“With Fate Guiding My Every Move” The Challenge Of Spelunky

Tommy Thompson
Department of Computing & Mathematics
University of Derby
Derby, UK
tommy@t2thompson.com

ABSTRACT

This paper aims at identifying the challenge of the video game *Spelunky* as a benchmark problem for Artificial Intelligence and Computational Intelligence methods. This is achieved by giving a thorough breakdown of the mechanics and design of the game, which are indicative of features previously established in research on complexity in 2D platforming games. We provide a series of theorems to indicate the challenge of Spelunky, noting that in the general case it not only sits within PSPACE but is at least NP-Hard. Our aim is to highlight this particular problem to the community to merit further discussion.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

1. INTRODUCTION

As a player begins a new run of the video game *Spelunky* [17], we are greeted with an opening narration by the protagonist: the Spelunker. This narration not only sets the mood for the player, but adds an air of mystery to the upcoming trials and tribulations that await. This opening, like the game world itself is generated at runtime[1]: resulting in different experience with each playthrough. This variation, achieved courtesy of a Procedural Content Generation (PCG) algorithm, combined with roguelike design principles are what give *Spelunky* such an air of discontent; a video game that is equally challenging as it is relentless. While on a surface level it shares many similarities with classical 2D platformer games such as *Super Mario Bros.* [9] and *Sonic the Hedgehog* [13] due to its reliance upon platformer mechanics, these features belie a much more complex task, which must be completed in one attempt. The mechanics, goals and secrets of *Spelunky* present an incredibly rich problem for players that not only drives newcomers to the cave entrance, but also helps build and maintain its dedicated fan base.

[1]The title of this paper is one of the many opening lines that are selected at random by the game.

Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015), June 22-25, 2015, Pacific Grove, CA, USA.
ISBN 978-0-9913982-4-9. Copyright held by author(s).

These challenges in-turn, are what make *Spelunky* an interesting domain to explore for Artificial Intelligence (AI) and Computational Intelligence (CI) methods. Recent work by the authors has focussed on the creation of a software API that permits AI/CI applications within the *Spelunky* domain. While this work is still in its infancy, it is important to indicate the relevance and challenge this domain presents. This consideration must be made given that *Spelunky*, while popular, is not as widely recognised as other games-based AI benchmarks such as *Super Mario Bros.* [5] or *Ms. Pac-Man* [8, 12]. As such, this paper is an effort to identify the complexity of this task in contrast to other benchmarks that would be considered similar in nature.

In this paper, we discuss the recurring features and mechanics of the *Spelunky* video game that impact the overall complexity of the task. Our contributions are providing a concise discussion of the gameplay mechanics and trappings of the *Spelunky* game, followed by preliminary analysis of the games challenges, largely denoting that the game sits at least within NP-Hard space. By considering relevant literature in the field of computational complexity of video games, we generate several reductions of the problems this game exhibits in an effort to establish theorems of complexity. We conclude with some discussion of how *Spelunky* compares to existing competition benchmarks such as *Super Mario Bros.* and *Ms. Pac-Man* and subsequently what potential challenges the game mechanics may bring for AI/CI methods.

We begin in section 2 by identifying related literature to the task of establish hardness of video games with similar mechanics, as well as the methods by which our complexity assessment is driven. In an effort to ensure the game is fully understood, we dedicate section 3 to a formal introduction to the *Spelunky* game; identifying the gameplay mechanics, rules and methods of progression. This will serve as means to identify to the reader elements of play that have an impact upon the complexity of the task. This knowledge is subsequently adopted in section 4 as we identify key elements of *Spelunky* gameplay that help us establish theorems for our complexity proof. We conclude with some discussion of what challenges the mechanics discussed in section 4 may bring for AI/CI methods in section 5.

2. RELATED WORK

The practice of establishing the computational complexity of games - be they board games, card games or video games - is an important one. It consolidates the challenge of a game in such a fashion that not only provides evidence of why humans continue to find these problems interesting, but more critically indicates to the research community the challenge presented by these problems as an optimisation task. This is largely encapsulated by works detailed in [3], which gives a thorough though by no means complete survey of the problem area.

Another excellent source is the survey conducted within [7] that provides a lengthy discussion of complexity classification with respect to 'puzzles', these are single-player games, ranging from paper-based or toy puzzles such as *n-Puzzle* and *Rush Hour* to video games of relatively small-scope such as *Minesweeper* and *KPlumber*. This survey also presents a useful introduction to the problem area for those new to computational complexity in games.

This paper is largely influenced by more recent work in establishing the complexity of video games. There is a significant body of work focussed on the classification of 'classic' video games dated from the 1980's and 1990's. This helps to establish that traditional 'platformer'[2] titles such as *Super Mario Bros.* and *Donkey Kong Country* [11] are at least NP-Complete [1, 2]. In addition, 2D platforming games typically associated with PC can vary in their complexity, with *Commander Keen* defined as NP-Hard while both *Prince of Persia* and *Lemmings* are P-SPACE-complete [4, 16]. In addition, efforts have been made to construct an ontology from which conclusions as to the difficulty and hardness can be established from the mechanics of a give game [15]. We refer to the theorems established in these papers for our analysis of *Spelunky*, given that its mechanics are largely influenced by these existing works.

