Estimates for the Branching Factors of Atari Games

Mark J. Nelson
American University
Washington, DC, USA
mnelson@american.edu

***Abstract*—The branching factor of a game is the average number of new states reachable from a given state. It is a widely used metric in AI research on board games, but less often computed or discussed for videogames. This paper provides estimates for the branching factors of 103 Atari 2600 games, as implemented in the Arcade Learning Environment (ALE). Depending on the game, ALE exposes between 3 and 18 available actions per frame of gameplay, which is an upper bound on branching factor. This paper shows, based on an enumeration of the first 1 million distinct states reachable in each game, that the average branching factor is usually much lower, in many games barely above 1. In addition to reporting the branching factors, this paper aims to clarify what constitutes a distinct state in ALE.**

***Index Terms*—branching factor, Atari, Arcade Learning Environment**

I. Introduction

Atari 2600 games have been a popular challenge domain for artificial intelligence research since the 2012 release of the Arcade Learning Environment (ALE). ALE wraps the Atari emulator Stella in a framework familiar to AI researchers: agents observe and take actions in an environment, sometimes receiving positive or negative reward as a result [1]. As of this writing, ALE supports 104 games.

Each supported game has been manually instrumented by the ALE developers. For example, ALE provides rewards to agents by reading changes in score from game-specific locations in the Atari RAM. Part of this instrumentation is the *minimal action set*, the set of actions that have any effect in the game. Currently supported games have minimal action sets as small as 3 and as large as 18. For example, in *Breakout* there are four actions: no-op (no input), left, right, and fire. The maximum 18 are: no-op, the fire button, the 8 directions that can be registered by the Atari joystick, and each of those 8 directions while also pressing the fire button.

A game's minimal action set is an upper bound on its *branching factor*, the number of new states that can be reached from a given state. Atari AI research – and other videogame AI research, for that matter – does not normally give much weight to (or even compute) branching factors. But they are commonly discussed in AI board-game playing. The difference in branching factors is cited as a reason that computer chess is harder than checkers, shogi harder than chess, and go harder than the other three [2], [3]. Multi-game engines such as Ludii also compute branching factor as an informational measure [4]. Does it give useful information for video games? A first step in investigating that question is to compute it.

A game's branching factor can be less – often significantly less – than its minimal action set, for two reasons. The first is that the state space of many games forms a graph, not a tree, so some distinct input sequences result in an identical state. For example, in *Tetris*, rotating a piece twice clockwise will often result in the same state as rotating it twice counter-clockwise. And in *Breakout*, there are many ways the player can move when the ball is in the air that will result in the same contact once the ball comes back down.

The second reason that the average branching factor can be smaller than the size of the action set is that Atari games may simply ignore some or all input at various times. For example, once the player initiates a jump in *Q*bert*, all input is ignored for a number of frames while the jump animation plays out. And in *Space Invaders*, the fire button has no effect when the player already has a laser-cannon shot in the air, since the game rules only allow one shot in the air at a time.

This paper provides average branching factor estimates for 103 of the 104 games supported by ALE.[1] These estimates are computed by exhaustively enumerating the first 1 million distinct states reachable in each game. The primary result is therefore Table II.

As a necessary prerequisite for enumerating those 1 million distinct states, this paper also clarifies what constitutes a "distinct state" in ALE, in the sense of *state* meant by game-tree search and Markov decision processes (MDPs). As discussed in Section II-A, existing literature has tended to either under- or over-specify state. In fact, a precise state for 6 of the games (those that use the Atari paddle controller) is not retrievable from the public API of ALE.

Finally, some lingering issues with determinism in ALE (a longstanding problem) are uncovered by the experiments here. They only impact the branching factor estimates of two games in a non-negligible way, but point to potential issues with reproducibility.

II. Methodology

The main results of this paper are the estimated branching factors given in Table II. This table is computed by enumerating the first 1 million distinct states in the games supported

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

[1] One is omitted because its initial state is broken; see Section II-D.

Figure 1. The two Atari input devices emulated by ALE: Joystick on the left, and paddle controller on the right. (Photographs by Evan Amos; released into the public domain.)

by ALE, and using those counts to estimate branching factor.[2] Carrying out this task requires: a definition of what constitutes a distinct *state*, a method for estimating *branching factor*, and a deterministic emulator. All three of these are surprisingly tricky to pin down.

