Methods of AI - Coding Task

Authors: Tillmann Brodbeck, Esther Chevalier, Sarah Neuhoff, Sören Selbach

Required Libraries

The program was developed and tested with `python 3.7.1` and the following libraries:

- `numpy (1.16.0)`
- `matplotlib (3.0.2)`
- `tkinter (8.6)`

In most cases any somewhat recent versions of these libraries should be fine.

The program has been tested on Windows 10 and macOS Mojave.

Usage

The main executable file is `gui.py`. The command to run the program is simply `python gui.py`.

Before running a search algorithm, the user needs to select a file containing the environment definition (warehouse file) and a file containing the problem to be solved (order file) via the two buttons at the top left of the GUI. After successfully selecting these files, the paths should be displayed in the two text cells below the buttons.
The files should have the format specified in the description of the programming task. There are no particular checks for correct syntax, the program might just crash if it receives invalid files.

The user can then choose one of the provided search algorithms from the first dropdown menu. If applicable, the user can also select the desired number of beams (in case of local beam search) or threads (in case of parallel hillclimbing) from the second dropdown menu.

To start the search, press the `Start` button. While the search is in progress, the program **will not be responsive**. When the search is finished, the found solution will be displayed in the large text area on the left.

Code Structure

The program is divided into several files:

File	Description
`gui.py`	Main executable file. Contains the logic for the graphical user interface. Always run this file.
`search.py`	Contains all local search algorithms except parallel hillclimbing, as well as an abstract class for search algorithms.
`searchutils.py`	Contains a function that returns the neighbors of a state, and a function that computes the value of a state.
`parallel_hillclimbing.py`	Contains parallel hillclimbing. Needs to be its own file for reasons of multiprocessing.
`listvar.py`	Contains a simple implementation of a traceable list.