Quality Diversity in the Amorphous Fortress: Evolving for Complexity in 0-Player Games

Anonymous Author(s)
Affiliation
Address
email

Abstract

We explore the generation of diverse environments using the Amorphous Fortress (AF) simulation framework. AF defines a set of Finite State Machine (FSM) nodes and edges that can be recombined to control the behavior of agents in the 'fortress' grid-world. The behaviors and conditions of the agents within the framework are designed to capture the common building blocks of multi-agent artificial life and reinforcement learning environments. Using quality diversity evolutionary search, we generate diverse sets of environments that exhibit dynamics exhibiting certain types of complexity according to measures of agents' FSM architectures and activations, and collective behaviors. QD-AF generates families of 0-player akin to simplistic ecological models, and we identify the emergence of both competitive and co-operative multi-agent and multi-species survival dynamics. We argue that these generated worlds can collectively serve as training and testing grounds for learning algorithms.

1 Introduction

Games with certain open-ended characteristics, such as sandbox simulation, management, or agentic multi-agent games, provide promising testbeds for learning agents Earle et al. [2021], Fan et al. [2022], Suarez et al. [2021]. The latter type, for example, allow for a range of potentially interesting interactions between artificial agents, often leading to emergent phenomena unforeseen by developers Guttenberg and Soros [2023].

The Amorphous Fortress framework-[Charity et al., 2023] is a simulation framework that uses finite-state machines (FSMs) to produce emergent AI behaviors. Drawing inspiration from games such as Dwarf Fortress and Rogue, AF defines a base reality consisting of a fortress, where multiple instances of FSM agents interact with each other. Roughly speaking, the FSM agents might be interpreted as simple caricatures as magical animals, with the ability to hunt, transform and breed, which behaviors are triggered by temporally/spatially conditions dependent upon the agent's state 1. Prior work shows that fortress environments can evolve the FSM entities towards "interesting" behaviors by maximizing (a proxy for) the complexity of graphs constituting FSMs. Given this objective, a hill-climber algorithm generates fortresses that exhibit symbiotic relationships between entities, where entity classes with both large and small FSM graphs are integral to the fitness in the fortress. From this, a variety of entity classes emerge, with diverse policies of agent behavior that depend on one another for deeper exploration of their own graphs.

In this paper, we extend previous work by optimizing both quality and diversity using metrics reflecting agent behavior and interaction. We implement the quality diversity (QD) algorithm MAP-Elites to evolve a grid of Amorphous Fortress fortress environments. Diversity is maintained in these fortresses by their defined behavior characteristics (BCs). These BCs are measured based on the agent action space class definitions and the ending state of the fortress after simulation. Quality of

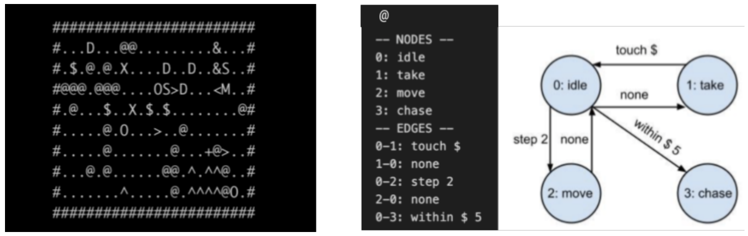


Figure 1: **Left: example of a random fortress** generated in the Amorphous Fortress framework. The fortress environment contains many instances of different entity class types. **Right: An example entity class FSM definition**. This FSM defines the action behavior an instance agent of this class can take within the fortress under the right conditions. Here, the agent is capable of random movement as well as 'chasing' (moving along a path toward) and 'taking' (removing from the fortress) an instance of the $ entity class.

the fortresses is maintained by evaluating what proportion of each entity class's finite-state machine definition was explored collectively during simulation. We use these metrics to generate a collection of fortresses that exhibit a range of specific behaviors during simulation

We argue that the diversity and complexity of environments generated by QD-AF can act as a general testbed for evaluating RL agents. Like prior work in Unsupervised Environment Design in RL ([Parker-Holder et al., 2022, Jiang et al., 2021, Dennis et al., 2020, Wang et al., 2020]), QD-AF retains some of the generality feedback from agent behavior. But it uses fixed agents comprising modular FSMs, eliding the non-stationarity of learning agents. By relying solely on agent behavior, our method is independent of the semantics of the environments in which agents are trained or deployed, and can generate diverse environments with varying potential interpretations. At the same time, we select diversity metrics that will naturally affect the difficulty of tasks for a learning agent within the fortress. Feedback from a learning agent, embodied in varying fortresses, could then be used to select between subregions of the pre-generated archive, potentially serving as a more stable and/or nimble environment curator than processes that mutate low-level environment parameters at run-time, and perform search/learning concurrently with RL players.

2 Background and related works

This extension of the Amorphous Fortress system emphasizes the themes of multi-agent open-ended environments and QD algorithms. The following subsections describes each theme in more detail along with previous works related to this experiment.

2.1 Open-ended systems

Simulation environments allow researchers to emulate real world events and phenomena in a controlled test framework. Open-ended simulation environments and environments that promote artificial life offer a multitude of challenges and emergent scenarios for AI to solve [Bedau et al., 2000, Stanley et al., 2017]. Examples of simulation environments have previously been studied by game AI researchers for developing both the artificial agents and the environments themselves. Charity et al. [2020] and Green et al. [2021] introduce minimal simulations of The Sims and RollerCoaster Tycoon environments respectively to generate novel and diverse layouts based on the game environment. Similarly, Earle [2020] introduces a training environment in the game SimCity and examines population behaviors of cellular automata in Conway's Game of Life. More recent works, such as Griddly, developed by Bamford [2021], and Maestro developed by Samvelyan et al. [2023], have developed custom open-ended environments to examine agent interactions—particularly for reinforcement learning (RL) tasks. Zhang et al. [2023] use human notions of interestingness in conjunction with Large Language Models to explore and facilitate open-ended learning. With Amorphous Fortress, we use an open-ended artificial simulation environment to investigate how agent learning models such as RL agents can interact with the generated agents in a specific environment setup.

