Prentice Isabella Jones
Lucia Saignac Perez

Executive Summary

For this project, we decided to break the process of solving the prompt into different steps. Firstly, we read over the prompt and determined what the different states, actions, transitions, and costs would be for this specific Temperature Control Markov Decision Process. Once we outlined the probabilities and states, we drew out the graph for what the MDP would look like so we could see what the start and end states were, what the probabilities are for different edge cases and how different transitions would affect the state and probability. After visualizing our MDP, we decided to outline the states and actions in terms of variables. We initially thought that there would be an action for each temperature change (ie +.5 degrees, -.5 degrees, etc); however; after analyzing this approach we realized that there were actually only two actions: turning the heat “on” or “off”. Once we established the actions, we went about calculating what the costs for the actions for this MDP would be. We determined, based on research outlined below, that we would associate the cost with the price that the energy would change by when either having the heat on or off. We then tackled the Bellman equations. We wrote out the equations on paper to see what the different components were for each different equation, and also included how edge cases would require slightly different equations. Once we understood this baseline knowledge, we planned out how we would approach the coding portion. We decided that we would create classes/objects for individual states, actions and transitions, which we could then reuse later on in our program. Then we decided to start coding the Bellman Equation out. After much coding and testing we finally got our Bellman equation function to work as expected. Once this was done, we parsed out the values from the last iteration and used those to determine the optimal policy. After getting output for all of these, we decided to test our code with different values to see how our output would change based on differing cost implementations. Finally, we cleaned up our code to make sure that all the classes and functions looked neat, and made sure that there were comment explanations for the different components of the project.

Objectives

1. To find the optimal policy for the user’s desired temperature of 22 degrees celsius.
2. To model this project as close to real-life as possible.

Formal Description of MDP Model

- Modeled After Using the Tuple → <S, A, P, R>
- States: {16℃, 16.5℃, 17℃, 17.5℃, 18℃, 18.5℃, 19℃, 19.5℃, 20℃, 20.5℃, 21℃, 21.5℃, 22℃, 22.5℃, 23℃, 23.5℃, 24℃, 24.5℃, 25℃}
- Actions: { On, Off } **Referring to the heat**
- Probabilities: Transition Tables

$P_{On}(S_{t+1} | S)$:

	16	16.5	17	17.5	. . . .	24	24.5	25
16	0.3	0.5	0.2	0	0	0	0	0
16.5	0.1	0.2	0.5	0.2	0	0	0	0
17	0	0.1	0.2	0.5	. .	.	.	.
17.5	0	0	0.1	0.2	. .	.	.	.
.								
.								
.								
.								
24	0	0	0	0	. .	0.2	0.5	0.2
24.5	0	0	0	0	. .	0.1	0.2	0.7
25	0	0	0	0	. .	0	0.1	0.9

$P_{Off}(S_{t+1} | S)$:

	16	16.5	17	17.5	. . . .	24	24.5	25
16	0.9	0.1	0	0	. . . .	0	0	0
16.5	0.7	0.2	0.1	0	. . . .	0	0	0
17	0	0.7	0.2	0.1	. . . .	0	0	0
17.5	0	0	0.1	0.2	. . . .	0	0	0
.								
.								
.								
.								
24	0	0	0	0	. . . .	0.2	0.1	0
24.5	0	0	0	0	. . . .	0.7	0.2	0.1
25	0	0	0	0	. . . .	0	0.7	0.3

