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

AI-2 Systems Project (Winter Semester 2024/2025)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

Topic:	Basic probabilities
Due on:	January 25, 2025
Version from:	October 9, 2024
Author:	Jan Frederik Schaefer
Important notes:	Earlier deadline for first results (see your solution repository)
	Ask for help if you are stuck (office hours, assignment room, . . .)
	Every assignment has a guide with tips – you can find it at [AG]

This assignment has to be solved individually (not as a team).
Using someone else's solution code, even as inspiration, is not allowed!

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 for the Wumpus. Ideally, this travel plan will get the Wumpus back to the cave soon, but it is not a requirement – any travel plan will be accepted.

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 travel plan will be evaluated based on the how likely the travel plan is to succeed and how long it would take to reach the cave (details in Section 2.3). 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:

```
CWBBB
RRRBB
WRRRR
BBRWB
MWRRR
```

We have the following cell types:

- `M`: Meadows.
- `B`: Trees (broad-leaf).
- `C`: Trees (coniferous).
- `R`: Rocks.

[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.

- `W`: Wumpus cave entrance. Your goal is to reach one of those.

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

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 some environments, it also remembers what cell type it observed in the cell that is west of the current cell. However, there is again a chance of misidentification.

2.3 Plans and their execution time

Your task is to make a plan for the Wumpus. A plan is a sequence of actions. You have the following actions available:

- `GO [north|east|south|west]`: The Wumpus should go 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).
- `DROP climbing-gear`: In some environments, the Wumpus carries climbing gear. This action tells the Wumpus to drop it (which allows it to be faster on most cell types).

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. Usually, crossing a cell takes 1 hour. With climbing gear, it takes 1.2 hours. However, crossing a cell with rocks (`R`) takes 4 hours, but with climbing gear it only takes 2 hours. 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. Let us consider the following map as an example:

```
CR
MW
```

If the Wumpus is initially the `C` cell and follows the plan `GO east, GO south`, then it will have to cross $\frac{1}{2}$ of the `C` cell (as it starts in the center) and $\frac{2}{2}$ of the `R` cell ($\frac{1}{2}$ to reach its center and $\frac{1}{2}$ to reach the edge of the `W` cell). In total, it will therefore take $0.5 + 4 = 4.5$ hours to reach the cave, assuming that the Wumpus has no climbing gear.

2.4 Success chance and expected time

Along with the plan, your agent should also provide the probability of the plan succeeding and the expected time it will take to reach the cave. Let us consider the following map as an example:

```
BWM
MBB
BMW
```

Let us assume that we know that the Wumpus is in a `B` cell. Every environment has a maximum time for the Wumpus to reach the cave. Let us assume that it is 3 hours for this example.

Now, the agent has to create some plan. Any plan is acceptable, but plans with a higher success chance and a lower expected time are better. We will use: `GO north`, `GO east`, `GO south`, `GO east`.

If the Wumpus is in the top left `B` cell, it will take 2.5 hours to reach a `W` cell (recall that the Wumpus can go outside the map). If the Wumpus is in the center `B` cell, it will take 0.5 hour to reach a `W` cell. If the Wumpus is in the right `B` cell, it will never reach a `W` cell. If the Wumpus is in the bottom left `B` cell, it would theoretically take 3.5 hours to reach a `W` cell, but as the maximum time is 3 hours, the plan would be considered unsuccessful in this case.

Now, the **success chance** is the probability that the Wumpus is in a cell, for which the plan is successful. In this case, the Wumpus is equally likely in each `B` cell, and the plan succeeds for two of them, so the success chance is $\frac{2}{4}$.

The **expected time** is the expected value of the time it takes to reach the cave, only considering the cases where the plan is successful. In this case, it would be $\frac{1}{2} \cdot 2.5 + \frac{1}{2} \cdot 0.5 = 1.5$ hours.

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:

```
{
    "initial-equipment": ["climbing-gear"],
    "map": "CWBBB\nRRRBB\nWRRRR\nBBRWB\nMWRRR",
```

```
    "max-time": 6,
    "observations": {"cell-west": "R", "current-cell": "B"}
}
```

Your agent should then respond with a plan, the probability of the plan succeeding, and the expected time it will take to reach the cave:

```
{
    "actions": [
        "GO␣south",
        "GO␣south",
        "GO␣east"
    ],
    "expected-time": 2.6,
    "success-chance": 1.0
}
```

The server will then evaluate your response and give you a rating. If the success chance and the expected time are correct, your rating will be computed by the formula

$$\texttt{rating} = \texttt{success-chance} \cdot \texttt{expected-time} + (1 - \texttt{success-chance}) \cdot \texttt{max-time}$$

where `max-time` is the maximum time for the environment.

The server remembers your last 1000 responses and computes the average rating (the lower, the better). If you computed the wrong value for `success-chance` or `expected-time`, the server will instead use `max-time` for the rating. However, afterwards you can see the problem on the server website 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 [CSP]. 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: January 25, 2025). Concretely, your repository should contain:

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

Furthermore, 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	`C M`	2	–	current-cell	0 if rating > 1.9 20 if rating ≤ 1.9 30 if rating ≤ 1.5
env-2.json	5×5	`C M`	5	–	current-cell	0 if rating > 4.0 30 if rating ≤ 4.0 40 if rating ≤ 3.0
env-3.json	5×5	`B C M`	5	–	current-cell	0 if rating > 4.0 40 if rating ≤ 4.0 50 if rating ≤ 3.0
env-4.json	5×5	`B C M R`	6	climbing-gear	current-cell	0 if rating > 5.0 45 if rating ≤ 5.0 55 if rating ≤ 4.0
env-5.json	5×5	`B C M R`	6	climbing-gear	current-cell, cell-west	0 if rating > 5.0 50 if rating ≤ 5.0 60 if rating ≤ 4.0 70 if rating ≤ 3.0
env-6.json	15×15	`B C M R`	24	climbing-gear	current-cell, cell-west	0 if rating > 20.0 50 if rating ≤ 20.0 75 if rating ≤ 15.0 80 if rating ≤ 13.0

References

[AG] *Guide for "Assignment 0 (Warm-Up, Variant A): Find Back to the Wumpus Cave".* URL: https://kwarc.info/teaching/AISysProj/WS2425/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/ws2425/a2.0.a-find-wumpus-cave/assignment.

[CSP] Jan Frederik Schaefer. *AISysProj server – Clients and server protocol.* URL: https://aisysprojserver.readthedocs.io/en/latest/clients.html.

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