DIMACS Technical Report 2004-11
April 2004

The Hardness of the Lemmings Game, or "Oh no, more NP-Completeness Proofs"

by

Graham Cormode[1]
Center For Discrete Mathematics and Computer Science,
Rutgers University, Piscataway NJ.
graham@dimacs.rutgers.edu

DIMACS is a collaborative project of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs, NEC Laboratories America and Telcordia Technologies, as well as affiliate members Avaya Labs, HP Labs, IBM Research and Microsoft Research. DIMACS was founded as an NSF Science and Technology Center.

ABSTRACT

In the computer game 'Lemmings', the player must guide a tribe of green-haired Lemming creatures to safety, and hence save them from an untimely demise. We formulate the decision problem which is, given a level of the game, to decide whether it is possible to complete the level (and hence to find a solution to that level). Under certain limitations, this can be decided in polynomial time, but in general the problem is shown to be NP-Hard. This can hold even if there is only a single lemming to save, thus showing that it is hard to approximate the number of saved lemmings to any factor.

1 Introduction

Lemmings is a popular computer game, originally released by Psygnosis[1] in the early 1990s. Since then, there have been many different versions, sequels and imitations. It has been converted to a large number of different formats, and has inspired many websites devoted to the game[2]. There is a USENET newsgroup for the game, alt.lemmings, and an associated FAQ, `http://www.utahdesign.com/lemmings/downloads/Lemmings-FAQ-v20.t` The cleverly constructed levels in the original Lemmings games have caused many players to ask themselves whether completing a particular level is even possible. Here we will show that deciding whether a particular level is possible is an NP-Complete problem. However, under some restrictions then the question is decidable in polynomial time.

This line of research fits into interest in the study of games, more specifically computer games (such as Tetris [DHLN03], Minesweeper [Kay00] and so on). Understanding the complexity of Lemmings adds to our knowledge of the computational complexity of such popular games. Lemmings has been proposed as a model challenge to Artificial Intelligence and motion planning systems [McC94]. As with other such challenges [HLH03], it turns out to be an example of the well-studied class of NP-Complete problems, which are not known to have efficient solutions. Hence the challenge is unlikely to be answered with methods that can guarantee fast (polynomial time) results. It is tempting to conclude that the ability to encode NP-Complete problems is correlated with the structure that makes certain puzzles appeal to us. This claim is somewhat extravagant, especially since the constructions which allow us to encode such problems are typically large, unwieldy and repetitive (consisting of a few gadgets, repeated over and over). Still it is of interest that a growing number of the most popular computer puzzle games (Tetris, Minesweeper and Lemmings) admit proofs of hardness. It remains of interest to find examples of popular games which are more tractable.

Layout. In Section 2, the Lemmings game is described, and in Section 3, this is formalized into the language LEMMINGS, which is shown to be in NP. The language is shown to be NP-Hard in Section 4, and this holds even when we are restricted to a single Lemming (Section 5). But, under different restrictions, the game can decided (and solved) efficiently in polynomial time, as shown in in Section 6. Related work is discussed in Section 7, and extensions are explored in Section 8.

[1] The company has since been absorbed into Playstation Europe. Some information is at `http://www.psygnosis.org/`

[2] See, for example, `http://www.utahdesign.com/lemmings/`, `http://www.deinonych.com/lemmings/`. Recently, a Javascript implementation of the game was created, but this was subsequently withdrawn for copyright reasons.

2 The Game

What follows is a concise description of the game 'Lemmings'. Some aspects are omitted or abbreviated in order to give as simple a presentation as possible[3].

Lemmings are simple creatures, who must be guided to safety by the player. They live in a two dimensional space and are affected by gravity. Lemmings begin walking in a certain direction, and if they encounter an obstacle that they cannot surmount, such as a vertical blockage greater than their own height, then they will turn around and walk back in the direction they came. If they encounter a hole, then they will simply fall down it. If a Lemming falls beyond a certain distance, then it will die. In the full game, there are many ways in which Lemmings can die, including drowning, stepping in fiery lava, being crushed by a trap, being blown up and so on; however, here we will focus on death by falling. The world is made up of various types of material in addition to air: Permeable, which can be dug into in a variety of ways, and Solid, which it is not possible to tunnel through in any way. Time in the world of Lemmings is discretized into time units such that every event happens at a time which is an integer multiple of the unit of time. Space is similarly discretized.

Skills. The player plays the game by giving "skills" to the Lemmings to affect their course and the course of their peers. There are eight skill types, which come in three classes, permanent, semi-permanent, and temporary. The permanent skills are:

- (1) Climber: a Lemming given the skill of Climber can scale tall vertical obstructions
- (2) Floater, meaning that a floater Lemming can fall unlimited distances without coming to harm on impact with the ground.

These skills remain with the Lemming for the duration of its life. Two skills are deemed semi-permanent:

- (3) Exploder, which causes the Lemming to explode after a short delay causing some collateral damage to any surrounding permeable material[4]. The Lemming is then dead, and is removed from play.
- (4) Blocker, where the Lemming halts and causes all Lemmings which walk into it to turn around[5].

Temporary skills affect the Lemming for a limited duration, after which the Lemming returns to its normal pattern. The temporary skills include:

[3] See also the original game manual, available from `http://www.kallex.de/lemmings/misc/instructions.txt`. There are also public domain demonstrations of the game which include a small number of levels for the player to experiment with, available from the websites listed previously.

[4] Those of a philosophical bent might enjoy arguing whether this has a permanent or temporary effect on the Lemming.

[5] This might appear to be a permanent skill, but experienced players of Lemmings know ways to turn blockers back into regular lemmings.

- (5) Builder. The Lemming that is made into a builder builds a bridge consisting of 12 steps,[6] upwards and in the direction that it is currently facing. Bridges are made of a permeable material, which can later be dug through.
- (6) Basher. The Lemming proceeds to "bash" horizontally in the direction that it is facing. If the material is permeable, then the Lemming will clear a path through the material which it and other Lemmings can walk though. The basher will stop bashing when it breaks through the permeable material to air, or when it encounters solid material.
- (7) Digger. The Lemming starts to dig vertically downwards, stopping when it breaks through into air or hits solid material.
- (8) Miner. Like digger and basher, the Lemming digs through permeable material. Miners dig in a diagonal direction, down and in the direction the Lemming was facing when it was made into a miner. Mining stops when the miner breaks through to air or solid material.

The game consists of a number of levels. Each level consists of some arrangement of material to make a two dimensional world, including a number of *entrances* from which Lemmings emerge, and an Exit, into which they must be guided. When a Lemming reaches the Exit, it is removed from the level. Each level has a number of parameters:

- The number of Lemmings which emerge from each entrance and the rate at which they emerge.
- The number of Lemmings which must be safely guided to the exit for the level to be successfully completed.
- A time limit within which the requisite number of Lemmings must be saved.
- For each skill type, the number of times each skill can be given to the Lemmings.

