



Introduction to Artificial Intelligence

Laboratory activity

Name: Adam Cristina-Ioana & Manolescu Matei

Group: 30234

Email: cristina.adam97@yahoo.com & mateimanolescu@yahoo.com

Assoc. Prof. dr. eng. Adrian Groza
Adrian.Groza@cs.utcluj.ro



Contents

1	Rules and policies	3
2	A1: Search	6
3	A2: Logics	8
4	A3: Planning	9
A	Your original code	11

Chapter 1

Rules and policies

Lab organisation.

1. Laboratory work is 20% from the final grade.
2. There are 3 deliverables in total.
3. Before each deadline, you have to send your work (latex documentation/code) at moodle.cs.utcluj.ro

Class: Introducere in Inteligenta Artificiala
Enrollment key: lia2017-2018

4. *Laptop policy*: you can use your own laptop as long you have Linux. One goal of the laboratory is to increase your competency in Linux. It is **your** task to set static IPs:

IP: 192.168.1.51¹
MASK: 255.255.255.0
GATEWAY: 192.168.1.2
DNS: 192.168.1.2
PROXY 192.168.1.2:3128

Wifi: Network: isg
Password: inteligentaartificiala

5. *Group change policy*. Maximum number of students in a class is 14.
6. *For students repeating the class*: A discussion for validating the previous grade is mandatory in the first week. I usually have no problem to validate your previous grades, as long you request this in the first week. Failing to do so, leads to the grade 1 for the laboratory work in the current semester.

Grading. Assessment aims to measure your knowledge and skills needed to function in realistic AI-related tasks. Assessment is based on your written report explaining the nature of the project, findings, and recommendations. Meeting the deadlines is also important. Your report is comparable to ones you would write if you were a consultant reporting to a client.

Grade inflation makes difficult to distinguish between students. It also discourages the best students to do their best. In my quest for “optimal ranking of the students”, I do not use the following heuristics:

Table 1.1: Lab scheduling.

Activity	Deadline
<i>Searching agents, linux, latex, python</i>	W_1
<i>Uninformed search</i>	W_2
<i>Informed Search</i>	W_3
<i>Adversarial search</i>	W_4
<i>Propositional logic</i>	W_5
<i>First order logic</i>	W_6
<i>Inference in first order logic</i>	W_7
<i>Knowledge representation in first order logic</i>	W_8
<i>Classical planning</i>	W_9
<i>Contingent, conformant and probabilistic planning</i>	W_{10}
<i>Multi-agent planing</i>	W_{11}
<i>Modelling planning domains</i>	W_{12}
<i>Individual feedback</i> to clarify the good/bad issues related to student activity/results during the semester.	W_{14}

- "He worked hard at the project". Our society do not like anymore individuals that are *trying*, but individual that *do* stuff. Such heuristic is not admissible in education, except the primary school.
- "I knew he could do much better". Such a heuristic is not admissible because it does not encourage you to spread yourself.
- 7 means that you: i) constantly worked during classes, ii) you proved competent to use the tool and its expressivity for a realistic scenario, iii) you understood theoretical concepts on which the tool rely on.
- 8, 9 mean that your code is large enough and the results proved by your experiments are significant.
- 10 means that you did very impressive work or more efficient that I expected or handled a lot of special cases for realistic scenarios.
- 5 means that you managed to develop something of your own, functional, with your own piece of code substantially different from the examples available.
- You obtain less than 5 in one of the following situations:
 1. few code written by yourself.
 2. too much similarity with the provided examples.
 3. non-seriosity (i.e. re-current late at classes, playing games, worked for other disciplines, poor/unprofessional documentation of your work, etc.)².
- You get 2 if you present the project but fail to submit the documentation or code. You get 1 if you do not present your project before the deadline. You get 0 for any line of code

²Consider non-seriosity as a immutable boolean value that is unconsciously activated in my brain when one of the above conditions occurs for the first time.

taken from other parts that appear in section *My own code*. For information on TUCN's regulations on plagiarism do consult the active norms.

If your grade is 0, 1, or 2, you do not satisfy the preconditions for participating to the written exam. The only possibility to increase your laboratory grade is to take another project in the next year, at the same class, and to make all the steps again.

However, don't forget that focus is on learning, not on grading.

Using Latex in your documentation. You have to show some competency on writing documentation in Latex. For instance, you have to employ various latex elements: lists, citations, footnotes, verbatim, maths, code, etc.

Plagiarism. Most of you consider plagiarism only a minor form of cheating. This is far from accurate. Plagiarism is passing off the work of others as your own to gain unfair advantage.

During your project presentation and documentation, I must not be left with doubts on which parts of your project are your work or not. Always identify both: 1) who you worked with and 2) where you got your part of the code or solution. You should sign the declaration of originality.

