Mario Kart is Hard

Jeffrey Bosboom Erik D. Demaine Adam Hesterberg Jayson Lynch
Erik Waingarten

Massachusetts Institute of Technology

Abstract

Nintendo's Mario Kart is perhaps the most popular racing video game franchise. Players race alone or against opponents to finish in the fastest time possible. Players can also use items to attack and defend from other racers. We prove two hardness results for generalized Mario Kart: deciding whether a driver can finish a course alone in some given time is NP-hard, and deciding whether a player can beat an opponent in a race is PSPACE-hard.

1 Introduction

Mario Kart is a popular racing video game series published by Nintendo, starting with Super Mario Kart on SNES in 1992 and since adapted to eleven platforms, most recently Mario Kart 8 on Wii U in 2014. The series has sold almost 100 million game copies by 2013 [2] and contains the best-selling racing game ever, Mario Kart Wii [4].

In this paper, we analyze the computational complexity of most Mario Kart games, showing that optimal gameplay is computationally intractable. Our results follow a series of recent work on the computational complexity of video games, including the broad work of Forisek [3] and Viglietta [5] as well as the specific analyses of classic Nintendo games [1].

In Mario Kart, each player picks a character and a race track. There are three modes of play: players race against each other (racing), a player races alone to finish in the fastest time possible (time trial), and players battle in an arena (battle). Each race track features its own set of stationary or moving obstacles and geometry. The race tracks feature sharp turns, uphills, downhills, and big jumps.

In addition, a player may acquire items. Items temporarily give players special abilities. Each Mario Kart game has its own set of items, but three items are common to all Mario Kart games: mushrooms, Koopa shells, and bananas. Mushrooms boost a driver's speed for a certain amount of time. Koopa shells are shot at other players and, upon contact, temporarily stun them, reducing their speed and control. Bananas can be dropped by players along the track, and any player who runs over a banana becomes temporarily stunned. Crucially, shells can destroy bananas.

In this paper, we consider a generalized version of time trial and racing. We allow race tracks to be any size and have carefully placed items on the track. In Section 2, we show that that time trial is NP-hard, that is, it is NP-hard to decide whether a lone player can finish a race track in time at most t. In Section 3, we show PSPACE-hardness for racing: it is PSPACE-hard to decide whether a player can win the race against even a single opposing player. Finally, Section 4 considers upper bounds.

Our proofs apply to nine of the eleven games in the series. The items used in our proofs are available in all games, but tracks in Super Mario Kart and Mario Kart Super Circuit are flat, while our proofs use three-dimensional tracks.

2 Time Trial is NP-Hard

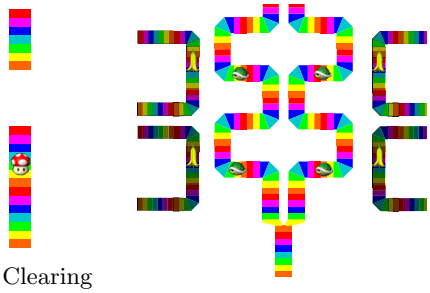
Theorem 1 *It is* NP*-hard to determine whether a driver can finish a given course in at most t time.*

We reduce from 3SAT. Given a boolean formula ϕ with variables $x_1, x_2, \ldots, x_n$, we build a level with the "Rainbow Road" style. The driver will first drive through each variable gadget where the branch they choose represents setting each variable to true or false. After setting all the variables, the driver will drive through each clause gadget. The driver will be able to complete the level in time if and only if the variable assignments chosen in the gadgets form a satisfying assignment for ϕ.

The variable gadget splits the road into two, one corresponding to setting x_i to true, and the other to setting x_i to false. Each path will contain a green shell for each instance x_i or $\bar{x_i}$ and travels over that corresponding location in the clause. This allows the player to shoot the green shell, removing the banana obstacle from the corresponding location in the clause; however, the roads are placed far enough the player cannot jump onto the other track.