3. SPELUNKY

Spelunky (Figure 1) is a 2D platforming game where the protagonist must complete a series of maps through increasingly hostile environments. As is suggested by the title, the player takes control of a spelunker avatar and upon entering the game, must delve deeper into the underground caverns to ultimately win their freedom.

Spelunky was originally released in 2009 as a freeware game developed in the GameMaker engine for Windows PC platforms by Derek Yu [17]. The game has since been rebuilt in the C++ language with enhanced graphics and is available on a variety of gaming platforms, such as PC, Mac OS, Sony PlayStation and Microsoft Xbox consoles. The full project and source code for the original, often referred to as *Spelunky Classic*, is available from Yu's video game company Mossmouth [17]. It is important to recognise that while there are two different implementations of the game the differences in mechanics are relatively minor and in most instances changes to the later version, known as *Spelunky HD*, make the game more challenging for the player. There are slight variations in control, but these do not cause a fundamental change to the gameplay mechanics.

[2]While the scope of platformer games that exist is fairly broad (Spelunky being one of them), we denote traditional platformers to be extending from the design templates established by the highly popular *Super Mario Bros.* series

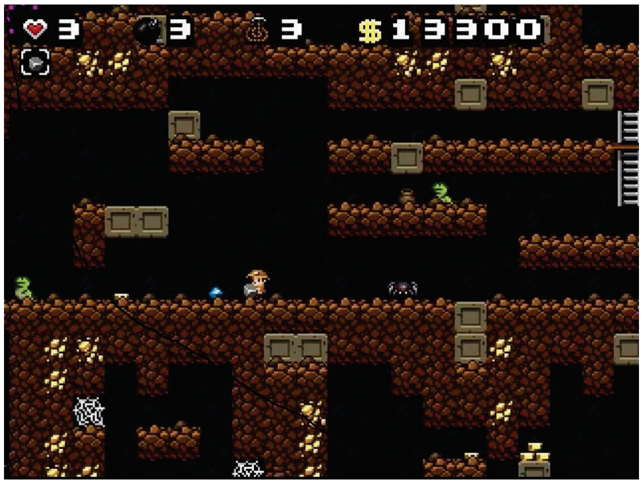
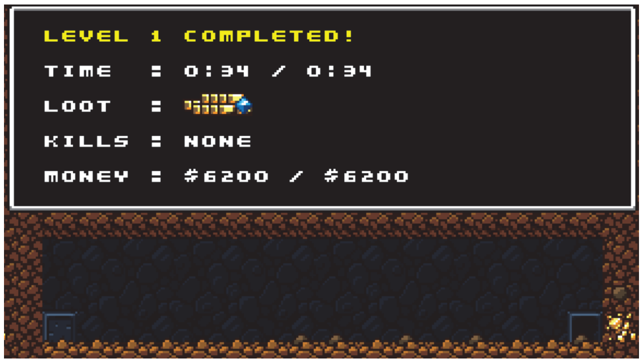


Figure 1: A screenshot from *Spelunky Classic*, where the player is currently navigating through 'The Mines' (world 1).



```
LEVEL 1 COMPLETED!
TIME  : 0:34 / 0:34
LOOT  :
KILLS : NONE
MONEY : $6200 / $6200
```

Figure 2: The scoreboard for a given playthrough after completing the first level. Note that the game records both time and dollar value of all treasures found.

In order to fully explore the challenges of *Spelunky*, this section will focus on the overall structure, core mechanics and particular features of the game that will prove relevant later in this paper.

3.1 Goals & Objectives

Despite the emphasis on money, a significant portion of the community that *Spelunky* has fostered often ignore the score and instead focus on the overall time taken to complete the entire game. While time taken is noted throughout a players progress, it is not recorded as the actual score of the game. However, while there are communities of players who aim to attain the highest score, there is an equally large online community who focus on achieving 'speed-runs', which attempt

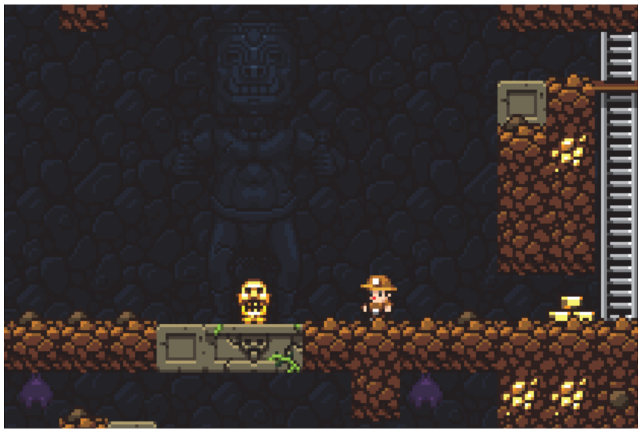


Figure 3: The golden idol is a special treasure that must be taken either to a level exit or a shopkeeper in order to be redeemed for dollar value. However, picking it up often causes environmental traps to spring.

to complete the game as fast as possible[3].

