POET: Open-Ended Coevolution of Environments and their Optimized Solutions

Rui Wang, Joel Lehman, Jeff Clune*, and Kenneth O. Stanley*
Uber AI
San Francisco, CA 94103
*co-senior authors
{ruiwang,joel.lehman,jeffclune,kstanley}@uber.com

ABSTRACT

How can progress in machine learning and reinforcement learning be automated to generate its own never-ending curriculum of challenges without human intervention? The recent emergence of quality diversity (QD) algorithms offers a glimpse of the potential for such continual open-ended invention. For example, novelty search showcases the benefits of explicit novelty pressure, MAP-Elites and Innovation Engines highlight the advantage of explicit elitism within niches in an otherwise divergent process, and minimal criterion coevolution (MCC) reveals that problems and solutions can coevolve divergently. The Paired Open-Ended Trailblazer (POET) algorithm introduced in this paper combines these principles to produce a practical approach to generating an endless progression of diverse and increasingly challenging environments while at the same time explicitly optimizing their solutions. An intriguing implication is the opportunity to *transfer* solutions among environments, reflecting the view that innovation is a circuitous and unpredictable process. POET is tested in a 2-D obstacles course domain, where it generates diverse and sophisticated behaviors that create and solve a wide range of environmental challenges, many of which cannot be solved by direct optimization, or by a direct-path curriculum-building control algorithm. We hope that POET will inspire a new push towards open-ended discovery across many domains.

CCS CONCEPTS

• **Computing methodologies → Artificial life**; *Evolutionary robotics*; *Reinforcement learning*; *Neural networks*;

KEYWORDS

Open-ended evolution, coevolution, evolution strategies, novelty search, artificial life

ACM Reference Format:
Rui Wang, Joel Lehman, Jeff Clune*, and Kenneth O. Stanley*. 2019. POET: Open-Ended Coevolution of Environments and their Optimized Solutions. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3321707.3321799

1 INTRODUCTION

While conventional machine learning (ML) and artificial life (alife) have persisted historically as relatively disconnected pursuits with differing ambitions, their separation is recently narrowing as some of the themes emerging from alife-inspired research begin to inform our understanding of long-term progress in machine learning as a field. ML tends to mark progress through specific benchmarks or challenge problems introduced by the community that are gradually conquered and replaced by new benchmarks. For example, image classification has progressed from the simple MNIST benchmark [33] to breakthrough achievements like human-level performance in ImageNet [11]. In reinforcement learning (RL) [59], modest beginnings in pole balancing [1] have given way to learning to play Atari from pixels [3]. In effect we see in hindsight trails of stepping stones laid by the community, each in turn solved as a singular objective.

What alife offers in this never-ending gauntlet is a lens on the question of the scalability of such a process over vast spans of progress. Beginning with the introduction of novelty search [38], we have seen in the field a steady loss of confidence in the idea that we can reliably set objectives, or even know what the right objectives are, and pursue them to ambitious ends [65]. This growing realization points back to the broader strategic approach to the field of machine learning as whole–we may need at some point to liberate our algorithms from static objectives, and even from externally-imposed benchmarks entirely, so that they may autonomously generate their own stepping stones to the farthest reaches of possibility.

At a practical level, in the years since the introduction of novelty search [38], a new field has gradually emerged around the idea of *quality diversity* (QD). The core idea in this field is to develop algorithms that return not only the best or a set of best performers on some problem, but instead a broad diversity of behaviors that are still as high in quality as possible given such diversity [9, 37, 42, 45]. For example, Cully et al. [9] evolve a wide diversity of gaits for a hexapod robot, where each gait is the best possible for a different way of walking, such as not using one leg or a pair of legs. This kind of *repertoire* is useful because it can then be used to select the best controller as circumstances change in the environment or if the robot becomes damaged.

While the practical side of QD is often emphasized in such results, there is also a more fundamental and ambitious potential lurking behind such algorithms, which is the possibility that they might become *open-ended*. Once we achieve algorithms that can continually gather an expanding set of functional and diverse behaviors, if such a process could continue indefinitely and with increasing complexity, QD begins to look like a conduit to open-ended evolution [2, 32, 55, 58, 60]. We might want such an open-ended process because the chain that leads from the capabilities of computers today to the most ambitious imaginable possibilities (e.g. general human-level intelligence) could stretch across vast and inconceivable paths of stepping stones. There are so many directions we could go, and so many problems we could tackle, that the curriculum that leads from here to the farthest reaches of computation is beyond the scope of our present imagination. QD is inspiring in part because it offers a potential path to algorithms that traverse those vast reaches. In the shorter term, the implication of such algorithms is that, because the stepping stones to ambitious solutions are unknown a priori, they may reach solutions that could not be reached by any other means–not even by a curriculum aimed at the same target.

Interestingly, the progress of recent years in QD has produced several fundamental insights that could advance the push towards open-endedness, but at the same time, there are caveats to each advance that ultimately limit its path to open-ended innovation. For example, novelty search [38], which is a key predecessor to QD, pushes indefinitely towards novelty, but provides no assurance of the quality of anything that is discovered. Novelty search with local competition (NSLC) [37] and MAP-Elites [9, 42] rectify this limitation by adding a quality pressure (hence ushering in QD in earnest), but they still inevitably hit the limit of what is possible to discover in the particular environment where they are run–there are only so many novel ways a hexapod can walk. Interestingly, the coevolution [10, 14, 26, 44, 62] of environments with their solutions could provide a natural remedy to the limitation of static environments. In particular, minimal criterion coevolution [5] offers a counterpoint to this limitation by allowing the environment itself to change, but it does so while dropping any explicit push towards novelty or quality, relying instead entirely upon genetic drift. Innovation Engines [43] highlight the opportunity to transfer high-quality solutions from one objective among many to another, and the combinatorial multi-objective evolutionary algorithm (CMOEA) extends the Innovation Engine to combinatorial tasks [24, 25]. Both show the circuitous nature of serendipitous stepping stones, yet both are ultimately limited by the number of objectives in the problem space in which they are run.

This cohort of algorithms offers a number of powerful ideas–pressure towards novelty, combining novelty and quality pressure, coevolution of environments and solutions, transfer of solutions among different objectives being optimized simultaneously–yet each seems to lack in isolation something essential to genuine open-endedness. In response, the Paired Open-Ended Trailblazer (POET) algorithm introduced in this paper harnesses the insight that the shortcomings of each of these ideas are naturally remedied by the strengths of the others. That is, if the environment might run out opportunities for new solutions, then we can coevolve new environments with the solutions. If the coevolution of solutions and environments is too reliant on drift, then we can add a push towards novelty in environments and quality in the solutions to them. If there are many environments with completely disjoint solutions, then we can try transferring those solutions among them.

