Super Mario

Martin Ivanov
ETH Zürich
Super Mario Crash Course

1. Goal
2. Basic Enemies
   - Goomba
   - Koopa Troopas
   - Piranha Plant
3. Power Ups
   - Super Mushroom
   - Fire Flower
   - Super Start
   - Coins
Reductions

\[ Y \leq_P X \]

• \( Y \) is polynomial-time reducible to \( X \)
• \( X \) is at least as hard as \( Y \)
• if \( X \) can be solved in polynomial time, then \( Y \) can be solved in polynomial time
• if \( Y \) can not be solved in polynomial time, then \( X \) cannot be solved in polynomial time
Computational Complexity Overview

1. **P**
   - multiplication
   - sorting

2. **NP**
   - integer factoring

3. **NP-complete**
   - sudoku
   - satisfiability

4. **NP-Hard**
   - traveling salesman

5. **PSPACE**
   - quantified boolean formulas
Satisfiability

- **Literal**: Boolean variable or its negation \( x_i \text{ or } \overline{x_i} \)
- **Clause**: A disjunction of literals \( C_j = x_1 \lor \overline{x_2} \lor x_3 \)
- **Conjunction**: \( C_1 \land C_2 \land C_3 \land C_4 \)

- **SAT** – given a conjunction of clauses, does it satisfy a truth assignment?
- **3-SAT** – special case of SAT where each clause contains exactly 3 literals

\[
\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)
\]

**Yes instance**: \( x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false} \)
Gadgets

- partial instances of problem $X$ that are used to “simulate” objects in problem $Y$
- used to construct reductions from one problem to another

- **Start Gadget**
  - can be used to initialize a specific state

Figure 8: Left: Start gadget for Super Mario Bros. Right: The item block contains a Super Mushroom
Finish Gadget

- accessible only if the player is in the desired state
Framework for NP-hardness

- The framework reduces from 3-SAT
  - allowed
  - not allowed
Variable Gadget

- must force the player to choose one of two paths
- entering from one literal does not allow traversal back into the negation of the literal

Figure 10: Variable gadget for Super Mario Bros.
Clause Gadget

- accessible from the literal paths
- the player can perform some action that “unlocks” the gadget
- the check path traverses every Clause Gadget in sequence

Figure 11: Clause gadget for Super Mario Bros.
Crossover Gadget

- must allow traversal via two passages that cross each other
- no leakage can occur from the vertical to the horizontal path
Super Mario NP-hardness

- 3-SAT $\leq_P$ MARIO

- **Theorem** It is NP-hard to decide whether the goal is reachable from the start of a stage in generalized Super Mario Bros.

- Related Work
  - The Legend of Zelda
  - Donkey Kong Country
  - Metroid
  - Pokemon
Nintendo Entertainment System

- 8-bit processor → 00001111
- running at 1.79 MHz
- 2048 bytes of general purpose RAM
- fixed memory locations used for all the critical game facts

Figure 1: 2048 bytes, a 64x32 image.
Automating NES games

- video screen, sound effects are ignored
- notion of winning $\rightarrow$ value going up
- lexicographic order

\[a < aa < aaa < ab < aba.\]

- World 1-2 $\rightarrow$ \(p=1, q=2\)
- World 2-1 $\rightarrow$ \(p=2, q=1\)
- \((p_1,q_1) < (p_2,q_2)\) if \(p_1=p_2\) and \(q_1 < q_2\)
- OR if \(p_1 < p_2\)
learnfun

- the objective function is deduced from the player’s inputs
- learnfun watches you play and figures out what it means to win
- find series of byte locations in memory that go up according to the lexicographic ordering
• uses the gained knowledge from learnfun to play the game
• finds the optimal sequence of inputs to satisfy the objective function

• Greedy Approach
  ▪ search space is $2^8$ different inputs, pick the best step
  ▪ single input rarely affects your progress
Motifs

- look 10 frames into the future
- use the best scoring 10-keystroke motif
- still bad at avoiding enemies and jumps
Time Travel

• pick 40 random futures (50-800 frames)
• pick (based on weight) which one to replay for the next 10 frames
• extend futures with random motifs when they become too short
• worst futures are replaced with new random futures
• reach consistency → do combinations that worked and are likely to work again
Backtracking

- local maximum
- improvement – save a checkpoint
- occasionally reset to the beginning and generate some other replacement futures
- if the original sequence is the best, backtracking does nothing
Performance

• 1 hour to calculate 1000 frames of output = 16 sec of gameplay
• most of the time is spent emulating NES code

• MARIONET
  ▪ network version of playfun
  ▪ utilizes multiple cores and potentially multiple computers to score futures
  ▪ master/slave

Figure 8: Utilization with 12 helpers and one master on a 6-core (12 hyperthreads) processor. Looks good. Keeps the bedroom warm.
Results

• Super Mario
• Pac-Man
• Bubble Bobble
• Tetris
Future Work

- parameter reduction
- unsupervised learning
- better backtracking
- multiple players, multiple games
Conclusion

• Nintendo Games are awesome and fun!
• can be used in serious topics
• produce real and interesting results

Thank you for your attention