HexSolver: A real-time solver for the Hexcells logic puzzle series

Source Code Documentation

James Durant
Department of Computer Science
University of Warwick

Supervised by Matthew Leeke
Year of Study: 3rd

2021/2022

1 Outline

This document is intended to provide an overview of the source code and resources of the project, installation instructions and how to run the project. Knowledge from the final report is assumed. Contained within the `HexSolver` directory is the source code, resources and figures for the project. The code consists of nine Python files:

1. `data.py`: automatically generates the perceptual hashes used in number and hint type classification. This file can also automatically generate machine learning training datasets using the Hexcells Infinite random level generator.
2. `grid.py`: defines the class-based representation of a Hexcells level.
3. `learn.py`: trains and tests agents on online or offline environments.
4. `main.py`: defines the graphical user interface (GUI).
5. `navigate.py`: interfaces between GUI actions and menu parsing algorithms.
6. `parse.py`: defines menu screen and level parsing algorithms.
7. `plot.py`: creates the learning plots shown in the final report.
8. `solve.py`: formulates and solves a Hexcells level as an ILP problem.
9. `window.py`: encapsulates a Hexcells game window.

Also provided are a number of resources used by the application:

1. `resources/{black, blue, counter, column, level_select, screen}`: custom levels and corresponding perceptual hashes for each dataset type.
2. `resources/models`: models of varying architectures, pre-trained on different level sizes (see `models.txt` for a description of each model).
3. `resources/models/logs`: logs containing training and validation accuracy histories from training each model.
4. `resources/models/figures`: the figures displaying training and validation accuracies shown in the final report (generated by `plot.py`).
5. `resources/levels`: three datasets, consisting of levels of different size, that can be used for training and testing models.
6. `resources/shapes`: reference images for matching cell and counter shapes.

2 Installation

Access to one or more of the games in the Hexcells series is required and can be bought on Steam. Most of the code is relatively lightweight and so there are no specific hardware constraints. However, if training models, a CUDA-enabled GPU is advised to reduce training times. Please note that this project was developed using 64-bit Windows 10. Linux and macOS are not supported due to how manipulating windows differs between the operating systems.

The project was written in Python3 and use third-party packages (those that do not come with a default Python installation). To install these packages, run `pip install -r requirements.txt` via the command line. The project was developed using Python 3.8.11 but any version >= 3.6 should work.

For solving, this project uses the GNU Linear Programming Kit (GLPK). The package is not installable using `pip`. Instead, download the GLPK distribution (any version is suitable), untar the `glpk-x.tar.gz` file (for example, using 7-zip) and move its contents (not the folder itself) to `resources/glpk`. The following file path to the solver executable should then be valid:

```
resources/glpk/w64/glpsol.exe
```

If you wish to use a different file path, please change the `GLPK_PATH` global variable in `solve.py`. The file path to the `Steam.exe` executable is assumed to be the following:

```
C:\Program Files (x86)\Steam
```

If this is not the case, please change the `STEAM_PATH` global variable in `parse.py`.

3 Usage

To launch the application, run `python3 main.py` from the command line (in the directory containing the code). In the top section of the GUI, the game to load can be selected by clicking on the appropriate radiobutton (if a game is already running, this will be detected). If the game has not been modified to disable the "falling triangles" particle effect (see below), please click the tickbox for this. To run the solver, click the "Solve" button after using the radiobuttons and dropdown menus to select a level, set of levels, an entire game or a randomly generated level.

At the bottom of the GUI, you can either create a new model or load an existing one. To load an existing model, select the "Load Model" radiobutton and click on the "Select Model" button to open a file dialogue box. Model hyperparameter values can

be entered and variations of the deep Q-learning algorithm can be toggled. Click the "Run" button, after selecting either online or offline learning via the dropdown menu, to run the model. If you wish the model to learn as it runs, make sure the "Train" checkbox is ticked. Please make sure that the level generator has been modified to support the agent that you are running. Also, make sure that the level dimensions match the dimensions of the levels that the agent was trained on.

It is advised that the steam overlay is turned off when using the application. This is because pop-ups (e.g., a friend playing a game) can be mistakenly identified when parsing. To do this, in the top-left of the Steam client, go to "Steam" $\rightarrow$ "Settings"; in the settings window, click on the "In-Game" tab and untick the "Enable the Steam Overlay while in-game" checkbox. When using the application, you cannot run Hexcells in fullscreen as you will not be able to use the GUI. Instead, a resolution slightly smaller than your screen's maximum resolution, or borderless fullscreen, is suitable. The application has been tested for the resolution of 1920×1080. Other resolutions will likely work but there may be edge cases that cause issues. If using training or testing models in online environments, please select "Easy" on the level generator screen (assuming that the necessary game code modifications have been made). Finally, please use light mode (the default).

4 Modifying Hexcells

This project utilises the `dnSpy` editor to load, edit and recompile Unity assemblies. If you are not training or testing any models, this step not necessary. However, you will need to tick the "Particle Effect" checkbox in the GUI. If you wish to train or test models, you will need to modify the level generator to generate suitable training environments.

1. Close any Hexcells windows currently running
2. Download the dnSpy assembly editor, extract the zip file (`dnSpy-net.win64.zip`) and run `dySpy.exe`
3. In the `dnSpy` window, go to "File" $\rightarrow$ "Open" and select

   ```
   C:\Program Files (x86)\Steam\steamapps\common\Hexcells Infinite\
   Hexcells Infinite_Data\Managed\Assembly-CSharp.dll
   ```

4. On the left-hand panel, expand the "Assembly-CSharp" dropdown, then "Assembly-CSharp.dll", and finally "{}"
5. On the left-hand panel, right click on "OldLevelGenerator" and select "Edit Class (C#)..."

6. In the "SpawnAHex()" method, delete "`if (num < this.chanceOfBlackHex)`" from line 33.
7. In the "GenerateLevel()" method, delete line 173: "`this.marvinHexcellsSolver.AddDiagonalColumnNumbers();`"
8. In the "SetRandomVariables()" method:
   (a) Replace lines 90-101 with "`this.heightLimiter = x;`" where $x = 12$ for small and medium levels, and $x = 11$ for large levels.
   (b) Replace line 89 with "`this.numberOfTilesToSpawn = x;`" where $x = 9$ for small levels, $x = 15$ for medium levels and $x = 28$ for large levels.
   (c) Replace line 85 with "`this.chanceOfFlowerHex = 0f;`"
9. In the "Start()" method:
   (a) Replace line 52 with "`this.SetupLevelNoTagging;`"
   (b) Replace lines 39-51 with "`this.GenerateLevel();`"
10. Click the "Compile" button in the bottom right-hand corner of the window containing the code.
11. In the `dnSpy` window, go to "File" $\rightarrow$ "Save Module".
12. Click the "OK" button in the new window. You can now close the `dnSpy` editor and run Hexcells.

To remove the particle effect displayed after uncovering a black cell:

1. Follow steps 1-4 from above.
2. On the left-hand panel, expand "HexBehaviour", right click on "DestroyClick()", and select "Edit Method (C#)..."
3. Deletes line 18 and 13.
4. Follow steps 10-12 from above.