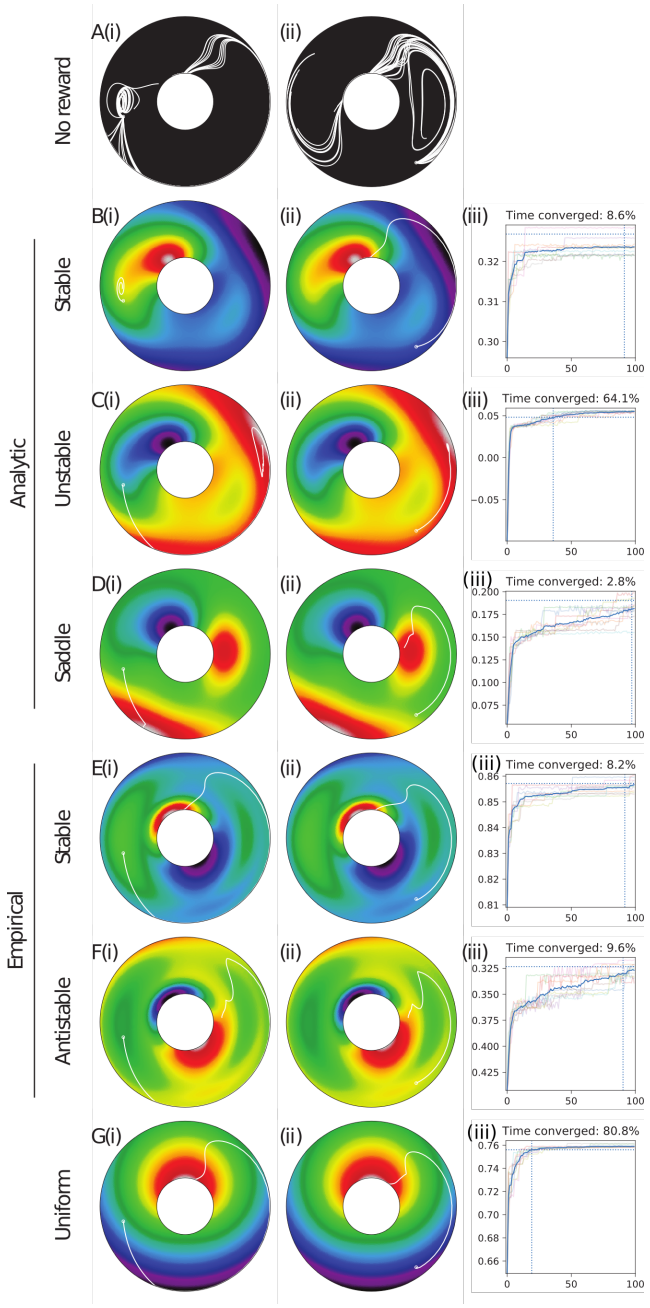


Figure 5: Different reward structures generated with the same task dynamics result in different evolved behaviors and different evolution dynamics. The reward structures are color-coded using the same mapping as Figure 1. Each row represents agents evolved on a different reward treatment over task dynamics set 15. Row (A) shows 25 randomly sampled agent trajectories without any reward structure. The first two columns (i-ii) depict state trajectories (white line) beginning with two different starting conditions (white circle). The last column (iii) depicts the fitness over time of 10 evolutionary runs (light colors). A thick blue trajectory represents the average across the runs. The horizontal dashed line represents the 95% convergence threshold. The vertical dashed line represents the average time of convergence.

## References


Clune, J. (2019). Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*.

Hillis, W. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1):228–234.

Izquierdo, E. and Buhrmann, T. (2008). Analysis of a dynamical recurrent neural network evolved for two qualitatively different tasks: Walking and chemotaxis. In Bullock, S., Noble, J., Watson, R., and Bedau, M., editors, *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 257–264, Cambridge, MA. MIT PRESS.

Kauffman, S. A. and Weinberger, E. D. (1989). The nk model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of theoretical biology*, 141(2):211–245.

Kirsch, L., van Steenkiste, S., and Schmidhuber, J. (2019). Improving generalization in meta reinforcement learning using learned objectives. *arXiv preprint arXiv:1910.04098*.

Kodjabachian, J. and Meyer, J.-A. (1998). Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects. *IEEE Transactions on Neural Networks*, 9(5):796–812.

Miller, J. F. (2020). Evolving developmental neural networks to solve multiple problems. In *Artificial Life Conference Proceedings*, pages 473–482. MIT Press.

Oh, J., Hessel, M., Czarnecki, W. M., Xu, Z., van Hasselt, H. P., Singh, S., and Silver, D. (2020). Discovering reinforcement learning algorithms. *Advances in Neural Information Processing Systems*, 33.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019). Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *CoRR*, abs/1901.01753.

Yang, G., Joglekar, M., Song, H., Newsome, W., and Wang, X. (2019). Task representations in neural networks trained to perform many cognitive tasks. *Nature Neuroscience*, page 297–306.