3.2 Game Structure

The world of Spelunky is separated into levels, which the player will typically enter from somewhere at the top. In order to proceed to the next level, the player must find the exit which is often hidden near the bottom [4]. It is not made apparent to the player where these exits are and they must explore the environment in order to find them. In addition, this issue is compounded by both mutators that impact the level as well as how levels are constructed. These issues are discussed in section 3.4.

A typical run of Spelunky requires the player to complete four worlds comprised of four levels each: ‘The Mines’, ‘The Jungle’, ‘The Ice Caves’ and ‘The Temple’. The last of these sixteen levels, level 4-4, is referred to as ‘Olmec’s Lair’ and requires the player to defeat the final boss in order to complete the game.

It is important to note that upon completing a given level, the player does not have the option to return back through the door they travelled from. As a result, players must continue to head towards the final objective and cannot backtrack to acquire items they may have seen in previous levels.

In addition, the ‘typical’ run as noted earlier is one of a number of routes that can be taken to complete the game. We briefly summarise these variants.

3.2.1 Hidden Areas

There are multiple hidden areas to be found in both versions of the game. The one area that exists in both *Spelunky Classic* and *Spelunky HD* is the ‘City of Gold’: a hidden level that typically replaces a segment of the fourth world known as ‘The Temple’. This environment is built entirely of gold, allowing the player to attain a massive increase in score. However, the City of Gold is only one of six hidden areas found in *Spelunky HD*[5]. Access to these hidden areas proves highly taxing given that hidden exits and special items must be found in order to reach them, with similar challenges found within the hidden areas themselves.

3.2.2 Shortcuts

It is possible to skip parts of the game in order to ease the challenge of the overall game. An assisting non-player character known as *The Tunnel Man*, will often greet the player upon completion of worlds 1 through 3, offering to create a shortcut that allows the player to skip all levels up to that point in future play. In order for a shortcut to be built, the player must provide a mixture of items and money to the Tunnel Man. The total cost of each path varies with each shortcut and payment is cumulative over time, allowing the player to eventually unlock a given shortcut if they continue to pay him. Once built, each shortcut is permanent and the player can use it at the beginning of a new game to avoid one or more worlds. However, it is important to note that using these shortcuts renders the subsequent play invalid for leaderboard scores. As such, many challenges[6] both in the Spelunky game itself and within the player community forbid the use of shortcuts.

3.3 Core Mechanics

Spelunky’s basic gameplay mechanics are similar to those of popular 2D platformers such as *Super Mario Bros.* [9] and *Sonic the Hedgehog* [13]. Allowing for the player to control the Spelunker avatar by walking, running and jumping across the environment. However, unlike the more popular examples, there are mechanics that distinguish *Spelunky* from its peers. The avatar shares a feature found within *Prince of Persia* in that the player can grab onto ledges if they are within proximity and cannot reach the top of a given platform. However, unlike *Prince of Persia*, the player cannot simply climb up to the platform and must instead jump to gain sufficient height to reach their goal.

Like Mario, the Spelunker can incapacitate or kill certain enemies by jumping on their head. While most enemies can be defeated by jumping on them, others may require the use of a weapon, such as the whip which is provided at the start of play. In addition, the player can pick up and use a variety of items that are provided throughout play to access hard to reach areas of eliminate enemy characters faster.

3.3.1 Items

What separates *Spelunky* from other popular 2D platformers is the ability to acquire and use items. In *Spelunky classic*, there are 43 different items that the player can use, with *Spelunky HD* extending this total to 51. These items fall under a number of categories:

[3]At the time of writing YouTube user ‘Pibonacci’ holds the world record of 1:55.353 which can be seen at `https://www.youtube.com/watch?v=rgCovly4uz4`

[4]This sole exception to this is the hidden ‘Mothership’ level in *Spelunky HD* that if discovered replaces level 3-4. In this instance players begin at the bottom and must work their way to the top.

[5]The complete set of hidden levels is comprised of The City of Gold, The Black Market, The Haunted Castle, The Mothership, The Worm and the secret 5th world entitled ‘Hell’.

[6]Both achievements delivered in Steam, PSN or Xbox Live as well as challenges to unlock secret items.

Consumable Items that can only be used one time. Each item has a limit on how many the player can carry. While items such as bombs and ropes can be carried in large quantities, others have a limit of one and a replacement must be found before they can be used again. Consumable items can often be replenished for cash in shops found throughout the game world. The exception being the Egyptian artefact 'Ankh' which can only be found and used once per game.

Accessories Items that once collected, are permanently owned by the player for the remainder of their playthrough unless they can be replaced. Typically, accessories are worn by the avatar in order to improve the strength and resilience of the player. For example, spring shoes can increase jump height while the cape reduces the rate at which the avatar falls through the air. Others allow for improved awareness of the environment, with the spectacles revealing treasures hidden in the rock and the compass adding an arrow to the display which permanently points to the location of the current exit door. It's important to note that if an accessory is picked up that affects the same mechanic, it will be replace the original. For example, should the player be using a cape and then pick up a jetpack (both of which impact in-air control), then the cape is replaced by the jetpack and a new cape must be found if the players wishes to use it.

