EECS 391 Written Homework 1

William Koehrsen wjk68

Tuesday, September 19

1. One example of an AI application is the real-time translation of smartphone camera images provided by Google Translate. Originally developed by Quest Visual as a stand-alone application known as Word Lens, this system translates text found in a camera image and overlays the translation over the original text on the smartphone display. The real-time text translation must solve numerous problems found in AI, including visual text capture through Optical Character Recognition (OCR), machine translation, image manipulation, and displaying information in an augmented reality overlay. The system relies on two different machine learning technologies: pattern matching using nearest neighbors for OCR and Google's translation model which uses a recurrent neural network to encode and decode sentences from one language to another. In terms of the PEAS intelligent agent model, the Word Lens system can be described as follows:
Performance Metric: The AI system is evaluated based on the usefulness of its translations. It is not intended to be used for handwritten text or paragraphs, but is mainly focused on printed, easily discernible lettering such as that found on signs. The translation does not need to be perfect, but it must convey the meaning of the sign in a language familiar to the user.
Environment: The environment in which the application operates consists of any possible text a user might capture and the smartphone itself. The environment is single agent, partially observable, stochastic, episodic, static, continuous, and unknown.
Actuators: The actions that can be taken by the Word Lens application are to overlay the translation on top of the original image. The system can do this by manipulating the pixels on the user's smartphone screen.
Sensors: The agent has access to the raw pixel data from the smartphone camera.

   Optical Character Recognition (OCR) generally works through one of two methods: pattern recognition and feature extraction. Pattern recognition takes the input image and compares the text against a dictionary of images that have already been labeled. The system then uses nearest neighbors or another clustering algorithm to find the most likely match for the letters in the image. Feature extraction is a more sophisticated technique that involves trying to identify the underlying patterns of the text in an image. For example, a vertical line that intersects at the top with a horizontal line is most likely a "T". Feature extraction programs often use deep learning techniques such as convolutional neural networks that are able to identify structures within an image. The Google Translate application uses a pattern recognition system that matches text in an image to a dictionary of existing fonts. This is done primarily for speed and because the application was designed to be used with printed text on signs which can be more easily matched to typefaces than handwriting.

   The translation itself relies on the Google Translation algorithm which, in 2016, was updated from a statistical machine translation engine to a deep neural network with a significant improvement in accuracy. In particular, the Google Translate engine is a recurrent neural network that uses Long Short-Term Memory (LSTM) cells. The statistical machine translation approach translated phrases one or several words at a time, ignoring the overall structure of the

sentence. A recurrent neural network can “remember” what is has read previously, enabling the translation of entire sentences at a time. Furthermore, the neural machine translation system enables Google to complete translations directly from one language to another rather than going from language 1 to English to language 2. The recurrent neural network takes as input the sentence in the original language, encodes it into a vector, and then sends the encoded vector to an attention function that can focus on different parts of the vector by assigning weights based on perceived importance. After passing through the attention function, the encoded vector is sent to a decoder where it is translated from a vector into a sentence in the desired language. The recurrent neural network is trained based on millions of examples of already translated phrases and sentences and is noticeably better than the statistical approach which could not account for the entire content of a sentence.

The final part of the application is overlaying the translation over the original text on the smartphone screen. As this is done in real-time, it requires scaling down the neural networks behind the translation to a reasonable size that can run on a smartphone. The system can overlay the translation on the display because it has already identified the location of the words through OCR. Real-time translation on a smartphone would have seemed like science fiction 10 years ago, but with the advent of more powerful CPUs and even GPUs on smartphones, and efficient scaled down neural networks, Google and Quest Visual were able to create a usable artificial intelligence translation system. Similar features such as instant speech translation have been integrated into the Google Translate application taking advantage of similar principles. The real-time image translation system is a successful implementation of AI that solves several of the more difficult problems in AI using machine learning techniques and improved hardware.

Sources: http://www.explainthatstuff.com/how-ocr-works.htmlhttps://www.technologyreview.com/s/423736/a-new-reality/
https://arxiv.org/pdf/1609.08144.pdf
https://techcrunch.com/2014/05/16/google-has-acquired-quest-visual-the-maker-of-camera-based-translation-app-word-lens/
https://www.blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/
https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html

2. **Exploring the subsurface oceans of Titan**

Performance Measure: Percentage of the oceans fully explored. Fully explored would be defined by the mission scientists based on the required level of information that needs to be collected to characterize the oceans.

