Thomas Huzij
tph2109@columbia.edu

Homework 1

2.4) For each of the following activities, give a PEAS description of the task environment and characterize it in terms of the properties listed in Section 2.3.2.

a. Playing soccer

To describe this task environment, the task must be broken down into the individual components of PEAS. The performance measure is the desirable quality one can aspire to when playing soccer. The performance measure can be scoring the most goals, maintaining possession of the ball for longer, or the number of successful passes. The environment dictates the conditions in which an agent would play soccer. The environment could include the type of field, the size of the field, the weather, and the other players. The actuators allow for the agent to perform actions in the scope of the task. The actuators can include limbs, such as legs and arms, for running and kicking, and gesturing respectively, the head for headbutting, and the voice to communicate with teammates and the referee. The sensors allow new information to be interpreted. The sensors can include the eyes, ears, and any touch sensation on the body.

The task is partially observable because the agent cannot see the whole field and what every other player is doing at any given time. It is multiagent because there are competing agents each trying to maximize the performance measure. It is stochastic because the agent cannot predict every aspect of next state from the current state, such as whether or not their ankle will twist, or it will start raining. It is sequential because a decision may affect all future decisions, such as whether possession is lost. It is clearly dynamic because the state of the game is constantly changing as the agent decides how to proceed in the game. It is continuous because the time and the location of the agent are continuous.

b. Exploring the subsurface oceans of Titan

This task can be broken down into the individual components of PEAS. The performance measure can be finding new life forms or compounds. The environment can be comprised of different features such as how hostile the oceans are to exploration or how conducive they are to generating life. The actuators are the parts of the space exploration apparatus used, such as the thrusters, the mechanical claw, the radio transmitter, and the propeller. The sensors can include the camera, the radio receiver, various sensors to interpret physical input from the environment such as the water current.

The task is partially observable because the entire ocean cannot be tracked at once. It is single agent because there is no sense of competition but rather the agent is alone in trying

to maximize its own performance measure. It is stochastic because the current state of the ocean as interpreted by the agent cannot necessarily allow the agent to predict the next state. It is sequential because each part of the ocean needs to be explored and thus it matters where the agent has already been. It is dynamic because the ocean is ever-changing while the agent deliberates about which way to explore next. It is continuous because both time and space are continuous in this task environment.

2.8) See README

3.19) What is an appropriate search strategy to find a path between two web page URLs? Is bidirectional search a good idea?

An obvious search strategy is to use breadth-first search. Treat the first web page as a node and all the web pages that it has links to as neighboring nodes. Using a FIFO queue, visit each linked web page one level of depth at a time, placing all of its successor web pages on the queue as well, until the goal web page is discovered. Bidirectional search can be a really good idea if the links between all the web pages are two-way (if page A has a link to page B, page B has a link to page A). This would create an opportunity for a basic bidirectional search. However, the Internet does not necessarily look like this. Most links to web pages are not reciprocated. However, bidirectional search only requires that we have a method to discover the predecessor node to any other node. If we had previously crawled the entire web, we would know exactly which pages link to any given page. This is precisely what search engines do to measure and rank the score of documents in their results. It would be terribly inefficient to do this from scratch each time but the result of the web crawl can be cached for future searches.

3.23) Trace the operation of A* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f, g, and h score for each node.

A*: $f(n) = g(n) + h(n)$, where $g(n)$ is the path cost to reach node n, and $h(n)$ is the cost to get from node n to the goal. Perform a uniform cost search where the the next node to expand is based on $f(n)$. Each time a node is expanded, add its successors to the queue of potential nodes.

Start: Lugoj
Goal: Bucharest

The following is a set of states for the queue that represents the trace of A* search. Note that certain nodes can appear in the same state twice because the search can return to a previously expanded city. Each state is numbered.