Weapons Weapons must be picked up by the player in order to be used. Melee weapons such as the whip and machete are close-range with varying damage and not useful against distant enemies. This is addressed courtesy of ranged weapons such as the boomerang and shotgun. Players can also adopt items as makeshift weapons by throwing them at enemies. Ironically, players cannot throw weapons at enemies and can only pick up and drop them.

3.4 Level Design

One large element of the challenge in *Spelunky* is that each level is driven by a procedural content generation algorithm, i.e. each level is built algorithmically at runtime, with some examples shown in Figure 5. This has a large impact on the perceived difficulty of the game for human players given the PCG ensures that the player never plays the same level twice (or rather, it reduces the probability of such an occurrence to be extremely unlikely). Despite the adoption of a PCG system, this issue is not as significant as is often perceived, given that the PCG system adheres to a strict series of rules in its construction that ensure a solution path exists for all generated instances. This element is further elaborated upon in section 4. This paper relies upon the authors own examination of the publicly available source code in conjunction with the excellent overview provided by Kazemi found in [6].

Once a level is established, it is populated from a collection of potential enemies. The range of enemies is broad given many are only encountered should the player visit particular worlds of the game. As a result, there are 25 different kinds of enemy in *Spelunky classic*, with the HD conversion extending this count to over 50. Enemy difficulty and strength varies, with some exhibiting simple pattern patrols or behaviours, while others actively pursue the player should they enter a given range. The outlier in this case is the 'Shopkeeper' a NPC responsible for manning stores that players visit to buy new items shown in Figure 4. The Shopkeeper remains friendly towards the player unless they either attack or attempt to steal from him. This results in the NPC becoming overly aggressive towards the player. In addition, this has a knock-on effect of making all subsequent Shopkeepers encountered in the game antagonistic towards the player and the level generator will place additional Shopkeepers near future level exits.

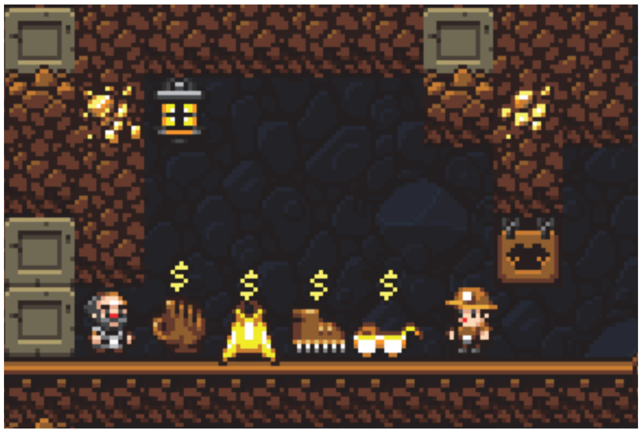


Figure 4: The shopkeeper allows the player to exchange money from treasures for specific items available in the store. These items change in each shop instance.

4. COMPLEXITY ANALYSIS

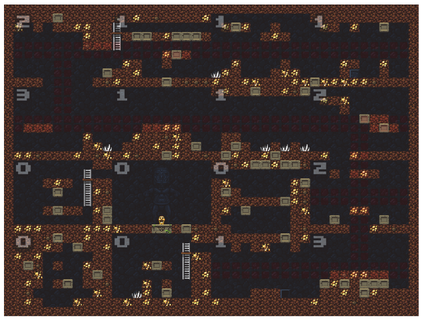
The focus of our analysis is to establish what is the most accurate classification of *Spelunky*. As discussed in [15], most titles that are deemed interesting for a human to play are typically found within the range of NP-complete to PSPACE-complete. As such, we are aware of the bounds in which *Spelunky* may be found within. For this analysis, we rely upon the pre-established meta-theorems discussed in [15] and the framework for proofs for other 2D platforming games found in [2]. Following from this work, we construct scenarios that can occur in the game world that provide evidence of our theorems. These examples should hold for a generalised version of *Spelunky*. This is achieved by constructing particular scenarios using the built-in level editor of *Spelunky Classic*.

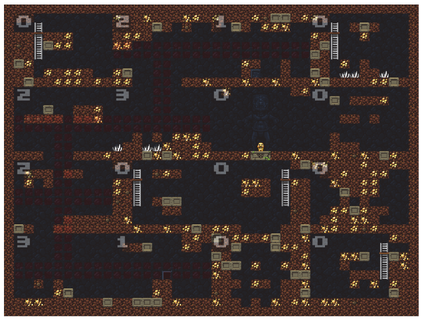
4.1 Existence Within PSPACE

It is important to acknowledge at this stage that both versions of *Spelunky* exist within PSPACE. This is established quite easily courtesy of discussion raised in [2], which states that generalisations of commercial single-player games are likely to exist within PSPACE. This is further corroborated in [15] courtesy of Savitch's theorem [10], given that enemies in *Spelunky* exhibit either deterministic or simple pseudo-random behaviour and that each level configuration can be stored in linear space.

4.2 Environment Traversal

We begin by considering the need in *Spelunky* to navigate the environment in order to reach an exit door. By adopting


(a) A level where the player must cross 10 'rooms' built by the generator.


(b) A level with incentive to explore courtesy of a golden idol in the centre.

