Trabalho prático individual nº 1

Inteligência Artificial + Sistemas Inteligentes I
Ano Lectivo de 2025/2026

24-25 de Outubro de 2025

I Important remarks

1. This assignment should be submitted via *GitHub* within 28 hours after the publication of this description. The assignment can be submitted after 28 hours, but will be penalized at 5% for each additional hour.

2. Complete the requested functions in module `"tpi1.py"`, provided together with this description.

3. Include your name and number and comment out or delete non-relevant code (e.g. test cases, print statements); submit only the mentioned module `"tpi1.py"`.

4. You can discuss this assignment with colleagues, but you cannot copy their programs neither in whole nor in part. Limit these discussions to the general understanding of the problem and avoid detailed discussions about implementation.

5. Include a comment with the names and numbers of the colleagues with whom you discussed this assignment. If you turn to other sources, identify them as well.

6. All submitted code must be original; although trusting that most students will do this, a plagiarism detection tool will be used. Students involved in plagiarism will have their submissions canceled.

7. The submitted programs will be evaluated taking into account: performance; style; and originality / evidence of independent work. Performance is mainly evaluated concerning correctness and completeness, although efficiency may also be taken into acount. Performance is evaluated through automatic testing. If necessary, the submitted modules will be analyzed by the teacher in order to appropriately credit the student's work.

II Exercices

Together with this description, you can find the `tree_search` module. You can also find attached the modules `cidades`, `strips`, and `blocksworld2`, containing the `Cidades(SearchDomain)`,

`STRIPS(SearchDomain)` and other related classes. These modules are similar to the ones initially provided for the practical classes, but with some changes and additions, namely:

- Loop prevention (by avoiding repeated states along a path) is implemented
- The `Cidades(SearchDomain)` is given fully implemented.
- The `STRIPS(SearchDomain)` is given fully implemented.
- The `blocksworld2` module describes a variant of the standard "blocks world" where specific blocks are handled with specific tools.

Don't change the `tree_search`, `cidades`, `strips` and `blocksworld2` modules.

The module `tpi1_tests` contains several test cases. If needed, you can add other test code in this module.

Module `tpi1` contains the classes `MyTree(SearchTree)`, `MyNode(SearchNode)` and `MySTRIPS (STRIPS)`. In the following exercices, you are asked to complete certain methods in these classes. All code that you need to develop should be integrated in the module `tpi1`.

1. Create a new method `search2()` in class `MyTree`, similar to the original search method, and add any other required code to make sure that nodes (class `MyNode`) have the attributes `depth`, `cost` and `heuristic`, with the usual meaning. In addition, make sure that, after the search is finished, the search tree (class `MyTree`) has attributes `terminal` and `non_terminal` (counters), also with the usual meaning, and `solution_cost`, with the cost of the found solution node.

2. In `MyTree`, implement the method `hybrid_add_to_open(lnewnodes)`, which manages the queue of open nodes according to the following criteria: in the early stages of the search, it behaves as uniform cost search; as soon as at least one node is added to the queue with cost larger than half of the heuristic value of the root node, the search switches to greedy mode. If there is a tie in the evaluation functions of two or more nodes, use depth and state as tiebreakers.

3. Develop a method `bipolar_search()` in `MyTree` that searches both forward from the initial state and backward from the goal state. When the node selected for expansion in one of these search poles contains a state already found in the other search pole, a solution is found. If this state appears several times in the other search pole, and the basic search strategy is blind (non informed), then the relevant node is the one with lowest depth. Otherwise, the relevant node will be the one with lowest cost. The returned solution is the concatenation of the forward and backward paths to the mentioned state.

4. The original implementation of the `actions(state)` method in the `STRIPS` class is rather naive. It was designed to be short and simple, but it is not efficient. The `actions2(state)` method is intended to be a better alternative, provided that you develop an efficient implementation of the `get_instanciations(op,state)` method in the `MySTRIPS` class. Scoring of this exercice will take into account computation time.

III Clarification of doubts

This work will be followed through `http://detiuaveiro.slack.com`. The clarification for the main doubts will be added here.

1. How will exercices be scored?

   Resposta: Ex. 1: 15%; ex. 2: 10%; ex. 3: 40%; ex. 4: 35%.