How hard is Wings of Vi?

MAX LINDBLAD
How hard is Wings of Vi?

An analysis of the computational complexity of the game

Wings of Vi

Max Lindblad

Supervisor: Danupon Nanongkai
Examiner: Örjan Ekeberg

2015-05-10
Abstract

Computational complexity theory is the study of the inherent difficulty of different computational problems. By determining the complexity class of a problem you can learn a lot about how hard the problem is to solve. For games, their complexity class determines sort of an upper limit to how hard they can be. All NP-complete games can be made to be both extremely difficult to play and to analyze. The purpose of this study is to analyze the computational complexity of the game Wings of Vi, where it is shown to be both NP-hard and in NP, and thus NP-complete.
## Contents

1. **Introduction**  
   1.1. Background  
   1.2. Project Specification  
   1.3. Method  

2. **Gameplay Mechanics**  

3. **Proof of membership to the complexity class NP**  
   3.1. Definition  
   3.2. Proof  

4. **Proof of NP-hardness**  
   4.1. Definition  
   4.1.1. Framework  
   4.2. Proof  

5. **Conclusion**  

6. **Discussion**  

7. **References**
1 Introduction

1.1 Background

The computational complexity of playing video games is a subject that has been touched upon several times in recent years \cite{1}\cite{2}\cite{4}\cite{5}. Mostly in the form of proving NP-completeness, or similar results such as PSPACE-completeness, for them. In the case of platformers the proof of NP-hardness was done through the reduction of the NP-complete problem 3-SAT (Boolean satisfiability problem) to the problem at hand. Examples of prior results would be that the generalized versions of the games Lemmings\cite{4} and Super Mario Bros\cite{1} are NP-hard, and that several games from the The Legend of Zelda series are PSPACE-complete \cite{1}. However, far from all games have had their complexity class determined and there has been little to no research beyond that in this area.

The main purpose of investigating the computational complexity of a problem is to determine what the efficiency limit is for algorithms that can solve the problem. Once the complexity is known then one can make a qualified judgement on whether it is viable to create such an algorithm. For example if the problem is NP-hard, it almost always means that a direct solution is infeasible and workarounds need to be found, such as in the form of approximation algorithms or heuristics. While the difficulty of a game can be estimated by simply playing it most of the time, there are some applications where a general algorithm could be useful. For example as a part of a good AI (artificial intelligence) or as an aid in game design.

There is also the factor of different problem instances, even if two games in essence are about solving the same NP-hard problem the problem instance for one of the games might be harder to solve than the other. For example, the subset sum problem of determining whether a set of integers contains a subset that sums to zero has been proven to be NP-complete, however in an instance such as the set \{1, 3, -1\} it is trivial to find a solution. Indeed, several popular games have been proven to be NP-hard yet are still considered “easy” by their player base. This is because the analysis of the computational complexity of games is about worst case scenarios, if a problem is proven to be NP-complete then there can exist problem instances that are very hard to solve, however this does not mean that all problem instances have to be.

1.2 Project Specification

In this project we will investigate the algorithm and complexity of the game Wings of Vi, a difficult 2D platformer. More specifically by attempting to answer the following question:

- Is it an NP-complete problem to solve the game Wings of Vi?

More specifically we will be looking at the decision problem of reachability, whether a goal point $t$ is reachable from a start point $s$. If determining reachability is NP-hard it means that there currently exist no algorithms that for all problem instances can efficiently calculate solutions to a level. Including all such variants, for example in the case of optimization problems such as fastest path or lowest damage taken.
We investigate Wings of Vi because although it is a difficult game for humans to play this does not necessarily mean that the underlying problem is difficult. It would be a very peculiar result if the game Wings of Vi is shown to not be NP-hard considering that other “easier” platformers, such as the Super Mario series, have been proven to be. In addition, determining NP-completeness for platformers is not something that has been done before to any greater extent.

1.3 Method

The general method used for proving NP-completeness of a problem is widely known. It consists of proving the following two statements [3]:

- The problem is in NP.

  This can be done by showing that it is possible for a deterministic Turing machine to verify the validity of a given solution in polynomial time. A Turing machine being a hypothetical device that manipulates symbols on a strip of tape according to a table of rules, and which can simulate the logic of any computer algorithm.

