Topological Analysis of Open-Endedness in Video Games

L. B. Soros[1,2,†], Nicholas Guttenberg[2,†]

[1]Barnard College, New York, NY, USA
[2]Cross Labs, Cross Compass Ltd., Tokyo, Japan

Abstract
This paper presents an automated methodology for analyzing specific properties of open-endedness in video games. We focus specifically on the phenomenon of door-opening states that expand a possibility space once they are encountered. Our analysis examines door-opening states both in terms of the physical properties of game levels and the cognitive abilities that a player must learn in order to progress through the game. Finally, we demonstrate our methodology on the NES homebrew game Blobquest.

Keywords
Video games, Open-endedness

1. Introduction

Juul (2002) divides games into two categories: games of emergence and games of progression. Games of emergence are games in which simple rules combine, leading to variation. Games of progression are games in which the player must perform a series of actions in order to complete the game. Some games can combine features of both emergence and progression – Juul cites Everquest as an example of such a game because of its combination of Dungeons-and-Dragons-style rules and scripted objective-based quests.

This paper is about games of progression examined through the lens of open-endedness in the field of artificial life. We focus in particular on a mechanism called a *door-opening state*, which expands a possibility space once encountered. The main contribution of this paper is an automated methodology for analyzing door-opening states in video games. The results of such analysis can reveal whether sequences of door-opening states in games are linear (suggesting a simple and straightforward kind of progression) or branching (suggesting more open-ended gameplay).

In this paper we will examine properties of open-endedness in video games by exploring the phenomenon of door-opening states in the video game BlobQuest (Fig. 1). We focus in particular on door-opening *skills*, meaning new skills become possible for the agent that weren't previously possible when some other requisite ability is acquired.

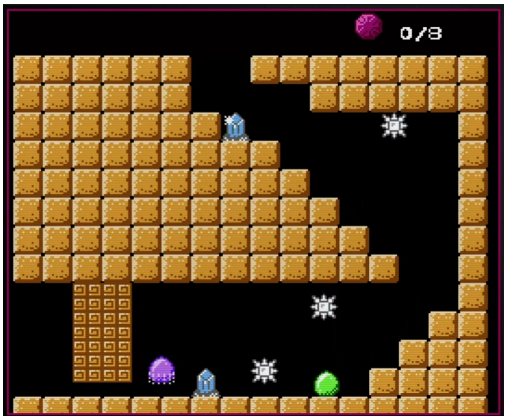

Figure 1: BlobQuest. The player (green blob) must collect rubies while avoiding obstacles such as the spiky balls. The player can only move left and right until acquiring power-ups. Here, collecting the purple power-up at bottom left will give the player the ability to jump, which increases the number of areas that are accessible to the player.

2. Background

According to Cook (2019) "the possibility space of a particular type of content (like a Minecraft world) is the set of all examples of that content we can imagine or describe". Possibility spaces are therefore a useful tool for thinking about creative processes in both artificial and natural systems. Taylor (2019) notes that the concept of possibility spaces is widely employed in particular when describing the products of evolutionary systems [4, 5, 6] and presents a formalism describing how possibility spaces are explored and expanded in such systems, specifically those that are more or less open-ended. In particular, Taylor notes the existence of *door-opening states* [7], the discovery of which "open up an expanded space of new

The Joint Workshop Proceedings of the 2022 Conference on Artificial Intelligence and Interactive Digital Entertainment
[†]These authors contributed equally.
lsoros@barnard.edu (L. B. Soros); ngutten@gmail.com (N. Guttenberg)

adjacencies", referring perhaps to Kauffman's (2003) theory of an *adjacent possible* that is constantly expanding and increasing the diversity of what can happen next.

Though Taylor explores these concepts in the context of open-ended evolution, there is nothing in particular that restricts their use to discussions of evolution; we claim herein that they are useful concepts for discussing open-ended systems in general, including video games. It has been established by Bogost (2008) that "In a video game, the possibility space refers to the myriad configurations the player might construct to see the ways the processes inscribed in the system work". In this paper, we explore the concept of door-opening states in the physical and cognitive spaces explored during video game play. Our eventual goal is to explore how open-ended various video games are and to create a sort of tool that can be used by designers for measuring the open-endedness of their games.

3. Related Work

The problem of identifying bottleneck states *in general* is an important problem in many research areas, and has been a focus of reinforcement learning (RL) research in particular. In the RL context, bottlenecks in an agent's observation space, for instance, might correspond to subgoals [10] or plan components [11]. However, in this section we review only works pertaining to automated bottleneck or state detection for for video games.