3 The Decision Problem

We shall formalize the description of a level to be a tuple:

$$\mathcal{L} = (limit, save, lems, start, width, height, grid, exit, skills)$$

The contents of the tuple are:

- $limit$, the time limit, in discrete time units. For technical reasons, we will insist that the time limit is bounded by a polynomial in the size of the level. We conjecture that this restriction is unnecessary, by claiming that if it is not possible to complete the level in time polynomial in the level size, then it is not possible to complete the level at all.
- $save$, the number of Lemmings which must be saved.

[6] Not to be confused with any other 12 Step programme that the reader may be familiar with.

- $lems$ is the number of Lemmings to be released into the level.
- $start$ is a vector of length $lems$. It consists of pairs $start[i] = (pos, t)$, meaning that the ith Lemming arrives in the level at position pos at time t.
- $width$ and $height$ describe the size of a bounding box located at the origin which contains all the objects in the level, so that every object occurs within this area.
- $grid$ is a rectangular array of dimension $width$ by $height$, which records the configuration of the level at time zero. For each location, it indicates whether that location is air, permeable, or impermeable.
- $exit$ gives the location of the Exit.
- $skills$ is a vector of length 8 which records for each of the eight skills listed above, how many are available to give to Lemmings.

A player of Lemmings has control over which skills to give to Lemmings at any given time. We can formalize this by defining a *move*, m to be a tuple $m = (t, x, y, l, s)$ where

- t is the time at which the move is made
- x, y give the location of the Lemming to which the skill is to be given.
- l is the identifier of the Lemming to which the skill is being given (so $1 \leq l \leq lems$).
- s is the identifier of the skill (in the range 1 to 8) which is to be given.

A strategy, $\mathcal{S}$ for a particular level, $\mathcal{L}$, is a sequence of moves, ordered by time. Each Lemming can be given at most one skill in each time step (it would not be of advantage to the player to give multiple skills in each step, since adding skills often overrides previous ones, eg. making a basher into a digger stops the lemming bashing and starts it digging). A *valid move* is a move from a strategy, such that, when the move is made, the named Lemming is at the location specified, the skill being given is available and the Lemming is capable of receiving the skill[7]. A *valid strategy* is one in which every move is valid. Note that multiple moves (on different Lemmings) may be made at the same time instant, in which case they are considered in their order in the strategy (the order should not matter). A *winning strategy* for a level $\mathcal{L}$ is a valid strategy which results in the target number of lemmings being saved in the time limit. Given a level of Lemmings, $\mathcal{L}$, the decision problem is, is there a winning strategy, $\mathcal{S}$. Denote as LEMMINGS the set of all levels for which the answer to the decision problem is "yes". We will also consider a particular subset of this set, restricted to levels where there is a single Lemming ($lems = 1$), and refer to this version as 1-LEMMINGS.

3.1 LEMMINGS $\in NP$

In order to show that LEMMINGS $\in NP$, it suffices to show that it is possible to check whether a given strategy $\mathcal{S}$ for a level $\mathcal{L}$ is a winning strategy, in polynomial time.

Lemma 1. LEMMINGS $\in NP$.

[7] For example, a Lemming which has been made a floater already cannot receive the same skill again.

Proof. The existence of a computer game of Lemmings essentially demonstrates that the strategy is a certificate for membership of the level in the language LEMMINGS. The computer game allows the user to specify the strategy as a clock marks out time and the Lemmings move around the level. At the end of the time limit, the game gives a positive response if the strategy was a winning one, and a negative response if the strategy was not winning.

This approach can be taken here: for each time step, a checker can apply any moves specified for that time step, ensuring that each one is valid, and advance the position of the Lemmings in accordance with their current position and direction. After the number of steps specified by the time limit, the checker halts and reports whether the strategy was a winning one or not.

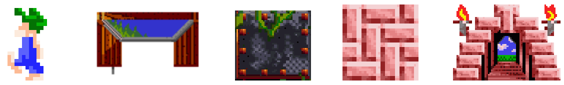
The fact that this verification process can be completed in polynomial time relies on the following facts: (1) Each timestep can be evaluated in time polynomial in the input size. This follows since the starting position of each Lemming is explicitly encoded, and it takes polynomial time to compute the new position of each lemming from its old location and direction. (2) Each move can be verified in polynomial time, provided that the current configuration for the time of the move is available. (3) The total number of timesteps simulated by the checker is polynomial in the input size. This relies on the time limit being bounded by a polynomial in the size of the level (poly in width and height). Since the grid description is linear in these parameters, then the number of timesteps is polynomial in the grid, and hence the input size. (4) The total number of moves is polynomial in the input size. This uses the observation that a strategy can be winning only if every move is valid, and there can be at most one valid move on each Lemming in each time step. The vector $start$ is linear in the number of Lemmings, and as noted above, the time limit is polynomial in the input size, and so combining these shows that the total number of valid moves, and hence the total number of moves in a winning strategy, is linear in the input size. □

4 LEMMINGS is NP-Hard

In order to demonstrate that LEMMINGS is NP-Hard, we shall give a poly-time reduction from 3SAT to LEMMINGS. The generated level of Lemmings will be solvable if and only if the corresponding instance of 3SAT is satisfiable.

4.1 Construction of the level

Given an instance of 3SAT, we shall construct a level of Lemmings. The level will be constructed by taking various pieces defined here (gadgets), and connecting them in a deterministic way so as to encode the structure of the 3SAT instance. The instance of 3SAT consists of: n boolean variables, $v_1 \dots v_n$; and m clauses $c_1 \dots c_m$, where each clause consists of a triple of literals, c_{i1}, c_{i2}, c_{i3}.



(a) (b) (c) (d) (e)

Figure 1: Object types used here: (a) A Lemming (b) Entrance (gives initial location of a Lemming) (c) A square of Solid material (d) A square of Permeable material (e) The exit