1. Lugoj ($f(n) = g(n) + h(n) = 0 + 244 = 244$)
2. Mehadia ($70 + 241 = 311$), Timisoara ($111 + 329 = 440$)

3. L (140 + 244 = 384), Drobeta (145 + 242 = 387), T (111 + 329 = 440)
4. D (145 + 242 = 387), T (111 + 329 = 440), M (210 +241 = 451), T (251 + 329 = 580)
5. Craiova (265 + 160 = 425), T (111 + 329 = 440), M (210 + 241 = 451), M (220 + 241 = 461), T (251 + 329 = 580)
6. T (111 + 329 = 440), M (210 + 241 = 451), M (220 + 241 = 461), Pitesti (403 + 100 = 503), T (251 + 329 = 580), Rimnicu Vilcea (411 + 193 = 604), D (385 + 242 = 627)
7. M (210 + 241 = 451), M (220 + 241 = 461), L (222 + 244 = 466), P (403 + 100 = 503), T (251 + 329 = 580), Arad (229 + 366 = 595), R (411 + 193 = 604), D (385 + 242 = 627)
8. M (220 + 241 = 461), L (222 + 244 = 466), P (403 + 100 = 503), L (280 + 244 = 524), D (285 + 242 = 527), T (251 + 329 = 580), A (229 + 366 = 595), R (411 + 193 = 604), D (385 + 242 = 627)
9. L (222 + 244 = 466), P (403 + 100 = 503), L (280 + 244 = 524), D (285 + 242 = 527), L (290 + 244 = 534), D (295 + 242 = 537), T (251 + 329 = 580), A (229 + 366 = 595), R (411 + 193 = 604), D (385 + 242 = 627)
10. P (403 + 100 = 503), L (280 + 244 = 524), D (285 + 242 = 527), M (292 + 241 = 533), L (290 + 244 = 534), D (295 + 242 = 537), T (251 + 329 = 580), A (229 + 366 = 595), R (411 + 193 = 604), D (385 + 242 = 627), T (333 + 329 = 662)
11. **Bucharest** (504 + 0 = 504), L (280 + 244 = 524), D (285 + 242 = 527), M (292 + 241 = 533), L (290 + 244 = 534), D (295 + 242 = 537), T (251 + 329 = 580), A (229 + 366 = 595), R (411 + 193 = 604), D (385 + 242 = 627), T (333 + 329 = 662), R (500 + 193 = 693), C (541 + 160 = 701)

4.1) Give the name of the algorithm that results from each of the following special cases:

a. Local beam search with k = 1

This picks one random spot to start from at the beginning and expands the search from there. This is known as the hill-climbing search algorithm

b. Local beam search with one initial state and no limit on the number of states retained

If there is no limit on the number of states retained in each iteration, then it is as if you can visit every successor of every state instead of just k successor states. This is ever-expanding, one-level at a time and most resembles breadth-first search.

c. Simulated annealing with T = 0 at all times (and omitting the termination test)

If T = 0, the probability of a “worse” successor” being accepted would be non-existent. This means the successors are randomly generated but they can only be accepted if and only if they are better than the current state. This is known as first-choice hill climbing.

d. Simulated annealing with T = ∞ at all times

If T = ∞, any randomly generated successor of a state will be accepted with a probability of 1, regardless of whether it is a better or worse state. This is known as the extremely inefficient purely random walk.

