CS 361 Final Project

4613 and 5A17

November 2019

1 Introduction

In this project, we will establish a lower bound on the computational complexity of the game *Popils* (a 1990s-era video game for the Sega Game Gear). The game features levels with destructible blocks in which the player character attempts to climb and rearrange the geometry of the level to reach the princess. There are a number of other mechanics, including warp portals, enemies, spikes, ladders, and indestructible blocks (see Fig. 1), but we attempt to use a minimal subset of these features in our proof. In order to establish the NP-completeness of *Popils*, we built gadgets which simulate the clauses and variables in an arbitrary instance of the 3SAT problem.

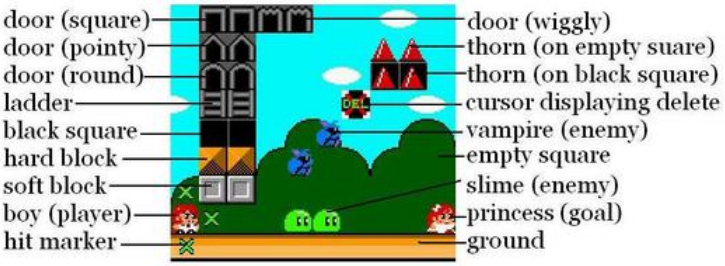


Figure 1: A compact diagram showing all the mechanics of *Popils* [4]. Of these, we only use the soft and hard blocks, ladders, black squares, player character, and goal (princess).

2 NP-Completeness

2.1 NP membership

We first need to define a certificate and a verifier for a *Popils* level. A certificate will consist of a series of legal actions, where each action can be either a movement or a block destruction. To verify the certificate, a verifier will step through each action, updating the *Popils* level as it goes. At each step, the

verifier will reject if the action is illegal (movement into a wall or destruction of an indestructible block). When it reaches the end of the certificate, the verifier accepts if the hero and princess occupy adjacent squares, and rejects if not.

Now that we have a certificate and verifier, we need to show that no certificate can exceed polynomial length in the size of the level. Let n be the number of squares in the level. Then, within the first n^2 actions in a certificate, a state must be reached that is a duplicate of a previous state. Without destroying any blocks, at most n unique states can be accessed (by moving to one of the n positions in the level). To avoid reaching a duplicate state, the player must then destroy a block. This opens up n additional states, once again corresponding to every position in the level. Thus the player can avoid duplicate states for the first (# of blocks) $\times$ (# of positions) actions. This is at most $n \times n = n^2$ actions, so a certificate of length greater than n^2 must have a duplicate state. If a satisfying certificate reaches a duplicate state, then another satisfying certificate with fewer steps can be generated by removing every action between the duplicates. If we repeatedly apply this length reduction, we eventually find a satisfying certificate of length less than n^2, so any solvable instance of *Popils* must have a certificate of polynomial length. Therefore, *Popils* is in NP.

2.2 NP-hardness

Given an instance of 3SAT ϕ with clauses $c_1, \ldots, c_m$ and variables $x_1, \ldots, x_n$, we construct a *Popils* level as follows.

The level consists of three main components, which are from bottom to top,

1. the truth assignments for each variable in the expression
2. a gadget for each clause of size $6 \times (2n + 3)$ blocks (Fig. 6)
3. a small "goal" section which sits at the top of the level, ensuring that the princess can only be reached by traversing the rest of the level.

Figure 2: Legend for our representation of a *Popils* level. Note that black support squares have been changed to white for visual clarity.

Fig. 3 shows a generic truth assignment gadget for n variables. In the constructed level the player always begins in the leftmost space that can be occupied. We represent the player with a red square in the figures. The gray (soft) blocks may be broken by headbutting them, which causes the entire column of blocks above them to drop down by one space. An unbroken block denotes a

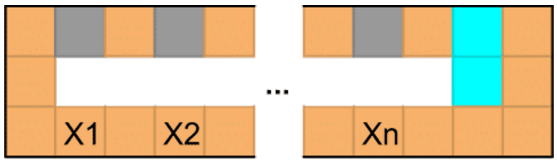


Figure 3: Subsection of a level for assigning truth values to n variables

"false" assignment for that variable while a broken block represents a "true" assignment. At any time, the player may walk to the right and climb the ladders to try to traverse the rest of the level, the layout of which will change depending on the chosen variable assignment. The player may return to the bottom of the level at any time and change the truth assignment, but this puts them back at the beginning of the stage so there is no way to traverse part of the level and "cheat" by altering the layout partway through to skip a section. Additionally, blocks may not be restored once broken, so if the player wishes to change an assignment from true to false, they must reload the level entirely.