In *Automatic Mapping of NES Games with Mappy* [12], linked maps of rooms in NES games are produced from input playthroughs of games. These maps are constructed by viewing the tiles on the screen as the game is being (re-)played and stitching the tiles together to form a cohesive map. To determine what is being shown on screen, Mappy analyzes the data stored in the NES Picture Processing Unit, which contains data about tilemaps, sprites, and other game features that are drawn on the screen. To determine whether the player has reached a transition between two rooms, Mappy takes into account whether the screen is scrolling and whether or not the player has control of the game. The subsequent MappyLand project [13] resulted in performance and quality of life improvements.

While Mappy and MappyLand are in many ways similar to the work introduced in this paper, there are significant differences. The most important difference is that Mappy constructs a graph of the true physical game map, while our method not only creates a graph of physical states but also a map of cognitive/mental/ability bottlenecks in agent play, as explained in the next section.

4. Methodology

One of the hypotheses in this work is that door-opening states correspond to bottlenecks in the game state, meaning there is some state that necessarily must be passed through in order to access successive states. It is important to note that there may be bottlenecks in the physical game state and also bottlenecks in terms of a gameplaying agent's cognitive abilities; if there were a power-up required to do something that would be a physical bottleneck, but not knowing how to do something is a mental bottleneck. Additionally, some (but not all) bottlenecks may be both physical and cognitive in nature. Figure 2 shows an example showing the difference between these two kinds of bottlenecks.

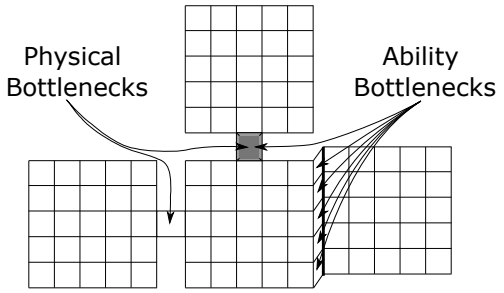

Figure 2: Toy example of physical and ability/cognitive bottlenecks. An imagined player starts in the middle room and must overcome physical and cognitive bottlenecks in order to leave the room. The passage to the left only presents a physical bottleneck because it doesn't require any special knowledge besides knowing how to walk to the left. The passage to the right contains a wall that must be climbed over, thus presenting a purely cognitive bottleneck (because the player must learn or figure out how to climb). The upward passage contains a pit that must be jumped over, presenting both a physical and cognitive bottleneck.

The goal of our analysis is to automatically construct a graph of physical and then mental door-opening states from human gameplay. In this paper we present two methods for automatically constructing the physical graphs. The initial step for both methods are the same. First, to form the graph of physical states, a human plays the game and the sequences of actions they take and the corresponding changes in game state are recorded. The methods then diverge.

4.1. Graph of physical states - Method 1

The first method to map out the physical states of the game is to use the RAM states directly[1]. We encode the RAM by looking at every 4-bit nibble, which corresponds

[1]Source code is available at tinyurl.com/39hcsp49

to the way many variables are stored within the game. Each nibble can take one of 16 values, which we one-hot encode. We remove the lowest-entropy nibbles across our set of playthroughs, retaining 941 features as a result. We then apply k-means clustering to obtain 1600 cluster centers, which are used to form the nodes of our game graph. The edges are made by observing all transitions between nodes that happened during our playthroughs.

4.2. Graph of physical states - Method 2

Using raw RAM to form the metric space for clustering has a number of problems. There are certain highly important bits (whether particular power-ups have been collected) which should determine whether a link can or cannot be traversed, but these features contribute only a small amount to the distance relative to things like full screen re-arrangements or the passive movement cycles of hazards.

To counteract this problem, we train a neural network to take as input pairs of RAM states that occur within the same playthrough, and give as output the probability distribution over their temporal distance in number of actions[2]. The penultimate 256d layer is used as an embedding vector, after which we again perform k-means clustering.

4.3. Mental link learning

The existence of a physical state bottleneck through which play must pass to reach some new set of states appears to be a kind of door-opening state, but it does not conform well to the idea of discovering 'ways to do things' that we would see in a more evolutionary open-endedness context. That is to say, there's a difference between something that attains a specific physical state in the world that creates new opportunities, and something that learns what is necessary to reliably attain that state. We would like to see if we can use our physical state graph in conjunction with a learning process in order to detect which physical bottlenecks might correspond also to mental bottlenecks, or to see if there may even be mental bottlenecks with no localized physical realization.