2.2 Multi-agent interactions

In some open-ended environments, multiple decision-making agents interact with each other co-operatively or competitively to achieve their common or opposing objectives. In the framework proposed by Grbic et al. [2021], multiple agents are interacting co-operatively to build complex block structures inspired by Minecraft. Deshpande and Magerko [2021] designed the application Drawcto, which uses multiple agents that are capable of co-creating interpretive open-ended artwork with human collaborators. Neural MMO [Suarez et al., 2021], is a platform for multi-agent RL over large agent populations in procedurally generated virtual maps. Lowe et al. [2017] introduce a method that successfully learns RL policies that require complex multiagent co-operation and coordination.

Moreover, the diverse interactions among these multiple agents could give rise to interesting emergent behavior within the given context. The work by Bansal et al. [2017] and Baker et al. [2019] introduce the simulation of diverse environments, where multiple agents engage with each other competitively, leading to the rise of intricate and complex emergent behaviors. Such emergent behaviors include offensive and defensive game playing strategies like blocking and kicking [Bansal et al., 2017] or object manipulation within the environment such as “box-surfing” and building barricades [Baker et al., 2019]. We employ multi-agent interactions to enhance the diverse and interesting emergent behavior of different entity class types within the fortress environment.

2.3 Quality diversity algorithms

Evolutionary algorithms are gradient-free optimization methods that randomly mutates pools of individuals to maximize a computable objective function [Norvig and Intelligence, 2002]. Novelty search replaces the objective function with a measure of an individual sample’s phenotypic/behavioral distance from the existing archive of discovered individuals, in effect uniform randomly sampling the search space Doncieux et al. [2019], often achieving the (held-out) objective given a sufficiently informative behavior distance metric. It has been used to generate diverse game AI components such as video game levels Beukman et al. [2022] and dungeons Melotti and de Moraes [2018].

Quality Diversity (QD) algorithms Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [Mouret and Clune, 2015] both optimize for a fixed objective while tessellating a behavioral search space and preventing competition between elites in different cells. MAP-Elites has been used to generate teams of agents for automated gameplaying [Guerrero-Romero and Perez-Liebana, 2021]. MAP-Elites has also been used to create, replicate and explore real-world adaptability by simulated agents in virtual open-ended environment as studied by Norstein et al. [2022]. Pierrot and Flajolet [2023] evolve repertoires of full agents to combine any RL algorithm with MAP-Elites to dynamically learn the hyperparameters of the RL agent. This approach not only alleviates the user’s workload but also enhances performance in the evaluated environments. We use MAP-Elites to generate the multiple finite-state machine agents that would be ideal use case environments for training agent learning models.

2.4 Amorphous Fortress 1.0 framework

The Amorphous Fortress framework, developed by Charity et al. [2023], is an artificial life simulation system has a hierarchy of 3 components: entities (the agent class of the system) the fortress object (the environment class of the system) and the engine (the “manager” and main loop of the simulation). Each entity of the Amorphous Fortress is defined by a singular ASCII character, a unique 4-bit identification hex number and a finite-state machine (FSM) specifying its behavior during simulation. The finite-state machine entity class definition is made up of a list of nodes and a set of edges. Each node in the FSM graph represents a potential action state an entity instance can be in. These actions define how an instance interacts with the environment. The edges define when an instance of the entity class can change states to another node and is dependent and prioritized based on conditions found during simulation. Table 1 shows the possible action nodes and Table 2 shows the possible conditional edges that can define an entity class for the Amorphous Fortress 1.0 System. At any time during the simulation, the entity is always in a state at one of the set nodes. At each timestep—a single update within the fortress environment—each connection is evaluated to move to the connecting node state based on whether the conditions are met, in order of priority defined internally. The agent will perform the action at its new current node on the next timestep.

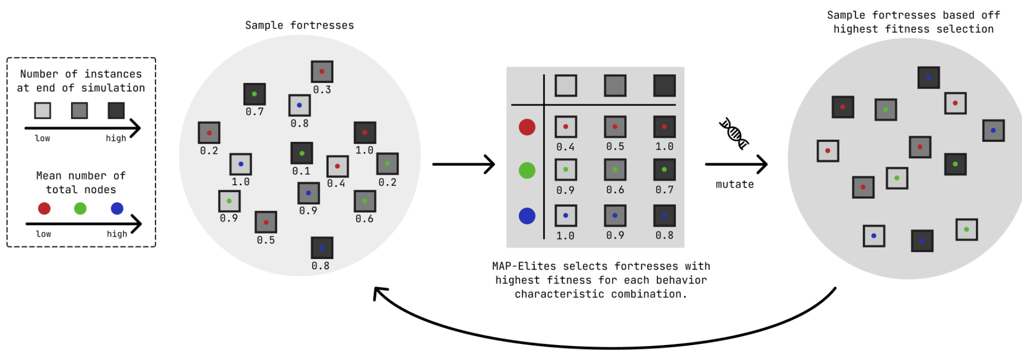


Figure 2: Diagram of the MAP-Elites algorithm applied to the Amorphous Fortress QD experiment

The fortress of the Amorphous Fortress contains the environment where the simulation takes place and stores general information accessible to all of the entities in the fortress. The borders and size of the fortress are defined initially with a set character width w and height h to enclose the entities. On initialization, the fortress generates each entity class FSM for each character defined at the start of the simulation. This global dictionary of entity classes allows any instance of an entity to add or transform different entity instances even if none exist on the map at initialization. The fortress maintains a list of currently active entity instances in the simulation and adds or removes them by their ID value. The fortress also maintains positional data about each entity to return for conditional checks (i.e. whether a particular position has an instance of an entity class located there.) The engine of the Amorphous Fortress system maintains the execution of the simulation and exports any log files or entity class data information such as the node and edges definitions. The fortress area itself where the entities interact is 15 spaces wide by 8 spaces tall—including the walls—allowing for a total traversable area of 78 tiles.

As an update from this first Amorphous Fortress framework, the *push* action node was modified to include an entity character target. The *move_wall* action node is also an addition from the previous iteration of the Amorphous Fortress system.