- Rewards (We use costs instead of rewards): c (On) = 3 , c (Off) = 0.01

Cost Model Analysis Research

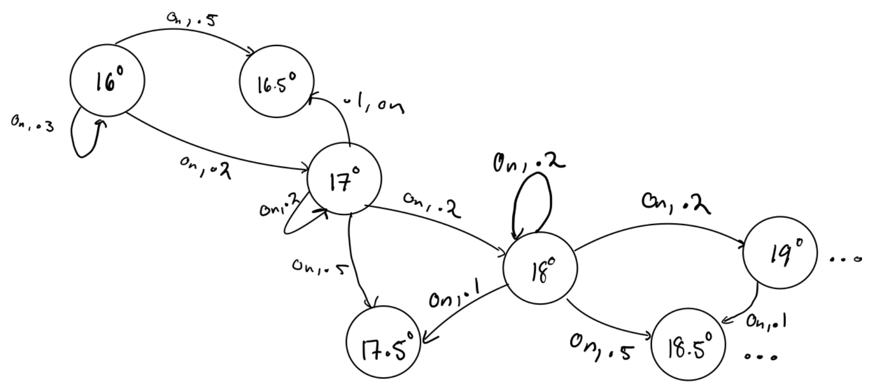
While there are different ways to implement a cost of an action whether that be based on kilowatt-hour, time, or money, we decided to represent the cost of an action in monetary terms. We want to model this project as close to real-life as possible for a user. Utility bills are a monthly recurrence and monitoring the costs of heating are important to the average user in California, United States. We decided to look at California as a model since both my partner and I are from there. Heating can be produced using natural gas, heat pump, electricity, or oil. However, in California for instance the most common heating method is natural gas with 90 percent of furnaces and water heaters running on propane or gas (Natural Resources Defense Council). According to three sources we researched, **it costs on average 3% of your total heating bill for that month to heat by one degree** (Energy Hub, Alps Comfort Air, Novak Heating). For this reason, the cost associated with the action of having the heating on is going to be 3. Looking at the heat being off, in the real world to not turn it on would be zero in monetary terms but this would not work with the implementation of our MDP. The calculations for the optimal policy would be incorrect since using zero in computations would not be truly reflective of the model. So, the cost for the heating being off will be 0.01 since it is a number close to zero but not zero itself. However, this number can vary slightly on a couple of factors: having an energy-efficient furnace and how cold it is outside.

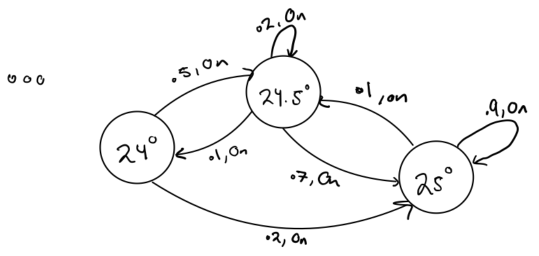
We had trouble finding data that spoke to the quantity of how many Californians have energy-efficient furnaces, but we had already established that most Californias are using natural gas which isn't that energy-efficient. There is currently a "State Implementation Plan" in place to ban natural gas furnaces and heaters (Los Angeles Times). Looking at the second factor, the cost of heating is dependent on the outside temperature. This could be anywhere from 16 degrees Celsius to 25 degrees Ceclius.