POET is the first algorithm to seize the opportunity to pull all these pieces together, with the result that increasingly diverse and complex environments can be generated at the same time as their solutions continually optimize to master them, leading to behaviors that would be difficult to discover in any other way.

In this introduction of POET, it is evaluated in a simple 2-D bipedal-walking obstacle-course domain in which the form of the terrain is evolvable, from a simple flat surface to heterogeneous environments of gaps, stumps and rough terrain. The results establish that (1) solutions found by POET for challenging environments cannot be found directly on those same environmental challenges by optimizing on them only from scratch; (2) neither can they be found through a curriculum-based process aimed at gradually building up to the same challenges POET invented and solved; (3) periodic transfer attempts of solutions from some environments to others–also known as "goal switching" [43]–is important for POET's success; (4) a diversity of challenging environments are both invented and solved in the same single run. POET in effect reveals a rich landscape of new opportunities to investigate algorithms that invent their own circuitous paths of problems and solutions to otherwise inaccessible levels of achievement, and even open-endedness.

2 BACKGROUND

This section begins with foundational ideas in QD and then reviews evolution strategies (ES) [47], which serves as the optimization engine behind POET in this paper (though other reinforcement learning algorithms could be substituted in the future).

2.1 Foundational Ideas

Population-based algorithms going back to novelty search (NS) [38] that encourage *behavioral* (as opposed to genetic) diversity [9, 24, 25, 42, 43, 45] have proven less susceptible to local optima, and thus naturally align more closely with the idea of open-endedness as they focus on *divergence* instead of *convergence*. These algorithms are based on the observation that the path to a more desirable or innovative solution often involves a series of waypoints, or *stepping stones*, that may *not* increasingly resemble the final solution, and are not known ahead of time.

Quality diversity (QD) algorithms [9, 37, 42, 45] elaborate on NS by keeping track of many different niches of solutions that are (unlike pure NS) being optimized simultaneously and in effect try to discover stepping stones by periodically testing the performance of offspring from one niche in other niches, a process referred to as *goal switching* [43]. An approach called the Innovation Engine [43] helps to cement the power of goal switching: It can evolve a wide range of images in a single run that are recognized with high-confidence as different image classes by a high-performing deep neural network trained on ImageNet [31]. In that domain, the Innovation Engine maintains separate niches for images of each class, and periodically checks whether the best performer in one class might be able to unseat the best performer in another. Interestingly, the evolutionary path to performing well in a particular class often must pass through other (oftentimes seemingly unrelated) classes that ultimately serve as stepping stones to recognizable objects [43]. POET will similarly harness goal-switching within divergent search.

It is also important to note, because POET will likely combine with diverse RL algorithms in the future, that the idea of promoting diversity and preserving stepping stones is also gaining prominence in the RL literature. Examples include acquiring complex skills in RL through diversity preservation by Eysenbach et al. [13], implementing an NS-archive-like mechanism to overcome reward sparsity by Savinov et al. [48], adding NS to ES [8], and the recent success of storing diverse state-space discoveries in a QD-like manner in Go-Explore [12] so that the search can return to them later.

One challenge remaining if conventional QD is to achieve open-endedness is that although it applies pressure for ongoing divergence in the solution space, the *environment* itself remains static, limiting the scope of what can be found in the long run. Eventually, for progress in ML to be truly automated, algorithms will need to generate their own *problems* as well as solutions. The eventual importance of generating problems recently has gained recognition across a variety of related fields, such as goal generation [15] and reverse curriculum generation in RL [16], intrinsically motivated goal exploration processes (IMGEPs) [17], the POWERPLAY search for an increasingly-general problem solver through new tasks [49], and Teacher-Student Curriculum Learning [39]. The field of *procedural content generation* (PCG) [52, 61] also offers inspiration for generating new challenges (usually focused on gaming).

From an evolutionary perspective, one way to think about learning environments evolving along with their solutions is through the *coevolution* [10, 14, 26, 44, 62] of the two. Following this principle, an important predecessor to the POET algorithm in this paper is the recent *minimal criterion coevolution* (MCC) algorithm [5], which explores an alternative paradigm for open-endedness through coevolution. In particular, it implements a novel coevolutionary framework that pairs a population of evolving *problems* (i.e. environmental challenges) with a co-evolving population of *solutions*. Unlike conventional coevolutionary algorithms that are usually divided between competitive coevolution [14] and cooperative coevolution [62], MCC introduces a new kind of coevolution that evolves two interlocking populations whose members earn the right to reproduce by satisfying a *minimal criterion* [35, 53, 54] with respect to the other population, as both populations are gradually shifting simultaneously. The result in a demonstration coevolving mazes and maze solvers is that the mazes increase in complexity and the neural networks continually evolve to solve them [5]. However, in MCC there is no force for *optimization* within each environment (or maze in the example experiment). That is, once a maze is solved there is no pressure to *improve* its solution; instead, it simply becomes a potential stepping stone to a solution to another maze. There is also no opportunity for improvements in one environment to transfer to another. In effect, MCC relies entirely on genetic drift, whereas POET explicitly adds pressure for higher quality, as well as the opportunity to transfer solutions through goal switching [43].

2.2 Evolution Strategies (ES)

In the POET implementation in this paper, ES plays the role of the optimizer (although other optimization algorithms should also work). Inspired by natural evolution, ES [46] represents a broad class of population-based optimization algorithms. The method referred to here and subsequently as "ES" is a version of ES popularized by Salimans et al. [47] that was recently applied with large-scale deep learning architectures to modern RL benchmark problems. This version of ES draws inspiration from Section 6 of Williams [64] (i.e. REINFORCE with multiparameter distributions), as well as from subsequent population-based optimization methods including Natural Evolution Strategies (NES) [63] and Parameter-Exploring Policy Gradients (PEPG) [51]. More recent investigations have revealed the relationship of ES to finite difference gradient approximation [34] and stochastic gradient descent [66].