3 Methods

3.1 MAP-Elites for Amorphous Fortress

For this paper, we implement the MAP-Elites QD algorithm Mouret and Clune [2015] to evolve multiple entity classes towards a diversity of emergent behaviors. The emergent behaviors of the entities defined within this system are dependent on interactions with other instances within the same fortress. Therefore a single cell of the MAP-Elites grid contains a fortress with its own set of entity class definitions. Figure 2 illustrates a small example of the evolution and evaluation process as the fortresses are placed in the MAP-Elites grid for the experiment (described in more detail later in this section).

Behavior Characteristics For a MAP-Elites implementation, the dimensions of the archived QD grid are known as the behavior characteristics (BCs). These BCs designate how the individuals from a population are separated and maintained for sample diversity and replaced within the cell to improve quality. For this paper, we define the following behavior characteristics that are used for experiment of this paper: a) the mean number of total instances in the fortress at the end of the simulation and b) the mean number of total nodes across all entity class definitions.

For behavior characteristic (a) based on the number of entity instances, this dimension is intended to explore the population of a fortress, whether the entity class combinations result in an overpopulation of entity instances, an extinction of all instances, or a stability or "equilibrium" of instances within the fortress. The values of this dimension can range from 0 to 156—the maximum number of instances allowed to exist in the fortress before it terminates based on an "overpopulation" condition.

Behavior characteristic (b), based on the collective class FSM size, looks to examine the "complexity" and "depth" of the entity classes; whether the combined set includes a majority of simple entity class definitions with only 1 action node or conversely with extremely large entity class definitions. The values of this dimension can range from 15 nodes—where each of the entity classes has only 1 node—to 1400 nodes—where each possible node is included every entity class FSM definition. The exploration of this dimension by the MAP-Elites algorithm will demonstrate the growth and utility of varying sized entity class FSMs.

Population An evaluated population consists of a fortress with a set of agent entity class definitions. The population size for this paper was 10 individuals per generation—9 sampled from elites and 1 randomly created similar to the initialization. The singular random fortress is injected into the set of mutant elites to encourage exploration within the MAP-Elites grid and to prevent the algorithm from reaching a local minimum during evolution. We parallelize the evaluation of these 10 individuals to speed up evolution. On initialization, all 15 of the entity class FSMs are defined for each fortress individual. In this step, the number of total nodes to be added over all FSMs in the fortress is sampled uniformly to encourage a larger spread of randomly initialized individuals over the MAP-Elites grid (along the n. nodes axis), thus allowing for more uniform exploration of cells as evolution progresses.

We randomly initialize fortresses so as to sample uniformly along the axis measuring number of aggregate FSM nodes. We first uniformly sampling this aggregate number, then split it into as many summands as there are entity types using an evenly weighted multinomial distribution, where each summand corresponds to the number of nodes to be assigned a given entity. It is possible in this setting for an entity type to be assigned more nodes than there are distinct node types; in this case, we (greedily) re-assign the surplus nodes to one or more non-overfilled entity classes, until no surplus nodes remain.

Mutation A fortress individual in the population is mutated by modifying its genotype: the class level definitions of the FSMs. Each fortress contains 15 entity classes, where each class can have a minimum of 1 action node and a maximum of 95 action nodes. This process is done similarly to the first Amorphous Fortress work by Charity et al. [2023], where separate coin-flip probabilities determine whether a node, edge, and/or entity instance in the fortress itself is added, removed, or altered. However, as a modification for the MAP-Elites experiment, the node mutation is adjusted to increase exploration within the grid. Unlike the previous experiment where a single node was modified per coin-flip chance, a range of nodes can be added, removed, or altered into another action node definition. For example, 10 nodes can be added to one entity class definition, while 4 are removed from another (or the same if randomly chosen again). Algorithm 1 shows a pseudocode algorithm for the mutation process of the evolution.

Fitness The fortress sample individuals are evaluated based on the ending state of the fortress. The fortress is simulated for 100 steps, where each instance of an entity class present in the fortress enacts the current action node of its FSM graph once per step and then evaluates the next action node to move to based on the state conditions it ends in. The fitness function of the MAP-Elites implementation of the system is similar to the hill-climber experiment from the original Amorphous Fortress work, which is based on the average proportion of nodes and edges that have been explored, i.e., the percentage of nodes over the whole entity class activated during simulation by all instances of said class. This fitness definition encourages each class entity to have the full possibility of its emergent behaviors demonstrated within the simulation. The final exploration of a class definition's nodes and edges are also aggregated over evaluation trials in case different behaviors occur due to different seed evaluations. The fitness function for a fortress is defined with the following equation: $f = e/t$ where f is the fitness value from 0 to 1, e is the total number of explored nodes—nodes that were activated during simulation—for all entity class definitions in the fortress individual and t is the total number of nodes—activated or un-activated—for each entity class in the fortress individual.

Entropy of the FSM definitions Entropy examines the distribution of the sizes across the entity class definitions. The values of this dimension ranges from 0 to 1, with 0 meaning all of the entity class FSMs have the same number of nodes and no variation, and 1 meaning the number of nodes are different for each entity class FSM definition. We use Shannon Entropy to calculate the entropic value of the FSM sizes with a b base N where N is the number of FSM size bins (which for this experiment is equal to the number of entity classes).

Figure 3 shows how the BCs, fitness value, and entropy value are calculated for any given fortress.

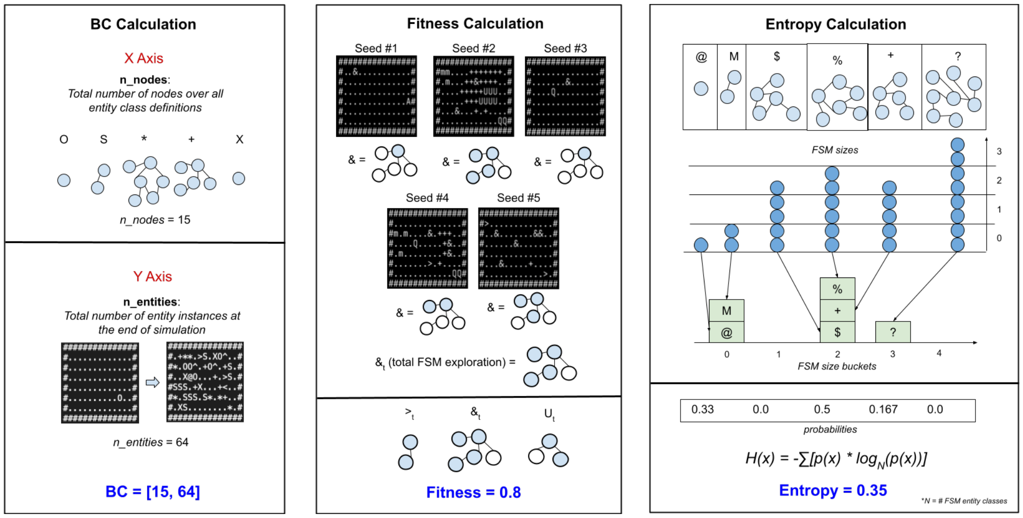


Figure 3: Diagram of the calculations for MAP-Elites BCs, fitness values, and entropy value