To do this, we make use of the idea of a world model [14], which predicts subsequent RAM states given current RAM state and action. We would like to use this as a proxy for discovery of 'how to' do something — the how-to comes from the human playthroughs, but we require the model to accurately predict the consequences of all actions taken along those trajectories. From this requirement and the current state of the model, we obtain a set of game states that are accessible from the starting state. The transitions starting from these accessible states are used as the data set for the next step of training the world model.

We can then look at the dynamics of how the set of accessible states changes as the model is trained, and see whether there are plateaus in training (difficult links or sets of links) that would correspond to bottlenecks. We can also see whether what is being learned is local to the accessible states, or if the model learns to predict unseen links elsewhere in the game. Finally, it can be interesting to compare multiple training runs to see if certain sets of links are consistently learned or fail to be learned in a correlated manner, representing something like an eigenskill.

This analysis is very dependent on the learning method and architecture of the world model, and so it can at best illuminate things which may correspond to mental bottlenecks (unlike a physical state analysis in which there is some ground truth about whether something actually is or isn't a bottleneck).

5. Case Study: BlobQuest

BlobQuest is an NES homebrew game created by Tom Livak[3]. It is essentially a metroidvania game wherein the player must collect rubies (which will allow the player to return to human form from blob form) while avoiding obstacles and navigating platforming levels. Importantly, the player can collect power-ups that imbue the blob with new abilities, enabling the exploration of new parts of game levels.

Figure 3 shows a plot of physical game states generated via Method 1, and Figure 4 shows physical game states generated via Method 2. Viewing screenshots allows us to verify hypotheses about what certain topological features in the graph correspond to. For instance, short cycles (i.e. two nodes connected via bidirectional links) might correspond to quick instances of backtracking. Alternatively, a more likely explanation is that certain game features such as an enemy moving back and forth on the screen (while the player might even be standing still) can cause repetitive changes in RAM, resulting in small cycles on the graph. Longer paths likely correspond to relatively longer play sequences that don't involve as much backtracking. Nodes with high degree might correspond to hubs in the game, where the available paths are determined by what power-ups the player possesses.

Figure 5 shows the mental link learning corresponding to Figure 3 (physical graph Method 1), and Figure 6 shows the mental link learning corresponding to Figure 4 (physical graph Method 2).

[2]Source code is available at tinyurl.com/5f2ptmv2

[3]BlobQuest is used with permission from Tom Livak.