Environment: Subsurface oceans of Titan, operators on Earth, other robotic probes in the oceans, communications satellites

Actuators: Robotic arms to manipulate samples, propellers or jets for locomotion, lights for illuminating surroundings, communications equipment for talking to Earth and other robotic vehicles

Sensors: Science instruments for gathering data, cameras capturing optical information, acoustic sensors for audible input, receiving equipment for incoming transmissions, location tracking system, sensors for monitoring condition of craft

The environment would be **partially observable, multiagent, stochastic, sequential, dynamic, continuous, and unknown**. This is the most difficult environment in which to operate!

Representation: the environment would need to be transformed into a map, showing areas that have already been explored, areas that need to be explored, and the craft's location within the map. This is a problem of Simultaneous Location And Mapping (SLAM) as discussed in class with regards to the DARPA Grand Challenge. The vehicle must create a representation of the environment while simultaneously tracking its own position in the environment. The map could be represented as a 3-D numeric or binary matrix of latitude, longitude, and altitude above the ocean floor. The matrix could also be flattened into a 1-D vector. The positions visited would have a value of 1, the positions yet to be visited 0, and the current location of the craft 2, or the positions could be recorded in binary using 2 bits. The fidelity of the map (the increments in latitude, longitude, and height) that would be recorded would be up to the engineers designing the mission. As the mission progressed, the matrix would be updated when the craft visited a particular [latitude, longitude, altitude] value. Other craft could also be represented on the matrix (perhaps taking a value of4). Recording elevation would allow for a more thorough exploration of the oceans because the craft could keep track of the depths it had visited. The final performance metric would then be the percentage of the oceans that had been thoroughly explored (thoroughly would be defined by the mission scientists).

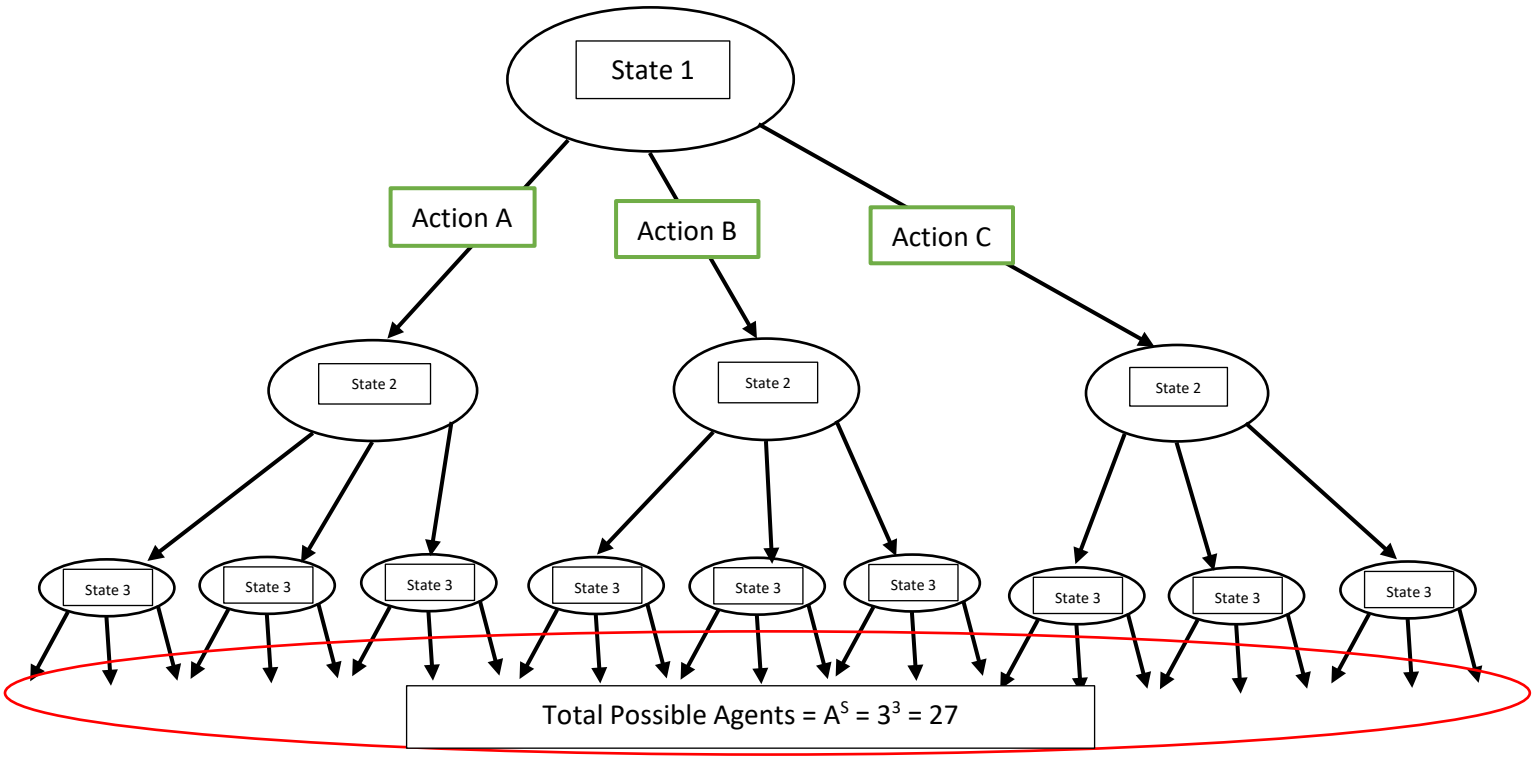
Bidding on an item at an auction
Performance Measure: Has item been purchased at a price below a specified threshold. The threshold could be the fair market value of the item or up to the human user
Environment: Other bidders (both human and other AI), auctioneer or auction platform (ebay)
Actuators: Screen for displaying current bid, communication system for submitting bids
Sensors: Keyboard entry/electronic input of current price