3.2 Experiment setup

Using the fitness function, AF mutation operators, and AF initialization schemes described above, we use standard MAP-Elites to iteration on a population of AF samples, evaluating them to determine fitness measures and characteristics, sorting them into their respective MAP-Elites grid cells, re-sampling from the grid to create the new population of samples and repeating for a set number of generations. In this experiment, we use a population sample size of 10 fortresses each with 15 entity class definitions, evaluate these fortresses across 10 randomly chosen seeds, and evolve the populations for 10,000 generations. We use a small number of samples for the population with significantly longer generation time to encourage the algorithm to explore more of the cells of the MAP-Elites grid. The initial population of fortresses is made up of entity class definitions with only a single uniformly random action node. We use two combinations of behavior characteristics in this paper to explore the QD space of the evolved fortresses as well as the emergent behaviors from the elite fortresses saved.

4 Results

Figure 7a shows the heatmap for the MAP-Elites grid with the fitness for this experiment measuring the percentage of the entity class FSM visited. Nearly every possible MAP-Elites grid cell is filled for this experiment. Fortresses with fewer total nodes had the highest fitness values in the grid—most likely because it becomes increasingly challenging to explore different nodes in larger graphs. Predictably, many cells that contained fortresses with high instance numbers were terminated from overpopulation. The number of total entity instances does not seem to be limited by the total number of nodes, with fortress environments that lead to either extremes of near extinction (0-1 instances) or overpopulation (156 instances.)

Measuring entropy Given this archive of individuals—optimized to maximize FSM exploration while diversifying along number of surviving entity instances and number of total FSM nodes—we investigate the distribution of total FSM nodes among entity classes. More specifically, we define a handful of FSM size buckets, and measure the entropy of the distribution of entity class FSMs among these buckets (Figure 7b). The entropy of the FSM size distribution is necessarily minimal at the extremes along the axis measuring number of nodes: only a set of minimum/maximum size FSMs can sum to a minimum/maximum number of total FSM nodes, where in either case, all FSMs must have the same size (falling into to the same FSM size bucket and minimizing distributional entropy).

Higher entropy FSM size distributions appear in the subsection of the axis containing fortresses with medium-high numbers of nodes. It is possible that more variably-sized sets of FSMs could be beneficial to maximizing fitness (in terms of FSM exploration during simulation), which might explain

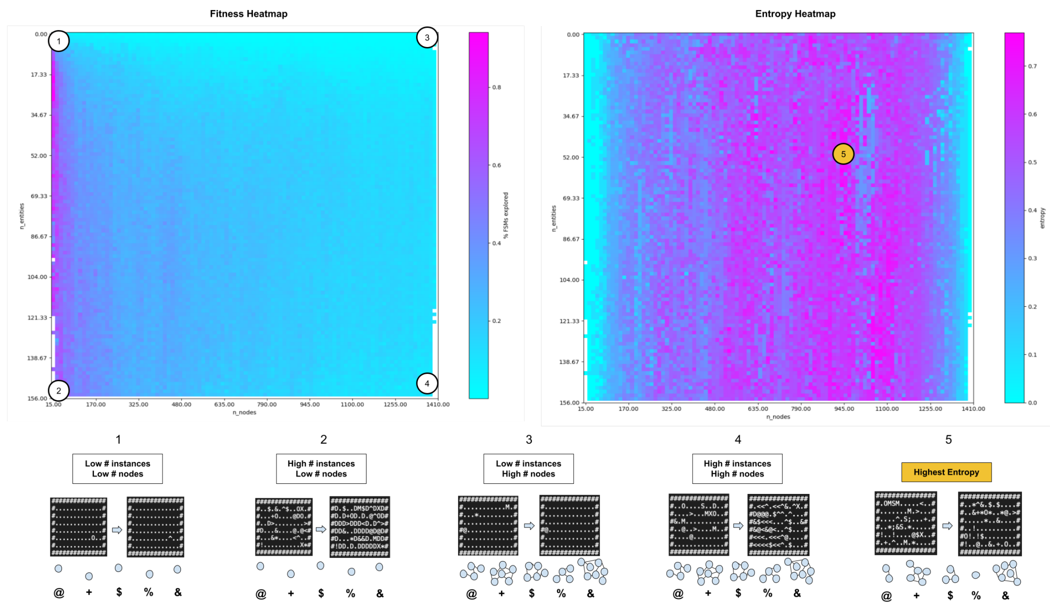


Figure 4: Results of the AF-QD experiment: exploring **number of instances** at the end of the simulation vs. **total number of nodes**. The graph on the left shows the heatmap with respect to the fitness measurement—the proportion of FSMs explored in the fortresses. Naturally, FSMs with fewer nodes (left) are more easily explored. The graph on the right shows the same archive of fortresses measured with the heatmap showing the measure of FSM entropy sizes.