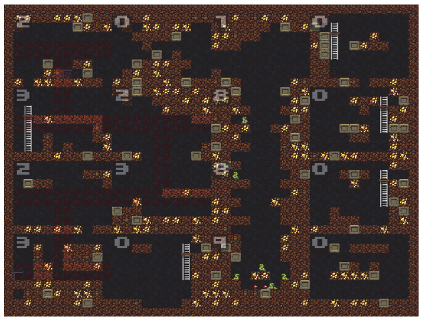

(c) An procedurally generated level that has a 'snake pit' at the bottom.

Figure 5: A collection of procedurally generated levels, where the main solution path is constructed (faintly visible as a red line), before 'padding' the rest of the world accordingly. These screenshots recorded courtesy of the tool developed in [6].

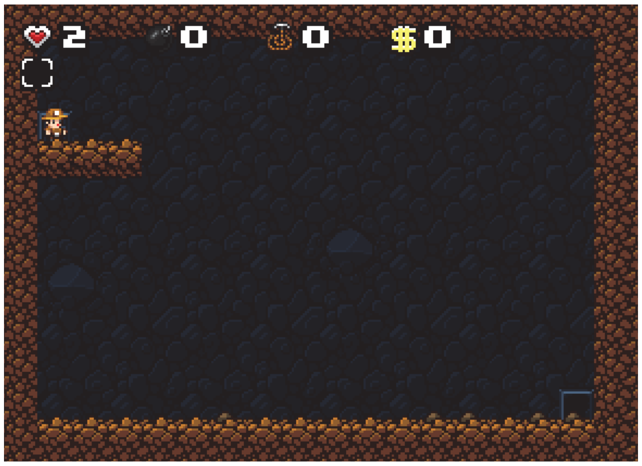


Figure 6: An example that satisfies the assertion of Theorem 1, whereby the spelunker must a single-use pathway to reach the exit.

Metatheorem 1 listed in [15], we can establish the following theorem:

Theorem 1. *The existence of both location traversal and single-use pathways classify Spelunky as NP-Hard.*

We can provide proof of this theorem courtesy of a reduction from a Hamiltonian cycle. Whereby nodes are locations the player must visit and edges are the single-use pathways that exist between them. *Spelunky* exhibits both traits: the former is achieved not only by the need for the player to discover the valid path through the level, but also the desire to collect gold in order to increase their score. Meanwhile the latter trait is exhibited in circumstances whereby the player can no longer reach a previously established area of the level.

If we consider the scenario shown in Figure 6 where the avatar must fall down a large drop while having only two units of health. Given the height, the spelunker will survive the fall but only have one unit of health remaining. Since

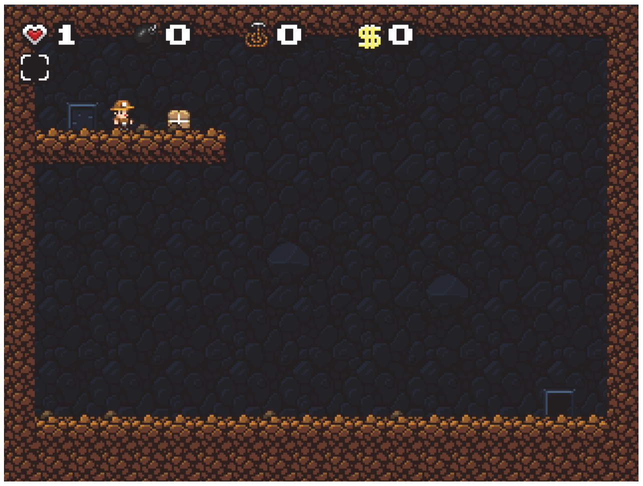


Figure 7: An example that satisfies the assertion of Theorem 2, the spelunker must use a collectible token (a parachute) in order to cross a 'toll road' that exists in the game or face certain death.

the player is not carrying any items to help climb back up, this maintains the case of single-use pathways.

Our theorem need not concern itself with whether a solution path exists in the context of a regular *Spelunky* game, given that one is always guaranteed by the generator. If we view the level space as a 4×4 grid, the generator will navigate each 'room' in the grid until a solution-path is constructed from the starting position in the top row, to the exit door, which will be placed on the bottom row. Only once a solution-path has been built are side rooms with no exits built into the environment [6]. As shown in Figure 5 this can result in solution paths of varying lengths. While the lower bound for this can be established as four rooms, it is noted in [6] that such solutions, which would require multiple top and bottom exits, typically result in a 'snake pit' as shown in Figure 5c, as the generator attempts to force a larger path to be constructed.