The environment would be **partially observable, multiagent, stochastic, sequential, dynamic, discrete, and known.** The environment is slightly simpler than in the previous example because the input, current price, can only take on discrete values, and the rules of the auction are completely known.

Representation: The entire environment could be reduced to a single vector with the current price of the item, the current bid of the AI agent, and the maximum price that should be paid. These could be recorded as simple numeric values. The vector would be updated as the price and current bid change and the maximum price would be persistent (unless the user has a change of mind midway through the auction). This is a simple representation that would still allow the agent to accomplish its goals because if the current price is greater than the current bid of the agent and below the maximum price, the agent should increase its bid by the minimum increment allowed over the current bid. At the end of the auction, the performance

could be assessed by comparing the bid of the agent to the current price of the item and the maximum price that was specified. If the bid of the agent equals the price that the auction ended at and is below the maximum price, then the AI agent has succeeded. More complex representations that take into account the actions of the other agents (how quickly other agents bid, or emotional reactions of human competitors) could inform the algorithm to buy the item at the lowest price possible.

3. Given a discrete, fully observable environment with a simple reflex agent, there will be $\mathbf{A^S}$ possible distinct agents. This can be determined by looking at a decision tree with each level of the tree being a state and the branches representing possible actions in the state. The decision tree for an environment with 3 states and 3 actions is as follows:



4. The total number of agents is $\boldsymbol{A^{S^{k+1}}}$**.** If each agent is given a memory where it can remember the past k states, then the agent becomes a model-based reflex agent as opposed to simple-reflex. This does not impact the number of actions or states available to the agent, but it does change the number of percept sequences. The formula for number of possible agents becomes A^P where P is the number of percept sequences. To determine the number of percept sequences, we need to consider both the possible memory sequences and the current state. With a memory of the past *k* steps and S possible states, the number of possible memory sequences is S^k. The number of total percept sequences is therefore S * S^k or S^{k+1}. As we have defined the number of agents as A^P where P is the number of percept sequences, the total number of agents

is $\boldsymbol{A^{S^{k+1}}}$. This makes sense because when the agent does not have a memory, this simplifies to A^Swhich is the number of simple-reflex agents. As an example, if there are 2 possible states with 2 actions and the agent remembers the past 3 states, there will be 2^3 = 8 memory sequences, $2^4 = 16$ percept sequences, and $2^{16} = 65536$ model-based reflex agents to program.