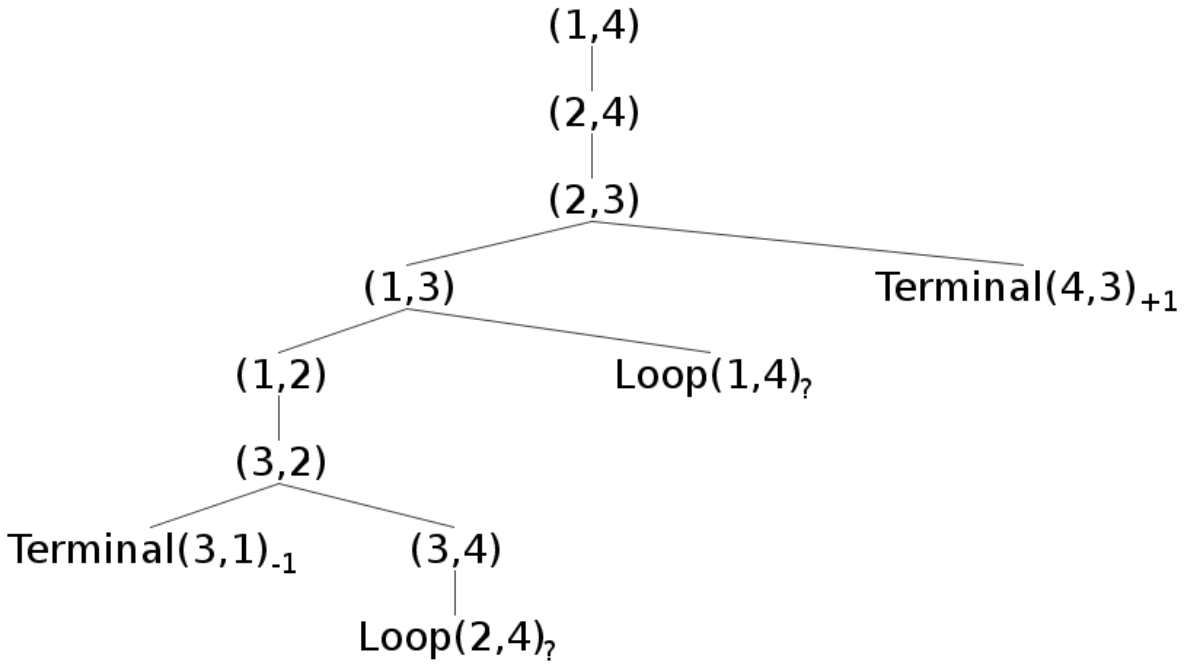
e. Genetic algorithm with population size N = 1

If the population size is 1, when attempting to reproduce, both parents states will have to be the same exact state. Thus when mating, regardless of the crossover point, the same state is returned. Then the only way the state could actually change is by random chance as a result of the mutation. This can be described as a purely random-walk over the set of possible states.

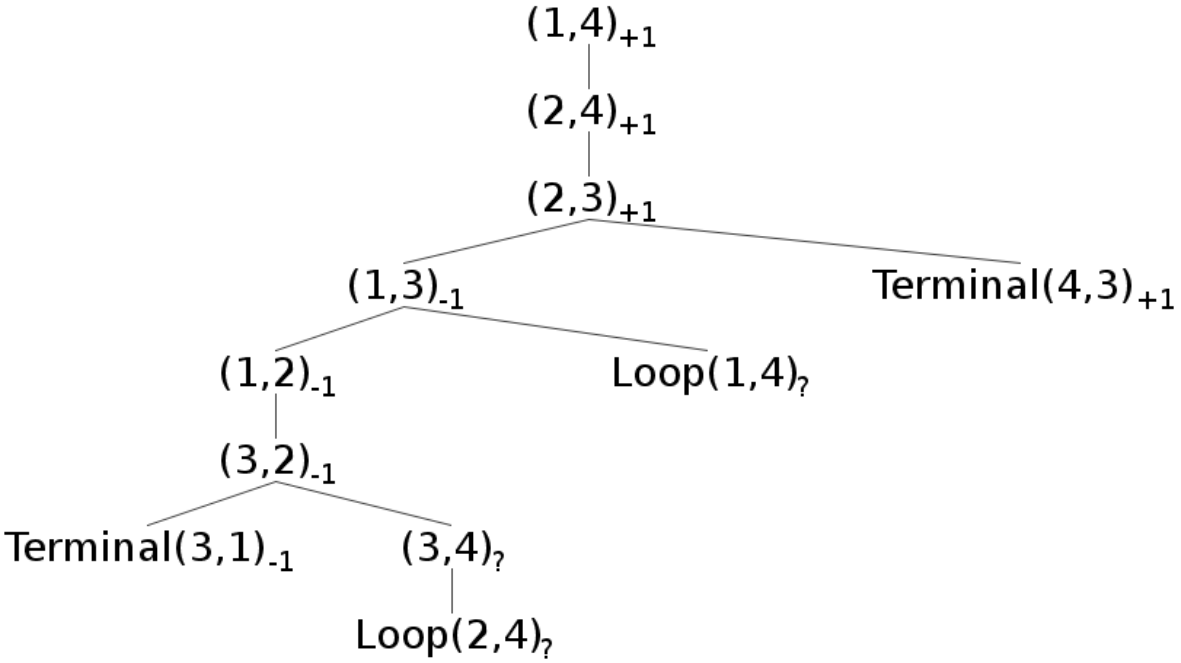
5.8) Consider the two-player game described in Figure 5.17

a. Draw the complete game tree

The drawing software I used did not allow me to draw circles or boxes. Instead, I prepended the word “Terminal” to represent terminal states and the word “Loop” to represent loop states. Additionally, the values of the states are denoted by subscripts on the state pair.



b. Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the “?” values and why.