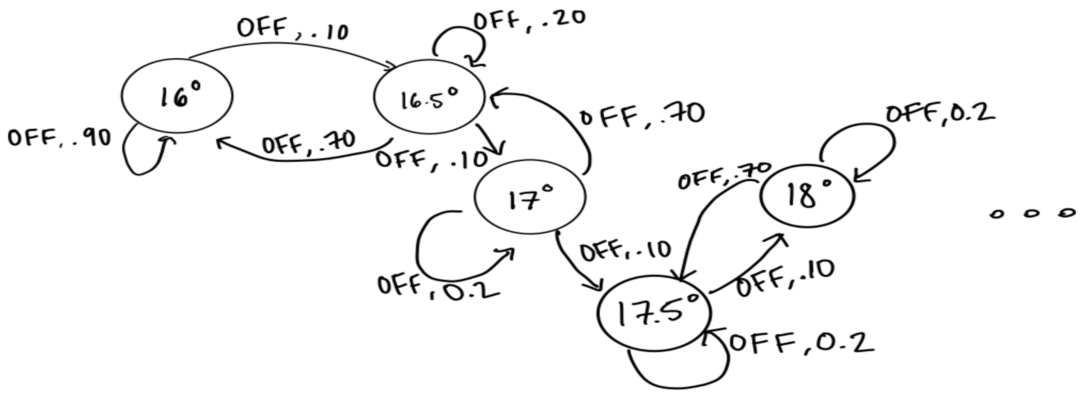
Optimal Policy

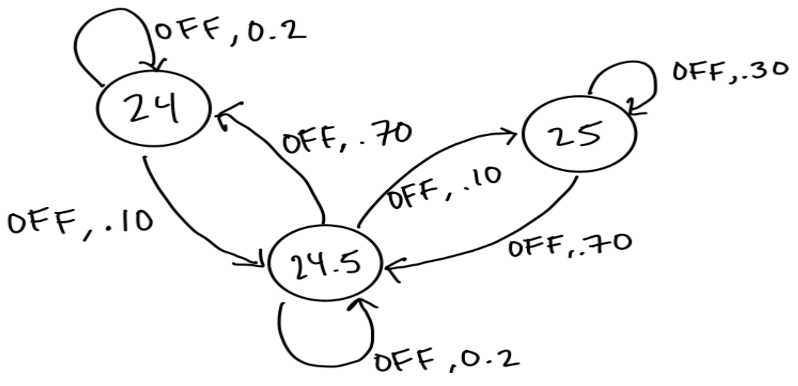
Project Phases

Design









Design continued…

Most of our time was spent in the design stage in order to make the transition to coding more seamless. After going through the lab specifications and regrouped the information into current circumstances, a goal state, probabilities associated with the heat being on and off, and edge cases to consider (See Appendix 1). Given this foundation, we next identified the states, actions, probabilities, and costs (See Cost Model Analysis for more detail). We found it helpful to represent the MDP Model visually with graphs as seen above to understand the transitions.

The next part of our design was to work on the Bellman equation which we learned in class can determine the minimal cost for an action if it makes the optimal decision at the current state and the subsequent ones, thus aiding in the search for the optimal policy. This was done out by hand as similar to the steps above (See Appendix 2).

Implementation

We decided to implement our logic using PyCharm and using a shared repository in GitHub since we were familiar with this workflow. Based on our graph in the design section we needed a way to represent states, actions, probabilities associated with actions, and transitions. As seen in class, we created separate classes for states, actions, and transitions then used containers for each one because multiple objects are needed in our process. In the container classes we also included magic methods for easy computation so our objects could act as built-in types.

The next step was to implement the Bellman equation that was originally done on paper. To be able to access the probabilities for the heating being on and off (not including the edge cases) we stored these in a dictionary to be easily accessible.

Testing

We implemented many different testing strategies in order to ensure that our functions were working correctly. Firstly, we used many print statements when writing our code for debugging purposes. We printed out specific values, lists, and dictionaries in order to determine which loop our code was producing errors. After multiple rounds of printing different values, we finally got the Bellman equation function to work correctly. Once the Bellman equation was functioning as it should, we began to test the equation for accuracy. Inputting different states, probabilities, and costs to see if the Bellman Equation was outputting the correct values, and whether those values began duplicating themselves at any point. There were some bugs in our code for certain iterations, like that the goal state of 22 degrees celsius must always be set to zero. We then made some changes to the code in order to take this fact into account. Once this was corrected, we began to see the correct output for all the iterations of the Bellman Equation. We also tested to see how changing the costs (using costs based on different factors energy consumption, the financial cost, and time) would affect the Bellman equation that we coded. Overall, these are all the different methods of testing that we used in order to make sure that our Bellman Equation was functioning correctly, and we used the same testing strategies when determining whether our optimal policy function was working correctly as well.