In the typical context of RL, we have an environment, denoted as $E(\cdot)$, and an agent under a parameterized policy whose parameter vector is denoted as w. The agent maximizes its reward, denoted as $E(w)$, as it interacts with the environment. In ES, $E(w)$ represents the stochastic reward experienced over a full episode of an agent interacting with the environment. Instead of directly optimizing w to maximize $E(w)$, ES seeks to maximize the expected fitness over a population of w, $J(\theta) = \mathbb{E}_{w \sim p_\theta(w)}[E(w)]$, where w is sampled from a probability distribution $p_\theta(w)$ parameterized by θ. The complete derivation of the ES of Salimans et al. [47], and used by POET is given in Section A.1 of Supplemental Information (SI), which also gives the ES step pseudocode, shown in Algorithm 1 of SI.

ES has exhibited performance on par with some of the traditional, simple gradient-based RL algorithms on difficult RL domains (e.g. DQN [41] and A3C [40]), including Atari environments and simulated robot locomotion [8, 47]. More recently, NS and QD algorithms have been shown possible to hybridize with ES to further improve its performance on sparse or deceptive deep RL tasks, while retaining scalability [8], providing inspiration for its hybridization within an CMOEA- and MCC-like algorithm in this paper.

3 THE POET ALGORITHM

POET is designed to facilitate an open-ended process of discovery within a single run. It maintains a population of environments (for example, various obstacle courses) and a population of agents (for example, neural networks that control a robot to solve those courses), and each environment is paired with an agent to form an *environment-agent pair*. POET in effect implements an ongoing divergent coevolutionary interaction among all its agents and environments in the spirit of MCC [5], but with the added goal of explicitly optimizing the behavior of each agent within its paired environment in the spirit of Innovation Engines [43] and CMOEA [24, 25]. It also elaborates on the minimal criterion in MCC by aiming to maintain only those newly-generated environments that are not too hard *and* not too easy for the current population of agents. The result is a *trailblazer* algorithm, one that continually forges new paths to both increasing challenges and skills within a single run. The new challenges are embodied by the new environments that are continually created, and the increasing skills are embodied by the neural network controllers attempting to solve each environment. Existing skills are harnessed both by optimizing agents paired with environments and by attempting to transfer current agent behaviors to new environments to identify promising stepping stones.

The fundamental algorithm of POET is simple: The idea is to maintain a list of active environment-agent pairs `EA_List` that begins with a single starting pair ($E^{\text{init}}(\cdot), \theta^{\text{init}}$), where E^{init} is a simple environment (e.g. an obstacle course of entirely flat ground) and θ^{init} is a randomly-initialized weight vector (e.g. for a neural network). POET then has three main tasks that it performs at each iteration of its main loop: (1) generating new environments $E(\cdot)$ from those currently active, (2) optimizing paired agents within their respective environments, and (3) attempting to transfer current agents θ from one environment to another.

Generating new environments is how POET continues to produce new challenges. To generate a new environment, POET simply mutates (i.e. randomly perturbs) the encoding (i.e. the parameter vector) of an active environment. However, while it is easy to generate perturbations of existing environments, the delicate part is to ensure both that (1) paired agents in the originating (parent) environments have exhibited sufficient progress to suggest that reproducing their respective environments would not be a waste of effort, and (2) when new environments are generated, they are not added to the current population of environments unless they are neither too hard nor too easy for the current population. Furthermore, priority is given to candidate environments that are most novel, which produces a force for diversification that encourages many different kinds of problems to be solved in a single run.

These checks together ensure that the curriculum that emerges from adding new environments is smooth and calibrated to the learning agents. In this way, when new environments do make it into the active population, they are genuinely stepping stones for continued progress and divergence. The population of active environments is capped at a maximum size, and when the size of the population exceeds that threshold, the oldest environments are removed to make room (as in a queue). That way, environments do not disappear until absolutely necessary, giving their paired agents time to optimize and allowing skills learned in them to transfer to other environments.

POET optimizes its paired agents at each iteration of the main loop. The idea is that every agent in POET should be continually improving within its paired environment. In the experiments in this paper, each such iteration is a step of ES, though any reinforcement learning algorithm could conceivably apply. The objective in the optimization step is simply to maximize whatever performance measure applies to the environment (e.g. to walk as far as possible through an obstacle course). The fact that each agent-environment pair is being optimized independently affords easy parallelization, wherein all the optimization steps can in principle be executed at the same time.

Finally, attempting *transfer* is the ingredient that facilitates serendipitous cross-pollination: it is always possible that progress in one environment could end up helping in another. For example, if the paired agent θ^A in environment $E^A(\cdot)$ is stuck in a local optimum, one remedy could be a transfer from the paired agent θ^B in environment $E^B(\cdot)$. If the skills learned in the latter environment apply, it could revolutionize the behavior in the former, reflecting the fact that the most promising stepping stone to the best possible outcome may not be the current top performer in that environment [42, 43, 57]. Therefore, POET continually attempts transfers among the active environments. These transfer attempts are also easily parallelized because they too can be attempted independently.

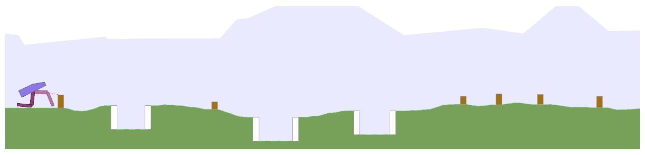


Figure 1: A landscape from the Bipedal Walker environment created by POET. Possible obstacles are stumps, gaps, stairs, and surfaces with different amounts of roughness.

Each step of POET repeats environment generation and optimization of paired agents. Transfer is attempted ever n steps ($n = 25$ in this paper's implementation). Full pseudocode and complete implementation details for POET are given in SI Section A.2.

As noted above, the independence of many of the operations in POET, such as optimizing individual agents within their paired environments and attempting transfers, makes it feasible to harness the power of many processors in parallel. In the implementation of the experiment reported here, each run harnessed 256 parallel CPU cores. Our software implementation of POET (available at https://github.com/uber-research/poet), allows such parallelization over any number of cores.

Provided that a space of possible environmental challenges can be encoded, the hope is that the POET algorithm can then start simply and push outward in parallel along an increasingly difficult frontier of challenges, some benefiting from the solutions to others.

4 EXPERIMENT SETUP AND RESULTS

An effective test of POET should address the hypothesis that it can yield an increasingly challenging set of environments, many with a satisfying solution, all in a single run. Furthermore, we hope to see evidence for the benefit of cross-environment transfers. The domain in this work is a modified version of the "Bipedal Walker Hardcore" environment of the OpenAI Gym [6]. Its simplicity as a 2-D walking domain with various kinds of possible terrain makes it easy to observe and understand qualitatively different ambulation strategies simply by viewing them. Furthermore, the environments are easily modified, enabling numerous diverse obstacle courses to emerge to showcase the possibilities for adaptive specialization and generalization. Finally, it is relatively fast to simulate.