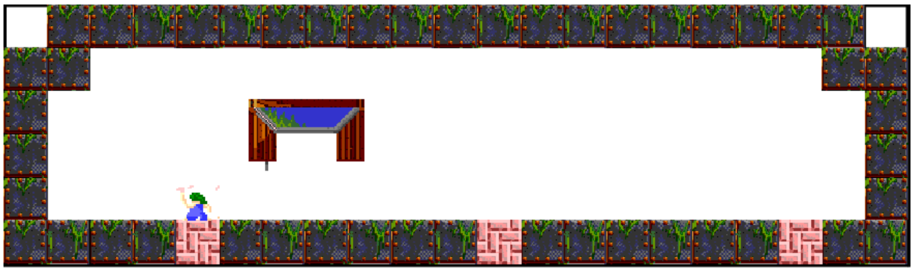


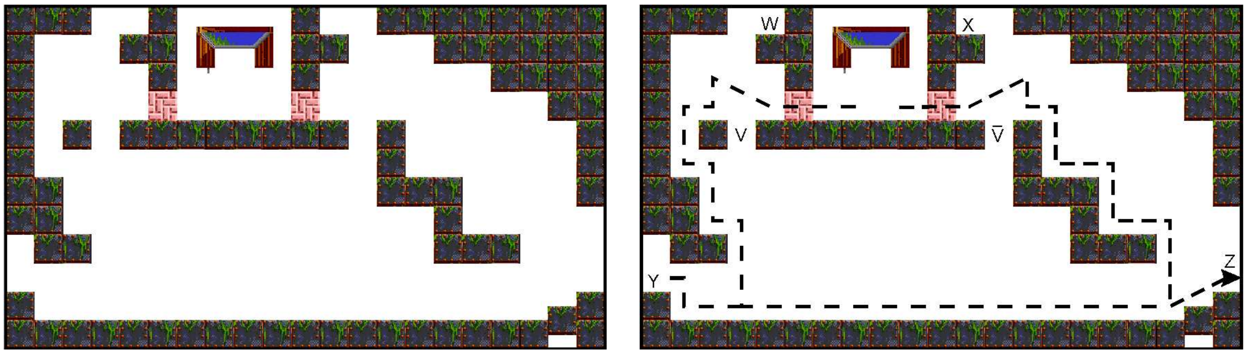
Figure 2: Clause Gadget

The transformation into LEMMINGS will use one Lemming to represent each clause, and one Lemming to represent each variable, so there will be a total of $m + n$ lemmings in the level. All $(m + n)$ Lemmings must be saved for the level to be completed successfully. The path taken by a clause Lemming will correspond to one of the literals in the clause which is satisfied, and the path taken by a variable Lemming will indicate whether the atom or its negation is satisfied. For this level, we will allocate n bashers, n builders and m diggers. No other skills will be available. The time limit will be set sufficiently long to allow a Lemming to walk the course necessary to get to the exit, which is linear in the level size. The level size will be polynomial in the size of the input instance of 3SAT.

Clause Gadget. Figure 2 shows the Clause gadget. At the start of the level, there is a Lemming placed inside each Clause gadget as indicated by the Entrance.

Proposition 1. *One digger skill must be used per clause gadget if all Lemmings are to be saved.*

This follows by observing that the lemming inside each clause gadget must be freed if it is to reach the exit. Of the three skills available, digger, basher and builder, the only one which can allow a Lemming to get out of the clause gadget is the digger skill. Using the digger skill will cause the Lemming to exit by one of the three available permeable areas. More strongly, the only way out is for the trapped Lemming to be given the digger skill. Since there are m digger skills and m clauses, it follows that the only way for all Lemmings to reach the exit is if one digger skill is used for each clause. Hence there will be no digger skills remaining to be used elsewhere.



(a) (b)

Figure 3: (a) Variable gadget (b) Possible paths through the gadget

Variable Gadget. The variable gadget is shown in Figure 3. At the start of each level, there is a Lemming inside each variable gadget.

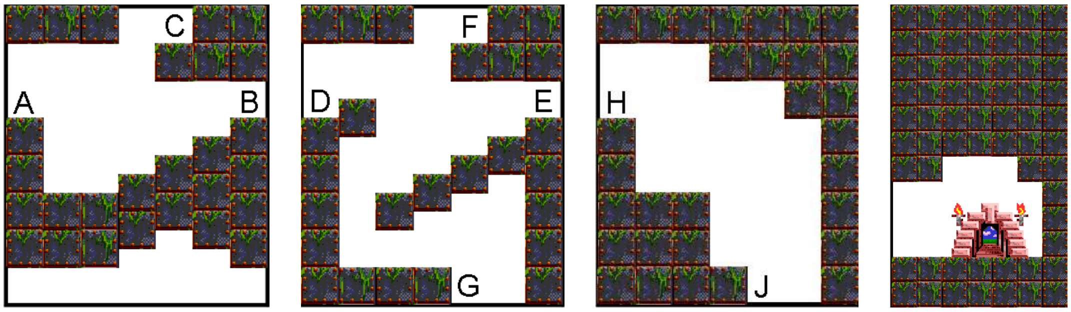
Proposition 2. *One basher and one builder must be used for each variable gadget if all Lemmings are to be saved.*

The only way to free the Lemming inside the variable gadget is to use a basher skill to break through the permeable material either to the left or to the right of the gadget. Once this has been done, then the Lemming will walk out in that direction. It will then fall to its death from the large drop unless a builder is used to build a bridge over the gap. Once a Lemming crosses the gap on a bridge, it will walk on and end safely at the bottom of the variable gadget. It will then exit on the right hand side of the gadget. Since there are n variable gadgets, and n builder and basher skills, all these skills will be consumed, one of each for each variable gadget, if all Lemmings are to be freed.

Semantics. Each variable clause is associated with one of the variables from the 3SAT instance. We shall use the location of the necessary bridge to infer a truth value of that variable. If the bridge is on the left side of the entrance, over V, then we infer that the variable is set to true; else, if the bridge is on the right side to cross over $\bar{V}$, then the variable is set to false.

Wiring. In Figure 4, three connecting gadgets are demonstrated, the Junction, Wire, and Corner gadgets. These are designed to allow lemmings to traverse them only in permitted ways. The following statements assume that no skills are deployed at any location within the gadgets.

Proposition 3.
(a) Any Lemming which enters a junction gadget at point A will exit at point B; any Lemming which enters at point C will exit at point B.



(a) (b) (c) (d)

Figure 4: (a) Junction Gadget (b) Wire gadget (c) Corner gadget (d) Accept gadget

(b) Any Lemming which enters a wire gadget at point D will exit at point E; any lemming which enters at point F will exit at point G.
(c) Any Lemming which enters a corner gadget at point H will exit at point J.
(d) Any Lemming which enters the accept gadget will reach the Exit, and successfully leave the level.

Conceptually, we will wire clause gadgets to variable gadgets, so that a literal in a clause is "wired" to the corresponding value of the variable in its clause. This means that if a Lemming exits a clause gadget by a path corresponding to some literal, say $\bar{x}$, then it will be guided by the wiring gadgets, to an entry point in the variable gadget, corresponding to $\bar{x}$.

This is achieved by the use of a "wiring block", which is an arrangement of junction, wire, and corner gadgets, which are arranged in a grid which is $3m + 3n$ gadgets wide and $2n$ gadgets tall. Each row of the wiring gadget conceptually corresponds to a value of a variable, either positive or negated. The first $3m$ columns correspond to the $3m$ literals in the clauses, and the last $3n$ columns complete the routing by forcing a Lemming "assigned" to a particular literal to enter the corresponding variable gadget.

Definition 1 (Wiring Block). *All locations in the wiring block are filled with wire gadgets, unless one of the following cases applies:*
If clause $c_{ij} = v_k$, then location $[3i + j, 2k - 1]$ is made to be a junction.
If clause $c_{ij} = \bar{v_k}$, then location $[3i + j, 2k]$ is made to be a junction.
For $k = 1$ to n, location $[3m + 3k - 2, 2k - 1]$ is made to be a corner.
For $k = 1$ to n, location $[3m + 3k - 1, 2k]$ is made to be a corner.