A. Atari state

Tree search and reinforcement learning algorithms both have a concept of *state*. A state for such algorithms is enough information to uniquely determine an environment's future dynamics. Given the same state, the same action sequence will produce the same sequence of successor states if the environment is deterministic; or it will produce the same distribution over state sequences if the environment is stochastic.

For one-player Atari games, the state can be completely specified by the 128 bytes of Atari RAM, plus the current position of the paddle controller, if the game uses the paddle controller. To the best of my knowledge, this paper is the first to use this definition of an Atari game state, which I claim to be the correct one. Computing it required patching ALE, since ALE doesn't expose paddle position in the public API.[3]

Previous definitions of Atari/ALE game state either under- or overspecify the state:

1) Underspecified game state: Much existing literature assumes that the Atari RAM is sufficient to capture the game state. For example, Machado *et al.* [5] mention that ALE allows querying either the current screen image or the current RAM, and call the RAM the "real state":

> This observation can be a single 210×160 image and/or the current 1024-bit RAM state. Because a single image typically does not satisfy the Markov property, we distinguish between observations and the environment state, with the RAM data being the real state of the emulator.

The RAM is in fact sufficient for many Atari games. The Atari 2600 has no external storage, network interface, etc., so the only place it can store data across frames is in RAM. However, it also polls every frame for player input from a physical controller. And some Atari controllers – one of which is used by ALE – maintain their own external state.

ALE supports two emulated player input devices: the joystick and the paddle (Figure 1). The joystick does not maintain external state. It returns one of the 18 possible actions each frame, with no interaction between frames (at least if we assume a player with sufficiently fast hands, such as an AI bot). Therefore the Atari RAM is sufficient to capure state when using the joystick.

The paddle, though, does maintain external state. It is a rotating potentiometer – a wheel that sends different voltages to the console depending on its position. Therefore, when playing on a real Atari console, the current position of the paddle in the physical world is part of the game state. The way ALE implements the paddle is that the actions "left" and "right" increase or decrease the the paddle's current rotation by a compile-time constant, `PADDLE_DELTA`, up to specified maximum and minimum values.[4] Therefore in the case of ALE as well, the paddle's rotation is needed in addition to the Atari RAM to have a complete game state.

Of the 104 ALE-supported games, 98 use the joystick, and six use the paddle. The six are: backgammon, blackjack, breakout, casino, kaboom, and pong. The original ALE paper [1] makes a point of noting the added complexity posed by paddle controllers as an example of how ALE captures some of the messiness of real-world decision-making, but subsequent papers often ignore them.

2) Overspecified game state: ALE also has a mechanism for serializing and deserializing state. This captures the entire state of the emulator and ALE itself. But it is *too* precise to correspond to what AI algorithms normally mean by state, and is therefore also not usable for this paper.

For example, serialized ALE states include the frame number, so if an otherwise identical game state can be reached at frame 5, or through a different action sequence at frame 6, using state serialization as the representation will treat these as distinct states. That is not normal practice in AI algorithms. Mechanisms such as discounted reward might prefer reaching the same state sooner rather than later, but the time a state is reached is not part of the *definition* of a state.

B. Branching factor

The term *branching factor* is used in a several different ways, all relating to the number of successor states reachable from a given game state. It can mean the *maximum* branching factor, or the *average* branching factor. Korf [6] also distinguishes an *edge branching factor* (average number of outgoing legal moves) from a *node branching factor* (average number of new states reached through such moves). The difference amounts to treating the game's state space as a tree vs. graph: the node branching factor avoids double-counting states reachable through multiple paths.

[2]More precisely, counting how many distinct states are reachable in n frames, until the first frame where $n \geq 1,000,000$.

[3]The patched version of ALE used in the experiments reported in this paper, which also applies a determinism fix discussed in Section II-C, is available at https://github.com/NelsonAU/Arcade-Learning-Environment.

[4]Paddle constants are defined in `src/environment/ale_state.hpp`.

Table I
MISMATCHES BETWEEN BFS AND ID ESTIMATES OF BRANCHING FACTOR (TO 10K STATES)

Game	BFS	ID
pitfall2	5.04	6.48
space_war	3.85	4.49
assault	4.51	4.55
solaris	7.61	7.64
ice_hockey	1.09	1.09
frostbite	1.10	1.10
seaquest	1.05	1.05
tic_tac_toe_3d	1.09	1.09
hangman	1.13	1.13
freeway	1.08	1.08

This paper estimates average node branching factors by a state-counting method. First, exhaustively count the number of distinct states that can be reached, through any sequence of actions, by a given frame number.[5] Then use this cumulative count to estimate the branching factor as follows.

