David Liu
CSE 415
Project - Option 1

Wicked Problem

1. Project Name: “The Dictator - The Formulation Game”
2. Name: David Liu
    a. Roles: Everything ;)
3. What does the program do?
    a. The program is a formulation that abides by the rules of a “Wicked Problem” and imitates where the money flows in a country ruled by a dictatorship. You play as the dictator where you choose one department to give money too per turn. This action raises the condition of the selected department but may also cause other departments to decay in condition. Your goal as the dictator is to eventually get to a state where all department conditions are either “Good” or “Excellent” delineated by an internal point system where the player can observe and make educated choices when they make their selections.
4. What were the techniques used and a brief description (half a page) of how each technique works?
    a. Since I chose Option 1 for my Project, the AI - Techniques I’ve implemented were:
        i. Uniformed Cost Search - Used to study the resulting state space of several varying formulations in an effort to evaluate the possible state space of the wicked problem in a uniformed procedural way. UCS helped assisted in providing a baseline of average states traversed in each formulation.
        ii. A-Star Search - Used to study the resulting state space of varying formulations but with additional information in the way it traverses to a goal state. Varying heuristics will help or hinder the search algorithm from finding a goal state by padding a state depending on certain attributes of the state. The technique works by investigating certain states that are more favored by lower heuristics (so it can be pushed out of the priority queue) in an efforts to find a goal state.
        iii. Heuristics - Implemented several functions that would help or hinder the AI in efforts to guide it to a goal state. It returns a value corresponding to the attributes of a given state. It would give a low value to favor that state or a high value that wouldn’t favor the given state.
5. either a screenshot or a transcript of an interesting sample session;

    Welcome to AStar By David Liu
    Initial State:

Military: Satisfactory with value 100
Food & Agriculture: Satisfactory with value 100
Science: Satisfactory with value 100
Government: Satisfactory with value 100
Housing: Satisfactory with value 100
Congratulations on successfully making a well-rounded dictatorship!
Solution path:

Military: Satisfactory with value 100
Food & Agriculture: Satisfactory with value 100
Science: Satisfactory with value 100
Government: Satisfactory with value 100
Housing: Satisfactory with value 100

Military: Satisfactory with value 100
Food & Agriculture: Good with value 200
Science: Satisfactory with value 100
Government: Satisfactory with value 100
Housing: Satisfactory with value 150

Military: Good with value 200
Food & Agriculture: Satisfactory with value 150
Science: Satisfactory with value 125
Government: Satisfactory with value 100
Housing: Satisfactory with value 150

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 100
Science: Satisfactory with value 150
Government: Satisfactory with value 100
Housing: Satisfactory with value 150

Military: Excellent with value 300
Food & Agriculture: Good with value 200
Science: Satisfactory with value 150
Government: Satisfactory with value 100
Housing: Good with value 200

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 150
Science: Satisfactory with value 100
Government: Good with value 200
Housing: Good with value 200

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 150
Science: Good with value 200
Government: Satisfactory with value 150
Housing: Good with value 200

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 100
Science: Satisfactory with value 150
Government: Good with value 250
Housing: Good with value 200

Military: Excellent with value 300
Food & Agriculture: Good with value 200
Science: Satisfactory with value 150
Government: Good with value 250
Housing: Good with value 250

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 150
Science: Satisfactory with value 100
Government: Excellent with value 350
Housing: Good with value 250

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 150
Science: Good with value 200
Government: Excellent with value 300
Housing: Good with value 250

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 100
Science: Satisfactory with value 150
Government: Excellent with value 400
Housing: Good with value 250

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 100
Science: Good with value 250
Government: Excellent with value 350
Housing: Good with value 250

Military: Excellent with value 300
Food & Agriculture: Satisfactory with value 100
Science: Excellent with value 350
Government: Excellent with value 300
Housing: Good with value 250

Military: Excellent with value 300
Food & Agriculture: Good with value 200
Science: Excellent with value 350
Government: Excellent with value 300
Housing: Excellent with value 300

Military: Excellent with value 300
Food & Agriculture: Excellent with value 300
Science: Excellent with value 350
Government: Excellent with value 300
Housing: Excellent with value 350
Length of solution path found: 15 edges
Total cost of solution path found: 15.0
24146 states expanded.
MAX_OPEN_LENGTH = 17096

