Practical Work/Assignment No. 1

Heuristic Search Methods: A* for Solving a Puzzle

Adversarial Search Methods: Minimax for Playing a Board Game

Theme/Topics

EIACD first practical assignment consists in the development of an assignment related to one of two possible main topics and choosing a specific subject for the assignment.

Topic 1: Heuristic Search Methods for One Player Solitaire Games

A solitaire game is characterized by the type of board and pieces, the rules of movement of the pieces (possible operators/plays), and the conditions for ending the game with defeat (impossibility to solve, maximum number of moves reached, time limit reached) or victory (solitaire solved), together with the respective score. Typically, in the event of a win, a score is awarded depending on the number of moves, resources spent, bonuses collected and/or time spent.

In addition to implementing a solitaire game for a human player, the program must be able to solve different versions/levels of this game, using appropriate search methods, focusing on the comparison between uninformed search methods (breadth-first search, depth-first search, iterative deepening, uniform cost) and heuristic search methods (greedy search, A* Algorithm, Weighted A*, …), with different heuristic functions. The algorithms employed should be compared with several criteria, with emphasis on the quality of the solution obtained, number of operations performed, maximum memory used, and time spent to obtain the solution. All games, if it is possible, should have variable board size and a set of puzzles to solve with different difficulty levels.

Students should focus on firstly developing a simple version of the game with a small size board, the algorithms and testing that everything works (algorithms are able to solve the simple small puzzles), before proceeding to more complex and bigger scenarios.

The application should have a text or graphical use interface to show the evolution of the board and interact with the user/player. You should allow a game mode in which the PC solves the solitaire alone using the algorithm and respective configuration as selected by the user. Optionally, you can allow a Human game mode in which the user can solve the game, while asking the PC for "hints".

Topic 2: Adversarial Search Methods for Two-Player Board Games

A board game is characterized by the type of board and pieces, the rules of movement of the pieces (operators) and the finishing conditions of the game with the respective score. In this work, the aim is to implement a game for two players and solve different versions of this game, using the Minimax search algorithm with αβ cuts and its variants.

Human-human, human-computer and computer-computer game modes should be developed, where the computer should exhibit different skills (levels of difficulty). Computer performance should be compared regarding the different skills (e.g., hard, medium, easy), corresponding to different evaluation functions, different depth levels of Minimax, different successor generation ordering and/or variants of the Minimax algorithm. Monte Carlo Tree Search with different configurations may also be use. Emphasis should be

placed on the analysis of the results of the computer players (wins, draws, losses, and other quality parameters, such as the number of plays to obtain the win/loss) and average time spent to obtain the plays. All games, if it is possible, should have different versions with variable board size.

Students should focus on firstly developing a simple version of the game with a small size board and simplified rules, the algorithms and testing that everything works (algorithms are able to show intelligent/correct moves in simple game situations), before proceeding to more complex and bigger scenarios.

The application to be developed must have a proper text or graphical user interface, to show the evolution of the board and interact with the user/player. Different game modes must be included, as explained above, allowing the selection of the game mode, type of each player, and skills of computer players. You should allow different skilled computer players to play against each other. You may also consider providing human players with movement "hints".

Programming Language

The game should be developed using the Python programming language and simple Python libraries.

Groups

Groups must be composed of 3 students. Groups should be composed of students from the same practical class. Individual groups or groups with students from different classes are typically not accepted. All students should be present in the checkpoint sessions and presentation/demonstration of the work. The establishment of groups composed of students from different classes is not advisable, given the logistic difficulties of performing work that this can cause.

Checkpoint Delivery

Each group must submit in Moodle a brief presentation (max. 5 slides), in PDF format, which will be used in the class to analyze, together with the teacher, the progress of the work. The presentation should contain: (1) specification of the work to be performed and how the work will be developed, (2) related work with references to works found in a simple bibliographic search (articles, web pages and/or source code), (3) formulation of the problem as a search problem (state representation, initial state, objective test, operators (names, preconditions, effects and costs), heuristics/evaluation functions) and (4) implementation work already carried out (development environment, data structures, file structure, algorithms, text/graphical interface among others).

Final Delivery

Each group must submit in Moodle two files: a presentation (max. 10 slides), in PDF format, and the implemented code, properly commented, including a "readme" file with instructions on how to compile, run and use the program. Based on the submitted presentation, the students must carry out a demonstration (about 10 minutes) of the work, in the practical class, or in another period to be designated by the teachers of the course.

The file with the final presentation should include, in addition to the aforementioned for the checkpoint, details on: (5) the approach (heuristics, evaluation functions, operators, ...) and (6) algorithms implemented (search, A*, minimax and extensions), as well as (7) experimental results, using appropriate tables/plots and comparing the various methods, heuristics, algorithms and respective parameterizations for different puzzle/game scenarios. The presentation may also include a slide of conclusions and another of references consulted, and materials used (software, websites, scientific articles, ...).

Suggested Problems

Topic 1: One Player Solitaire Games

1A) Bird Sort 2 - Color Puzzle (small puzzles, flexible size):

https://play.google.com/store/apps/details?id=com.globalplay.birdsort2.color.puzzle&hl=en

1B) Wood Block - Puzzle Games (smaller boards, flexible size, known sequence of pieces):

https://play.google.com/store/apps/details?id=com.block.puzzle.free.wood&hl=en

1C) Space Block – Roll the Block (small puzzles, flexible size)

https://play.google.com/store/apps/details?id=com.zawor.spaceblock&hl=en&gl=US

Topic 2: Two-Player Adversarial Games

2A) Jungle Game (smaller boards, flexible size, with less pieces):

https://brainking.com/en/GameRules?tp=56

2B) Dameo (smaller boards, flexible size, with less pieces)

https://en.wikipedia.org/wiki/Dameo

2C) Lines of Action (smaller boards, flexible size, with less pieces)

https://boardgamegeek.com/boardgame/3406/lines-of-action

https://www.boardspace.net/loa/english/index.html