Proposition 4. *Any Lemming which enters the Wiring block at the top of column $[3i + j]$ will leave the Wiring block in column $[3m+3k+1]$ if $c_{ij} = v_k$, and will leave in column $[3m+3k+2]$ if $c_{ij} = \bar{v_k}$, providing no moves are made on Lemmings in the wiring block.*

Immediately above the wiring block are placed m clause gadgets in columns $1 \ldots 3m$. Below the wiring block are placed n variable gadgets in columns $3m + 1 \ldots 3m + 3n$. The

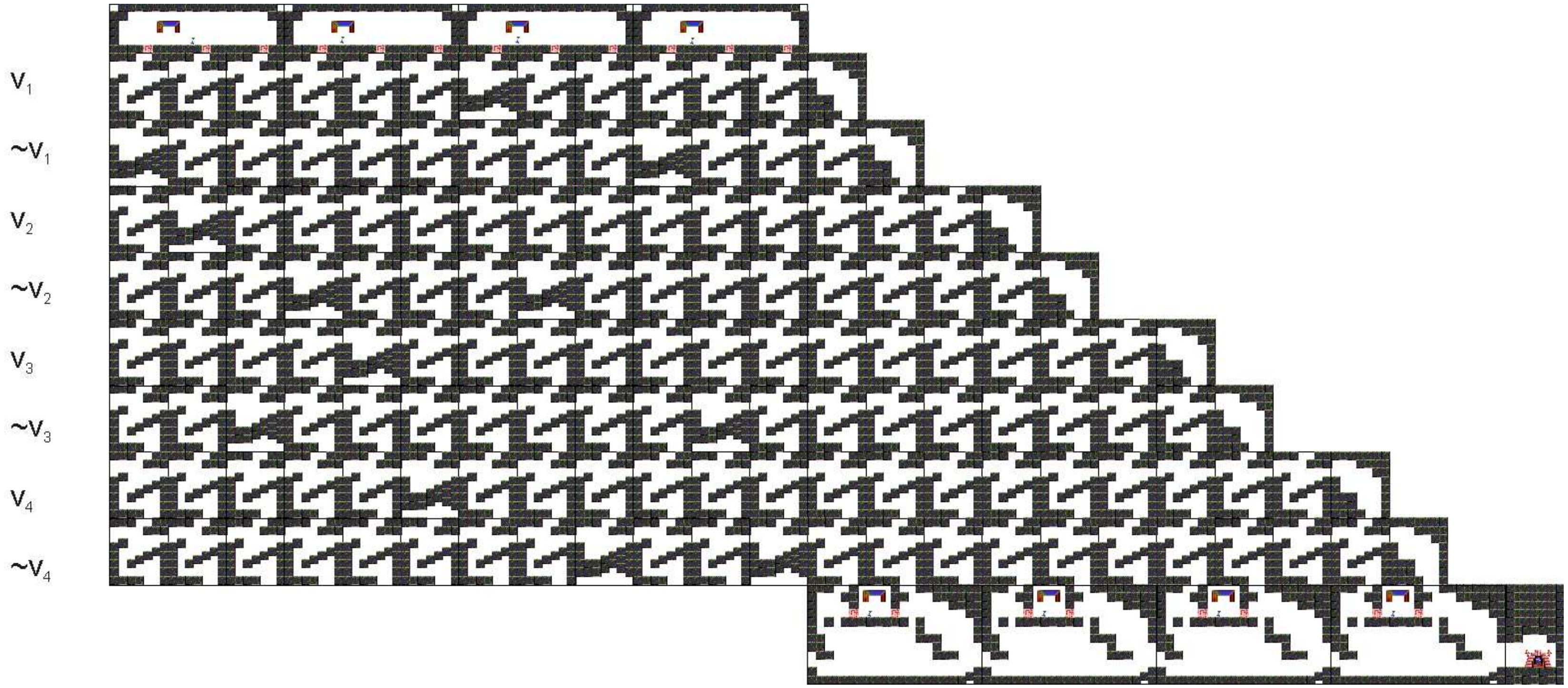


Figure 5: Lemmings level encoding the formula $(\bar{v_1} \vee v_2 \vee \bar{v_3}) \wedge (\bar{v_2} \vee v_3 \vee v_4) \wedge (v_1 \vee \bar{v_2} \vee \bar{v_4}) \wedge (\bar{v_1} \vee \bar{v_3} \vee \bar{v_4})$

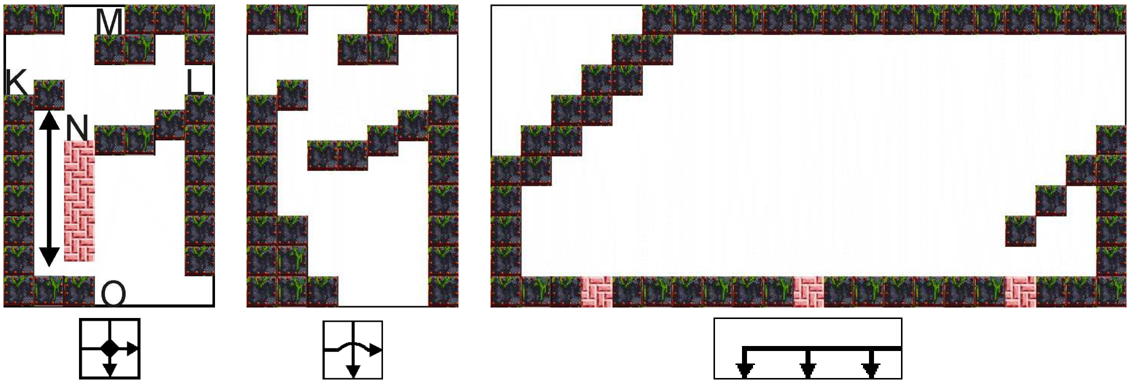
clause gadget columns $3i-2\dots 3i$ corresponds to clause c_i, and the variable gadget in columns $3m+3i-2\dots 3m+3i$. The accept gadget is placed at location $[3m+3n+1, 2n+1]$. An example layout is shown in Figure 5.

4.2 Proof

To show the NP-Hardness, we first show that (1) any satisfying assignment of 3SAT corresponds to a winning strategy for the corresponding level of LEMMINGS; then show that (2) any winning strategy for a derived level of LEMMINGS must correspond to a satisfying assignment for the 3SAT formula.