4.3 Items Allow For Crossing 'Toll Roads'

If we consider Metatheorem 2.a found in [15, p.600], it notes that a game can be denoted as NP-Hard should a game feature "collectible tokens, toll roads and location traversal". While we have previously established the notion of location traversal courtesy of Theorem 1, we must address the notion of a toll road: a part of the game world whereby consumption of a collectible token must occur in order for the avatar to traverse it. A collectible token is an item that must be found in-game and used for some purpose. If the player does not have the required token, then they cannot traverse the space. With this in mind, we identify the following:

Theorem 2. *The existence of consumable items that allow the Spelunker to traverse hazardous environments only once classify Spelunky as NP-Hard.*

This assertion can be satisfied courtesy of the example shown in Figure 7. While similar to Figure 6, the Spelunker cannot successfully traverse the gap, given he only has one unit of health. This will result in death. However, a parachute has been provided that will allow for safe passage but can only be consumed once. This creates a toll-road given that the Spelunker cannot return back to the original point due to a lack of resources. It is worth noting that it is noted in [15] that some tokens can be cumulative; allowing for the player to carry several of the same type of token. The example in Figure 7 does not meet this description given it is a consumable item (established in section 3.3.1) with a carry-limit of one.

4.4 Bombs and Ropes Act As 'Keys'

Similar to the toll-road problem, Metatheorem 3 of [15] identifies door and key mechanic: whereby the player can collect 'keys'. Keys are items that enable the opening of 'doors', thus enabling the player to continue through a particular problem. Much like tokens, keys can must be collected and can be cumulative, allowing the player to carry multiple keys at once. We can consider many of the consumable items in *Spelunky* as either keys or tokens, given that they enable access to parts of the game world that would otherwise not be possible. While we have already shown items can be used as tokens in Theorem 2, we present Theorem 3, inspired by Metatheorem 3.b of [15].

Theorem 3. *The existence of consumable items such as bombs and ropes to create 'doorways' to enable access to unreachable areas, combined with the need for location traversal make Spelunky NP-Hard.*

Players will often find themselves in circumstances where they have placed themselves in a difficult position. Falling down parts of the path which are in fact dead ends is a common occurrence and in some cases may be difficult to return back to their starting point (see the 'snake pit' shown in Figure 5c as an example). However, the player can use certain consumable items to open 'doorways' in the world thus preventing the player from ending the game prematurely given they have effectively lost.

Examples that satisfy out theorem can be found in Figures 8 and 9 showing the use of bombs and ropes respectively. In Figure 8, we see that the player is effectively blocked from reaching the exit due to a wall. The use of the one bomb in the Spelunker's possession, placed in the appropriate location, allows for the wall to be blown apart, thus creating a permanently open doorway between the two locations. Meanwhile, Figure 9 achieves a similar effect through the use of ropes, whereby the Spleunker deploys a rope in position such that it enables access to the exit door.

A further argument could be made to adopt this theorem in the context of shortcuts provided by the *Tunnel Man* mentioned in section 3.2.2. In this case the player must repeatedly provide the NPC with gold and consumable items. By this token, we could also consider gold cumulative keys, given that it enables access to permanent doorways that avoid entire segments of the game.

5. DISCUSSION

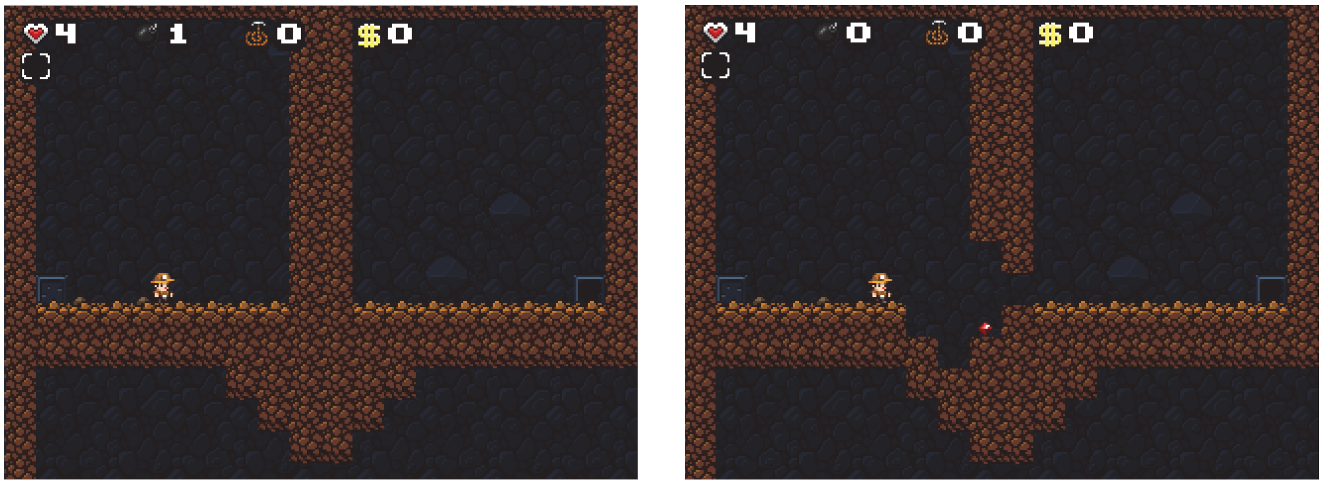
In section 4, we have denoted that both iterations of *Spelunky* exist within PSPACE and Theorems 1 through 3 help classify the problem as NP-Hard. These theorems have focussed largely on generalised instances of the *Spelunky* problem, rather than focussing analysis of the 'core' game, as players attempt to navigate the complete sequence of levels. This could lead to further clarification of this work and discussion of the challenges this game presents.