4.1 Environment and Experiment Setup

The agent's hull is supported by two legs (the agent appears on the left edge of figure 1). The hips and knees of each leg are controlled by two motor joints, creating an action space of four dimensions. The agent has ten LIDAR rangefinders for perceiving obstacles and terrain, whose measurements are included in the state space. The 14 other state variables include hull angle, hull angular velocity, horizontal and vertical speeds, positions of joints and their angular velocities, and whether legs touch the ground [6].

Guided by its sense of the outside world through LIDAR and its internal sensors, the agent is required to navigate, within a time limit and without falling over, across an environment of a generated terrain that consists of one or more types of obstacles. These can include stumps, gaps, and stairs on a surface with a variable degree of roughness, as illustrated in figure 1. Reward is given for moving forward, keeping the hull straight, and minimizing motor torque:

$$\text{Reward per step} = \begin{cases} -100, & \text{if robot falls} \\ 130 \times \Delta x - 5 \times \Delta \text{hull\_angle} & \\ -0.00035 \times \text{applied\_torque}, & \text{otherwise.} \end{cases}$$

The episode immediately terminates when the time limit (2,000 time steps) is reached, when the agent falls, or when it completes the course. We define an environment as *solved* when the agent both reaches the far end of the environment and obtains a score ≥ 230 (meaning the walker is reasonably efficient).

The three-layer neural network controller setup, which follows Ha [21], and the ES hyperparameters are given in SI Section A.3.

4.2 Environment Encoding and Mutation

Intuitively, the system should begin with a single flat environment whose paired policy can be optimized easily (to walk on flat ground). From there, new environments will continue to be generated from their predecessors, while their paired policies are simultaneously optimized. The hope is that a wide variety of control strategies and skill sets will allow the completion of an ever-expanding set of increasingly complex obstacle courses, all in a single run.

To enable such a progression, a simple encoding represents the search space of possible environments. There are five types of obstacles that can be distributed throughout the environment: (1) stump height, (2) gap width, (3) step height, (4) step number (i.e. number of stairs), and (5) surface roughness. Three of these obstacles, e.g. stump height, are encoded as a pair of parameters (or *genes*) that form an interval from which the actual value for each instance of that type of obstacle in a given environment is uniformly sampled. As will be described below, in some experiments, some obstacle types are intentionally omitted, allowing us to restrict the experiment to certain types of obstacles. Table 2 in SI Section A.3 gives the parameters for environmental initialization and mutation. When selected to mutate for the first time, obstacle parameters are initialized to the corresponding initial values shown in Table 2 in SI Section A.3. For subsequent mutations, an obstacle parameter takes a *mutation step* (whose magnitude is given in Table 2), and either adds or subtracts the step value from its current value. The value of any given parameter cannot exceed its *maximum value*.

Because the parameters of an environment define a distribution, the actual environment sampled from that distribution is the result of a random *seed*. This seed value is stored with each environment so that environments can be be reproduced precisely, ensuring repeatability. (The population of many environments and the mutation of environments over time still means that training overall does not occur on only one deterministic environment.) With this encoding, all possible environments can be uniquely defined by the values for each obstacle type in addition to the seed that is kept with the environment.

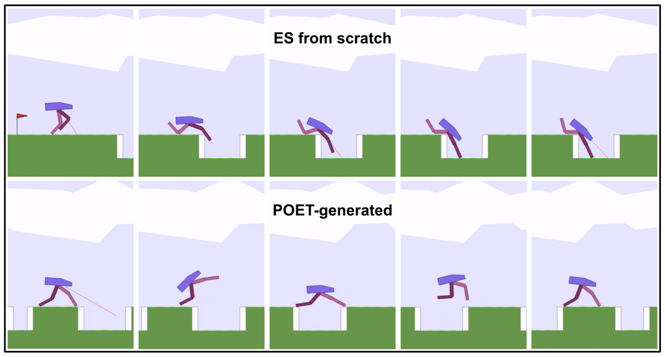
Any environments that meet the eligibility condition for reproduction are allowed to mutate to generate a child environment. In our experiments, this condition is that the paired agent of the environment achieves a reward of 200 or above, which indicates that the agent can reach the end of the terrain (though slightly below the full success criterion of 230). To create children, the set of eligible parent environments is sampled uniformly to choose a parent, which is then mutated to form a child that is added to the list. This process is repeated until a maximum number of children is reached (512 in our experiments). Each child is generated from its parent by independently picking and mutating some or all the available parameters of the parent environment and then choosing a new random seed. The *minimal criterion* (MC) $50 \leq E^{\text{child}}(\theta^{\text{child}}) \leq 300$ then filters out child environments that appear too challenging or too trivial for the current capability level of agents. In case the number of child environments that satisfy the MC is more than the maximum number of children admitted per reproduction, those with lower scores (indicating more room to improve) are admitted until the cap is reached. Algorithmic details are in Algorithm 3 (SI Section A.2.1).

4.3 Results

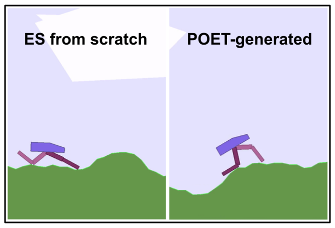
Many of the behaviors reported in this section can be seen in the video at https://youtu.be/D1WWhQY9N4g.

An important motivating hypothesis for POET is that the stepping stones that lead to solutions to very challenging environments are more likely to be found through a divergent, open-ended process than through a direct attempt to optimize in the challenging environment. Indeed, if we take a particular environment evolved in POET and attempt to run ES (the same optimization algorithm used in POET) on that specific environment from scratch, the result is often premature convergence to degenerate behavior. The first set of experiments focus on this phenomenon by running POET with only *one* obstacle type enabled. That way, we can see that even with a single obstacle type, the challenges generated by POET (which POET solves) are too much for ES on its own.