6. Brief demo instructions:
    a. Launch the command prompt.
    b. Change directory until you reach the location of all the python files.
    c. Type in: “python3 Int_Solv_Client.py dliu206_WickedProblem_one” without the quotations to run the formulation for you to play.
        i. There are several other formulations with extension two to play with more consistent operators.
7. code excerpt showing some interesting part(s) of your Python code and some explanation of it
    a.
```
MC_combinations = [(100, random.randint(-50, 0), random.randint(0, 50), 0, 0), (0, 100, 0, 0, random.randint(0, 50)),
                   (0, 0, 100, random.randint(-50, 0), 0), (0, random.randint(-50, 0), random.randint(-50, 0), 100, 0),
                   (random.randint(-50, 0), 0, 0, 0, 100),
                   (random.randint(-25, 0), random.randint(-25, 0), random.randint(-25, 0),
                    random.randint(-25, 0), random.randint(-25, 0))]
```
    b. Since Wicked Problems are defined as not having a definitive formulation or solution. The random variables in formulation one pertains to decay of external factors.
8. brief description of what each team member learned in this project
    a. I learned how powerful heuristics could be in finding a goal state.
9. what you would like to add to your program if you had more time
    a. I would like to add machine learning to find an optimal path and compare it to the optimality of some of the best heuristics I could find.

10. citations for any references you used in the project. This should include the names and URLs of any websites that you used and whose ideas or other resources were incorporated into your project. In each case, describe in a sentence what role that website played in your project and what you incorporated from it.
    a. "What's A Wicked Problem? | Wicked Problem." *Stonybrook.edu*. N. p., 2019. Web. 31 May 2019.
        i. This website guided me in understanding what a Wicked Problem was and how I could possibly explore it.

Legend -

- Formulation 1 - Random fluctuating operators that represent decay. A formulation that meets the requirements of the above problem with 6 operators (1 for each department and 1 operator for not allocating money to an department) and 4 conditions thresholded goal value at 200. All departments start with conditional quality 100 and they aren't allowed to select an action that makes them have less than 100 quality in any department.
    - Branching Factor: 6 states
    - Number of possible states: infinity
- Formulation 2 - Operators that are stagnant and aren't random in a range. Everything else matches up with formulation 1.
    - Branching Factor: 6 states
    - Number of possible states: 14400 states
- Formulation 3 - The loss of the self-allocating money operator making it 5 total operators but still retaining 4 conditions. Everything else matches up with formulation 2. Branching Factor:
    - Branching Factor: 5 states
    - Number of possible states: 2880 states
- Formulation 4 - The increase of the goal state so all departments have to reach a conditional quality of 300.
    - Branching Factor: 6 states
    - Number of possible states:14400 states
- H1 - Heuristic 1 - Average - The A-Star algorithm is used to realize the state space of how quick it is to find a goal state using the average of the departmental conditions to traverse the state space.
- H2 - Heuristic 2 - Hamming - The A-Star algorithm is used to realize the state space of how quick it is to find a goal state using the number of the departments not yet having reached the goal threshold condition.
- H3 - Heuristic 3 - High Sum - The A-Star algorithm is used to realize the state space of how quick it is to find a goal state using the sum of all departmental conditions to traverse the state space.

- H4 - Heuristic 4 - Inverse Average - The A-Star algorithm is used to realize the state space of how quick it is to find a goal state using the average of the formula 1 / Average to traverse the state space.
- H5 - Heuristic 5 - Inverse Sum - The A-Star algorithm is used to realize the state space of how quick it is to find a goal state using 1.0 / sum of all the departmental conditions to traverse the state space.
- H6 - Heuristic 6 - Reverse Hamming - The A-Star algorithm is used to realize the state space of how quick it is to find a goal state using the number of the departments having reached the goal threshold condition.
- UCS - Uniformed Cost Search - Traversed the state space to find a goal state without the use of a heuristic.