Budget

Costs to consider for a financial estimate of how much it would take to implement this project would be for a software engineer and the technical setup required like a laptop. We are going to base the budget from California since that is also the same origin for our Cost Analysis Research. The average software engineer makes about 60 USD per hour (Zippia). If one were to hire him full-time for a week (40 hours) which should be enough time for someone who has experience in the tech industry with AI, that would cost 2,400 USD. For a laptop one should budget 600 USD (considering options from Hewlett-Packard and Windows not Mac). All in all for the development and software alone one should budget 3,000 USD.

Conclusions

Some challenges we had included choosing the costs for actions, designing the Bellman equation, and implementing the optimal policy function. Choosing the costs for the on/off actions was difficult for us because we had trouble finding sources and analyzing them. We understood that the cost of the off function would be minimal but we had trouble using the numbers and data we collected on the internet to combine that into one singular number that would define the cost. Another difficulty we faced with this assignment was designing the Bellman equation in our code. We were stuck with how we could store and access the previous iterations, to then use and the value input for the current iteration. We were stuck on this for a while, and then decided to use a dictionary to store the values, as searching and retrieval is more efficient with dictionaries. We also ran into issues when trying to implement our optimal policy function. We wrote it out on paper and understood the different components but again had trouble accessing the values from the value iteration with the Bellman Equations. Overall, the overarching problem that we had was being able to transform what we had written out on paper into well-written, understandable code; however, after much testing we were able to create functions that worked as expected.

Nevertheless, our implementation may be different from others because of how we implemented the Bellman Equation. This may be different because we decided to use key-value pairs in a dictionary to store values from different iterations. This is overall a beneficial implementation because when dealing with large data sets, it is more efficient to retrieve values from a dictionary over other data structures.

Examining the Lab Project itself we think it was interesting to use the Markov Decision Process and topics that we learned in Lecture since we go to apply them directly to a real-life problem. Although, we feel that the project could have more of a real-life application. For instance, building a game like PacMan or 2048 from the ground up might be more interesting while at the same time maintaining AI concepts of Heuristics and Optimization strategies (AI UC Berkeley, Maksutov). If one is worried that these might be too common then I believe some changes could be made to existing ones. We feel that when there is an intersection of information we learn about in courses and a problem that employs those concepts that is directly related to us is where the excitement happens. For instance, in a previous course of ours where we learned to program in C language our end project was building a server. This was highly interesting to many students since we were able to use what we learned during the semester and program something we use everyday.

Appendix

Appendix 1

Step 1:

- identify tasks
  - ↳ some may require software development

- goal temp = 22°C
- actual temp = 16°-25° C (in .5° intervals) [R1]
- time = 00:00-01:30 (30 min intervals)
- Heat on (30 mins) → temp += .5° (50% of the time)
  - → temp += 1° (20% of the time)
  - → temp = temp (20% " ")
  - → temp += -.5° (10% " ")

Edge Cases:

- Heating on
  - 16°:
    - 30% temp will not change $P(16° \mid 16°, HeatOn) = .3$
    - 50% temp will increase by .5° c $P(16° \mid 16.5, HeatOn) = .5$
    - 20% temp will increase by 1° c $P(16° \mid 17°, On) = .2$
  - 25°:
    - 10% temp will go down by -.5° c $P(25° \mid 24.5°, On) = .1$
    - 90% temp will stay the same $P(25° \mid 25°, On) = .9$
  - 24.5°:
    - 70% temp will go up by .5° C $P(24.5° \mid 25°, On) = .7$
    - 20% temp will stay the same $P(24.5° \mid 24.5°, On) = .2$
    - 10% temp will go down by -.5° $P(24.5° \mid 24°, On) = .1$
- Heating off
  - every 30 mins
    - 70% temp goes down by -.5°c
    - 10% temp goes up by +.5° c

- 20% temp stays the same

Edge cases:

- 16°c :
  - 90% of the time temp stays the same $P(16^{\circ} | 16^{\circ}, \text{off}) = .9$
  - 10% of the time the temp goes up $P(16^{\circ} | 16.5^{\circ}, \text{off}) = .1$