5. A state space of a problem is defined on page 67 as "the set of all states reachable from the initial state by any sequence of actions." For the 8-puzzle, only half of all possible states are reachable from any given initial state (page 71). The number of all possible states for the 8-puzzle is $9! = 362880$ and the number of possible states reachable from any given initial state is 181440. The number of possible state reachable from a given state for a 15 puzzle is $\frac{16!}{2} = 1.05\ x\ 10^{13}$. To store each state would require a vector of 9 digits. This could be encoded in binary where each number in the vector would be 4 bits. The vector would thus be 36 bits. For the 8 puzzle, multiplying the number of vectors times the memory for a single vector would be $6.53\ x\ 10^6$ bits or 0.779 MB. The memory for the 15-puzzle would be 42.8 TB.

   For a single state, what must be represented is the position of all eight number tiles and the blank tile. This requires a vector with 9 numbers. If each number is encoded as a 4-bit binary number, then each state requires 36-bits to represent.

6. The state space for a 2 x 2 Rubik's cube can be determined by considered the available options for the cube. There are 8 corners, meaning there are 8! ways to arrange the 8 corners. For each of the eight corners, any of the 7 can be independently moved at one time (one corner must remain fixed in space). Each corner can be rotated in 3 directions so the number of possible moves for the corners is 3^7 (A^S). The total number of states would therefore be $8! * 3^7$ except that we do not want to count duplicate states. To account for duplicate orientations of the cube, we need to divide the total number of states by 24, or the number of face tiles on the cube. The final answer is therefore

$$S = \frac{C * A^{C-1}}{F} = \frac{(8! * 3^7)}{24} = 3674160$$

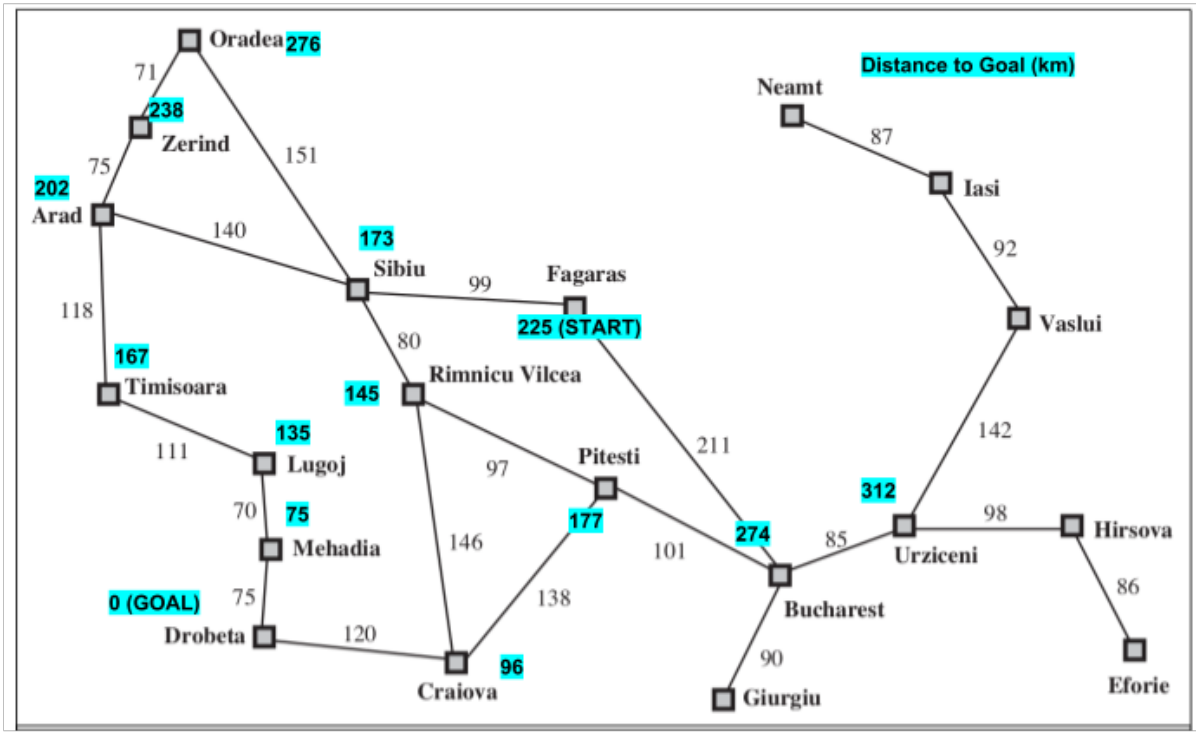
   The cube essentially has 8 tiles, because it has 8 corners that can be rotated. Each corner has three actions and three possible orientations or states.

