Assignment 0 (Warm-Up, Variant A): Find Back to the Wumpus Cave

AI-2 Systems Project (Summer Semester 2024)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

Topic:	Basic probabilities
Due on:	June 29, 2024
Version from:	April 16, 2024
Author:	Jan Frederik Schaefer
Important notes:	To be solved individually
	Earlier deadline for first results (see your solution repository)
	Do not use someone else's solution code (even as inspiration)
	Ask for help if you are stuck (office hours, assignment room, . . .)
	Every assignment has a guide with tips – you can find it at [AG]

1 Task summary

The Wumpus, a mythical creature that repeatedly shows up in the AI systems project, has left its cave and got lost. Your task is to make a travel plan that will quickly get it back to the cave. Your implementation will get different problem instances from the AISysProj server and has to send back its travel plan. That lets the server evaluate your agent and can help you with debugging. The assignment repository [AR] has a script that already implements the server interaction for you.

Didactic objectives

1. Develop an algorithm to solve a non-trivial problem,
2. implement a small software project from scratch,
3. get hands-on experience working with basic concepts from probability theory,
4. get to know the AISysProj setup and workflows.

Prerequisites and useful methods

1. The basics of probability theory: condition probabilities, independence, expected values, Bayes' rule, …

2 Navigating the Wumpus world

The Wumpus got lost. Your task is to implement an AI agent that makes a plan for it to get back to the Wumpus cave. We have a server to help evaluate your agent. It will send you a map along with some other information and your agent will have to respond with a plan to find back to the cave.[1] The server has different environments, corresponding to different difficulty levels. For the easier environments, not everything mentioned in this section is relevant.

2.1 Maps

Maps of the Wumpus world are encoded as a string, where each line corresponds to a row of cells and each character describes the properties of an individual cell. Here is an example map:

```
BWCCP
BWWMM
PCPCM
MWBPM
WMMMM
```

We have the following cell types:

- `M`: Meadows.
- `B`: Trees (broad-leaf).
- `C`: Trees (coniferous).
- `P`: Pits.
- `W`: Wumpus cave entrance. Your goal is to reach one of those.

We assume that everything outside the map are meadows (`M`).

[1] The server part may seem daunting, but we provide an example implementation in Python. All you have to then do is implement a function.

2.2 Observations

We do not know where the Wumpus is (after all, the Wumpus got lost). Therefore, we assume that, a priori, the Wumpus is equally likely in any of the cells on the map. However, the Wumpus can make some basic observations to narrow down the possibilities:

- It observes what type of cell it currently is in (`M` or `B` or …). Unfortunately, the Wumpus has bad vision and misidentifies trees 20 % of the time (i.e. if the Wumpus is in a `B` cell, there is a 20 % chance that it instead thinks it is a `C` cell and vice versa).
- In the more advanced environments, the Wumpus may perceive an echo if there are pits nearby. If the Wumpus is in a pit, it will always perceive an echo. If there is no pit anywhere, the Wumpus will never perceive an echo. Otherwise, the Wumpus will perceive an echo with a probability of $1/d^2$, where d is the Euclidean distance to the nearest pit (measured in cells, and we measure from cell center to cell center). For example, if nearest pit is one cell east and two cells north of the Wumpus, the Wumpus will perceive an echo with a probability of 20 % as $d = \sqrt{1^2 + 2^2}$.

2.3 Plans

Your task is to make a plan for the Wumpus. A plan is a sequence of actions of the form `GO [north|east|south|west]`. For each action, the Wumpus goes one cell in the indicated direction. Note that the world is not limited to the map and the Wumpus can go to cells outside the map (recall that every cell outside the map is a meadow).

Ideally, the Wumpus reaches one of the Wumpus cave entrances while following the plan. How long it takes for the Wumpus to reach the entrance depends on the cell types. We assume that the Wumpus starts in the center of a cell and only has to reach the edge of a cell with a cave entrance. Usually, crossing a complete cell takes 1 hour. However, climbing out of a pit takes 4 hours (but entering a pit does not take any time at all).

Let us consider the following map as an example:

```
CW
MP
```

If the Wumpus is initially in the `M` cell and follows the plan `GO east, GO north`, then it will have to cross $\frac{1}{2}$ of the `M` cell (as it starts in the center), which takes 0.5 hours. It will then have to climb out of the pit to reach the edge of the `W` cell, which takes 4 hours. So it will take $0.5 + 4 = 4.5$ hours in total.

If the Wumpus is initially in the `P` cell and follows the plan `GO north`, then it will have to climb out of the pit to reach the edge of the `W` cell, which takes 4 hours. So when starting in the pit, it will still take the full 4 hours to get out (even though we start in the center of the cell).

