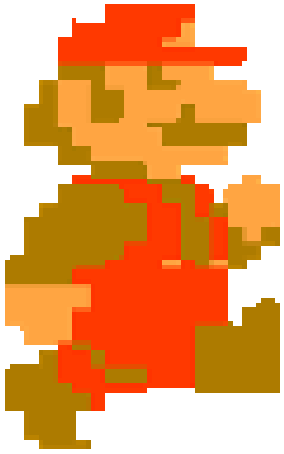

Optimizing Mario Adventures in a Constrained Environment

Sanyam Jain

September 24, 2023

0.1 Objective

This Constrained Optimisation Problem (COP) includes an environment of Super Mario Bros. where our agent (Mario) travels several worlds and stages to find princess! However, its goal is to collect as many coins, energy boosters, power-ups, life, and avoid obstacles. To formalise, for our project, Mario will travel worlds but its task is to travel as maximum distances, make as less movements, and make minimum moves in order to collect coins and finish the level. In simpler words, **the objective is to guide Mario within a game environment to maximize the number of collected coins and maximising the distance traveled in order to finish level**. This optimization problem is subject to constraints that include completing the task within minimum time, with the minimum number of rebirths (deaths), and with as few actions (movements) as possible. We will further formalise objective function, inputs, outputs, models and complexity analysis in following sections.

0.2 Inputs

0.2.1 Game Environment (Game State)

The game state in the Super Mario Bros. environment (Kauten (2018)) is a data structure that encapsulates all the relevant information about the current state of the game. Following are the key components that comprises the Game Environment:

- Mario's Position and Status: This includes Mario's (player's) position on the screen, which is often represented as a pair of coordinates (e.g., x, y). Additionally, Mario's status information is included, such as his current size (small, big, or with a power-up), whether he is jumping or standing, and his animation frame.
- Enemy Positions and States: Information about the positions and states of enemies in the game. This may include details about the type of enemy, its position, its movement pattern, and whether it is alive or defeated.
- Object Positions: The positions of various in-game objects, such as coins, power-ups (e.g., mushrooms, fire flowers), platforms, and obstacles. These positions are essential for Mario to interact with the game world.

- Level Map: A representation of the game level's layout, often in the form of a grid or a graph. This map defines the locations of walls, platforms, pits, and other level features. Mario's position and the positions of objects and enemies are typically tracked relative to this map.
- Score and Statistics: Information about the player's current score, the number of coins collected, and other relevant statistics like remaining lives and time. The score may be a cumulative value that increases as Mario collects coins or defeats enemies.
- Game Status: Details about the overall game status, such as whether Mario has completed the level, whether he has lost a life (e.g., through an enemy collision or falling into a pit), and whether the game has ended due to a win or loss condition.
- Game Configuration: Some game state representations include configuration settings and parameters, such as time limits, maximum lives, and other game-specific rules.

The game environment is also described in the Table 1 as key elements.

Table 1: Mario Game Dictionary

Key	**Type**	**Description**
coins	int	The number of collected coins
flag_get	bool	True if Mario reached a flag or ax
life	int	The number of lives left, i.e., {3, 2, 1}
score	int	The cumulative in-game score
stage	int	The current stage, i.e., {1, ..., 4}
status	str	Mario's status, i.e., {'small', 'tall', 'fireball'}
time	int	The time left on the clock
world	int	The current world, i.e., {1, ..., 8}
x_pos	int	Mario's x position in the stage (from the left)
y_pos	int	Mario's y position in the stage (from the bottom)

0.2.2 Time Constraint

Mario must complete the task within the specified time limit, denoted as T. The total time spent should not exceed this limit:

$$\sum_{i=1}^{n} \text{time}_i \leq T$$

Where:

n is the total number of movements or actions Mario takes,
time_i is the time associated with the i-th action.