7. A. There are infinite states if x and y are continuous because even a differential movement in either direction would result in a change of state. If x and y are discrete integers, then there would be x * y possible states. There are infinite paths to the goal because any path can be made arbitrarily long.
   B. The shortest distance between two points is a straight line on a 2D plane. Therefore, traveling from one vertex to any other vertex must travel along straight lines that touch at least some of the other vertices. To navigate around an obstacle in the environment, the shortest path is to follow the outline of the obstacle, which will touch at least one vertex of the polygon.

The state space consists of the start and the end points, plus the total number of vertices of the shapes. There are 35 possible states including the start and goal state.

8. A. Breadth-first search is a special case of uniform-cost search only when the path costs are equal to every node. As breadth-first search expands all the nodes at a given level of the graph tree without consideration for path costs, it will be equivalent to a uniform-cost search if all the path costs are the same. Uniform cost expands the lowest cost path first, and if all the path costs are equal, it will expand all the nodes at a given level of the tree before moving down to the next level because none of the paths are superior.
B. Depth-first search is a special case of best-first search where the evaluation function is $f(n) = -depth(n)$. DFS expands the deepest node first and this evaluation function will place the node with the greatest depth on the top of the priority queue.
C. Uniform-cost search is a special of A* search where the heuristic is equal to 0 for all nodes. This implies that the estimated path from the node to the goal is unknown and the nodes will be ordered on the priority queue by the path cost to reach the node. The definition of Uniform-Cost search is that nodes are placed on the queue in order of path cost, so A* search will be equivalent under the condition that h(n) = 0 for all nodes.