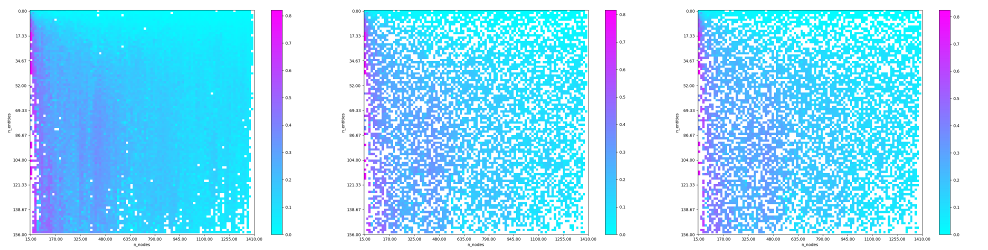
their prominence in this section of the archive. This prominence is striking given the evenly-weighted multinomial distribution used to distribute FSM nodes among entity classes from a fixed number of aggregate number of FSM nodes, which normally lead to low-entropy FSM size distributions (because each FSM is likely to be assigned roughly equal numbers of nodes). On the other hand, fitness in this section of the archive is low, such that it may be unlikely very much such selection pressure has been applied, making further analysis necessary to come to firm conclusions along these lines.

Because our domain is stochastic—in particular, re-simulating the same fortress (with the same entity class FSMs and initial entity instances and starting positions) will result in different random movement actions from any agent in a 'move', 'push', or 'chase' state—we re-evaluate the the fortresses in the archive using new random seeds and re-insert fortresses into a fresh archive. The results of these re-evaluations (for a single trial) are visualized against the archive resulting from QD search in Figure 5. In Table 3, we repeat the re-evaluation process for 10 archives, each generated by a separate QD search. We then consider the aggregate archive of overall best elites before and after re-evaluation.

5 Discussion

Our results show promise in generating a diverse set of environments for learning agents. Our fitness function emphasizes environments in which entities exhibit a diversity of behaviors (i.e. explore as many nodes in their constituent FSMs as possible). By combining this objective with behavior characteristics measuring the overall size of FSMs (via number of nodes), we seek to generate a set of environments that may act as a curriculum for a future embodied learning agent, which would have to navigate and perhaps (indirectly) model the varyingly complex behavioral policies governing the activity of NPCs in the fortress.

After the QD search process, we additionally evaluate the diversity of the entity classes within each fortress, measured as the entropy over the distribution of entity class FSM sizes. We note that high entropy—exhibited in a large swatch of the network with a medium-high number of nodes—corresponds to sets of entities with variably sizes FSMs. When such individuals are fit (and FSMs



(a) Original archive after QD search.

(b) Archive after re-evaluation with new random seeds.

(c) Archive after re-evaluation with new random seeds and longer episodes.

Figure 5: Using the archive from a single trial of QD search (left) (with n. entities and n. nodes as BCs), we observe that **re-evaluating on new seeds** (middle) and with **longer episodes** (right) leads to increasingly "holey" archives, due to random variation in the number of surviving entities after each episode.

have few ineffective nodes/edges), we can guarantee that different types of entities will exhibit diverse behavior. In this case, a hypothetical learning agent will be forced to adapt to a diversity of behavior profiles, increasing the richness of its task.

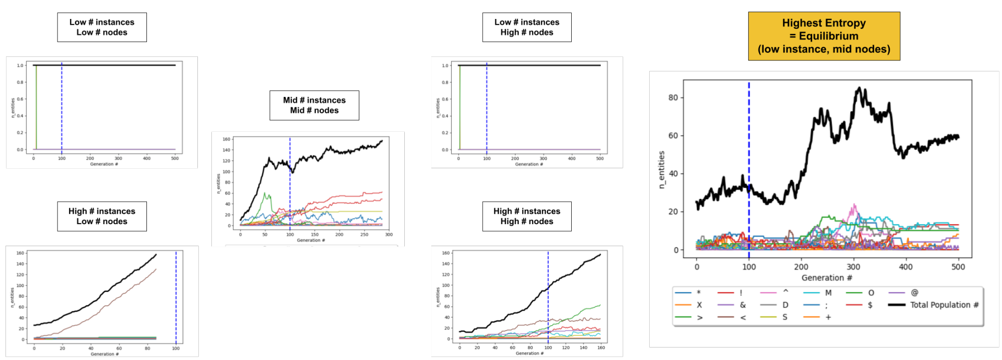


Figure 6: **Intra-fortress population**. For select fortresses in the archive, we measure the number of instances over time, across entity types. The vertical dashed blue line is the number of iterations the fortress was simulated for during evolution. The cell with the highest entropy of entity class definitions achieves an ecological equilibrium. This behavior is distinct from the extremes of the archive, which either overpopulate or rarely add any additional entities.

We observed an interesting phenomena within the MAP-Elites cells concerning the population numbers of each of the entities. Ideally, we were aiming to find fortresses with balanced interactions between entities. The cells found at the extreme points of the archive (i.e. cells with lowest and highest possible behavior characteristics) exhibited uncooperative behavior between the instances. Fortresses with lower instance numbers refused to populate and caused a stagnation in the fortress. Conversely, fortresses with higher instance numbers quickly overpopulated. However, fortress individuals found in the middle of the archive had more "equilibrium" and cooperation. The fortress in the exact middle of the archive achieved much more diversity in terms of the entity class population; some entity classes having a sudden growth in instance number before dying off, while others slowly expanded their presence over time. This "equilibrium" was most noticeable in the fortress individual that demonstrated the highest entropy between entity class FSM sizes. This fortress showed a near perfect balance between all entity classes; neither dominating nor diminishing in numbers. The entities found in this fortress find a "harmony" of co-existence where the ecosystem does not find itself in danger of overpopulation nor extinction. From this, we can conclude that having a diversity of entity class sizes leads to better balance of entity populations and allows for more exploration of co-operative class behaviors.

The main weakness of our results is the generally low fitness, which indicates that much of the larger FSMs generated by our system could be pruned to drastically smaller size without having any effect on environment dynamics. We hypothesize that this lack of FSM exploration is the result of limited compute resources. In particular, 100 steps of simulation is not likely enough to explore FSMs with up to 94 nodes. Or, the small map size of generated environments may make prohibit more interesting large-scale dynamics. We observe the QD score to still be rising steadily after 10k iterations, such that further evolution would be beneficial. Qualitatively, we see that certain fortresses in the archive maintain varying equilibria between entity types over long time horizons, potentially showing how to optimize for environments facilitating novel dynamics for lifelong learning agents.