Proof of (1). For each variable, we consider its value in the satisfying assignment. If it is true, then we include in our strategy a move which makes the variable Lemming for that variable bash through the permeable material to its left, and then build a bridge over the gap at V, before walking to the exit, following the route indicated in Figure 3 (b). Else, it bashes through the permeable material to the right, and builds a bridge over $\bar{V}$, and follows the alternate route to the exit. This ensures that all variable Lemmings are saved. For each clause Lemming, we consider the clause that it corresponds to, and pick one of the literals in the clause which is satisfied—by the fact that this is a satisfying assignment, then there must be at least one. Then we make a move that makes the clause Lemming dig through the permeable material corresponding to the chosen literal: if c_{i1} is chosen, then the left material is cleared; if c_{i2} then the central material; and the right material if c_{i3} is chosen. Then, by the properties of the wiring block, that Lemming will walk to the variable gadget corresponding to that variable, and enter in the column corresponding to the value of the literal in the clause (location W if the variable is true, X if it is false). Since this literal is true, then the variable Lemming for that variable will have already built the bridge, allowing the clause Lemming to land safely on the bridge, and then walk out of the variable gadget at Z. It will then pass through all other variable gadgets, entering at Y and leaving at Z, until it reaches the accept gadget, where it can exit safely. Thus all Lemmings are saved, and all skills are used up. □

Proof of (2). This proof argues that the above type of winning strategy is the only type which can be used. In order to release all the Lemmings from their initial locations in gadgets, all the skills allocated for the level must be used. Hence, no skills can be used in the wiring block. Since every variable Lemming must climb across a bridge to exit safely, then there must be exactly one bridge covering a gap (V or $\bar{V}$) in each variable gadget, which can be interpreted as an assignment of the corresponding variable. Similarly, since each clause must be exited by the clause Lemming in exactly one location, this can be interpreted as picking a literal from each clause which is true. Since there are no spare skills, the clause Lemmings have no choice but to proceed to entries into variable gadgets, corresponding to one of the literals in their starting clause. If that literal is set to true, then the Lemming will survive, as shown above. But if that literal is false, then there can be no bridge over the corresponding gap, and so the Lemming cannot survive. Hence for



(a) (b) (c)

Figure 6: Gadgets for 1-LEMMINGS: (a) One Way (b) Wire (c) 3-split

all Lemmings to survive to the Exit, then every clause must have a literal in it which is satisfied, and so every winning strategy can be interpreted as a satisfying assignment to the original 3SAT instance. □

Combining this result with Lemma 1 shows that deciding membership of LEMMINGS is NP-Complete.

5 Hardness of Lemmings levels with a single Lemming

The above reduction is mildly unsatisfying, since it relies on placing many Lemmings in specific places at the start of the level, which is unusual in the computer game. This inspires us to consider the problem of 1-LEMMINGS, the subset of LEMMINGS where there is only a single Lemming. That is, levels where $lems = save = 1$. We will show that deciding the solvability of these levels is also NP-Hard by a reduction from 3SAT. This effectively replaces the above proof, but the construction is more complicated, and builds on structures and ideas introduced above. A greater amount of "wiring" is needed to complete the construction, as well as some new gadgets.

5.1 Gadgets for 1-LEMMINGS

The main gadgets used are shown in Figure 6. The most important new piece is the one-way. We also show small icons to represent the gadgets, these will be used to illustrate the level construction later. We assume that only digger skills are available to be used, since this will be enforced in the reduction.

Proposition 5. *If a Lemming passes through the one way gadget from M, then it will leave the gadget at O, and it cannot enter the gadget again without dying.*

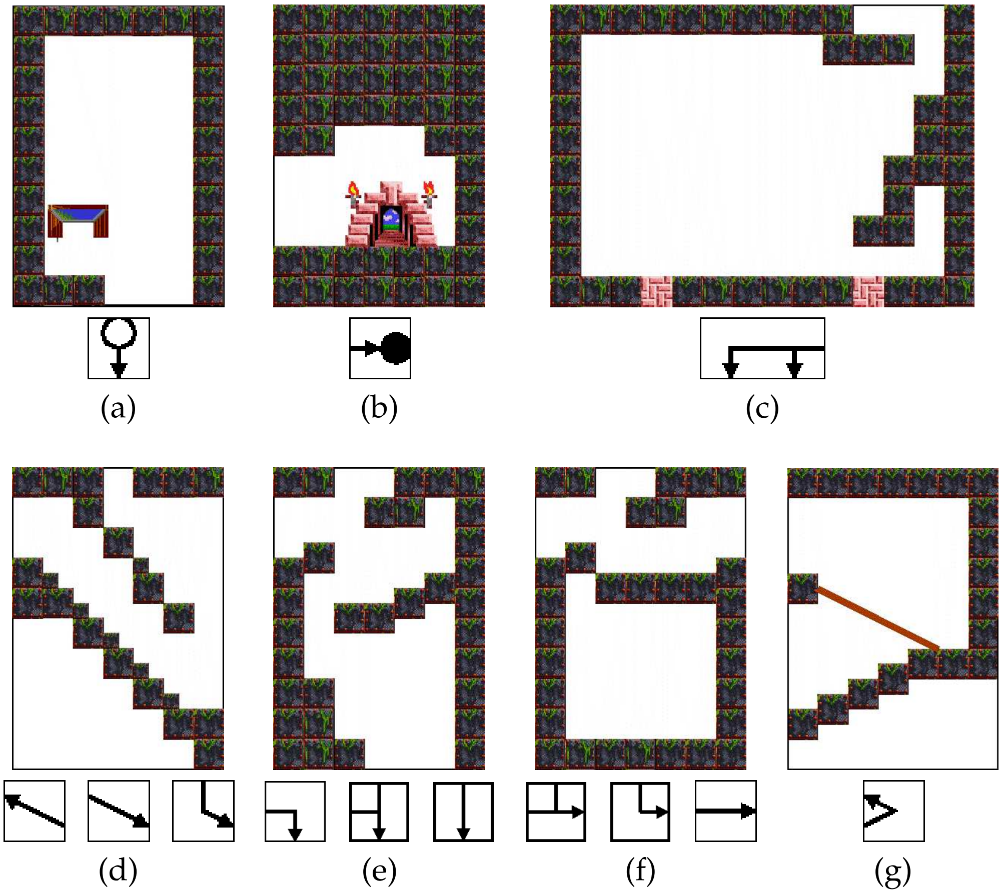


Figure 7: (a) Entry (b) Exit (c) Two way split (d) Stairs (e) Down Junction (f) Right Junction (g) Flipper

Proof. The drop, illustrated by a double headed arrow, is just too far for the lemming to fall and survive. Suppose the Lemming enters at M. Then it will reach N. If it does not dig, then it will fall down and die. So to survive, the Lemming must become a digger at N, and so safely reach the ground. It will then exit at O. If the lemming subsequently re-enters the one way gadget at K (or M or L), then it cannot avoid falling down the hole that has been created at N, and so it will die. But if the Lemming never entered at M, then on entering at K, it can safely walk across and exit at L (the Lemming could also exit at O, but this will not help the Lemming escape). In this, we use the assumption that the only skill available to Lemmings is digging. The construction using this gadget will ensure this. □

The second gadget is a new wire. It is basically the same as the previous wire gadget, except that it has been increased in height to match the height of the one-way gadget. We use it to allow the Lemming to travel downwards or horizontally, but when the Lemming is traveling horizontally it cannot exit downwards; similarly, when it is traveling down it cannot exit horizontally. The third gadget (Figure 6 (c)) is the three way split. This is very similar to the clause gadget from before: once the Lemming has entered from the right, it can only exit by choosing one of the three positions to dig through. Some further gadgets are required: these are straightforward. The Entry (Figure 7 (a)) occurs once in any layout, and marks the initial position of the Lemming, which must exit downwards. Similarly, the Exit (Figure 7 (b)) can also only occur once, and must be entered from the left. Lastly, the 2-Way Split (Figure 7 (c)) is very similar to the 3-Way split, except that it is entered from the top instead of the side.