- The problem is NP-hard.

  Proof of NP-hardness is done by showing that it is possible to perform a reduction from a verified NP-complete problem to the problem at hand in polynomial time. A reduction here being the transformation of a problem into another problem, with the purpose of proving that the first problem is at least as difficult as the second. If there exists such a reduction then an algorithm that could solve this specific decision problem efficiently would also be able to do so with all other NP-complete problems.

2 Gameplay Mechanics

In the game Wings of Vi the player controls a character named “Vi” through a series of stages [7]. The goal of the game being to complete the final stage. Each stage can be seen as being further split into a series of save points which allow the player character to restore health and save his/her progress. The player character dies whenever its health is reduced to zero as a result of contact with hazards or falling outside the level. The exact amount of damage taken depends on the chosen difficulty, the difficulty also changes the number of usable save points. Whenever the player character dies they are spawned with full health at the latest used save point. Fig 1 shows an example of a stage section from the game.
At the start of the game the player can only walk and single jump. However as the game progresses the player unlocks more abilities. The following is a list of unlockable abilities:

- Flutter (makes Vi able to float in the air for a short time)
- Double Jump (makes Vi able to perform a second jump mid-air)
- Dash (makes Vi able to collect dash feathers, which can be used to quickly move into one of the 8 directions)
- Demon Form (makes Vi able to turn into a flying demon for a limited amount of time)
- Attack (makes Vi able to attack enemies in 8 different direction depending on the directional keys pressed)

Most actions are not exclusive. Vi can for example both attack and move at the same time, and the direction of a jump can be steered by the use of the movement commands. However there are exceptions to this such as in the case of jumping during a flutter, which interrupts the flutter.

Important to note is that progress is impossible without picking up the abilities and that it is impossible to skip or lose any abilities.

There also exists a number of different environmental objects throughout the game which the player character can interact with. These include switches that open gates and mine carts which the player can ride on (usually over large gaps).
Most gameplay elements have no randomness to them, with a few exceptions amongst some of the encountered enemies. For example; all bosses, large enemies that the player needs to defeat in order to progress, choose their next attack by random and the attacks themselves may have some randomness to them. However we will be not be including them in our analysis.

3 Proof of membership to the complexity class NP

3.1 Definition

A player of Wings of Vi has control over what actions Vi should take at any given time. We can formalize this by defining a move, $m$ to be a tuple $m = (d, l, a)$ where:

- $d$ is the time at which the move is made.
- $l$ is the duration for which the move is performed (a duration of 0 describes an instant press and release).
- $a$ is a list of identifiers for the actions which together describe the move.

Examples of actions would be those mentioned in section 2 about gameplay mechanics. A solution $S$ for a particular stage, described by a game state $g$, would then be a set of moves.

Now it follows to prove that such a solution $S$ is verifiable in polynomial time.

3.2 Proof

Lemma 1. The decision problem of reachability is in NP for the game Wings of Vi.

Proof. We consider a function $F$ which calculates a new game state $g$ with the help of two parameters, the current game state $g$ and a move $m$. The return value of the function being the new game state after the move has been performed in the current game state. If a given solution $S$ contains $n$ moves this function is called $O(n)$ times, each call using the return value of the last call to the function as the parameter $g$. After the final call to the function the area around the goal point $t$ (an area of fixed size) is then checked for the presence of the player character in the returned game state. If the character is present inside that area the solution is valid.

Note that as we have chosen not to consider gameplay elements with random components the function $F$ is deterministic, that is for each game state $g$ and move $m$ there is only one possible end state.

As we have determined that a deterministic Turing machine will verify solutions in polynomial time in respect to the size of the move list the next step is to show that the list of moves will at worst be polynomial in size in respect to the problem space. However, we can determine that this will also be true because in Wings of Vi there exist no gameplay mechanics where there is a reason to go to them more than once \[^7\]. For example:

- Switches are single use.
- Enemies serve no other purpose than to hinder.
• Objects that restore health only do so once.
• Respawning reverts all progress.

This means that the only reason to pass through an area more than once is to reach a location which has not been visited yet. Any given solution which does not have redundant moves (leading to the same game state) must therefore be polynomial in size in respect to the number of gameplay elements. Thus the decision problem of reachability is in NP for the game Wings of Vi.

