Blind Snake: Comparative Analysis of Heuristics

Luka Nedimović
luka.nedimovic@dmi.uns.ac.rs
Faculty of Sciences, University of Novi Sad, Novi Sad
Faculty of Computing, Union University, Belgrade

Abstract

In this short paper, we aim to conduct a comparative analysis of heuristics for the **Blind Snake** game. We experiment with four approaches, namely **Zig-Zag**, **Zig-Zag-Swap**, **Biased Random Walk** and **Coiling**.

1 Introduction

Blind Snake game is formally defined as follows:

Imagine that you are playing a snake game on your old Nokia phone that has a screen that has width $\mathbf{A}$ and height $\mathbf{B}$. You have a small $\mathbf{1x1}$ snake that can move up, down, left, and right, and you start from some random point on your screen. Also, there is some apple (also sized $\mathbf{1x1}$) in some other random place on the screen. If you hit the right border of the screen and try to move right, your snake will appear on the left border in a symmetrical position. A similar thing would happen if your hit the top border of the screen (your snake will appear at the bottom), left border of the screen (it will appear on the right), and bottom border (it will teleport to the top). Your goal is to eat the apple as in the usual "Snake" game, but there are some important limitations of the game:

- You are playing blindly (with closed eyes).
- You don't know the size of the screen ($\mathbf{A}$ is unknown, $\mathbf{B}$ is unknown, and the area $\mathbf{S} = \mathbf{A} * \mathbf{B}$ is also unknown).
- All you can do is only to send commands to the game by pressing "up", "down", "left", and "right" buttons.
- You don't know your initial position.
- You don't know your current position in any point in time in the game.
- You don't know the position of the apple.
- You can't cheat in the game and can't look at the screen at all.
- You only know when you find the apple (your snake's position hits the apple's position) – because your phone would signal a sound (your ears are not closed).
- Once you find 1 apple, the game immediately finishes and you win.
- If you use more than $\mathbf{35S}$ left/right/up/down commands (where $\mathbf{S} = \mathbf{A} * \mathbf{B}$ – also unknown value to you, but known to the game engine), you lose the game.
- While you don't know the size of the screen, it can be literally any size, for example: $\mathbf{1x20}$, $\mathbf{100x1}$, $\mathbf{5000x5000}$ – are all correct screen sizes. The only limitation is that $\mathbf{A}$ and $\mathbf{B}$ are positive integers.

In short, the game is formatted as the standard snake game, however the initial position, current position and the position of the apple are unknown. Additionally, lack of $\mathbf{A}$ and $\mathbf{B}$ means that is impossible to trace the borders, so the game formally regresses onto a problem of a probabilistic nature.

2 Observations and Proposed Approaches

Because of the torus effect, it is possible to reformulate the problem such that the snakes position is, indeed, the top-right corner of the border. This makes trivial searching options, often seen in dynamic programming and graph competitive problems, such as only going DOWN and RIGHT (interchangeably), viable for the majority of use cases. Toroidal shape of board does not allow any form of position tracking, such as storing the visited cells.

However, there are methods that use more steps in general, but achieve a higher success rate. Intuitive options include patterns that are, intrinsically, of spreading dimensionality, such as a coiling pattern, which creates a circle of growing radius and ascertains there are no apples within.

In this short analysis, we will propose 4 different strategies to address the problem.

Zig-Zag. As described above, a standard DOWN-RIGHT movement pattern that repeats.
Zig-Zag-Swap. A new strategy that follows the Zig-Zag, but has a probability of changing from DOWN-RIGHT to DOWN-LEFT. The same probability holds for changing back to original Zig-Zag (i.e. DOWN-RIGHT) approach.
Biased Random Walk. Intuitively speaking, when moving, direction change is not that prevalent, meaning that it could be useful to introduce a bias to current direction. For example, if the snake moves upwards, we can introduce a bias of 50% that lets it stay on that direction, and other direction (DOWN, LEFT, RIGHT) sample from the leftover probability. In our implementation, we introduce the bias of 50%.
Coiling. Creating the expanding circle around the initial point. In our example, we implement a circle that turns in LEFT-DOWN-RIGHT-UP directions, i.e. in counter-clockwise manner.

3 Experiments

3.1 Experimental Settings

Each experiment is conducted through respective script, where script's name comes in the format `sweep_{strategy_name}.sh`. Result of each experiment is stored within respectively passed in bash script, in our case being simply named `sweep_{strategy_name}.csv`.

3.2 Experiment Descriptions

We conduct two kinds of experiments for each strategy, first one being a (limited) full search with shots, and the other one being the large search with limited number of runs.