The extra wiring pieces are also mostly straightforward. The same gadget may be used for different purposes and so may be represented by multiple icons. The stairs (Figure 7 (d)) allow the Lemming to ascend or descend half the height of the gadget. Additionally, if the Lemming enters from the top then it exits to the right. A down junction (Figure 7 (e)) ensures that whether the Lemming enters from the top or from the left, it exits from the bottom. Similarly, the right junction (Figure 7 (f)) ensures that whether the Lemming enters from the top or the left, it exits from the right. Versions of the same gadgets exist in "mirrored" form: a (left-right) mirror image of one of the above icons corresponds to the (left-right) mirror image of the gadget. Lastly, the flipper requires a little extra explanation, since it requires an additional property of the Lemmings game. The gadget includes a thin bridge (Figure 7 (g)). When the Lemming walks towards this from the left, it can step across it, but when it turns and comes back from the right then it walks up the bridge. Hence, the Lemming entering from the left at the bottom leaves from the left at the top. (Note that the same effect can be achieved by allocating just enough bridge building skills so that the Lemming can make the bridge itself after turning around; this requires a little extra argument, so for this presentation we give the version where the bridge is already in place).

5.2 Reduction from 3SAT

A similar approach is used as before: the main part of the reduction comes from building a wiring block consisting of two kinds of gadgets (wires and one-ways), with gadgets allowing the Lemming to choose a path corresponding to a satisfying assignment of the original problem (using 2-way and 3-way splitters to make the choices), and wiring to link everything together. In this extended abstract we give a brief, informal description of the construction.

Construction. Given a 3SAT instance with n variables and m clauses, the reduction constructs a level of Lemmings containing a central wiring block with $4m$ rows, each of which containing $2n$ gadgets. The Lemming traverses this grid multiple times. Initially, the Lemming passes through the grid vertically n times. Each vertical pass gives the Lemming a choice of two routes: in the ith pass, it can choose between column $2i+1$ and $2i+2$ using a 2-way split. Choosing column $2i+1$ corresponds to setting the ith variable in the instance to true, and choosing $2i+2$ sets it to false. At the bottom of each pair of columns, the two

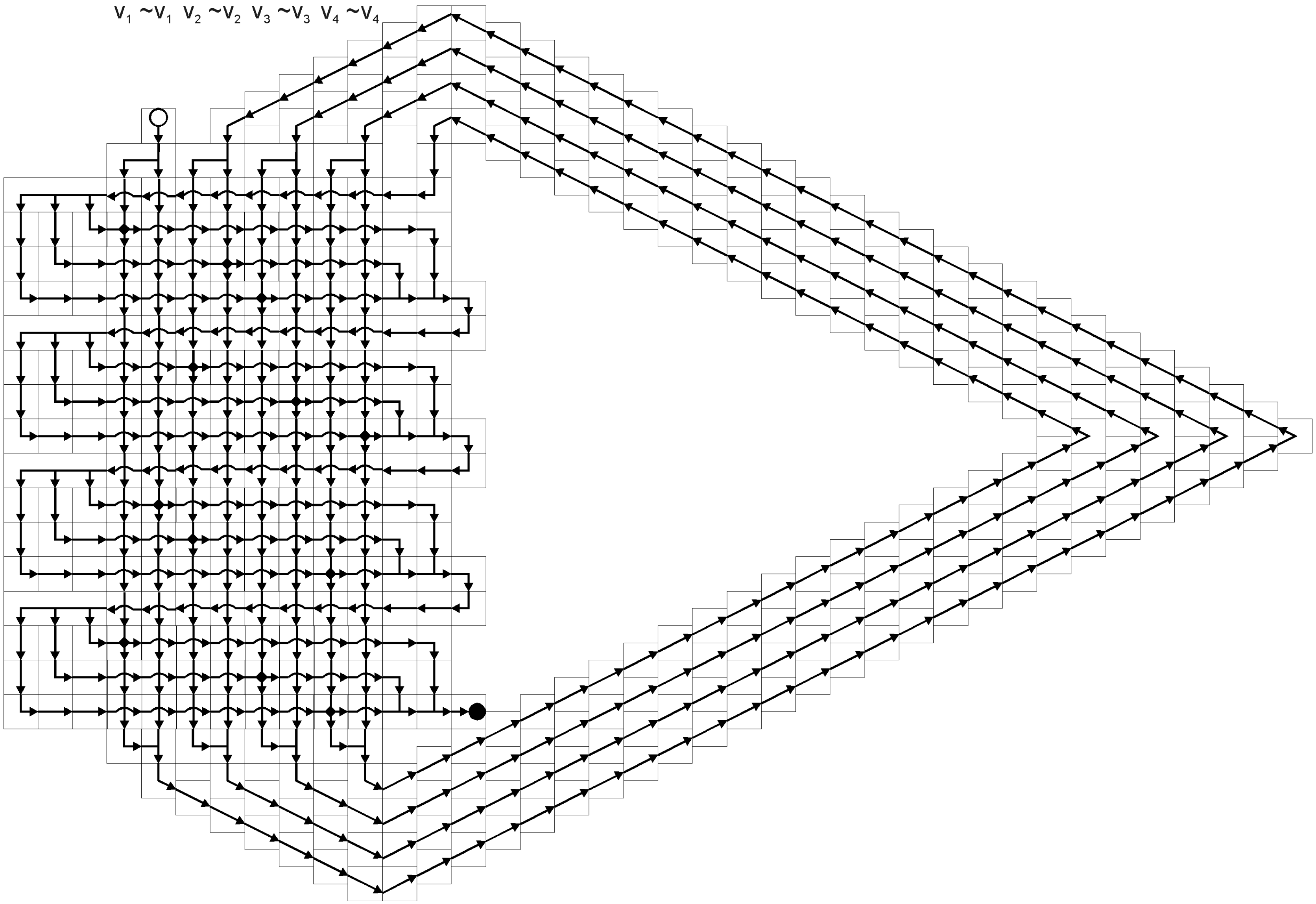


Figure 8: 1-LEMMINGS level encoding the formula $(\bar{v_1} \vee v_2 \vee \bar{v_3}) \wedge (\bar{v_2} \vee v_3 \vee v_4) \wedge (v_1 \vee \bar{v_2} \vee \bar{v_4}) \wedge (\bar{v_1} \vee \bar{v_3} \vee \bar{v_4})$

paths are merged again using junction gadgets, and then the path is looped around the block to the top, in order to set the $i + 1$st variable. After the nth variable has been set, the Lemming begins horizontal traversals of the grid. There are $4m$ rows, arranged into groups of 4 consecutive rows for each clause. The first row in each group is used to move the Lemming from the right side of the grid to the left, where it enters a 3-way split. Each exit from the split corresponds to choosing one of the literals in the clause that is satisfied. For the jth clause, this results in the Lemming walking across row $4j + 2$ if the first literal is chosen; $4j + 3$ for the second literal; and $4j + 4$ for the third. These paths are all merged on the right hand side of the wiring block, and the Lemming is directed to row $4j + 5$ for its journey to the left side again, where it will deal with the $j + 1$st clause. At the end of the mth row is the exit.