6 Future work

From the engineering side, the fortress engine could likely be drastically accelerated if it was implemented in a batched, GPU-compatible manner, similar to the recent trend in RL environments which has allowed for orders of magnitude increases in simulation speed [Lange, 2023, Freeman et al., 2021].

For example, one archive dimension could measure how many times the “take” node is enacted by entities could encourage the evolution of fortresses with more or less aggressive entities. We would also like to examine the compressibility (e.g. via a simple gzip algorithm) or predictability (e.g. by a a neural network trained with supervised learning) of environment rollouts generated by a given fortress definition. We expect that such measures will provide a reasonable estimate of a hypothetical learning agent’s ability to model and/or adapt its behavior to a given fortress [Gomez et al., 2009], and could thus be used to validate the effectiveness of our FSM-based complexity metrics (themselves being cheaper to compute) and/or supplant them (if necessary).

A different line of future work—emphasizing the QD-AF paradigm as a design tool in its own right—will involve developing a mixed-initiative online system in which users are free to design their own fortresses and entity class definitions. The MAP-Elites fortress illumination process could create “casts” of generated characters for the user to include, acting as a recommendation engine. These generated entity classes would be selected to highlight and enhance the potential behaviors singular “main character” entity within the fortress (e.g. by leading to particular activity in the main character’s FSM). Future systems could then augment QD-AF with learned models of human preference and style gathered consensually via such an interface. Users could even be invited to narrativize the emergent dynamics of fortresses in natural language, opening the door for training models converting human narrative and first-hand experience into environments—with real stakes and incentives beyond next-token prediction—for learning agents.

7 Conclusions

We utilize QD methods to create an archive of diverse grid-world environments using the mechanics of Amorphous Fortress, with an eye toward generating diverse training sets for learning agents. By searching for diverse fortresses in terms of number of surviving entities at the end of a simulation, we guarantee that a hypothetical learning agent will be exposed to a variety of environment states. In the archives generated by QD search, we find a large swath of environments which avoid extinction or population explosion to maintain equilibria that appear robust to stochasticity and longer episode lengths. We select for fortresses with well-explored FSMs to prohibit the growth of ineffective FSM components. By diversifying the aggregate size of entity FSMs within an individual fortress, we seek to provide a set of environments containing a smooth increase in the complexity of agent behavior profiles. Since the complexity of activity within agent FSMs in this work is limited (we suspect) by scale and compute budget, future versions will seek to batch simulation, allowing for faster evolution on equal hardware. The proxies for complexity proposed here can be compared against predictability of rollouts by supervised models, or learnability for embodied deep RL agents, embodying species’ with pre-defined predator/prey dynamics between FSM-based agents to provide reward.

References

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

Christopher Bamford. Griddly: A platform for ai research in games. *Software Impacts*, 8:100066, 2021.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Mark A Bedau, John S McCaskill, Norman H Packard, Steen Rasmussen, Chris Adami, David G Green, Takashi Ikegami, Kunihiko Kaneko, and Thomas S Ray. Open problems in artificial life. *Artificial life*, 6(4):363–376, 2000.

Michael Beukman, Christopher W Cleghorn, and Steven James. Procedural content generation using neuroevolution and novelty search for diverse video game levels. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1028–1037, 2022.

M Charity, Dipika Rajesh, Rachel Ombok, and Lisa B Soros. Say "sul sul!" to simsim, a sims-inspired platform for sandbox game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 182–188, 2020.

M Charity, Dipika Rajesh, Sam Earle, and Julian Togelius. Amorphous fortress: Observing emergent behavior in multi-agent fsms. *arXiv preprint arXiv:2306.13169*, 2023.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Manoj Deshpande and Brian Magerko. Drawcto: A multi-agent co-creative ai for collaborative non-representational art. 2021.

Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx. Novelty search: a theoretical perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 99–106, 2019.

Sam Earle. Using fractal neural networks to play simcity 1 and conway's game of life at variable scales. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2020.

Sam Earle, Julian Togelius, and Lisa B Soros. Video games as a testbed for open-ended phenomena. In *2021 IEEE Conference on Games (CoG)*, pages 1–9. IEEE, 2021.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax-a differentiable physics engine for large scale rigid body simulation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Faustino J Gomez, Julian Togelius, and Juergen Schmidhuber. Measuring and optimizing behavioral complexity for evolutionary reinforcement learning. In *International Conference on Artificial Neural Networks*, pages 765–774. Springer, 2009.

Djordje Grbic, Rasmus Berg Palm, Elias Najarro, Claire Glanois, and Sebastian Risi. Evocraft: A new challenge for open-endedness. In *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*, pages 325–340. Springer, 2021.

Michael Cerny Green, Victoria Yen, Sam Earle, Dipika Rajesh, Maria Edwards, and Lisa B Soros. Exploring open-ended gameplay features with micro rollercoaster tycoon. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2021.

Cristina Guerrero-Romero and Diego Perez-Liebana. Map-elites to generate a team of agents that elicits diverse automated gameplay. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021.

Nicholas Guttenberg and LB Soros. Designing emergence in games. In *ALIFE 2023: Ghost in the Machine: Proceedings of the 2023 Artificial Life Conference*. MIT Press, 2023.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

Robert Tjarko Lange. evosax: Jax-based evolution strategies. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 659–662, 2023.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Alexandre Santos Melotti and Carlos Henrique Valerio de Moraes. Evolving roguelike dungeons with deluged novelty search local competition. *IEEE Transactions on Games*, 11(2):173–182, 2018.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Emma Stensby Norstein, Kai Olav Ellefsen, and Kyrre Glette. Open-ended search for environments and adapted agents using map-elites. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 651–666. Springer, 2022.

P Russel Norvig and S Artificial Intelligence. A modern approach. *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems*, 90: 33–48, 2002.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022.

Thomas Pierrot and Arthur Flajolet. Evolving populations of diverse rl agents with map-elites. *arXiv preprint arXiv:2303.12803*, 2023.

Mikayel Samvelyan, Akbir Khan, Michael Dennis, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Roberta Raileanu, and Tim Rocktäschel. Maestro: Open-ended environment design for multi-agent reinforcement learning. *arXiv preprint arXiv:2303.03376*, 2023.

Kenneth O Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge you've never heard of. *While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself*, 2017.