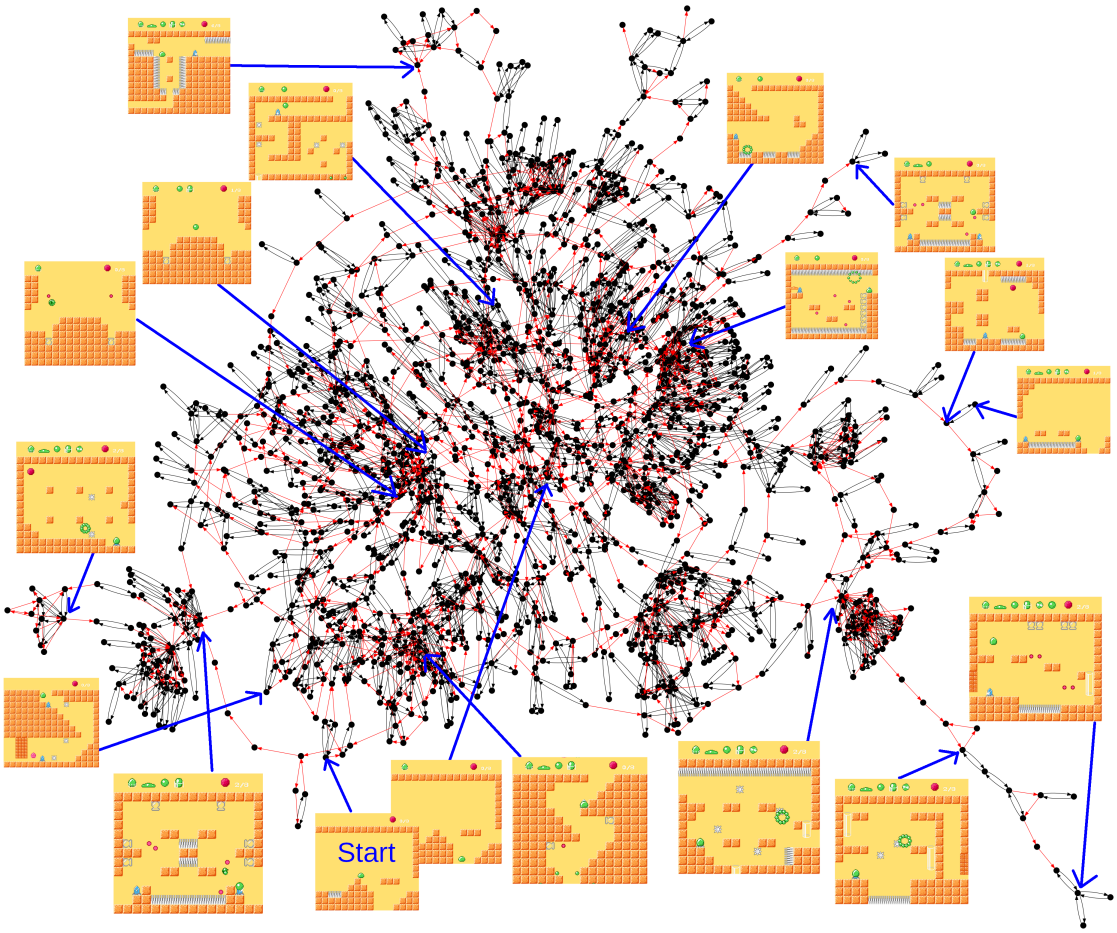

Figure 3: Graph of physical game states in BlobQuest, Method 1, 1600 nodes. The topology of the graph is determined via human gameplay; actions are recorded along with corresponding changes in RAM and then k-means clustering is used to group the game states into clusters. Links are formed between nodes when a player action leads from one game state to another. Red links are unidirectional (indicating no backtracking, or irreversibility) and black links are bidirectional.

6. Discussion

Successful video game play is a function of both an agent (the player) and its environment (the game). Static analysis of just the game levels (such as in VGLC [15]) does not afford us what we want to examine because of the complex relationship between environment and agent, which form a coupled system. For instance, tall pipes in the NES game Super Mario Bros. only present a bottleneck because Mario must figure out how to jump high (a cognitive skill) in order to overcome them. The difficulty of tall pipes is not inherent to the pipes themselves, but is relative to Mario's embodiment and abilities.

This paper presented a case study of an initial foray into automatically analyzing the open-endedness of simple video games. It is important to note that the current method is sensitive to a number of parameters and design decisions. For example, changing the number of clusters created by the clustering algorithm will affect the density of the generated graph, with fewer clusters increasing legibility (because the graph itself is less cluttered) but decreasing the degree to which links map to human-interpretable changes in play.

The eventual goal of this work is to fully automate the game analysis process instead of relying on human gameplay to generate graphs of physical states. Preliminary research explored the possibility of using a modified Monte Carlo tree search to play the game, and while this approach worked for some games, the modified MCTS agent was not able to play BlobQuest well. The high amount of backtracking needed to successfully play Blobquest may explain this outcome, and games with less backtracking

Figure 4: Graph of physical game states in BlobQuest, Method 2, 1600 nodes.

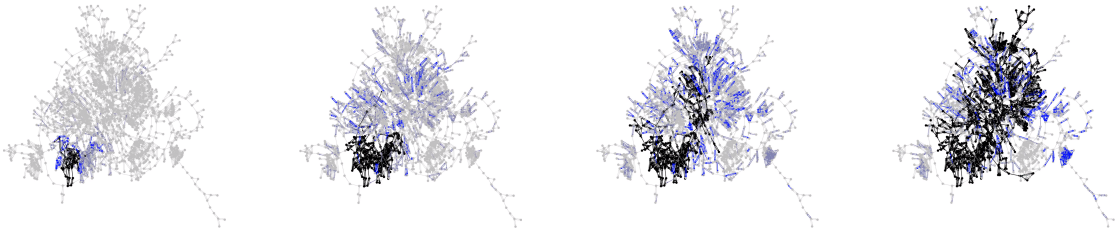

Figure 5: Progression of link learning for the 1600-node graph in Figure 3. Black links are links that are correctly predicted and connected to some state that is accessible to already-predicted states. Blue links are links that are correctly predicted but not connected to an accessible state (showing generalization).