To save the Lemming, and hence complete the level, the Lemming must safely get through the wiring grid. The wiring grid is set according to the instance of 3SAT. Remember, this grid is an array of $2n$ by $4m$ gadgets. If clause $c_{ij} = v_k$, then location $[2k + 2, 4i + j + 1]$ is made to be a one-way gadget else it is a wire. If clause $c_{ij} = \bar{v_k}$, then location $[2k + 1, 4i + j + 1]$ is made to be a one-way gadget else it is a wire. For $1 \leq k \leq 2n$ and for $1 \leq i \leq m$, location $[k, 4i + 1]$ is made to be the mirror image of a wire. To complete the level of LEMMINGS, we allocate $3m$ digging skills and no other skills, and a sufficient time limit to allow the Lemming to walk around the grid in the intended manner, and get to the exit (this is linear in the size of the level). Observe that the size of the level is again linear in mn, and hence is polynomial in the input size. The same example 3SAT instance as before is illustrated under this transformation in Figure 8.

Theorem 1. *The constructed level of 1-*LEMMINGS *can be completed if and only if the corresponding instance of* 3SAT *is satisfiable.*

Proof sketch. The proof proceeds as before, in showing that a satisfying assignment to the variables corresponds to a way to save the Lemming, and that if the Lemming is saved then this corresponds to a satisfying assignment to the 3SAT instance. The crucial part is the use of the one-way gadgets. If we set variable v_k to be true, then we pass vertically through all the one-way gadgets corresponding to literals in clauses containing $\bar{v_k}$. This means that in the pass through any of those clauses, we cannot select $\bar{v_k}$ as the satisfied variable, since this path is now closed to us. A symmetric argument applies for setting v_k to false. We assume that any clauses containing both a variable and its complement are removed: such a clause is trivially satisfiable in any assignment, and so does not affect the satisfiability of the original instance. So we can only pass through rows where the one-way in that row has not been used already (observe that each row in the wiring block contains exactly one one-way gadget). This gives a one-to-one correspondence between paths through the block and assignments to the variables (in the full proof, we must argue that there is no way to avoid or circumvent the intended path). It therefore follows that the Lemming can be saved, and the level is completable, if and only if the instance is satisfiable. □

Since this reduction can be computed in time polynomial in the size of the 3SAT instance, we can conclude that 1-LEMMINGSis also NP-Complete.

Corollary 1. *It is NP-Hard to approximate the number of Lemmings that can be saved in levels of* LEMMINGS*.*

Suppose we could approximate the number of Lemmings saved to any factor. Then, given a level of 1-LEMMINGS, we could use this approximation to tell us whether the Lemming could be saved or not: if the answer is non-zero, then it can be saved, while if it was zero, then it cannot be save. Hence, the answer cannot be approximated without solving the original problem, and so it is NP-Hard to approximate.

6 Efficiently Decidable Versions of Lemmings

Given that the general game of Lemmings is NP-Hard, it is natural to ask under what restrictions is the game more easily decidable? Under restrictions on the skills available, then the question is efficiently decidable (that is, in P).

Theorem 2. LEMMINGS *with only permanent skills available is in P.*

Proof. The theorem follows because a Lemming given a permanent skill cannot affect the path of any other Lemming (unless they deploy some temporary or semi-permanent skill). Further, they cannot affect the level around them. So, the level can be modeled by a directed graph where each node is a tuple $(loc, dir, climb, float)$. The four components represent the current location of a Lemming (loc), the direction that it is facing (dir), whether or not it is a climber ($climb$), and whether or not it is a floater ($float$). Additionally, there is a distinguished node representing a dead Lemming, which has no outgoing edges. The edges in the graph represent the next move made by a Lemming of that type: if a Lemming is walking to the right, then an edge links $(loc, right, climb, float)$ to $(loc', right, climb, float)$, where loc' is the location reached by walking to the right from loc. If there is a wall to the right, then the edge instead links to $(loc, left, climb, float)$ if $climb$ is false; if $climb$ is true, then it will link to $(loc', right, true, float)$ where loc' is the location reached by climbing one unit up the wall. If a Lemming will fall too far and die (since it has just walked off a high ledge), then an edge links that node to the "dead" node. Additionally, at each node there are edges corresponding to giving the Lemming a new skill: for every node $(loc, dir, false, float)$ there is an edge to $(loc, dir, true, float)$, representing giving a Lemming the climbing skill; and for every node $(loc, dir, climb, false)$, there is an edge to $(loc, dir, climb, true)$, representing giving the climbing skill. Note that each node in the graph has degree at most three: if no action is taken, then the next move of the Lemming is determined; else, the Lemming can be given at most two skills, which take it to a new node. Also observe that once a skill is given, it is never lost: the graph is partitioned into four parts, consisting of nodes with the pair of values $(climb, float)$ taking on the four different combinations of true and false. There are edges between $(false, false)$ and $(true, false)$ and $(false, true)$, and between these two and $(true, true)$, and all other edges (apart from those to "dead") are internal. Lastly, note that the graph is linear in the size of the board, since there are at most eight nodes for each location (eight nodes representing all combinations of dir, $climb$ and $float$).

To decide whether the level is solvable, we perform the following operations on the graph. First, for each starting position, determine if there is a path to the exit which stays within the $(false, false)$ cluster. This is straightforward, since there is only one outgoing edge for each node in this cluster. Then remove the Lemmings which reach the exit unaided from consideration. If the number of such Lemmings, x, surpasses the number of Lemmings that must be saved on that level, $save$, then it is trivially solvable. Else, consider the remaining Lemmings, and the two subgraphs induced by removing nodes with $climb$ set to true, and by removing nodes with $float$ set to true. The two subgraphs model strategies where Lemmings are given at most one of the climbing and floating skills. In each subgraph, for each starting position of the remaining Lemmings, compute which Lemmings have a path to the exit; this is straightforward to do in time polynomial in the size of the graph. Let the number of Lemmings which have a path in both graphs be d, let the number of Lemmings which have a path to the exit only in the subgraph with $float$ set false be c, and the number of Lemmings in the subgraph with $climb$ false be f. Denote by C the number of climber skills available, and by F the number of floater skills available. This leaves a set of Lemmings that cannot be saved by being given only one skill. From this set, compute the number of Lemmings, e, which can get to the exit if they are given both floater and climber skills. The following table defines some derived variables:

Variable	**Value**	**Meaning**
C'	$C - \min(c, C)$	Climber skills remaining after saving up to c climbers
F'	$F - \min(f, F)$	Floater skills remaining after saving up to f floaters
d'	$\max(d - \lvert C' - F' \rvert, 0)$	Number of Climber and Floater skills remaining after balancing the number of remaining skills.
D'	$\min(C', F') - \lceil \frac{d'}{2} \rceil$	Number of both Climber and Floater skills remaining after saving up to d Lemmings with climber and floater skills.