- 25°c :
  - 30% of the time temp stays the same $P(25^{\circ} | 25^{\circ}, \text{off}) = .3$
  - 70% of the time temp goes down $P(25^{\circ} | 24.5^{\circ}, \text{off}) = .7$

Appendix 2

$$V(16) = \min\Big[c(on) + \sum P(16|16,on)\,V(16) + P(16.5|16,on)\,V(16.5) + P(17|16,on)\,V(17)$$
$$c(off) + \sum P(16/16, off)\,V(16) + P(16.5|16,off)\,V(16.5)\Big]$$

$$V(16.5) = \min\Big[c(on) + \sum P(\overset{\text{going down by .5°C}}{16}|16.5)\,V(16) + P(\overset{\text{stay the same}}{16.5}|16.5)\,V(16.5) + P(\overset{\text{go up by .5°C}}{17}|16.5)\,V(17) + P(\overset{\text{go up by 1°C}}{17.5})16.5) \overset{*V(16.5)}{}$$
$$c(off) + \sum P(16|16.5)\,V(16) + P(16.5|16.5)\,V(16.5) + P(17|16.5)\,V(17)\Big]$$

$$V(17) = \min\Big[c(on) + \sum P(16.5|17)\,V(16.5) + P(17|17)\,V(17) + P(17.5|17)\,V(17.5) + P(18|17)\,V(18)$$
$$c(off) + \sum P(16.5|17)\,V(16.5) + P(17|17)\,V(17) + P(17.5|17)\,V(17.5)\Big]$$

$$\vdots$$

$$(24) = \min\Big[c(on) + \sum P(23.5|24)\,V(23.5) + P(24|24)\,V(24) + P(24.5|24)\,V(24.5) + P(25|24)\,V(25)$$
$$c(off) + \sum P(23.5|24)\,V(23.5) + P(24|24)\,V(24) + P(24.5|24)\,V(24.5)\Big]$$

$$V(24.5) = \min\Big[c(on) + \sum P(24|24.5)\,V(24) + P(24.5|24.5)\,V(24.5) + P(25|24.5)\,V(25)$$
$$c(off) + \sum P(24|24.5)\,V(24) + P(24.5|24.5)\,V(24.5) + P(25|24.5)\,V(25)\Big]$$

$$V(25) = \min\Big[c(on) + \sum P(24.5|25)\,V(24.5) + P(25|25)\,V(25)$$
$$c(off) + \sum P(24.5|25)\,V(24.5) + P(25|25)\,V(25)\Big]$$

Works Cited

Alps Comfort Air. “How Much Does It Cost to Increase the Heater by One Degree?”, https://alpscomfortair.com/much-cost-increase-heater-one-degree/#:~:text=What%20that%20means%20is%20if,are%20many%20variables%20to%20this.

EnergyHub. “How Much Is One Degree Worth?" EnergyHub Blog, www.energyhub.com/blog/how-much-is-one-degree-worth/#:~:text=For%20example%2C%20the%20DOE's%20Energy,thermostat%20down%20during%20the%20winter.

Novak Heating. “Cost of Raising the Thermostat 1 Degree.” Novak Heating Blog, www.novakheating.com/cost-of-raising-the-thermostat-1-degree/#:~:text=The%20cost%20of%20a%20thermostat,bill%20of%20about%203%20percent.

Los Angeles Times. “California moves to ban natural gas furnaces and heaters by 2030.”, 23 Sept. 2022, www.latimes.com/business/story/2022-09-23/california-moves-to-ban-natural-gas-furnaces-and-heaters-by-2030.

Zippia.“Software Engineer Salary in California.”, https://www.zippia.com/software-engineer-jobs/salary/california/.

AI UC Berkeley. "Project Overview." AI Berkeley., ai.berkeley.edu/project_overview.html.

Maksutov, Rinat. Towards Data Science. “A Puzzle for AI.” Towards Data Science, https://towardsdatascience.com/a-puzzle-for-ai-eb7a3cb8e599.