Joseph Suarez, Yilun Du, Clare Zhu, Igor Mordatch, and Phillip Isola. The neural mmo platform for massively multiagent research. *arXiv preprint arXiv:2110.07594*, 2021.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeff Clune, and Kenneth O Stanley. Enhanced poet: open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9940–9951, 2020.

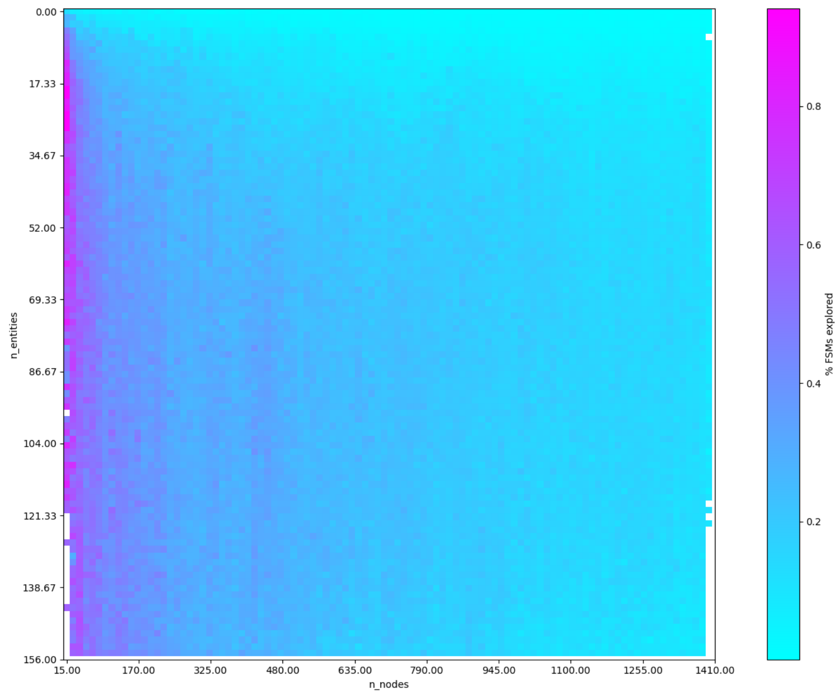
Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. Omni: Open-endedness via models of human notions of interestingness. *arXiv preprint arXiv:2306.01711*, 2023.

Table 1: Entity FSM action node definitions

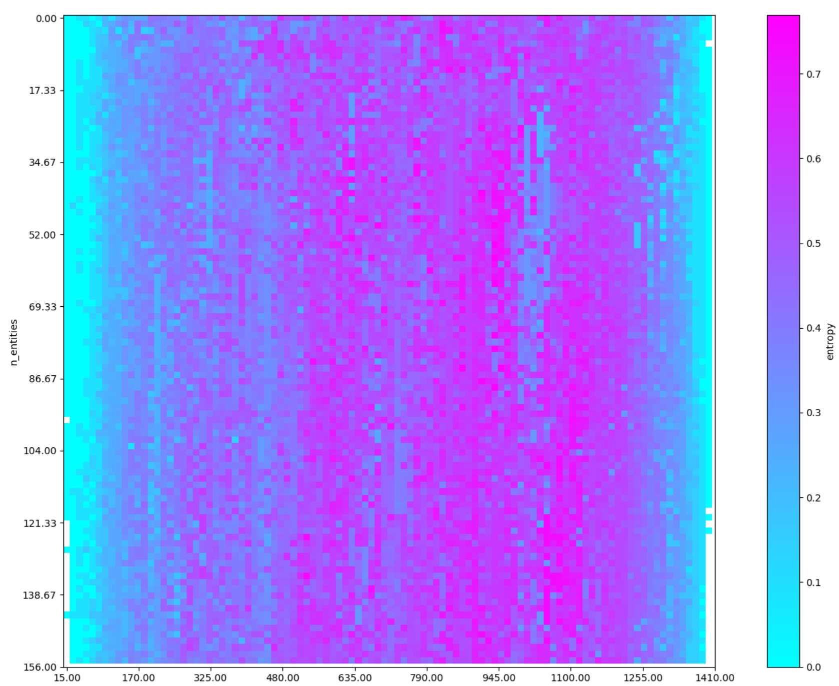
Action Node	Definition
idle	*the entity remains stationary atfferently-sizedsame position*
move	*the item moves in a random direction (north, south, east, or west)*
die	*the entity is deleted from the fortress*
clone	*the entity creates another instance of its own class*
push (c)	*the entity will attempt to move in a random direction and will push an entity of the specified target character into the next space over (if possible)*
take (c)	*the entity removes the nearest entity of the specified target character*
chase (c)	*the entity will move towards the position of the nearest entity of the specified target character*
add (c)	*the entity creates another instance from the class of the specified target character*
transform (c)	*the entity will change classes altogether to an entirely different entity class - thus changing its FSM definition entirely*
move_wall (c)	*the entity will attempt to move in a random direction unless there is an entity of the specified class at that position - otherwise it will remain idle*

Table 2: Entity FSM conditional edge definitions (ordered by least to greatest priority)

Action Node	Definition
none	*no condition is required to transition states*
step (int)	*every x number of simulation ticks the edge is activated and the node transitions*
within (char) (int)	*checks whether the entity is within a number of spaces from an instance of another entity with the target character*
nextTo (char)	*checks whether the entity is within one space (north, south, east, or west) of another entity of the target character*
touch (char)	*checks whether the entity is in the same space as another entity of the target character*