It's important to acknowledge that the analysis provided in section 4 provides us with evidence that suggests *Spelunky* is in-line with its peers that are established benchmarks within the artificial intelligence community. Both *Super Mario Bros.* and *Pac-Man* have been explored at length in [2] and [15] respectively. In both cases, these games have been classified as NP-Hard. There is no mention of whether there is any discernible difference between *Pac-Man* and *Ms. Pac-Man* which differ largely through non-deterministic ghost execution. Given that the discussion raised in [15] is focussed largely on the maze construction and more abstract behaviour of the ghosts (moving from hunt to evade states based on power pill consumption), it would appear that both these existing benchmarks share the same complexity as *Spelunky.* In addition, popular AI benchmark *Starcraft* is proven (rather simply) to be NP-Hard in [15], but it is noted that the real complexity is expected to be EXP-Hard due to the fact it is a competitive two player game.

What such complexity analysis fails to address is the challenges that the *Spelunky* domain creates for AI applications that distinguish it from its contemporaries. As noted previously, *Spelunky* shares many mechanics and tropes with classic 2D platformers such as *Super Mario Bros.* However, as is evident throughout section 3, there are many facets of the design that distinguish *Spelunky* from commercial games previously adopted as AI/CI benchmarks.

5.1 PCG For Level Construction

Perhaps the critical difference is the use of a PCG system to build levels at runtime. This introduces a new element to gameplay: the need to search for the exit. This is an issue not found in either *Super Mario Bros.*, where the flagpole



(a) Spelunker has one cumulative token, a bomb, but cannot reach the exit.

(b) By using the bomb, the Spelunker creates a path that continues to be accessible after items consumption.

Figure 8: An example that satisfies Theorem 3, in that a ‘doorway’ is built by using a cumulative key, in this case a bomb.

is always at the far right of the level, or *Ms. Pac-Man* for which there is no exit, given the collection of pills is the condition by which a level is completed. This compounds the existing problems faced by research conducted previously in the Mario AI Competition, originally detailed in [14], given that any player (be they autonomous or human) only know that the exit is beneath them. They must subsequently navigate the map to find the exit. While *Super Mario Bros.* faces a similar problem in that the exit is ‘somewhere to the right’, the game operates within a view frame which is fixed on the vertical axis. Meaning the player need only concern themselves with moving this frame along the horizontal axis in order to reach the flagpole. *Spelunky* provides a more challenging problem given not only must players move the view frame along both horizontal and vertical and axis in order to discover the exit location. This exit, as noted in section 4.2, will always appear on the bottom row of the 4×4 room matrix, thus providing some general direction for any search process to adopt. Despite this, the non-linear arrangement of the map creates dead-ends and force backtracking that may not be possible as mentioned in Theorem 1. Furthermore, the use of PCG as a level construction mechanic adds a new wrinkle to an already challenging task: there is no guarantee that the exit will be in the same place the next time the player visits this level. As such, an effort must be made to ensure the search process explores the environment while adopting knowledge accrued from the current permutation of the level.

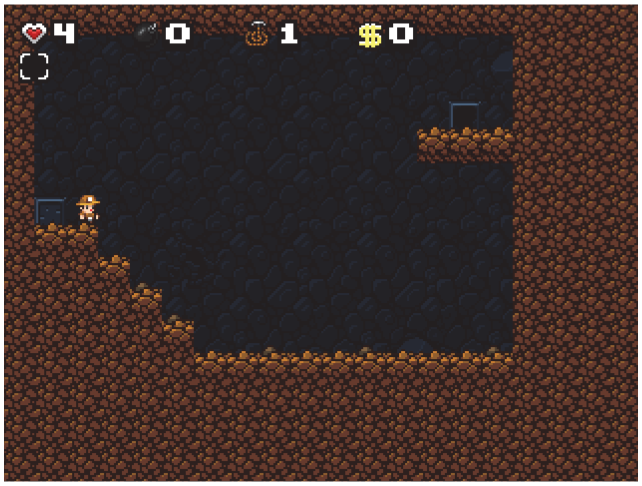
5.2 Treasures as Resources

The collection of treasures in *Spelunky* compound the navigation challenge, given that they are often in areas off the path to the exit. In addition, as discussed in section 3.1, collection of treasures not only impacts score, but provides currency for valuable resources that enable more effective navigation. This distinguishes the game from *Super Mario Bros.*, where similar items are always in fixed positions and coin collection occurs periodically throughout a linear path. In addition, coins in Mario eventually provide players an opportunity *Spelunky* does not: replaying segments of the level upon death.