9. Map with straight line distances determined from Google Maps (https://goo.gl/w3oXFH)

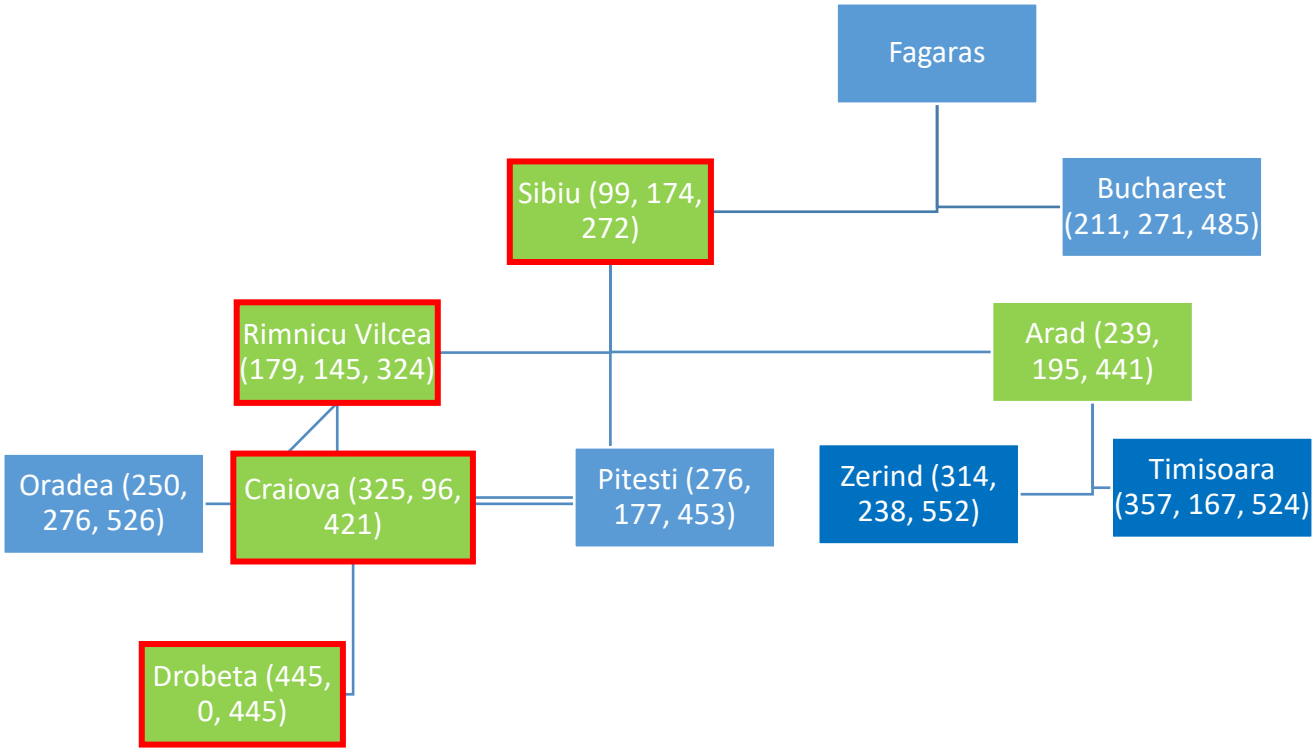


City	Level	g(n)	h(n)	f(n)
Fagaras	0	-	-	-
Sibiu	1	99	173	272
Bucharest	1	221	274	485
Rimnicu Vilcea	2	179	145	324
Arad	2	239	202	441
Oradea	3	250	276	526
Craiova	3	325	96	421
Pitesti	3	276	177	453
Zerind	3	314	238	552
Timisoara	3	357	167	524
Drobeta	4	445	0	445

The nodes expanded in order are **Sibiu, Rimnicu Vilcea, Craiova, Arad, and then Drobeta.**

The final path is **Fagaras, Sibiu, Rimnicu Vilcea, Craiova, Drobeta.**

The diagram below shows nodes expanded in green and the final path selected outlined in red. The values in parenthesis are (g(n), h(n), f(n)) for each node.



10. Give the name of the algorithm that results from each of the following cases:
    A. Local beam search with k = 1 is equivalent to **hill climbing search.** On each iteration, only one node, the node with the lowest evaluation function, will be kept in memory, and the algorithm will choose the best successor move at each state resulting in a climb up the hill towards the best state. The algorithm does not look beyond one successor ahead and is therefore susceptible to becoming stuck in a local minimum.
    B. Local beam search with one initial state and k = ∞ retained at each step is equivalent to **breadth first search**. Local beam with k = ∞ will evaluate all of the successor states for the current state and will not remove any of the successor states no matter their total cost. This is equivalent to breadth-first search will expands every node at a given depth and orders the priority queue by first-in, first-out.
    C. Simulated annealing with T=0 is equivalent to **first-choice hill climbing search** (gradient ascent). In simulated annealing, successor moves are chosen at random. If a move decreases the cost (i.e. is better than the current state), it is selected and moves that increase the cost (i.e. they are worse than the current state) are selected with probability $e^{\Delta E/T}$. If T=0, then nodes that increase the cost function will have no chance of being selected and the algorithm will choose the first node that results in a better state. The algorithm will never choose actions that increase the cost function and therefore is susceptible to becoming stuck in a local minimum rather than finding the global minimum.
    D. Simulated annealing with T=∞ is equivalent to a **random walk.** In simulated annealing, successor moves are chosen at random. If a move decreases the cost, it is selected, and moves that increase the cost are accepted with probability $e^{\Delta E/T}$. If T = ∞, then this probability = 1 and all moves are equally likely to be selected making the selection of the next state purely random.
    E. A genetic algorithm with N = 1 is equivalent to a **random walk**. In a genetic algorithm, successor states are generated by combining two parents as opposed to taking one action from a current state. If the size of the population is 1, then the two parents will be identical and the crossover operation will copy the information of the one parent. The mutation operation will the introduce a small change in state, resulting in a random walk, or a random evolution of the single individual. The fitness function does not factor in with a population size of 1 because there are no individuals to order by fitness.