4 Proof of NP-hardness

4.1 Definition

In order to prove NP-hardness for the decision problem of reachability for Wings of Vi we will perform a reduction from the known NP-complete problem of Boolean satisfiability (SAT), more specifically the sub-variant 3-satisfiability (3-SAT) which has also been proven to be NP-complete [6]. The Boolean satisfiability problem itself is, when provided with a Boolean formula, determining whether there exists an assignment to the variables such that the formula evaluates to true. If the Boolean formulas are restricted to being in conjunctive normal form where each clause can contain at most three literals then the problem becomes that of 3-satisfiability. Recall that a problem in 3-SAT is a collection of clauses where all the clauses need to be true for the Boolean formula to be true and where each clause is true as long as any of its (at most three) containing literals are true (a literal being either a propositional variable or its negation).

Our motivation for choosing to perform the reduction from 3-SAT is the existence of a general framework, described in detail in the following section, for proving that such a reduction is possible for platformers. However, note that a problem being NP-hard would mean that every NP-complete problem is reducible to it. It is impossible for there to be a reduction from 3-SAT without there also existing reductions from all the other NP-complete problems, such as subset sum or travelling salesman, although the difficulty of making such a reduction may vary. Thus, the only reason to choose one reduction over another is simplicity, and we deemed a reduction from 3-SAT to be the simplest option.

4.1.1 Framework

Earlier works that have touched upon the computational complexity for platformers have often used a general framework for proving NP-hardness [1][5]. The framework describes which fundamental units, called gadgets, that are both required and sufficient in proving the existence of a reduction from 3-SAT for a platformer. Each gadget having the purpose of simulating the behavior of a specific element of the 3-SAT problem.
The required gadgets are:

- Start gadget, the start point \( s \) as detailed in the definition of the decision problem of reachability.
- Finish gadget, the goal point \( t \) as detailed in the definition of the decision problem of reachability.
- Variable gadget, a one-time non-reversible choice that depending on the value chosen unlocks certain Clause gadgets.
- Clause gadget, a path which is only traversable if a specific choice was made for one of the three connected Variable gadgets.

As the paths may cross over each other one last gadget is also required:

- Crossover gadget, a way for two different paths to cross without the player being able to switch between them.

Otherwise it would be possible for a certain choice in a Variable gadget to unlock more Clause gadgets than intended.

Together the gadgets enable constructions of the type shown in Fig 2 which each correlate with a 3-SAT problem instance.

![Fig 2. General framework for proving NP-hardness for platformers.](image)

These constructions define a subset of the decision problem of reachability, where the problem becomes one of choosing exclusive paths in such a way that it is later possible to traverse a check path and reach the goal. The choice between two exclusive paths correlates with the assignment of a true or false value to a propositional variable (represented by the Variable gadget). While the traversing of the check path can be seen as the verification of whether the Boolean conjunction of the clauses (represented by the Clause gadgets) holds true. The start and finish points (represented by the Start and Finish gadgets respectively) simply serve to make the problem into one of reachability. Answering the decision problem of reachability under these circumstances is therefore the same as answering the respective problem in 3-SAT.
For proof of NP-hardness it thus suffices to show that it is possible to construct the above mentioned gadgets within the game’s engine.

4.2 Proof

**Lemma 2.** The decision problem of reachability is NP-hard for the game Wings of Vi.

**Proof.** We show that it is possible to implement the gadgets detailed in the section above using the gameplay elements from Wings of Vi. We assume that Vi has unlocked all abilities except Demon Form.

![Fig 3. Start gadget for Wings of Vi.](image)

![Fig 4. Finish gadget for Wings of Vi.](image)

The Start and Finish gadgets are displayed in **Fig 3** and **Fig 4** respectively. The player character starts with a save at the save point shown in the Start gadget, and the goal is to reach the save point pictured in the Finish gadget.

![Fig 5. Variable gadget for Wings of Vi.](image)
The Variable gadget for Wings of Vi can be seen in Fig 5. It has two entrances, one from each of the paths possible in the previous Variable gadget, and by taking either the left or right path down the player character assigns a “value” to the current Variable gadget. Limitations on jump height for Vi means that once a path has been chosen the other path becomes unreachable.