The main portion of the level features variable gadgets nested inside clause gadgets, with each variable vertically aligned above its truth assignment block. Within each clause, every variable will be represented by one of three block configurations, as seen in Fig. 5. The ladders for a negated variable will enable the player to climb through the clause by default, but if the assignment block below is broken for that variable (meaning it will be set to "true") the blocks will fall and be unhelpful. Similarly, the non-negated variable gadget is useless unless it is made to drop down a space by a true assignment. The gadget for variables which are missing from the clause have two separated ladders so as to be of no consequence regardless of their assignment. The ladders ensure that the player won't be blocked from reaching later variables, but they do not help the player climb vertically.

Variables are separated from one another by a column of hard blocks and support blocks to ensure that they are properly isolated and the player can't move directly from one set of ladders to another. Every clause can be entered and exited only from the right side, which has a set of ladders. Clause gadgets are stacked vertically in ascending order (so c_i sits on top of c_{i-1}). The player can only reach the exit ladder for a clause if at least one variable has a truth value that allows vertical traversal of its ladders. This reflects the nature of the Boolean expression, where each variable within a clause can independently make the whole clause true. A fully constructed clause gadget can be seen in Fig. 6.

Finally, the princess sits atop the last clause, surrounded on all sides by unbreakable blocks except for the platform she is standing on, which is breakable. The player can reach that block if and only if they traversed every prior clause, as the only way to reach the top of the level is with the provided ladders. The player may headbutt the breakable block to shatter it, and stand on top of the topmost ladder once that space has been cleared. At that point, the player will

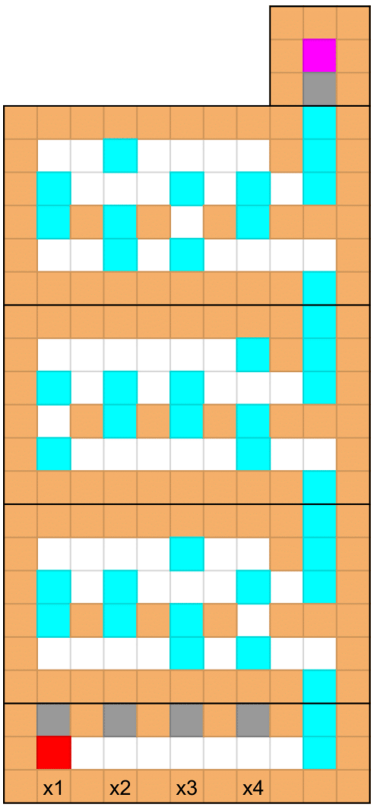


Figure 4: This is the representation of ϕ before we assign any truth values to its variables. All of the variables are false by default because their blocks are unbroken.

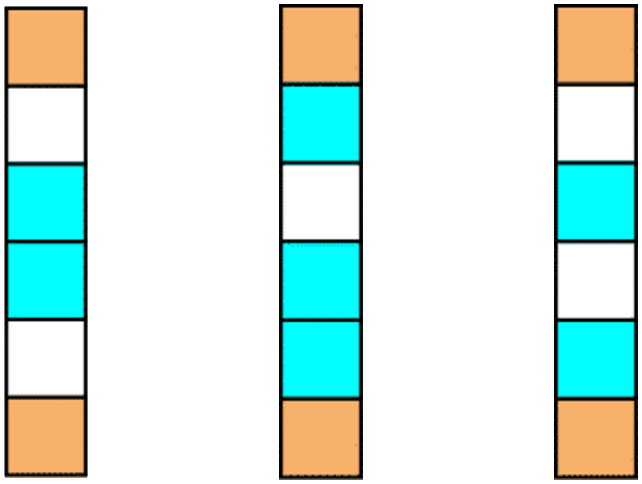
be adjacent to the princess, which the verifier will accept.

The goal section, as well as an example of the full level construction for the Boolean expression

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee x_4)$$

is found in Fig. 4.

With this construction, it is clear that the level is traversable (i.e. the goal is reachable) if and only if the Boolean expression which generated it is satisfied. Furthermore, these levels require a precise number of blocks and scale linearly with the number of variables and clauses in the expression. Hence, the 3SAT problem is reducible to *Popils* in polynomial time. Therefore, *Popils* is NP-Hard, and since we also showed it is a member of NP, it is by definition NP-complete.



(a) Non-negated variable (b) Negated variable (c) Missing variable

Figure 5: These gadgets are made from three types of material. The tan squares are "hard blocks" (which are unbreakable) the white squares are "black squares" (which support other blocks), and the cyan squares are "ladders" (which are climbable). See Fig. 1 for in-game representations of these materials.