Formulation	**H1**	**H2**	**H3**	**H4**
Formulation 1	Length of solution path found: 9 edges 86.6 states expanded. MAX_OPEN_LENGTH = 104.2	Length of solution path found: 9 edges 86.6 states expanded. MAX_OPEN_LENGTH = 104.2	Length of solution path found: 74 edges 7667 states expanded. MAX_OPEN_LENGTH = 10275	Length of solution path found: 9.6 edges 286.6 states expanded. MAX_OPEN_LENGTH = 214
Formulation 2	Length of solution path found: 19 edges 505 states expanded. MAX_OPEN_LENGTH = 557	Length of solution path found: 9 edges 68 states expanded. MAX_OPEN_LENGTH = 65	Length of solution path found: 19 edges 671 states expanded. MAX_OPEN_LENGTH = 725	Length of solution path found: 19 edges 11340 states expanded. MAX_OPEN_LENGTH = 12004
Formulation 3	Length of solution path found: 9 edges 76 states expanded. MAX_OPEN_LENGTH = 64	Length of solution path found: 9 edges 68 states expanded. MAX_OPEN_LENGTH = 55	Length of solution path found: 9 edges 76 states expanded. MAX_OPEN_LENGTH = 64	Took too long to compile.
Formulation 4	Length of solution path found: 15 edges 9293 states expanded.	Length of solution path found: 15 edges 1223 states expanded.	Length of solution path found: 15 edges 24146 states expanded.	Took too long to compile. Assume longer than 24146

	MAX_OPEN_LENGTH = 6587	MAX_OPEN_LENGTH = 790	MAX_OPEN_LENGTH = 17096	states expanded.

Formulation	**H5**	**H6**	**UCS**
Formulation 1	Length of solution path found: 8.2 edges 190.6 states expanded. MAX_OPEN_LENGTH = 173.4	Took too long to compile.	Length of solution path found: 9 edges 240.4 states expanded. MAX_OPEN_LENGTH = 183.8
Formulation 2	Length of solution path found: 9 edges 148 states expanded. MAX_OPEN_LENGTH = 104	Length of solution path found: 9 edges 154 states expanded. MAX_OPEN_LENGTH = 116	Length of solution path found: 9 edges 146 states expanded. MAX_OPEN_LENGTH = 99
Formulation 3	Length of solution path found: 9 edges 142 states expanded. MAX_OPEN_LENGTH = 81	Length of solution path found: 9 edges 135 states expanded. MAX_OPEN_LENGTH = 88	Length of solution path found: 9 edges 136 states expanded. MAX_OPEN_LENGTH = 79
Formulation 4	Length of solution path found: 15 edges 24146 states expanded. MAX_OPEN_LENGTH = 17096	Length of solution path found: 15 edges 2761 states expanded. MAX_OPEN_LENGTH = 1541	Length of solution path found: 15 edges 1922 states expanded. MAX_OPEN_LENGTH = 804

Notes:

- Formulation 1 data is calculated by taking the average of 5 executions.

My findings:

- Bad heuristics can exponentially decrease the optimality of finding a goal state (using UCS as a comparison)

- Good heuristics can exponentially increase the optimality of finding a goal state (using UCS as a comparison)
- Hamming outperforms any other heuristic performed
- Heuristics retain admissibility as long as they never over-estimate the distance to a correct goal state.
- Mathematically calculating the values associated with a state isn't as appealing of a heuristic as to one that measures how close it is to a goal state. (Hamming)
- The state space is calculated using factorials which drastically increase or decrease the size of states needed to be traversed. A small state space problem doesn't retain the same complexity of a problem but is harder to solve.
    - Increasing/Decreasing the complexity of this given wicked problem, I was able to evaluate the state space by decreasing the number of operators or increasing the goal threshold.
    - Branching factor was calculated by the max number of operators able to be executed at a given state.
    - The total number of states in a state space was evaluated by: (num operators - 1)! * (num departments)!
        - Reducing an operator in Formulation 3 eliminated a * 5 multiplier in the state space or 11520 states.
    - Since the decay is never fixated in formulation 1, the conditional quality of the 5 departments creates a large state space.
    - Raising the goal state drastically increases the number of states needed to be traversed in state space.
- Some heuristics faired better than UCS but some fared worse than UCS
    - Based on the state space (Formulation 3), you can see that heuristics 1,2, and 3 faired better than a state space traversal with no additional information. On the other hand, you can see that heuristics 1,2, and 3 did not fair that well in formulation 2.
    - Making heuristics also takes computational time to justify the use of a heuristical function in path-finding. Since all of my heuristic functions traverse the states is O(n) time, it's still optimal for path-finding but I could imagine that more complex heuristics would require larger trade-offs.