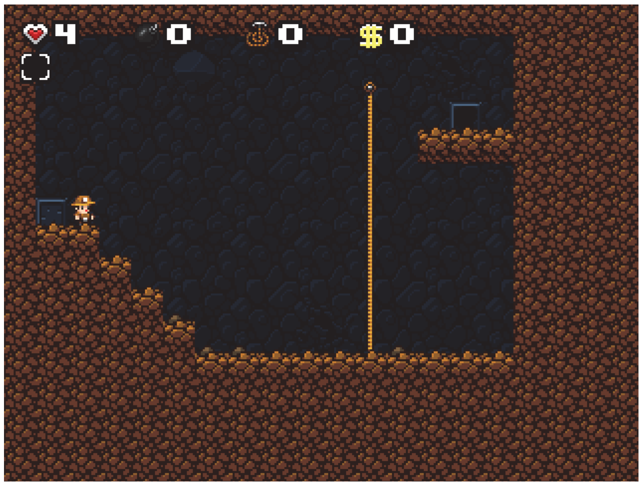
We argue that treasures in *Super Mario Bros.* merely create temporary distractions that can easily be ignored should the player not deem it necessary. This is evident in the most successful *Super Mario* bots detailed in [5, 14], where the emphasis was purely on navigation and enemy avoidance. By comparison, treasure in *Spelunky* is a valuable resource that players ignore at their peril. In addition, it is littered treasure throughout a disjointed environment full of dead-ends and traps. This presents an interesting multi-objective optimisation problem which is similar, albeit more complex than *Ms. Pac-Man*, since in the latter the entire level is visible in a single view-frame. This means that all pills and fruit are immediately visible upon their insertion into the game world and the player can begin to plan accordingly against the four NPCs found in the maze. Furthermore, completion of the level in *Ms. Pac-Man* is reliant upon treasure collection and does not carry time-constraints. By comparison, players only become aware of the existence of treasures and indeed enemies in *Spelunky* once they have seen them in the view-frame. Planning to acquire treasures and factor enemies into that decision making process can only occur once their existence has been confirmed by the player. Meanwhile, other treasures only become apparent once parts of the terrain are destroyed. This reinforces the relationship between collecting treasures and purchasing resources: given that many treasures can only be acquired by first spending some that has previously accrued on items.

5.3 Terrain Deformation and Path Refinement

Lastly, we refer to Theorems 2 and 3 in that the adoption of items as cumulative keys to permit access to areas of the game map that would otherwise not be possible. This opens up another key issue that separates *Spelunky* from its contemporaries, in that the navigation issues already discussed in this section can actually be reduced by smart adoption of items. This introduces an entirely new optimisation task: minimising the distance travelled to reach the exit in contrast to the amount of resource required to achieve this re-



(a) The Spelunker is carriyng one rope, while currently unable to reach the exit door.



(b) By using a rope, the Spelunker creates a path that continues to be accessible after items consumption. Though it is worth noting the player can still kill themselves if they do not use this path wisely.

Figure 9: An example that satisfies Theorem 3, in that a 'doorway' is built by using rope, which we consider a cumulative key.

duction. This presents a new challenge that is largely at odds with the previous task: exploring the game world to find treasures to pay for the items we wish to use.

6. CONCLUSION

In this paper we have identified the core mechanics and gameplay rules of 2D platformer video game *Spelunky*. This effort is driven largely by a desire to classify the 'hardness' of generalised instances of the game and raise this problem for further discussion. In addition, this paper identifies that *Spelunky* is expected to be at minimum as complex as existing benchmarks for artificial and computational intelligence techniques.

7. ACKNOWLEDGEMENTS

The author wishes to thank Daniel Scales for contributions and feedback on this paper. In addition, thanks must be given to the contributors of the Spelunky Wiki[7], which acted as a useful reference for differentiating the content of *Spelunky Classic* and *Spelunky HD*.

8. REFERENCES

[1] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta. Classic nintendo games are (np-) hard. *arXiv preprint arXiv:1203.1895*, 2012.

[2] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta. Classic nintendo games are (computationally) hard. In *Fun with Algorithms*, pages 40–51. Springer, 2014.

[3] E. D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *Mathematical Foundations of Computer Science 2001*, pages 18–33. Springer, 2001.

[4] M. Forišek. Computational complexity of two-dimensional platform games. In *Fun with Algorithms*, pages 214–227. Springer, 2010.

[5] S. Karakovskiy and J. Togelius. The mario ai benchmark and competitions. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):55–67, 2012.

[6] D. Kazemi. Spelunky generator lessons, 2013. [Online; accessed 08-February-2015].

[7] G. Kendall, A. J. Parkes, and K. Spoerer. A survey of np-complete puzzles. *ICGA Journal*, 31(1):13–34, 2008.

[8] Midway Games. Ms. Pac-Man. Namco Bandai Games, 1982.

[9] Nintendo EAD. Super Mario Bros. Nintendo, 1985.

[10] C. H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[11] Rare Ltd. Donkey Kong Country. Nintendo, 1994.

[12] P. Rohlfshagen and S. M. Lucas. Ms pac-man versus ghost team cec 2011 competition. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 70–77. IEEE, 2011.

[13] Sonic Team. Sonic the Hedgehog. Sega, 1991.

[14] J. Togelius, S. Karakovskiy, and R. Baumgarten. The 2009 mario ai competition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

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

[16] G. Viglietta. Lemmings is pspace-complete. In *Fun with Algorithms*, pages 340–351. Springer, 2014.

[17] Yu, Derek. Spelunky. Mossmouth, 2009. [Online; accessed 08-February-2015].

[7]spelunky.wikia.com/wiki/Spelunky_Wiki