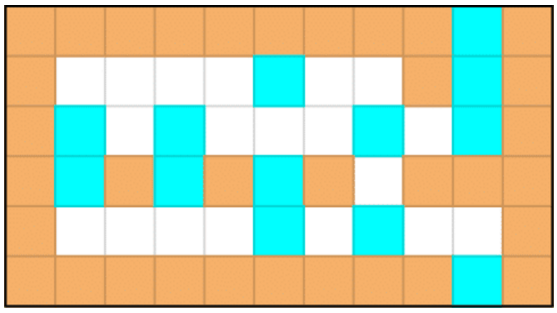
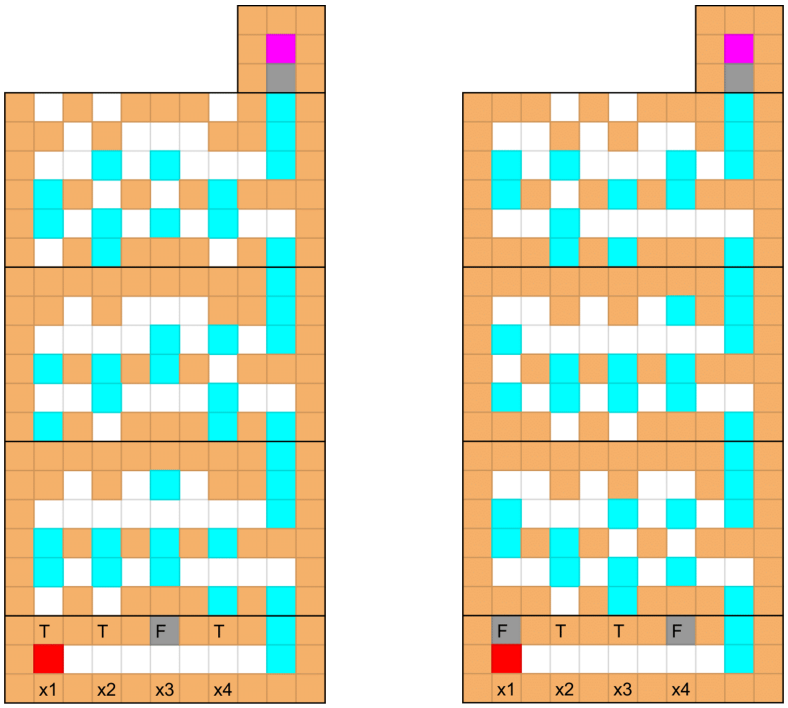


Figure 6: A gadget representing a single clause in a Boolean expression. This clause is $(x_1 \vee x_2 \vee x_3)$ and is part of an expression containing four variables.



(a) A satisfying assignment for ϕ (b) A non-satisfying assignment for ϕ

Figure 7: The expression is satisfied if and only if the level becomes traversable with that assignment (i.e. the player is able to reach the princess). Note that there is no way for the player to climb through the subsection of the level for the third (uppermost) clause in the right-hand figure because the expression isn't satisfied.

3 Further Work

In general, deeper exploration of this topic would involve further restrictions on the number of employed mechanics and the types of elements in the constructed level. For example, our construction relies heavily on the gravity mechanic to link the state of a variable across clauses. While this is a core mechanic to the game, it may be possible to prove NP-completeness with a level of finite height.

Conversely, there are many mechanics that we did not utilize which could increase the complexity of the full game. At minimum, warp doors, enemies, and the movement of the princess are all promising avenues for further research.

An alternative direction could involve a different means of forming the reduction. Rather than making an initial truth value assignment to each variable, we could create (and then chain together) AND and OR gadgets using the mechanics and block types from the game. This would lend itself particularly well to accommodating constraints on the dimensions of the level as our current model necessarily scales with the size of the Boolean expression, which can have any number of clauses.

References

[1] *Magical Puzzle Popils*. URL: `https://segaretro.org/Magical_Puzzle_Popils`. (game info).

[2] *Magical Puzzle Popils (Game Gear) Playthrough*. URL: `https://www.youtube.com/watch?v=2c5qgU0tvlk`. (example playthrough).

[3] *Play Magical Puzzle Popils - Sega Game Gear online*. URL: `http://game-oldies.com/play-online/magical-puzzle-popils-sega-game-gear`. (online emulator).

[4] *Popils*. URL: `https://en.wikipedia.org/wiki/Popils`. (game mechanics).