As you can see, any state that contains a successor with a +1 value somewhere down the branch gets the +1 value (this is because player A went first and gets to go first). Otherwise, the state gets a value of -1. The only exception to this is if the state does not have any successors in its branch that have either a +1 or a -1 value. This can only mean the branch will eventually lead to a loop where the ultimate score of the game is not yet known, so it gets a value of ?. Once you’ve reached a branch that can only lead to a loop, the fate of the state is set. Otherwise, it is obvious that an optimal player will always choose the branch that will lead to a win over a branch that will lead to a loop.

c. Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?

The standard minimax algorithm uses what is essentially depth-first search to traverse the tree and compute the minimax values. This would fail because once the depth-first algorithm reaches a loop state, it will loop indefinitely and never terminate. To fix this, before moving to a successor, check to see if it has appeared in the depth-first traversal before. If it has, then it is clearly part of a cycle. Mark that successor so that it is never traversed again, and follow a different branch instead. It is clear that if a loop is entered, the second time node “Node(1,4)” is seen, it will be obvious that it is leads to a cycle, and it will be ignored. The same is true for node “Loop(2,4)”. Thus the modified algorithm will actually terminate, but it is not guaranteed to give optimal decisions. For instance, if not taking the loop will lead to a loss, the player will want to take the loop so they never have to lose.

d. This 4-square game can be generalized to n squares for any n > 2. Prove that A wins if n is even and loses if n is odd.

For this n-square game to be 2-player, n >= 2. The 2-square game is always a draw because it's impossible to move. In the 3-square game, player A loses because as soon as A moves, B hops over them and ends up in a winning state. In the 4-square game, it is clear from the game tree in (b) that player A wins. Going forward, if it is a 5-square game, the first move that player A makes and the first move that player B make put them in positions that exactly resemble the 3-square game. At this point, player A can either move back or forward. Moving forward will lead to a loss, just like in the 3-square game. However, if A tries to move back, B can move forward even further and force the 3-square game again. However this time, A would have no choice but to move forward and B would win. Any larger n that is also odd will follow the same principle (if a 5-square game eventually boils down to a 3-square game, a 7-square game will eventually boil down to a 5-square game, which we know is a loss for A, and so on with larger n values). This exact same reasoning can be applied to an n-square game when n is even. A 6-square game becomes a 4-square game after A and B's first moves, and so on.

5.12) Describe how the minimax and alpha-beta algorithms change for two-player, non-zero-sum games in which each player has a distinct utility function and both utility functions are known to both players. If there are no constraints on the two terminal utilities, is it possible for any node to be pruned by alpha-beta? What if the player's utility functions on any state differ by at most a constant k, making the game almost cooperative?

For non-zero-sum games, we cannot simplify the value of a state to just a single value because the values are not always opposite. Instead, just like a multiplayer game, a vector of values must be used to represent the value of each state to each user. In the case of a two-player game, a vector of two values, also known as a pair, should suffice to represent the minimax values for each player. The player whose turn it is would simply choose the branch that obtains the vector with the higher value in the corresponding vector index for that player. If there are no restraints on the two terminal utilities, then it is no longer possible to perform alpha-beta pruning. This is because the point of alpha-beta pruning is to eliminate branches that cannot possibly improve the score of one player However, this only applies to zero-sum games because in non-zero-sum games, any one branch can actually be optimal for both players to choose and thus it makes no sense to prune from the perspective of only one user. However, if the player's utility functions on a state differ by at most a constant k, then the amount of optimality for the move can be calculated and thus an optimal result might be the same for both players. This creates a cooperative play situation and it would make sense to prune branches that do not produce optimal results for both players.