Describe clearly the starting point of your solution. List explicitly any code re-used in your project. List explicitly any help (including debugging help, design discussions) provided by others (including colleagues or teaching assistant). Keep in mind that it is your own project and not the teaching assistant's project. Learning by collaborating does remain an effective method. You can use it, but don't forget to mention any kind of support. Learning by exploiting various knowledge-bases developed by your elder colleagues remain also an effective method for "learning by example". When comparing samples of good and poor assignments submitted by your colleagues in earlier years try to identify which is better and why. You can use this repository of previous assignments, but don't forget to mention any kind of inspiration source.

The assignment is designed to be individual and to pose you some difficulties (both technological and scientific) for which you should identify a working solution by the end of the semester. Each semester, a distinct AI tool is assigned to two students. You are encouraged to collaborate, especially during the the installation and example understanding phases (W_1 - W_4). The quicker you get throughout these preparatory stages, the more time you have for your own project.

Class attendance. I expect active participation at all activities. Keep in mind the exam can include any topic that was covered during class, explained on the board, or which emerged from discussions among participants. Missing lab assignments or midterm leads to minimum grade for that part. You are free to manage your laboratory classes - meaning that you can submit the project earlier - as long as you meet all the constraints and deadlines.

Chapter 2

A1: Search

Algoritmul A* este un algoritm care cauta drumul avand costul minim. Calculeaza drumul de la nodul start pana la succesori si apoi estimeaza costul de acolo pana la nodul de finish. La baza algoritmului sta formula $f = g + h$, unde g reprezinta costul de la inceput pana la succesori, iar h reprezinta euristica.

Algoritmul A* cu ponderi reprezinta un algoritm bazat pe algoritmul A*, cu diferenta ca se va adauga o pondere atat pentru euristica cat si pentru cost. Noua functie va deveni: $f = \text{ponderi1} * g + \text{ponderi2} * h$. Prin acest algoritm vrem sa vedem care componenta a functiei are cel mai mare impact in eficienta algoritmului pe labirinturi de diferite dimensiuni.

Concluzii

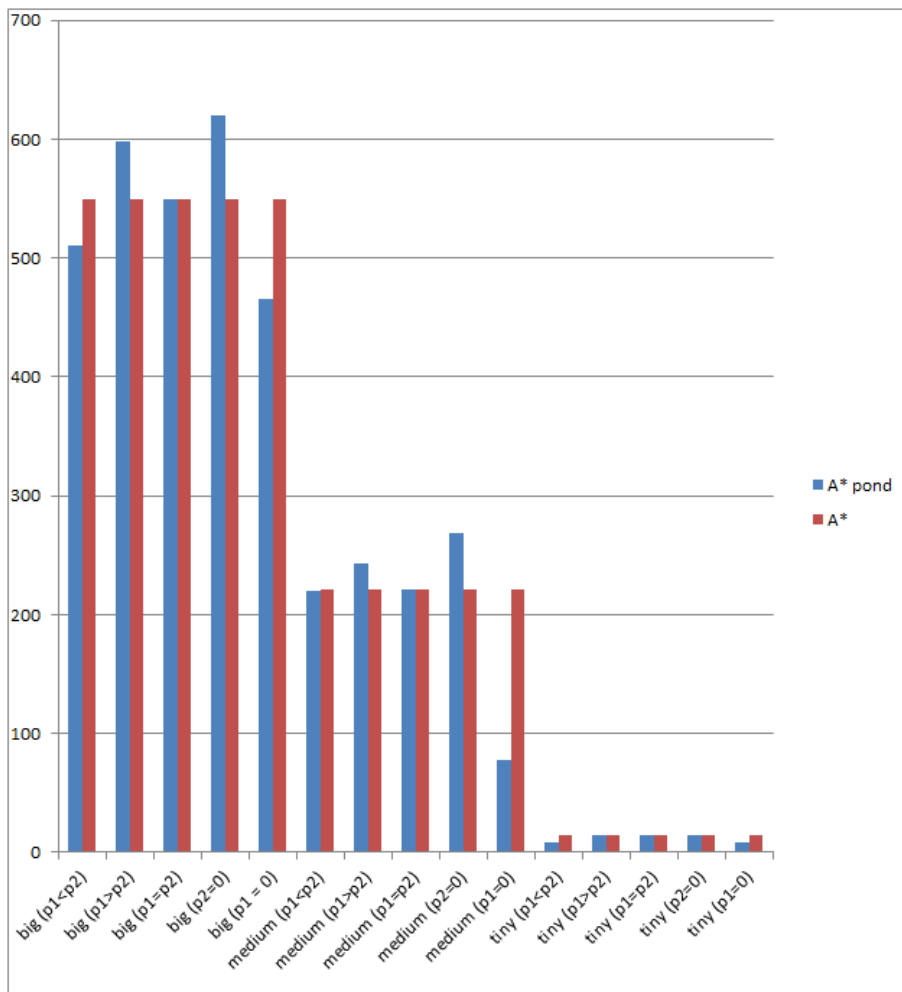
Pentru realizarea algoritmului A* cu ponderi am incercat sa transmitem cele 2 ponderi ca parametri in interiorul functiei `aStarPonderiSearch` (vezi anexa pentru cod). Cele 2 ponderi am vrut sa le initializam in momentul in care apelam functia din terminal. Pentru testarea pe mai multe cazuri ar fi trebuit sa definim o functie care ar fi apelat comanda din terminal de mai multe ori cu ponderi diferite, ducand la pornirea mai multor jocuri.