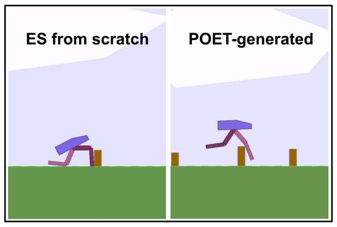
In separate runs, POET generated environments with varying gaps, surface roughness, stump heights, and staircase configurations. We then chose challenging environments for each one of these that were generated and solved by POET. For one environment from each of the four single-obstacle types of environments, we ran ES five times with different initialization and random seeds up to 16,000 ES steps (which is twice as long as Ha [21] gives agents in the same domain with ES; in our experience all ES runs converge long before this point). Such ES-optimized agent consistently get stuck at local minima in these environments. The *maximum* scores out of the five ES runs for the gap, roughness, stump, and staircase environments are 17.9, 39.6, 13.6, and 24.0 respectively, far below the success threshold of 230 that POET exceeds in each case. The result is similar for a hybrid environment from a multi-obstacle run of POET with roughness, gaps, and stumps, where ES achieves a maximum score of just 19.2 on a POET-generated environment where POET exceeds 230 (figure 5 in SI Section A.4). Statistical testing (detailed in SI Section A.4) confirms that in all cases the scores of agents optimized by ES alone are very unlikely to be from a distribution near the POET-level solutions ($p < 0.01$). Note that ES had previously been shown to be a competent approach [20] on the original Bipedal Walker Hardcore environment in OpenAI Gym, which, as illustrated later in Table 1, consists of similar, but much less difficult obstacles. The implication is that the challenges generated and solved by POET are significantly harder than the original Hardcore environment and ES alone is unable to solve them. Several examples of POET-solved environments where ES gets stuck are shown in figure 2. SI Section A.4 includes additional visualizations and statistics illustrating this result.



(a) ES-/POET-generated agents attempting gaps



(b) ES-/POET-generated agents on rough surfaces



(c) ES-/POET-generated agents attempting stumps

Figure 2: POET creates challenging environments and corresponding solutions that cannot be obtained by optimizing randomly initialized agents by ES. As illustrated in the top row of (a) and the left panels of (b) and (c), agents directly optimized by ES converge on degenerate behaviors that give up early in the course. In contrast, POET not only creates these challenging environments, but also learns agents that overcome the obstacles to navigate effectively, as shown in the bottom row of (a) and right panels of (b) and (c).

One interpretation of POET's ability to create agents that can solve challenging problems is that it is in effect an automatic curriculum builder. Building a proper curriculum is critical for learning to master tasks that are challenging to learn from scratch due to a lack of informative gradient. However, building an effective curriculum given a target task is itself often a major challenge. Because newer environments in POET are created through mutations of older environments and because POET only accepts new environments that are not too easy and not too hard for current agents, POET implicitly builds a curriculum for learning each environment it creates. The overall effect is that it is building many overlapping curricula simultaneously, and continually checking whether skills learned in one branch might transfer to another.

A natural question then is whether the environments created and solved by POET can also be solved by an explicit, *direct-path curriculum-building control* algorithm. To test this approach, we first collect a sample of environments generated and solved by POET, and then apply the direct-path control to each one separately to see if it can reach the same capabilities on its own. In this control, the agent is progressively trained on a sequence of environments of increasing difficulty that move towards the target environment. This kind of incremental curriculum is intuitive and variants of it appear in both the evolutionary and ML literature when a task is too hard to learn directly [4, 18, 22, 27, 28]. The sequence of environments starts with an environment of only flat ground without any obstacles (which is easy enough for any randomly-initialized agent to quickly learn to complete). Then each of the subsequent environments are constructed by slightly modifying the current environment. More specifically, to get a new environment, each obstacle parameter of the current environment has an equal chance of staying the same value or *increasing* by the corresponding mutation step value in Table 2 (in SI; the same step sizes used by POET) until that obstacle parameter reaches that of the target environment.

	Top Value in range of Stump Height	Top Value in range of Gap Width	Roughness
This Work	≥ 2.4	≥ 6.0	≥ 4.5
Reference	2.0	3.0	1.0

Table 1: Difficulty level criteria. The difficulty level of an environment is based on how many conditions it satisfies out of the three listed here. The *reference* in the second row shows the corresponding top-of-range values used in the original Hardcore version of Bipedal Walker in OpenAI Gym [30].

In this direct-path curriculum-building control, the agent moves from its current environment to the next when its score in the current environment reaches the reproduction eligibility threshold for POET, i.e. the same condition for when an environment reproduces in POET. The control algorithm optimizes the agent with ES (just as in POET). It stops when the target environment is reached and solved, or when a computational budget is exhausted. To be fair, each run of the control algorithm is given the same computational budget (measured in total number of ES steps) spent by POET to solve the environment, which includes all the ES steps taken in the entire sequence of environments along the direct line of ancestors (taking into account transfers) leading to the target.

The direct-path curriculum-building control is tested against a set of environments generated and solved by POET that encompass a range of difficulties. For these experiments, the environments have three obstacle types enabled that can be combined in the same obstacle course: gaps, roughness, and stumps. To provide a principled framework for choosing the set of generated environments to analyze, they are classified into three difficulty levels. The difficulty level of an environment is based on how many conditions it satisfies out of the three listed in Table 1. In particular, a *challenging* environment satisfies one of the three conditions; a *very challenging* environment satisfies two of the three; and satisfying all three makes an environment *extremely challenging*. It is important to note that these conditions all merit the word "challenging" because they all are much more demanding than the corresponding values from the original Hardcore version of Bipedal Walker in OpenAI Gym [30] used in Ha [21] (denoted as *reference* in Table 1).

In this experiment, each of three runs of POET takes up to 25,200 POET iterations with a population size of 20 active environments, while the number of sample points for each ES step is 512. These runs each take about 10 days to complete on 256 CPU cores. The mean (with 95% confidence intervals) POET iterations spent on solving challenging, very challenging, and extremely challenging