Observe that, if a game's average node branching factor were b, the cumulative number of distinct game states s observed by frame f should be approximately:

$$s \approx \sum_{i=0}^{f} b^i$$

This holds exactly if the game tree is uniform, because there will be exactly b^f new states reachable at frame f. Otherwise, it approaches equality as s grows large. We can rewrite this equation such that solving for b reduces to finding the (positive real) root of a polynomial:

$$(1 - s) + b + b^2 + \ldots + b^f = 0$$

Such problems can be solved quickly to high precision by a number of computational root-finding methods. The results in this paper use the `uniroot` method of R v4.1.0.

C. Determinism

In principle, Atari games are deterministic [5], [7]. In practice, ALE has often not been very deterministic, due to a mixture of bugs and the complexity of emulation. Bellemare notes: "ALE determinism has always been brittle at best".[6]

The patched version of ALE used in paper includes a significant patch from Jesse Farebrother that improves ALE's determinism.[7] It does solve most determinism problems, but a few games still behaved strangely in preliminary experiments.

To test the reproducibility of estimating branching factor by exhaustively counting distinct states, I compared the results of textbook versions of iterative deepening (ID) and breadth first search (BFS) on a validation run counting the first 10,000 distinct states. The two search methods agreed on 93 of 103 games. They disagreed on the 10 games shown in Table I, sorted from largest to smallest disagreement.

For understanding branching factor specifically, this table is perhaps not too concerning. Ninety-three games (those not in the table) have exact agreement between ID and BFS. Of the ten games that differ, the branching factor difference is below 0.1 in eight, and below 0.01 in six. Two do have noticeably different estimates: `pitfall2` and `space_war`.

The overall results for those two games are therefore worth taking with a grain of salt. In addition, though it seemingly does not pose a large problem for branching factor estimates specifically, the fact that iterative deepening and breadth-first searches may produce different results on a deterministic domain is somewhat worrying for the reproducibility of other analyses of the Atari state space.

D. Broken initial states

Two games, `trondead` and `klax`, have an additional issue. Both start in a kind of "dead" state: the initial RAM is not changed by any action (neither game uses the paddle controller, so RAM is sufficient for state). That would imply the games have only one total state, the initial one. This seems to be caused by the emulator not having properly initialized the RAM yet. The game `trondead` was fixed by taking one no-op action before starting the real experimment. The game `klax` seems to require more involved fix-up, so was excluded.

III. RESULTS AND DISCUSSION

This paper's main goal is to compute the branching factor estimates in Table II. These results are computed by counting the distinct states reachable in each game from the initial state, using breadth-first search, until the frame at which the count exceeds 1 million. The experiments took about 86.5 core-hours of CPU time on an Intel Xeon E5-2650 v4 @ 2.20GHz (average of about 50 core-minutes per game).

The estimated branching factor is very low in a large number of games. The median is 1.19, and 79 of 103 games (77%) are below 2.0. And in a few dozen games, the branching factors barely exceed 1.0.

Such low branching factors suggest that there may not be significant decisions to be made every frame. An open question is what specifically that means. Some existing work has proposed making decisions less often than every frame, using a hyperparameter called frame-skip [8]. The results here suggest frame-skip may be justified, but don't directly prove it. Future work might look at whether branching factor correlates with optimal choice of frame-skip. Alternately, if we first find the optimal frame-skip for a game, we might recompute effective branching factors for the slower "real" decision cycle.

A different explanation is that quick-reaction decision-making *is* sometimes needed, and so we can't simply use a more granular decision-making cycle – but only sometimes. The branching factors reported here are averages; it

[5]The state-counting code used for this paper, along with data and analysis scripts, is available in the `cog2021` branch of https://github.com/NelsonAU/ale_countstates/

[6]GitHub issue comment, January 12, 2020. https://github.com/mgbellemare/Arcade-Learning-Environment/issues/291#issuecomment-573483143

[7]https://github.com/JesseFarebro/Arcade-Learning-Environment/tree/rng

Table II
ESTIMATED BRANCHING FACTORS (TO 1MM STATES)

