Twisty Puzzle Analysis version 2.2.1

by Sebastian Jost

december 2020 - december 2020

Dependecies

non-standard library modules:

- vpython
- colored
- lxml
- sympy
- keras

What can it do?

This version focusses on the 3d animation using `vpython`

It also implements a console interface to control the animation. Overall it can do the following things (commands for these features are in brackets):

Setup of puzzles

- import .ggb files into vpython 3d points (`import`)
- save a puzzle imported like that in a new .xml file (`savepuzzle`)
- visually define moves by clicking on the points (`newmove`, `endmove`)
- automatically calculate the rotation axis for every move and animate it (`move`)
- rename existing moves (`rename`)
- delete exisiting moves (`delmove`)
- load a previously saved puzzle with it's moves from a .xml file (`loadpuzzle`)
- list all or just a single defined move (`listmoves`, `printmove`)

playing with the puzzles

- change shape of the puzzle by snapping it to a sphere, cube or it's initial shape (`snap`)
- scramble the puzzle randomly (`scramble`)
- reset the puzzle to solved state (`reset`)
- train an AI using Q-Learning to learn to automatically solve any sufficiently easy puzzle. harder puzzles can be partially solved (`train_q`, `move_q`, `solve_q`)
- plot the success of the AI during training (basic plot, no legend, acis labels or any explanation) (`plot`)
- control animation speed (`sleeptime`)

Additions in version 2.2.1

- added compression of states in q-table by saving the entire puzzle state in a single integer. This reduces memory requirements roughly by a factor 1/4.

  Not fully implemented yet. Some parts of the code would need major changes to work with the compressed states.

  Since performing actions on the compressed states is much slower, the training time would increase drastically just to save some memory.

---

fixed issues:

---

possible improvements

Train the AIs not on permutations of the color points but on permutations of the pieces instead. This doesn't decrease the state space size but it could decrease the computational cost for each applied move and decrease the size of the neural network.

Both of which would accelerate the training and massively decrease the time it takes to solve puzzles with the implemented A* algorithm.