0.2.3 Rebirth Constraint

Mario's rebirths are subject to a maximum limit, denoted as R. The total number of rebirths should not exceed this limit:

$$\sum_{i=1}^{m} \text{rebirth}_i \leq R$$

Where:

m is the total number of rebirth instances,
rebirth_i is a binary variable indicating whether Mario is reborn in instance i.

0.2.4 Moves Constraint

The number of movements Mario can perform is constrained by a maximum value, denoted as A. The total number of moves should not exceed this limit:

$$\sum_{i=1}^{n} \text{moves}_i \leq A$$

Where:

n is the total number of movements Mario takes,
action_i is the number of moves or actions associated with the i-th movement.

We also add a constraint of moves to check which decides how many moves without increase in fitness until termination or catastrophic death.

0.3 Model Definition

Based on the availability of input, output and model, there is a classification for Optimisation, Modelling and Simulation (Section 1.1, Introduction to Evolutionary Computing, A.E. Eiben, J.E. Smith). However, we already know this is a COP because we have output specified as the best path that Mario traverses such that it is travels more and collects maximum coins in

its adventure, keeping with constraints of time, actions or movement done, etc. So the output is specified however we do not know exactly what input combinations will give this. Further, for a given instance of environment, we can always know (using a formula or calculations) how many coins it has collected so far at the moment, distance traveled, movements or actions performed, number of rebirths, and time elapsed so far at the moment.

0.4 Objective Function

The objective of this Constrained Optimization Problem (COP) in the Super Mario Bros. environment is to maximize the total value of collected coins and maximising the total distance traveled in order to finish the level faster. The optimization problem is subject to the constraints. These constraints ensure that Mario completes the task within the specified time, rebirths within the limit, and performs actions or moves within the maximum allowed moves, while seeking to maximize the total coin value collected.

Maximize the total coin value collected and maximising the total distance traveled:

$$\begin{aligned} \text{Maximize:} \quad \text{fitness} = \sum_{i=1}^{n} (C_i \cdot \text{coins}_i + D_i \cdot \text{distance}_i) \\ - P \cdot (300 - \text{time}_i) \end{aligned}$$

Where:

- C_i represents the coin reward for the i-th step.
- coins_i represents the number of coins collected at the i-th step.
- D_i represents the distance reward for the i-th step.
- distance_i represents the distance traveled at the i-th step.
- P represents the time penalty.
- time_i represents the remaining time on the clock at i-th step.

0.5 Decision Variables

Decision variables are typically the elements that we can adjust or optimize to achieve the desired objectives while satisfying the given constraints. In

other words, Decision variables represent the choices or decisions that can be made in the context of optimizing Mario's adventures in the constrained environment. For this COP, the Decision variables are Distance Traveled, Coins Collected, Number of Rebirth, Allowed Game Time, and Number of Moves.

0.6 Mathematical Formulation

$$\text{Maximize:} \quad \text{fitness} = \sum_{i=1}^{n} (C_i \cdot \text{coins}_i + D_i \cdot \text{distance}_i) - P \cdot (300 - \text{time}_i)$$

Where:

C_i represents the coin reward for the i-th step.
coins_i represents the number of coins collected at the i-th step.
D_i represents the distance reward for the i-th step.
distance_i represents the distance traveled at the i-th step.
P represents the time penalty.
time_i represents the remaining time on the clock at the i-th step.

Subject to the following constraints:

$$\sum_{i=1}^{n} \text{time}_i \leq T$$
$$\sum_{i=1}^{m} \text{rebirth}_i \leq R$$
$$\sum_{i=1}^{n} \text{action}_i \leq A$$

Where:

T is the maximum allowed time for completing the game.
R is the maximum number of rebirth instances allowed.
A is the maximum number of moves or actions that can be performed.

0.7 Output

The expected output of this COP is a path that Mario has taken in a Game where it collects as many coins with maximum distance traveled under some constraints in order to finish level faster.