Game	Actions	Branching factor	Game	Actions	Branching factor
haunted_house	18	7.70	atlantis	4	1.19
solaris	18	7.40	lost_luggage	9	1.19
double_dunk	18	7.15	up_n_down	6	1.17
zaxxon	18	5.01	riverraid	18	1.16
boxing	18	3.92	chopper_command	18	1.13
assault	7	3.61	frostbite	18	1.12
video_pinball	9	3.42	mario_bros	18	1.11
yars_revenge	18	3.17	ice_hockey	18	1.11
road_runner	18	3.17	galaxian	6	1.11
laser_gates	18	3.11	venture	18	1.10
gravitar	18	3.11	krull	18	1.10
darkchambers	18	3.10	videocube	18	1.10
sir_lancelot	6	2.98	hero	18	1.10
trondead	18	2.87	battle_zone	18	1.09
space_war	18	2.86	surround	5	1.08
enduro	9	2.66	seaquest	18	1.08
word_zapper	18	2.66	kaboom	4	1.08
pitfall2	18	2.63	hangman	18	1.07
jamesbond	18	2.60	othello	10	1.07
asteroids	14	2.42	pacman	5	1.07
robotank	18	2.32	space_invaders	6	1.07
private_eye	18	2.32	flag_capture	18	1.07
tennis	18	2.18	tetris	5	1.07
centipede	18	2.14	defender	18	1.06
earthworld	18	1.95	pong	6	1.06
bank_heist	18	1.88	videochess	10	1.06
basic_math	6	1.88	time_pilot	10	1.06
air_raid	6	1.81	et	18	1.05
backgammon	3	1.61	video_checkers	5	1.04
adventure	18	1.59	turmoil	12	1.04
crossbow	18	1.59	bowling	6	1.04
entombed	18	1.58	wizard_of_wor	10	1.04
skiing	3	1.58	kangaroo	18	1.04
berzerk	18	1.54	star_gunner	18	1.04
phoenix	8	1.49	beam_rider	9	1.04
asterix	9	1.48	pooyan	6	1.03
montezuma_revenge	18	1.45	amidar	10	1.03
donkey_kong	18	1.41	mr_do	10	1.03
carnival	6	1.41	ms_pacman	9	1.03
pitfall	18	1.39	kung_fu_master	14	1.03
miniature_golf	18	1.39	qbert	6	1.03
journey_escape	16	1.38	crazy_climber	9	1.03
human_cannonball	18	1.35	name_this_game	6	1.03
elevator_action	18	1.35	frogger	5	1.02
breakout	4	1.30	casino	4	1.02
demon_attack	6	1.30	blackjack	4	1.02
alien	18	1.25	freeway	3	1.02
tutankham	8	1.24	king_kong	6	1.01
fishing_derby	18	1.24	koolaid	9	1.01
keystone_kapers	14	1.22	gopher	8	1.01
superman	18	1.20	tic_tac_toe_3d	10	1.01
atlantis2	4	1.19			

is likely that some games have significantly more frame-to-frame variation in the branching factor than others. Future work might look at additional ways of summarizing the state space. Average branching factor is a single-number summary of the rate of growth of a game's state space. Other measures might look at variability or asymmetry of growth.

Finally, the relationship between branching factor and difficulty is open. In board game AI, higher branching factor is often taken to mean a more complex game (at least for AI!). Is this true for videogames? My intention is to put forward some numbers as a way of starting a discussion on that question.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under Grant IIS-1948017. Computing resources were provided by the American University High Performance Computing System, which was funded in part by the National Science Foundation under Grant BCS-1039497.

Thanks to Amy Hoover, David Dunleavy, and the anonymous reviewers for helpful suggestions.

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[2] J. Schaeffer, "Checkers: A preview of what will happen in chess?" *ICCA Journal*, vol. 14, no. 2, pp. 71–78, 1991.

[3] H. Matsubara, H. Iida, and R. Grimbergen, "Natural developments in game research: From chess to shogi to go," *ICCA Journal*, vol. 19, no. 2, pp. 103–112, 1996.

[4] D. J. N. J. Soemers, E. Piette, M. Stephenson, and C. Browne, *Ludii User Guide*, version 1.1.13, 2021. [Online]. Available: https : / / ludii . games / downloads / LudiiUserGuide.pdf.

[5] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *Journal of Artificial Intelligence Research*, vol. 61, pp. 523–562, 2018.

[6] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97–109, 1985.

[7] M. J. Hausknecht and P. Stone, "The impact of determinism on learning Atari 2600 games," in *Proc. AAAI Workshop on Learning for General Competency in Video Games*, 2015, pp. 19–20.

[8] A. Braylan, M. Hollenbeck, E. Meyerson, and R. Miikkulainen, "Frame skip is a powerful parameter for learning to play Atari," in *Proc. AAAI Workshop on Learning for General Competency in Video Games*, 2015, pp. 10–11.