Limited full search. A **shot** is simply a generation of a board. If we have $\mathbf{N}$ shots, that means we are generating a new board $\mathbf{N}$ times, for the same width and height. Maximal width and height are set to **100**, while the number of shots is set to **16**.

Large search. We conduct large search on boards of shape **1000x1000**, for **100** sampled runs. Due to the inefficiency of Zig-Zag and Zig-Zag-Swap, we only perform this search for Biased Random Walk and Coiling strategies.

3.3 Experiments Results

In the following table, we display the metrics for the aforementioned simulations:

Strategy	Passed	Failed	Success Rate (%)
Zig-Zag	137,439	22,545	85.91
Zig-Zag-Swap	137,339	2,2645	85.85
Biased Random Walk	158,508	1,476	99.08
Coiling	158,046	1,938	98.79

Table 1: Limited full search results

Strategy	Passed	Failed	Success Rate (%)
Biased Random Walk	98	2	98
Coiling	100	0	100

Table 2: Large search results

On the following plots we may see the percentage of steps utilized per board size.

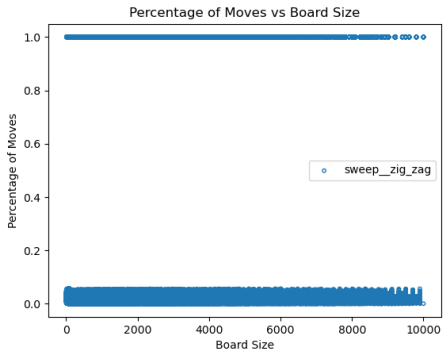

Figure 1: Zig-Zag

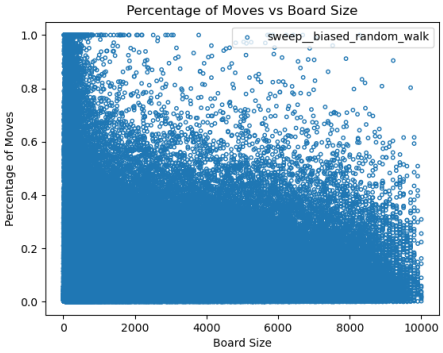

Figure 3: Biased Random Walk

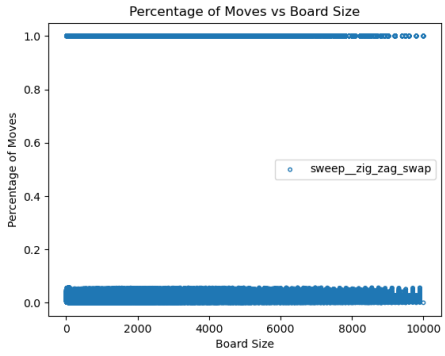

Figure 2: Zig-Zag Swap

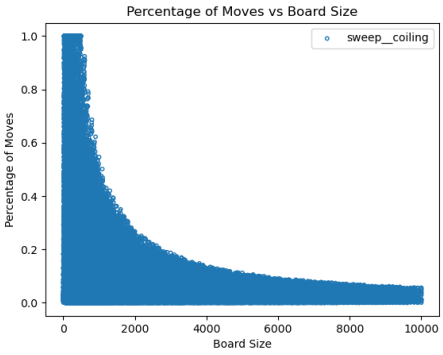

Figure 4: Coiling

It is easy to notice that Zig-Zag and Zig-Zag-Swap methods lead to $\tilde{8}5\%$ success rate, however their utilization of moves signifies that they will never reach their target after first $\tilde{5}\%$ of moves. This indicates the infinite loop they enter.

However, the Biased Random Walk (BSR) and Coiling methods show different statistics. Namely, BSR shows relatively consistent, but slow decay in number of moves, whilst the coiling shows the biggest progress, akin to **inverse proportionality curve**. Performance of the coiling proves itself to be the best, without about 5 - 10% percent of moves possible utilized on the maximum board size.

Coiling also sometimes fails on the boards of shape **1 x N** or **N x 1**, by entering the infinite loop and not being able reach the apple, especially if it is on the first or the last position on the board.

4 Conclusion

In this short paper, we experiment with 4 different heuristics for solving the Blind Snake game. Experimental evidence suggests that the coiling is the most optimal strategy for the boards of greater sizes, whilst the Biased Random Walk achieves great accuracy with the greater amount of moves. The Zig-Zag and Zig-Zag-Swap strategies show similar performances, meaning that the direction swap is countered by the toroidal effect, with looping existent over complete range of sizes.