Deoarece nu am reusit sa implementam aceasta solutie am ales sa luam succesiv mai multe valori pentru ponderi, acoperind toate cazurile de interes. Rezultatele obtinute pot fi vizualizate in tabelul si graficul de mai jos.

Drept concluzie, putem observa faptul ca pe aceste labirinturi cea mai eficienta metoda este cea in care se tine cont exclusiv de euristica, adica ponderea lui g este 0.

Table 2.1: Rezultate A* cu ponderi vs A*.

Tip	A* ponderi	A*	Pond1	Pond2	Concluzie
<i>BigMaze</i>	510	549	1	2	A* cu ponderi mai bun
<i>BigMaze</i>	598	549	4	2	A* mai bun
<i>BigMaze</i>	549	549	5	5	Acelasi rezultat
<i>mediumMaze</i>	220	221	1	2	A* cu ponderi mai bun
<i>mediumMaze</i>	243	221	4	2	A* mai bun
<i>mediumMaze</i>	221	221	5	5	Acelasi rezultat
<i>tinyMaze</i>	220	221	1	2	A* cu ponderi mai bun
<i>tinyMaze</i>	220	221	4	2	Acelasi rezultat
<i>tinyMaze</i>	220	221	5	5	Acelasi rezultat
<i>bigMaze</i>	620	549	5	0	Acelasi rezultat ca la uniform
<i>mediumMaze</i>	269	221	5	0	Acelasi rezultat ca la uniform
<i>tinyMaze</i>	15	14	5	0	Acelasi rezultat ca la uniform
<i>bigMaze</i>	466	549	0	5	Functioneaza ca BestFirst
<i>mediumMaze</i>	78	221	0	5	Functioneaza ca BestFirst
<i>tinyMaze</i>	8	14	0	5	Functioneaza ca BestFirst



Chapter 3

A2: Logics

Chapter 4

A3: Planning

Bibliography

Appendix A

Your original code

This section should contain only code developed by you, without any line re-used from other sources. This section helps me to correctly evaluate your amount of work and results obtained. Including in this section any line of code taken from someone else leads to failure of IS class this year.

```
def aStarSearch(problem, heuristic=nullHeuristic):
    """
    Search the node that has the lowest combined cost and heuristic first.
    To test, try the following command:

    python pacman.py -l bigMaze -z .5 -p
    SearchAgent -a fn=astar,heuristic=manhattanHeuristic
    """

    time.sleep(2)
    fringe = util.PriorityQueue()
    fringe.push( (problem.getStartState(), [], 0),
                heuristic(problem.getStartState(), problem) )
    expanded = []

    while not fringe.isEmpty():
        node, actions, curCost = fringe.pop()

        if(not node in expanded):
            expanded.append(node)

            if problem.isGoalState(node):
                return actions

            for child, direction, cost in problem.getSuccessors(node):
                g = curCost + cost
                fringe.push((child, actions+[direction], curCost + cost),
                            g + heuristic(child, problem))

    return []
```

```

import sys
first_arg = sys.argv[1]
second_arg = sys.argv[2]

def aStarPonderiSearch(problem, heuristic=nullHeuristic, pond1=first_arg, pond2=second_arg):
    fringe = util.PriorityQueue()
    fringe.push( (problem.getStartState(), [], 0),
                 heuristic(problem.getStartState(), problem) )
    expanded = []

    print(pond1, pond2)

    while not fringe.isEmpty():
        node, actions, curCost = fringe.pop()

        if(not node in expanded):
            expanded.append(node)

        if problem.isGoalState(node):
            return actions

        for child, direction, cost in problem.getSuccessors(node):
            g = curCost + cost
            fringe.push((child, actions+[direction], curCost + cost),
                        pond1*g + pond2*heuristic(child, problem))

    return []

```

Intelligent Systems Group