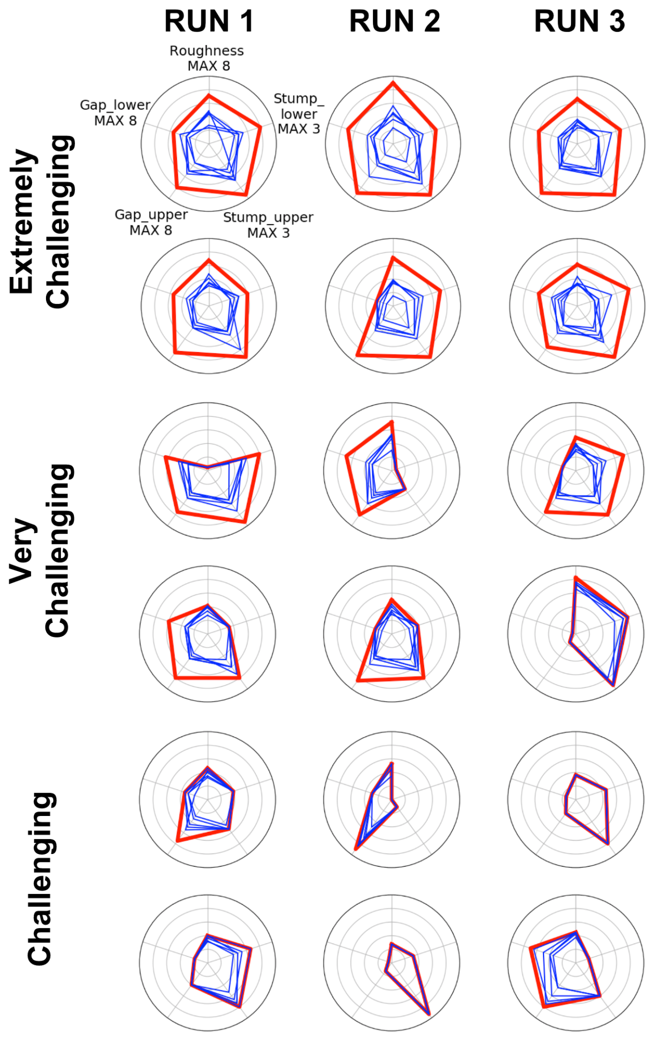


Figure 3: POET versus direct-path curriculum-building controls. Each rose plot depicts one environment that POET created and solved (red pentagon). For each, the five blue pentagons indicate what happens in control runs when the red pentagon is the target. Each blue pentagon is the closest-to-target environment solved by one of the five independent runs of the direct-path control algorithm. The five vertices of each pentagon indicate the environment parameters (see key, upper-left). The value after MAX in the key is the maximum value at the outermost circle for each type of obstacle. Each column contains sample solved environments from a single independent run of POET.

environments starting from the iteration when they were first created are 638 ± 133, 1,180 ± 343, and 2,178 ± 368, respectively. Here, one POET iteration refers to creating new environments, optimizing current paired agents, and possibly (i.e. every 25 iterations) attempting transfers (i.e. lines 4-25 in Algorithm 2 in SI). The more challenging environments clearly take more effort to solve.

Figure 3 compares the POET environments and the direct-path curriculum-building control algorithm through a series of *rose plots*, each of which compares the configuration of an environment solved by POET (red pentagons) with the closest that the direct-path control could come to that configuration (blue pentagons). For each red pentagon there are five such blue pentagons, each representing one of five separate attempts by the control to achieve the red pentagon target. The five vertices of each pentagon indicate roughness, the lower and upper bounds of the range of the gap width, and those of the stump height, respectively. Each column in figure 3 consists of six representative samples (red pentagons) of environments that a single run of POET up to 25,200 iterations created and solved. As the rows descend from top to bottom, the difficulty level decreases (all are randomly sampled from targets generated and solved at each difficulty level in each run). Figure 3 clearly shows that attempts by the direct-path control consistently fail to reach the same difficulty level as environments that POET generated and solved.

To more precisely quantify these results, we define the normalized distance between any two environments, E_A and E_B as $\frac{1}{\beta}||\frac{e(E_A)-e(E_B)}{e(E_{\text{Max}})}||_2$, where $e(E)$ is the genetic encoding vector of E, and E_{Max} is a hypothetical environment (for normalization purpose) with all genetic encoding values maxed out. (The maximum values are roughness $= 8$, Gap_lower $=$ Gap_upper $= 8$, Stump_lower $=$ Stump_upper $= 3$.) The constant $\beta = \sqrt{5}$ is simply for normalization so that the distance is normalized to 1 when $E_A = E_{\text{Max}}$ and E_B is a purely flat environment (roughness zero) without any obstacles (i.e. an all-zero genetic encoding vector). Based on this distance measure, we can calculate the median values and statistics of distances between target environments (created and solved by POET) and the corresponding closest-to-target environments that the control algorithms can solve.

An analysis of the results is based on the *distance between the closest environment reached by the direct-path control and the POET-generated target.* First, we can examine whether that distance is significantly greater the higher the challenge level. Indeed, Mann-Whitney U tests show that the difference between such distances between challenging and very challenging environments is indeed very significant, as is the difference between such distances between very challenging and extremely challenging ($p < 0.01$ for both). This test gives a quantitative confirmation of the intuition that these challenge levels are indeed meaningful, and the higher they go, the more out of reach they become for the control. Second, a single-sample t-test (which was previously used when comparing POET to the ES-alone control, detailed in SI Section A.4) can also here provide confidence that the distances from the targets are indeed far in an absolute sense: indeed, even at the lowest challenge level (and for all challenge levels), the one-sample t-test measures high significance ($p < 0.01$) between the distribution of distances from the target and the target level. Qualitatively, the main result is that POET creates and solves environments that the control algorithm fails to solve at very and extremely challenging difficulty levels, while the curriculum-based control algorithm can sometimes (though not always) solve environments at the lowest challenge level. One implication of these results is that the direct-path curriculum-building control is valid in the sense that it does perform reasonably well at solving minimally challenging scenarios. However, the very and extremely challenging environments that POET invents reach significantly beyond what the direct-path curriculum can match.

In aggregate, the results from the direct-path curriculum-building control help to show quantitatively the advantage of POET over conventional curriculum-building. In effect, the ability to follow multiple chains of environments in the same run and transfer skills among them pushes the frontier of skills farther than a single-chain curriculum can push, hinting at the limitations of preconceived curricula in general.

A fundamental problem of a pre-conceived direct-path curriculum (like the control algorithm above) is the potential lack of necessary *stepping stones*. In particular, skills learned in one environment can be useful and critical for learning in another environment. Because there is no way to predict where and when stepping stones emerge, the need arises to conduct transfer experiments (which POET implements) from differing environments or problems [43].

To give a holistic view of the success of transfer throughout the entire system, we count the number of *replacement attempts* during the course of a run, which means the number of times an environment took a group of incoming transfer attempts from all the other active environments. The total number of such replacement attempts in RUN 1, RUN 2, and RUN 3 (labelled in Figure 3) are 18,894, 19,014, and 18,798, respectively, out of which, 53.62%, 49.26%, 48.89%, respectively, are successful replacements. Note that each replacement attempt here encompasses both the direct transfer and proposal transfer attempts. These statistics show how pervasively transfer permeates (and often adds value to) the parallel paths explored by POET.