0.8 Complexity Classification

This problem is classified as a constrained optimization problem. It belongs to the class NP because verifying whether a proposed solution (Mario's path) is valid can be done efficiently (in polynomial time). The specific algorithm, such as Evolutionary Computation, can be employed to solve this NP problem. For example, it would be more accurate to say for his optimization problem, which could be something like "Is there a path for Mario that collects at least X coins while satisfying all the constraints?" can be verified in polynomial time, making it a problem in NP.

0.9 Related Work and Review

This problem is widely studied with different techniques including Genetic Algorithm and NeuroEvolution (Mautschke, 2019; Bytyçi et al., 2020). Among these approaches, Genetic Algorithm leverages evolutionary principles to optimize Mario's gameplay sequences, evolving action plans over generations. On the other hand, NeuroEvolution employs artificial neural networks to evolve intelligent controllers for Mario, allowing him to adapt and learn in dynamic environments. These methods have shown significant promise in enhancing Mario's performance within constrained game environments, achieving near-optimal solutions through evolutionary and neural network-based strategies.

Genetic Algorithms (GAs) optimize Mario's gameplay by evolving action sequences over generations, representing potential paths that undergo mutation and crossover for enhanced performance. Fitness evaluation considers coin collection, distance traveled, and constraints, ultimately guiding Mario to collect more coins efficiently in constrained game environments. GAs simulate natural selection, starting with random sequences and selecting the best for reproduction. Over time, these sequences improve, optimizing Mario's gameplay strategy (Tønder & Olsen (2013)).

NeuroEvolution: NeuroEvolution focuses on evolving artificial neural networks (ANNs) as controllers for Mario. It begins with a population of ANNs

with random weights. These networks control Mario's actions and are evaluated based on their ability to help Mario achieve goals like coin collection and obstacle avoidance. Successful ANNs are preserved and mutated, gradually improving Mario's decision-making abilities through neural network evolution. NeuroEvolution adapts Mario's behavior to various challenges, making it a valuable tool for optimizing gameplay(Andersen et al. (2018)).

In the landscape of AI research applied to Super Mario Bros., Reinforcement Learning (RL) has emerged as a pivotal technique for enhancing Mario's gameplay capabilities. RL methodologies, such as Q-learning and Deep Q-Networks (DQNs), have garnered substantial attention due to their data-driven approach, enabling Mario to adapt and learn optimal strategies through interactions with the game environment. These RL-based methods equip Mario with the ability to make decisions based on rewards and penalties, which proves highly effective in navigating dynamic and evolving in-game scenarios. This adaptability, coupled with the capacity to discover novel gameplay strategies over time, positions RL as a vital component in the quest to optimize Mario's performance in Super Mario Bros. (Engelsvoll et al. (2020)).

References

Andersen, P.-A., Goodwin, M., & Granmo, O.-C. (2018). Deep rts: a game environment for deep reinforcement learning in real-time strategy games. In *2018 ieee conference on computational intelligence and games (cig)* (pp. 1–8).

Bytyçi, E., Hulaj, A., & Aliu, D. (2020). Combination of genetic algorithm and neural network for independent game playing. In *2020 international conference on intelligent data science technologies and applications (idsta)* (pp. 106–109).

Engelsvoll, R. N., Gammelsrød, A., & Thoresen, B.-I. S. (2020). *Generating levels and playing super mario bros. with deep reinforcement learning using various techniques for level generation and deep q-networks for playing* (Unpublished master's thesis). University of Agder.

Kauten, C. (2018). *Super Mario Bros for OpenAI Gym.* GitHub. Retrieved from `https://github.com/Kautenja/gym-super-mario-bros`

Mautschke, O. (2019). Evolutionary algorithms for controllers in games. *AI for Games - Oliver Mautschke*.

Tønder, L. S., & Olsen, O.-P. (2013). *Multi-objective neuroevolution in super mario bros.* (Unpublished master's thesis). Institutt for datateknikk og informasjonsvitenskap.