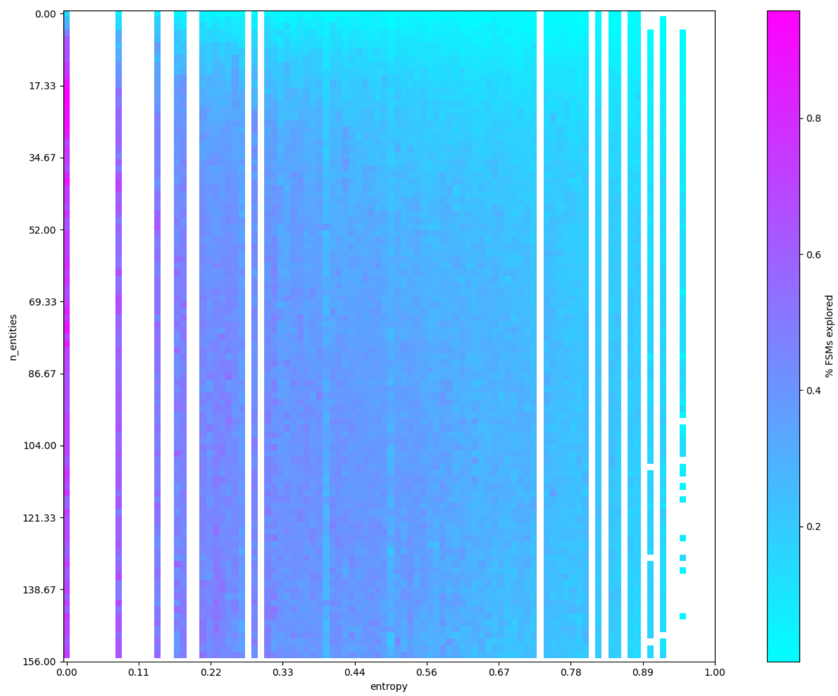


(a) Proportion of FSMs explored in fortresses. Naturally, FSMs with fewer nodes (left) are more easily explored.

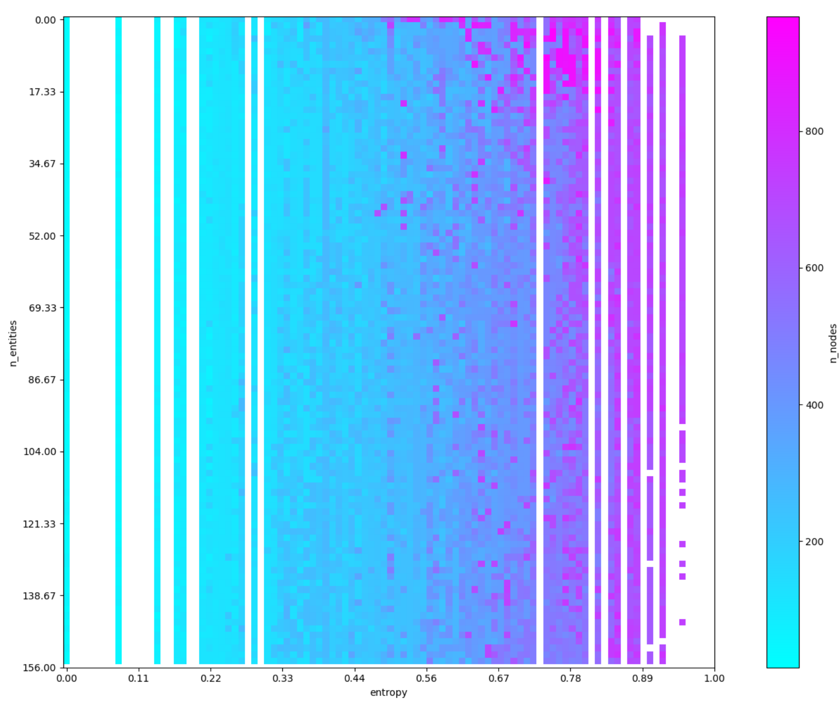


(b) Entropy of distribution of FSM sizes across entity types. Fortresses with very few/many nodes (left/right) across all FSMs must have low entry because all entity FSMs are necessarily small/large. Fortresses with a medium-high number of nodes—allowing for diverse FSM sizes between entities—exhibit high entropy. This suggests that sets of differently-sized FSMs are more likely to result in thorough FSM exploration.

Figure 7: Archive of fortresses resulting from maximizing proportion of FSMs explored while maintaining diversity in terms total size of FSMs and number of entity instances present in the fortress at the end of simulation.



(a) Proportion of FSMs explored in fortresses. Low entropy fortresses (left)—in which all entities have similar FSM size—allow for the most thorough exploration.



(b) Number of nodes over all entity types. Naturally, minimal FSMs lead to the fittest low-entropy fortresses (left), while higher entropy FSM size distributions require more nodes overall (right).

Figure 8: Archive of fortresses resulting from maximizing proportion of FSMs explored while maintaining diversity in terms of number of entity instances present in the fortress at the end of simulation, and entropy of the distribution of FSM sizes across entity types.

Algorithm 1: Mutation function for the Fortress

```
Input: node_prob, edge_prob, instance_prob
node_r = random();
edge_r = random();
instance_r = random();
/* Mutate random entity class nodes */
while node_r < node_prob do
    i = random(0,2);
    e = random(fortress.ent_def);
    n = random(log_f()) if i == 0 then
        fortress._delete_nodes(e, n);
    else if i == 1 then
        fortress._add_nodes(e, n);
    else if i == 2 then
        fortress._alter_nodes(e, n);
    node_r = random();
/* Mutate random entity class edges */
while edge_r < edge_prob do
    i = random(0,2);
    e = random(fortress.ent_def);
    if i == 0 then
        fortress._delete_edge(e);
    else if i == 1 then
        fortress._add_edge(e);
    else if i == 2 then
        fortress._alter_edge(e);
    edge_r = random();
/* Mutate random entity instances in the fortress */
while instance_r < instance_prob do
    i = random(0,1);
    e = random(fortress.entities);
    if i == 0 then
        fortress._remove_entity(e);
    else if i == 1 then
        x, y = random(fortress.pos);
        fortress._add_entity(e, x, y);
    instance_r = random();
```

behavior characteristics	new seeds	n. episode steps	best score	QD score	archive size
n. entities, n. nodes	no	100	0.941	2,235	9,986
	yes	100	0.941	2,197	9,975
		500	0.941	2,086	9,962
n. entities, FSM size entropy	no	100	0.958	2,156	6,974
	yes	100	0.958	2,005	6,951
		500	0.958	2,011	6,950

Table 3: **Re-evaluation of elites with new random seeds and longer episodes.** After aggregating (re-evaluated) elites from 10 trials, we see that the stochastic nature of our environment leads to some variance, with some shrinking of the archive and decrease in QD score.