While transfer is pervasive, that does not in itself prove it is essential. To demonstrate the value of transfer, a control is needed: We relaunched another three POET runs, but with all the transfers *disabled* (which we call *POET without transfer*). In this variant, POET runs as usual, but simply never tries to transfer paired solutions from one environment to another. We can then calculate the *coverage* of the environments that are created and solved by POET and the control, respectively, following a similar metric as defined in Lehman and Stanley [37]: We first uniformly sample 1,000 *challenging*, 1,000 *very challenging*, and 1000 *extremely challenging* environments. For each of the total 3,000 sampled environments, the distance to the nearest one of the *challenging*, *very challenging*, or *extremely challenging* environments created and solved by POET (and by the control, respectively) is calculated. Note that the better covered the environment space is, the lower the sum of all such nearest distances will be. The result is that the coverage of environments created and solved by POET is significantly greater than by POET without transfer ($p < 2.2e{-}16$ based on Mann-Whitney U test). In an even more dramatic statistic showing the essential role of transfer in open-ended search, in POET without transfer, *no extremely challenging environments are solved at all.* Figure 7 in SI Section A.4.1 illustrates this stark contrast.

Finally, one of the primary hypothesized benefits of POET is its ability to produce a broad diversity of different problems with functional solutions in a single run. All three POET runs created and solved sufficiently diverse environments to cover all three challenge levels. For example. the environments (depicted as red pentagons) shown in each column of Figure 3 are created and solved in a *single* run of POET. Each such column exhibits diversity in the values and/or value ranges in roughness, gap width of gaps, and height of stumps. Figure 8 in SI Section A.4.2 gives a more visual depiction of the range of environments solved in a single run. The diversity of environments also implies diverse experiences for the agents paired with them, who in turn thereby learn diverse walking gaits. This diversity then supplies the stepping stones that fuel the mutual transfer mechanism of POET.

5 DISCUSSION AND FUTURE WORK

POET is an attempt to move further down the road towards open-ended systems. While the road remains long, the rewards for machine learning of beginning to capture the character of open-ended processes is potentially high. First, as the results show, there is the opportunity to discover capabilities that could not be learned in any other way, even through a carefully crafted curriculum targeted at the desired result. In addition, a diversity of such results can be generated in a single run, and the problems and solutions can both increase in complexity over time. Furthermore, POET is self-generating multiple curricula simultaneously, all while leveraging the results of some as stepping stones to progress in others.

POET becomes more interesting the more unbounded its problem space becomes. The present space of 2-D walking environments is limited by the maximal ranges of the genomes describing the environment. Much more flexible, or even unbounded environment encodings, such as an indirect encoding like compositional pattern-producing networks (CPPNs) [56], can potentially enable POET to traverse a far richer problem space.

While ES optimizes solvers in this paper for various tasks under POET, and underlies the main transfer mechanism, POET can in principle be instantiated with other evolutionary algorithms and in fact with any RL or optimization algorithm, including those outside evolutionary computation–an intriguing opportunity to merge evolutionary computation with other fields to achieve open-endedness. Policy gradient methods [50], Q-learning [41], genetic algorithms, other variants of ES, diversity-promoting algorithms such as QD [9, 37, 42, 45] and NS [38], including NS-/NSRA-ES [8], and many other such algorithms are all viable alternatives.

POET could substantially drive progress in the field of meta-learning, wherein neural networks are exposed to many different problems and get better over time at learning how to solve new challenges (i.e. they learn to learn). Meta-learning requires access to a distribution of different tasks, and that traditionally requires a human to specify this task distribution, which is costly and may not be the right or best distribution on which to learn to learn. Gupta et al. [19] note that the performance of meta-learning algorithms critically depends on the distribution of tasks they meta-train on, and POET can offer a divergent engine to create such task diversity. POET can also be extended to take a more explicit approach to optimizing for generality, such as in CMOEA [24].

Finally, it is exciting to consider for the future the rich potential for surprise in all the possible domains where POET might be applied. For example, 3-D parkour was explored by Heess et al. [23] in environments created by humans, but POET could invent its own creative parkour challenges and their solutions. The soft robots evolved by Cheney et al. [7] would also be fascinating to combine with ever-unfolding new obstacle courses. The scope is broad for imagination and creativity in the application of POET.

REFERENCES

[1] Charles W. Anderson. 1989. Learning to Control an Inverted Pendulum Using Neural Networks. *IEEE Control Systems Magazine* 9 (1989), 31–37.

[2] Mark Bedau. 2008. The Arrow of Complexity Hypothesis (Abstract). In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, Seth Bullock, Jason Noble, Richard Watson, and Mark Bedau (Eds.). MIT Press, Cambridge, MA, 750. http://www.alifexi.org/papers/ALIFExi-abstracts-0010.pdf

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The Arcade Learning Environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)* 47 (2013), 253–279.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 41–48.

[5] Jonathan C. Brant and Kenneth O. Stanley. 2017. Minimal Criterion Coevolution: A New Approach to Open-Ended Search. In *Proceedings of the 2017 on Genetic and Evolutionary Computation Conference (GECCO)*. 67–74.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540

[7] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. 2013. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*. ACM Press, New York, NY.

[8] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. 2018. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. In *Advances in Neural Information Processing Systems (NeurIPS) 31*. 5032–5043.

[9] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. 2015. Robots that can adapt like animals. *Nature* 521 (2015), 503–507. https://doi.org/10.1038/nature14422

[10] Edwin D De Jong. 2004. The incremental pareto-coevolution archive. In *Genetic and Evolutionary Computation Conference*. Springer, 525–536.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), 248–255.

[12] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. *arXiv preprint arXiv:1901.10995* (2019).

[13] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2018. Diversity is All You Need: Learning Skills without a Reward Function. *arXiv preprint arXiv:1802.06070* (2018).

[14] S.G. Ficici and J.B. Pollack. 1998. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. *Artificial life VI* (1998), 238.

[15] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. 2018. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*. 1514–1523.

[16] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse Curriculum Generation for Reinforcement Learning. In *Proceedings of the 1st Annual Conference on Robot Learning*. 482–495.

[17] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. 2017. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190* (2017).

