Assignment No. 1

Heuristic Search Methods for Problem Solving

Adversarial Search Methods for Games

Optimization Methods/Meta-Heuristics

Theme

IART's first practical assignment consists in the development of a program related to one of three possible topics.

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), 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.

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. 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.

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".

Topic 3: Metaheuristics for Optimization/Decision Problems

An optimization problem is characterized by the existence of a (typically large) set of possible solutions, comparable to each other, of which one or more are considered (globally) optimal solutions. Depending on the specific problem, an evaluation function allows you to establish this comparison between solutions. In many of these problems, it is virtually impossible to find the optimal solution, or to ensure that the solution found is optimal and, as such, the goal is to try to find a local optimal solution that maximizes/minimizes a given evaluation function to the extent possible.

In this work, the aim is to implement a system to solve an optimization problem, using different algorithms or meta-heuristics, such as hill-climbing, simulated annealing, tabu search, and genetic algorithms (you may include other algorithms or variations of these). Multiple instances of the chosen problem must be solved, and the results obtained by each algorithm must be compared. Different parameterizations of the algorithms should be tested and compared, in terms of the average quality of the solution obtained and the average time spent to obtain the solutions.

The application to be developed should have an appropriate visualization in text or graphic mode, to show the evolution of the quality of the solution obtained along the way and the final (i.e. local optimal) solution, and to interact with the user. You should allow the selection and parameterization of the algorithms and the selection of the instance of the problem to be solved.

Programming Language

Any programming language and development system can be used, including, at the language level, Python, C++, Java, C#, Prolog, among others. The choice of language and development environment to be used is the entire responsibility of the students. In previous years, Python, C++ and Java were the languages selected by most of the students.

Groups

Groups must be composed of 3 students (exceptionally 2). Individual groups or groups composed of 4 students are not accepted. Groups should be composed of students attending the same practical class (alghouth exceptions are possible). All students must be present in the checkpoint sessions and presentation/demonstration of the work. Groups composed of students from different classes are discouraged, given the logistic difficulties of performing work that this can cause.

Checkpoint

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 (definition of the game or optimization problem to be solved), (2) related work with references to works found in a 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 function) or optimization problem (solution representation, neighborhood/mutation and crossover functions, hard constraints, evaluation functions), and (4) implementation work already carried out (programming language, development environment, data structures, 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) implemented algorithms (search algorithms, minimax, metaheuristics), as well as (7) experimental results, using appropriate tables/plots and comparing the various methods, heuristics, algorithms and respective parameterizations for different scenarios/problems. The presentation shall 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) Unequal Length Mazes https://erich-friedman.github.io/puzzle/unequal/

1B) Chess Snake Puzzles - https://erich-friedman.github.io/puzzle/snake/

1C) Exactly 1 Mazes - https://erich-friedman.github.io/puzzle/exactly1/

1D) Robot Mazes - https://erich-friedman.github.io/puzzle/robot/

Topic 2: Two-Player Adversarial Games

2A) Cirquarles – https://boardgamegeek.com/boardgame/334385/cirquarles

2B) Gekitai - https://boardgamegeek.com/boardgame/295449/gekitai

2C) Iro - https://boardgamegeek.com/boardgame/342251/iro

2D) Splinter - https://splinterboardgame.blogspot.com/2021/06/splinter-is-two-player-abstractstrategy.html

2E) Three Dragons - https://boardgamegeek.com/boardgame/306972/three-dragons

2F) Lines of Action - https://boardgamegeek.com/boardgame/3406/lines-action

Topic 3: Optimization Problems

Solve simplified versions of the following problems:

3A) Traffic Signaling (data)

3B) Optimize a Data Center (data)

3C) Street View outing (data)

3D) Mentorship and Teamwork