In Fig. 6 it is possible to see the Clause gadget for Wings of Vi. The gadget consists of three switches, which are only accessible via their respective path from a Variable gadget, and a gate that will normally block the path through the room. There is only one way of opening the gate and that is by activating one of the switches in the room (upon which all are toggled to the used position). It is only necessary for one of the switches to be triggered in order to make the room traversable, thus simulating a Boolean disjunction.

**Remark 2.4.2:** The exact limitations of switches are not detailed within the game. However, in the case that there is a limit to how many switches can be assigned to the same gate it is a simple fix to change the clause gadget to have three separate gates on different levels. Each switch could then be assigned to its own gate.
The final gadget is the Crossover gadget which is shown in Fig 7. It consists of two entrances and two exits. The entrances are at the left and top, and the exits are at the right and bottom. The two paths in turn are from left to right and top to bottom. If Vi enters from the left she can take the mine cart in order to traverse the row of portals (the purple-black circles) which would otherwise kill her, and end up at the right exit. However it is not possible for her to go up due to limitations on jump height (the same which prevent the revisiting of Variable gadgets), and the path down is blocked by alternating rows of portals which require quick horizontal movement (which she by default lacks) in order to bypass. If Vi enters from the top she can pick-up the stack of three dash feathers (the white object) which allows her to dash three times, providing her with the quick horizontal movement needed to go down. However the dashes are not enough to get her across the row of portals to the left or right.

**Fig 7.** Crossover gadget for Wings of Vi.

As we have shown all gadgets to be possible in Wings of Vi it follows from the framework shown in section 4.1 that the decision problem of reachability is NP-hard for the game Wings of Vi.

**5 Conclusion**

**Theorem 1.** The decision problem of reachability is NP-complete for the game Wings of Vi.

**Proof.**

In section 3.2 we proved the decision problem of reachability to be in NP for Wings of Vi. In section 4.2 we proved the decision problem of reachability to be NP-hard for Wings of Vi. Thus it follows that the decision problem of reachability is NP-complete for the game Wings of Vi.
6 Discussion

If no error was made with the proofs then the result should go for all generalized stages of Wings of Vi which contain no random elements. There exists a large amount of precedent for the NP-hardness proof and it thus stands on strong theoretical ground. Therefore, while it is conceivable that the NP-hardness proof contains errors, they should all be minor and rectifiable without influencing the end result. A more likely fault would be that the proof could have been more succinct, as closer investigation of some prior work has led me to believe \[2\]. Even so, the framework detailed here allows for easy modification in the case that sub-variants to the problem are to be analyzed. The same confidence for the result cannot be had with the NP-membership proof as most prior work in this area has not dealt with problems whose solutions have as much potential variance. As such it is possible, although unlikely, that the NP-membership proof is inadequate and/or partially redundant.

Important to note is that the problem is solely proven to be NP-complete for the general case. In practice there might be restrictions that prevent the problem from being NP-hard, for example in the case that the stages are limited in size (perhaps as a result of memory constraints). For reasonable stages, such as those that are within the game itself, there is likely to exist an efficient algorithm capable of accurately predicting whether a given stage is beatable. However, unless $P=NP$, there does not exist an efficient general algorithm and as such it is impossible to create AI or analysis tools for Wings of Vi that will always work efficiently.

As we have proven that Wings of Vi is NP-complete it follows that it is a part of the same complexity class as many other platformers, such as Super Mario Bros. Thus, if the non-generalized Wings of Vi is deemed harder by players than the non-generalized version of another NP-complete platformer it is simply because of the difference in how the stages are constructed between the two games. As the games are both NP-complete it must be possible to construct problem instances in both of them that are extremely difficult so solve.

If further analysis on the computational complexity of Wings of Vi would be done the obvious questions to answer would be:

- Is the decision problem of reachability NP-complete also for generalized stages which do contain random elements?
- What natural restrictions to the problem could make the decision problem of reachability not NP-hard?
- Is the decision problem of reachability also NP-hard for each different set of gameplay elements that can be seen in the game?

However answering these questions is beyond the scope of this report and we leave those questions open for future work.
7 References