To make the assignment easier, we have a maximum time M for each environment. If reaching a cave entrance takes longer than M (or the plan finishes before an entrance has been reached), then we instead assume that the time is M.

3 Evaluation on the server

You should evaluate your implementation with the AISysProj server: https://aisysproj.kwarc.info/. You will get JSON requests from the server and have to respond with your plan. For example, a request could have the following content (in this assignment you can ignore the `initial-equipment`):

```
{
    "map": "BWCCP\nBWWMM\nPCPCM\nMWBPM\nWMMMM",
    "observations": {"echo": true, "current-cell": "M"},
    "initial-equipment": [],
    "max-time": 6
}
```

Your agent should then respond with a plan and the expected time until a Wumpus cave is reached:

```
{"actions": ["GO␣west", "GO␣north", "GO␣south"], "expected-time": 5.016129032258064}
```

The server remembers your last 1000 responses and computes the average expected time to rate your agent (the lower, the better). If you submit the wrong `expected-time`, the server will instead use `max-time`. However, afterwards you can see the problem on the server with the correct expected time and some other information that could help you with debugging.

The server has different environments that you can use (some are easier than others). Your repository for this assignment will contain one configuration file for each environment, which contains, among other things, your agent name and a password. The assignment repository [AR] contains an example implementation in Python along with instructions on how to use it, so that you do not have to worry about the technical aspects of communicating with the server. If you want to use a different programming language, you can find the protocol specification at ??. We are happy to help you with implementing it if you are prepared to donate your code for others.

4 What to submit

Your solution should be pushed to your git repository for this assignment. For this warm-up assignment, we have an early deadline. At this deadline, the repository should contain all the code you have so far. It should be enough to get at least 1 point in one of the environments on the server. Otherwise, we might assume that you are not actually interested in the project and give your spot to someone else.

Your grade will be based on your final submission (deadline: June 29, 2024). Concretely, your repository should contain:

1. all your code for solving this assignment,
2. a `README.md` file explaining
   1. dependencies (programming language, version, external libraries and how to get them),
   2. how to run your code,
   3. the repository structure,
   4. anything else we should know,
3. a solution summary (see [SoS] for more details – it should describe the main ideas, not document the code).

Furthoremore, you should run your code so that the server has an evaluation for your agent.

5 Points and environments

The total number of points for this assignment is 100. You can get up to 20 points for the quality of the submission (README, evaluation, ...). Furthermore, you can get up to 80 points for the performance of your agent according to the server. Each environment allows you to get a certain numbers of points if your agent performs well enough. When grading, we will only consider the environment in which you would get most points (we do not add up results from different environments). That means that you do not have to run your code on every environment and you can get full points if you only run it on the most difficult one where you can get up to 80 points.

Note: It is okay if your agent is a bit “lucky” and gets a slightly higher rating than it usually does. We will use the best rating on the server for your grade, assuming that we can reproduce a similar performance (i.e. if you get a 5.39 rating on the server and we get a 5.48 rating that is okay – but if you get a 5.39 rating on the server and we get a 6.79 rating when testing it, we might ask some questions).

Below is a table that summarizes the properties of the different environments and how many points you can get for each. For each environment, you have a config file.

Config file	Map size	Cells	Max. time	Equipment	Observations	Points
env-1.json	5×5	`M P`	2	–	current-cell	0 if rating > 1.9
						20 if rating ≤ 1.9
						30 if rating ≤ 1.6
env-2.json	5×5	`B M P`	6	–	current-cell	0 if rating > 4.5
						25 if rating ≤ 4.5
						40 if rating ≤ 3.5
env-3.json	5×5	`B M P`	6	–	current-cell, echo	0 if rating > 4.5
						40 if rating ≤ 4.5
						50 if rating ≤ 3.5
env-4.json	5×5	`B C M P`	6	–	current-cell, echo	0 if rating > 4.5
						45 if rating ≤ 4.5
						55 if rating ≤ 3.5
env-5.json	15×15	`B C M P`	24	–	current-cell, echo	0 if rating > 18.0
						55 if rating ≤ 18.0
						75 if rating ≤ 16.5
						80 if rating ≤ 15.0

References

[AG] *Guide for “Assignment 0 (Warm-Up, Variant A): Find Back to the Wumpus Cave”.* URL: https://kwarc.info/teaching/AISysProj/SS24/assignment-2.0.A-guide.pdf.

[AR] *Repository for Assignment 0 (Warm-Up, Variant A): Find Back to the Wumpus Cave.* URL: https://gitlab.rrze.fau.de/wrv/AISysProj/ss24/a2.0.a-find-wumpus-cave/assignment.

[SoS] *Solution Summary.* URL: https://gitlab.rrze.fau.de/wrv/AISysProj/admin/general/-/blob/main/solution-summary.md.