Having computed all these quantities, the level is solvable if and only if

$$x + \min(c, C) + \min(f, F) + \min(d, C' + F') + \min(e, D') \geq save$$

Each term represents the number of Lemmings saved by, respectively, no intervention, giving climber skills, giving floater skills, giving either climber or floater skills, and giving both climber and floater skills. All of these quantities can be computed in time polynomial in the size of the input, hence this restricted version of Lemmings is decidable in polynomial time. □

It is worth noting that although levels of Lemmings with only floaters and climbers do exist, they tend to be early, "training levels" in the games. [8] It remains open to consider other restrictions based on the available skills.

7 Prior Work

There has been some academic study of the game of Lemmings in the past. One notable example is an unpublished report by John McCarthy [McC94]. There, he proposes that the game should be studied as a new "drosophila for Artificial Intelligence". That is, solving levels of Lemmings should be used as a challenge for Artificial Intelligence systems. This comes from the complexity in the game, combined with the predictability (for most purposes, the response of the game is completely deterministic and hence completely predictable). The results in this paper put this in an interesting context: the fact that deciding whether a level is completable is NP-Complete suggests that it will be a big challenge for any system to guarantee to solve any level. Any system claiming to be able to solve arbitrary Lemmings levels must be able to solve arbitrary NP-Complete problems, and consequently is likely to take worst case time exponential in the size of its input.

Other games have received more attention. Most related to this paper is the study of the game Tetris [DHLN03], which showed that the game was NP-Complete by transformation from 3-Partition. Other computer games that have been studied include Minesweeper [Kay00], Reflections [Kem], Sokoban [DZ99, Cul98, DH01], and Lunar Lockout [HLH03][9]. Mostly, these show that the games considered are NP-Hard or NP-Complete; in some cases they are also shown to be PSPACE Complete. For more details of this area, see the survey by Demaine [Dem01].

In addition to generalizing the games to allow for arbitrarily large game boards, certain other generalizations are necessary to prove hardness of some of the above—Tetris requires all pieces to be specified in advance, MineSweeper is for a decision problem that does not occur naturally in the game, Lunar Lockout requires a new type of game piece which is not permitted to move. In this paper, only constructions permitted in the regular game of Lemmings have been used, with no additions or restrictions, to answer a question that many players ask themselves when confronted with a new level.

Although in the spirit of puzzle games like Sokoban and Lunar Lockout, Lemmings has a somewhat different flavor, since there are (multiple) independently moving objects to be concerned with. In some ways, this made the first reduction a little simpler, since it is easier to set up a correspondence between n objects in the game and n objects in the problem to be reduced from. But there was extra challenge in ensuring that Lemmings stick to their assigned roles, and do not interact with each other in unintended ways, and in showing that the game was still hard with only one Lemming.

[8] One exception is the level of Lemtris, which occurs late in the game of Lemmings Revolution. Here Lemmings emerge from one of four locations, and face a perilous drop if they are not given the floater skills. It is interesting to note that although the game of Tetris is known to be NP-Complete [DHLN03], this particular level is trivially possible.

[9] Play online at `http://www.johnrausch.com/PuzzleWorld/app/lunar_lockout/lunar_lockout.htm`

8 Conclusions

A natural extension of this work is to consider other popular (computer) games and to characterize their computational complexity (see [Dem01] for some examples). But remaining with Lemmings, there are several extensions and variations that remain to be studied completely.

There have been several sequels to Lemmings, most of which have sought to improve on the relative simplicity of the original by adding many and varied different traps, skills, and objects in the levels. Still, so long as the key skills and types of material are available, then we can still construct transformed instances of 3SAT, and show that the game is NP-Complete.

One can imagine variations in the rules which would require a new proof. For example, consider Lemmings played under strong gravity: here, Lemmings cannot increase their vertical position, but can only walk along horizontally or fall downwards (then builders would build horizontally, say). In this scenario, the game can still be shown to be NP-Complete in the general case, by some small modifications of the gadgets used. Such a transformation could serve as the template for hardness proofs for other games, such as those based on directing inanimate objects which move under gravity and momentum only (typically marbles) towards a collection point. It is not clear if the same holds for a single object (1-LEMMINGS under heavy gravity). Another restriction would be to consider a one-dimensional version of the game. It seems intuitive that such a restriction would be tractable, but this depends on how the rules are applied in one-dimension, since there is no immediate natural version. The complexity of the game under other restrictions as to the nature of skills available or the number of Lemmings remains open. It also remains a possibility that LEMMINGS could be shown to be P-Space complete, encoding stored bits using the presence or absence of bridges in the level. In this version it may be necessary to allow an unbounded quantity of certain skills, such as builders and bashers.

Acknowledgments

I owe thanks to Mike Paterson for a discussion during which we formulated the construction for showing that 1-LEMMINGS NP-Hard. Martin Demaine suggested looking for efficiently decidable versions of the problem. I would especially like to thank my family, for the many times when as a teenager my dinner got cold as I struggled to complete a level of Lemmings. I can now claim that this was done in the name of research and scientific discovery.

References

[Cul98] J. Culberson. Sokoban is PSPACE-complete. In *Proceedings of the International Conference on Fun with Algorithms (FUN-98)*, pages 65–76, 1998.

[Dem01] E. D. Demaine. Playing games with algorithms: algorithmic combinatorial game theory. In *Mathematical Foundations of Computer Science (MFCS)*, volume 2136 of *LNCS*, pages 18–32, 2001.

[DH01] E. D. Demaine and M. Hoffmann. Pushing blocks is np-complete for non-crossing solution paths. In *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG)*, pages 65–68, 2001.

[DHLN03] E. Demaine, S. Hohenberger, and D. Liben-Nowell. Tetris is hard, even to approximate. In *COCOON: Annual International Conference on Computing and Combinatorics*, 2003.

[DZ99] D. Dor and U. Zwick. SOKOBAN and other motion planning problems. *CGTA: Computational Geometry: Theory and Applications*, 13, 1999.

[HLH03] J. R. Hartline and R. Libeskind-Hadas. The computational complexity of motion planning. *SIAM Review*, 45(3):543–557, 2003.

[Kay00] R. W. Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer*, 2000.

[Kem] D. Kempe. On the complexity of the reflections game. `http://www.cs.washington.edu/homes/kempe/publications/reflections.p`

[McC94] J. McCarthy. Partial formalizations and the lemmings game, 1994.