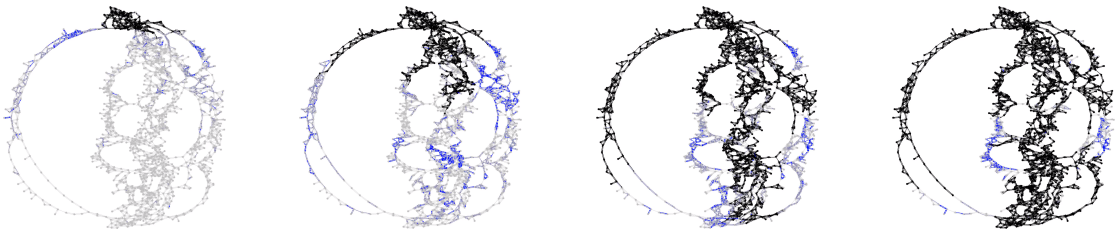

Figure 6: Progression of link learning for the 1600-node graph in Figure 4. As above, black links are links that are correctly predicted and connected to some state that is accessible to already-predicted states and blue links are links that are correctly predicted but not connected to an accessible state.

may be possible to analyze fully automatically.

An open question is whether there exist door-opening states that don't correspond to bottlenecks or vice versa. Additionally, future work will explore different representations for game topology. For example, one possibility might be to represent the game states as a manifold rather than a graph. Other future work could explore more formally how cognitive bottlenecks are formed and detected.

References

[1] J. Juul, The open and the closed: Games of emergence and games of progression., in: Computer Games and Digital Cultures Conference Proceedings, Tampere University Press, 2002.

[2] M. Cook, Tutorial: Generative & possibility space, https://www.possibilityspace.org/tutorial-generative-possibility-space/, 2019. Accessed: 08-03-2022.

[3] T. Taylor, Evolutionary Innovations and Where to Find Them: Routes to Open-Ended Evolution in Natural and Artificial Systems, Artificial Life 25 (2019) 207–224. URL: https://doi.org/10.1162/artl_a_00290. doi:10.1162/artl_a_00290. arXiv:https://direct.mit.edu/artl/article-pdf/25/2/207/1896728/artl_a_00290.pdf.

[4] H. P. de Vladar, M. Santos, E. Szathmáry, Grand views of evolution, Trends in Ecology & Evolution 32 (2017) 324–334.

[5] M. A. Boden, Creativity and alife, Artificial Life 21 (2015) 354–365.

[6] W. Banzhaf, B. Baumgaertner, G. Beslon, R. Doursat, J. A. Foster, B. McMullin, V. V. De Melo, T. Miconi, L. Spector, S. Stepney, et al., Defining and simulating open-ended novelty: requirements, guidelines, and challenges, Theory in Biosciences 135 (2016) 131–161.

[7] T. Taylor, M. Bedau, A. Channon, D. Ackley, W. Banzhaf, G. Beslon, E. Dolson, T. Froese, S. Hickinbotham, T. Ikegami, et al., Open-ended evolution: Perspectives from the oee workshop in york, Artificial life 22 (2016) 408–423.

[8] S. A. Kauffman, The adjacent possible, https://www.edge.org/conversation/stuart_a_kauffman-the-adjacent-possible, 2003. Accessed: 08-03-2022.

[9] I. Bogost, The rhetoric of video games, in: K. Salen (Ed.), The Ecology of Games: Connecting Youth, Games, and Learning, The John D. and Catherine T. MacArthur Foundation Series on Digital Media and Learning, The MIT Press, Cambridge, MA, 2008, p. 117–140.

[10] T. D. Kulkarni, A. Saeedi, S. Gautam, S. J. Gershman, Deep successor reinforcement learning, arXiv preprint arXiv:1606.02396 (2016).

[11] M. C. Machado, M. G. Bellemare, M. Bowling, A

laplacian framework for option discovery in reinforcement learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 2295–2304.
[12] J. Osborn, A. Summerville, M. Mateas, Automatic mapping of nes games with mappy, in: Proceedings of the 12th International Conference on the Foundations of Digital Games, 2017, pp. 1–9.
[13] J. C. Osborn, A. Summerville, N. Dailey, S. Lim, Mappyland: fast, accurate mapping for console games, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 17, 2021, pp. 66–73.
[14] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, Advances in neural information processing systems 31 (2018).
[15] A. J. Summerville, S. Snodgrass, M. Mateas, S. O. n'on Villar, The vglc: The video game level corpus, Proceedings of the 7th Workshop on Procedural Content Generation (2016).