[18] Faustino Gomez and Risto Miikkulainen. 1997. Incremental Evolution of Complex General Behavior. *Adaptive Behavior* 5 (1997), 317–342. gomez:ab97

[19] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. 2018. Unsupervised Meta-Learning for Reinforcement Learning. *arXiv* (2018). arXiv:1806.04640 http://arxiv.org/abs/1806.04640

[20] David Ha. 2017. Evolving stable strategies. http://blog.otoro.net/. (2017).

[21] David Ha. 2018. Reinforcement Learning for Improving Agent Design. *arXiv preprint arXiv:1810.03779* (2018).

[22] Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

[23] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. 2017. Emergence of Locomotion Behaviours in Rich Environments. *CoRR* abs/1707.02286 (2017). arXiv:1707.02286 http://arxiv.org/abs/1707.02286

[24] Joost Huizinga and Jeff Clune. 2018. Evolving Multimodal Robot Behavior via Many Stepping Stones with the Combinatorial Multi-Objective Evolutionary Algorithm. *arXiv preprint arXiv:1807.03392* (2018).

[25] Joost Huizinga, Jean-Baptiste Mouret, and Jeff Clune. 2016. Does Aligning Phenotypic and Genotypic Modularity Improve the Evolution of Neural Networks?. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference (GECCO)*. 125–132.

[26] Edwin D De Jong and Jordan B Pollack. 2004. Ideal evaluation from coevolution. *Evolutionary computation* 12, 2 (2004), 159–192.

[27] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. 2018. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. *arXiv preprint arXiv:1806.10729* (2018).

[28] Andrej Karpathy and Michiel Van De Panne. 2012. Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence*. Springer, 325–330.

[29] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[30] O. Klimov. 2016. BipedalWalkerHardcore-v2. https://gym.openai.com. (2016).

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[32] W. B. Langdon. 2005. Pfeiffer – A Distributed Open-ended Evolutionary System. In *AISB'05: Proceedings of the Joint Symposium on Socially Inspired Computing (METAS 2005)*, Bruce Edmonds, Nigel Gilbert, Steven Gustafson, David Hales, and Natalio Krasnogor (Eds.). 7–13. http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/wbl_metas2005.pdf

[33] Yann LeCun and Corinna Cortes. 1998. The MNIST database of handwritten digits. (1998).

[34] Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. 2018. ES is More Than Just a Traditional Finite-difference Approximator. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. ACM, New York, NY, USA, 450–457. https://doi.org/10.1145/3205455.3205474

[35] Joel Lehman and Kenneth O. Stanley. 2010. Revising the evolutionary computation abstraction: minimal criteria novelty search. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation (GECCO '10)*. ACM, New York, NY, USA, 103–110. https://doi.org/10.1145/1830483.1830503

[36] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary Computation* 19, 2 (2011), 189–223. http://www.mitpressjournals.org/doi/pdf/10.1162/EVCO_a_00025

[37] Joel Lehman and Kenneth O. Stanley. 2011. Evolving a diversity of virtual creatures through novelty search and local competition. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 211–218.

[38] Joel Lehman and Kenneth O. Stanley. 2011. Novelty Search and the Problem with Objectives. In *Genetic Programming Theory and Practice IX (GPTP 2011)*.

[39] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2017. Teacher-Student Curriculum Learning. *arXiv preprint arXiv:1707.00183* (2017).

[40] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*. 1928–1937.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[42] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[43] Anh Nguyen, Jason Yosinski, and Jeff Clune. 2016. Understanding Innovation Engines: Automated Creativity and Improved Stochastic Optimization via Deep Learning. *Evolutionary Computation* 24, 3 (2016), 545–572.

[44] Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. De Jong. 2012. *Coevolutionary Principles*. Springer Berlin Heidelberg, Berlin, Heidelberg, 987–1033. https://doi.org/10.1007/978-3-540-92910-9_31

[45] Justin K Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. 3, 40 (2016).

[46] Ingo Rechenberg. 1978. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*. 83–114.

[47] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[48] Nikolay Savinov, Anton Raichuk, Raphael Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. 2018. EPISODIC CURIOSITY THROUGH REACHABILITY. *arXiv preprint arXiv:1810.0227* (2018).

[49] Jürgen Schmidhuber. 2013. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology* 4 (2013), 313.

[50] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[51] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. 2010. Parameter-exploring policy gradients. *Neural Networks* 23, 4 (2010), 551–559.

[52] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games*. Springer.

[53] L.B. Soros, Nick Cheney, and Kenneth O Stanley. 2016. How the Strictness of the Minimal Criterion Impacts Open-Ended Evolution. In *ALIFE 15: The Fifteenth*

International Conference on the Synthesis and Simulation of Living Systems. 208–215.

[54] L.B. Soros and Kenneth O Stanley. 2014. Identifying Necessary Conditions for Open-Ended Evolution through the Artificial Life World of Chromaria. In *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. 793–800.

[55] Russell K Standish. 2003. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications* 3, 02 (2003), 167–175.

[56] Kenneth O. Stanley. 2007. Compositional Pattern Producing Networks: A Novel Abstraction of Development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems* 8, 2 (2007), 131–162.

[57] Kenneth O Stanley and Joel Lehman. 2015. Why Greatness Cannot Be Planned. (2015).

[58] Kenneth O. Stanley, Joel Lehman, and Lisa Soros. 2017. Open-endedness: The last grand challenge you've never heard of. *O'Reilly Online* (2017). https://www.oreilly.com/ideas/open-endedness-the-last-grand-challenge-youve-never-heard-of

[59] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1.

[60] Tim Taylor, Mark Bedau, Alastair Channon, and David Ackley et al. 2016. Open-Ended Evolution: Perspectives from the OEE Workshop in York. *Artificial life* 22, 3 (2016), 408âĂŞ423.

[61] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.

[62] R. Paul Wiegand, William C. Liles, and Kenneth A. De Jong. 2001. An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation (GECCO'01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1235–1242. http://dl.acm.org/citation.cfm?id=2955239.2955458

[63] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. 2008. Natural evolution strategies. In *Evolutionary Computation, 2008*. 3381–3387.

[64] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[65] Brian G. Woolley and Kenneth O. Stanley. 2011. On the deleterious effects of a priori objectives on evolution and representation. In *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, Dublin, Ireland, 957–964. https://doi.org/doi:10.1145/2001576.2001707

[66] Xingwen Zhang, Jeff Clune, and Kenneth O. Stanley. 2017. On the Relationship Between the OpenAI Evolution Strategy and Stochastic Gradient Descent. *arXiv preprint arXiv:1712.06564* (2017).