The clause gadget is a set of branching tracks with bananas in them. When a green shell is shot to a literal in a clause, it will remove a banana, allowing the clause to be driven through without delay.

Crossover gadgets are simple: roads can pass over and under other roads in three dimensions. To pre-



(a) Clearing Gadget (b) Variable gadget and literals

vent players from shooting shells or bananas during crossovers (maintaining the invariant that players shoot items only inside variable gadgets), we use a clearing gadget.

The clearing gadget, shown in Figure 1a, prevents players from carrying items past it. In a clearing gadget, players must use the mushroom to cross the gap in the track. As players can only hold one item at a time, using the mushroom requires discarding shells or bananas, ensuring players have no items after the clearing gadget.

3 Racing is PSPACE-Hard

Theorem 2 *It is* PSPACE*-hard to decide whether Player 1 has a forced win in a two-player Mario Kart race from given starting positions for the players.*

We prove this problem PSPACE-hard by reduction from the game QSAT. Player 1 will set the existentially-quantified variables and Player 2 will set the universally-quantified variables. We construct the track similarly to the NP-hardness proof. Player 1 will shoot shells from an elevated road to clear bananas from the clause gadgets. Player 2, who sets the universally-quantified variables, is on a separate elevated road throwing bananas into clause gadgets. The roads pass above each other so each player can observe the other player set the variables. This way, we get the alternating behavior while setting variables. The path Player 1 takes is slightly shorter than Player 2's path. If Player 1 can get through the clauses without hitting any bananas, they will win. If Player 1 runs over any bananas and slips, Player 2 will be able to win.

The clause gadget is a road that splits into one road per literal, as in the NP-hardness proof. The literals of existentially-quantified variables are initially blocked by a banana; literals of universally-quantified variables are initially empty.

Player 1's variable gadgets are the same as in the NP-hardness proof; each gadget forks to make the player choose between setting x_i or $\overline{x_i}$ to true, then passes by all the clauses containing that literal so the player can shoot a shell down to remove the banana from that literal instance.

Player 2's variable gadgets have the same structure, but the player instead sets literals to false by shooting bananas (picked up from item boxes in the clause) down into literal instances in the clause gadgets. The same clearing gadget from the NP-hardness proof is used to prevent either player from carrying a shell or banana past a variable gadget.

The elevated roads have appropriate space-filling padding to ensure one player has set their variables before the next player must make a choice. This padding is oriented such that each player can observe the other player's variable choices before having to make their own. Neither player can delay making a choice when it is their turn to set a variable because slowing down to do so allows the other player to win by continuing anyway.

After all variables have been set, Player 1 drives through the clause gadgets while Player 2 drives along a space-filling road slightly longer than the road through the clause gadgets. If all clauses are satisfied (have at least one literal branch without a banana), Player 1 passes unobstructed; otherwise, Player 1 must drive through at least one banana and slip, which causes Player 1 to slow down. In this case, Player 2 wins.

4 Conclusion

In practice, players in Mario Kart generally make forward progress on the track, other than short aberrations caused by attacks. This assumption implies a polynomial bound on the length of solutions, which in turn implies that our results are tight: time trial is NP-complete and racing is PSPACE-complete. Without this assumption, however, we only know containment in PSPACE and EXPTIME, respectively, and it is plausible that we could establish corresponding hardness.

References

[1] Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Vi glietta. Classic Nintendo games are (computationally) hard. *Theoretical Computer Science*, 586:135–160, 2015.

[2] Sam England. The 7 top racing games of all time. http://www.arnoldclark.com/newsroom/015-the-7-top-racing-games-of-all-time, 2013.

[3] Michal Forisek. Computational complexity of two-dimensional platform games. In *Proceedings of the 5th International Conference on Fun with Algorithms*, pages 214–227, 2010.

[4] Guinness World Records. Best-selling racing videogame. http://www.guinnessworldrecords.com/world-records/best-selling-racing-video-game/, 2014.

[5] Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54(4):595–621, 2014.