Introduction to AI   Test 1   Summer 2024

Not an open-book test. No cheat-sheet. Only you and the writing utensil. You have 2 hours to solve these problems. Some problems might seem impossible, but that is to be expected. If you cannot completely solve it, writing down how you would solve it would give you substantial points, whereas writing only an answer will give you zero. Choose one (or the first two for question 1) sub-question(s) to solve for each question. The maximum score you can get is 40. The lowest combination will be picked if you choose to solve extra problems (for fairness, I will, however, grade all of them for your reference).

Name:____

1. Consider the discrete step maze of size $N \times M$ with a path from the entry point to the exit. You can draw a non-trivial example along with your solution. We know that random walk guarantees to solve this kind of maze (if we let it run for long enough). [10 points]

   (a) Will the algorithm still guarantee to find the solution if **random restart** is imposed? Prove it mathematically. (Hint: If the probability of tossing a coin and getting a head is p, what does that tell us about the existence of the world where we toss a coin and get a head?) [5 points]

   (b) Systematic search like Breath-First Search (BFS) or Depth-First Search (DFS) also guarantees finding the solution in this maze. Why would someone still use random walks to solve the maze? Show that there is/is no advantage to using random walk in place of the systematic search. In other words, find the time complexity, space complexity, and expected run time (you can assume that you have a decent computer that allows sequential operation, or you can think of the case where you only have an 8-bit computer with 32KB of memory) for random and systematic search. (Hint: One of the algorithms here has the worst time complexity in a linear family.)[5 points]

   (c) We know that random walk is inefficient. However, the guarantee of the existence of the solution is really useful. Suppose we want to leverage the random walk to make it more efficient. Think of this as a similar approach to greedy and A* search where we have the ”knob” to turn the way we use heuristic like $U(x) = \alpha f(x) + \beta g(x), \alpha + \beta = 1$. Here, suggest a way to leverage the random walk using the trade-off between $f(x)$ and $g(x)$ and define those two functions in this scenario. Furthermore, define an algorithm that utilizes the $U(x)$ while still being probabilistic (like random walk). (Hint: Maybe for the first part, you want to explore all the paths, but if you find a ”promising” path, you might not want to leave that path but also want to give the benefit of the doubt due to the lack of exploration. Or if you have an idea to do this without the suggested utility function, you can write it down too.) [10 points]

2. Suppose you have a function $f : \mathbb{R} \to \mathbb{R}$ that is very costly to evaluate as a whole (think of this as a simplified version of loss landscape for training neural network), and you want to optimize it. The naive way would be to use the hill climbing method with **random restart**. However, that is similar to a random walk to solve the maze in the sense that a random initialization point is the key to escape the local optima. [10 points]

   (a) Suggest a way to solve this problem using systematic search. In other words, how can we optimize a function or search for the optima, like BFS or DFS, to solve the maze? If you think this is impossible or you can't think of the way, discuss why it is hard or impossible to do so. (Hint: Think about the search space. How large would the search be? Can BFS/DFS reduce the search space? Can you construct a space that is suitable with BFS/DFS? You can safely assume some behavior of the function f, like continuity or smoothness, to help with your argument.) [10 points]

   (b) We have discussed the heuristic search where we can use the Euclidean distance from the entrance to the exit to solve the maze. Can we do something similar here? Can we use a heuristic to find the function's optima? What would be a good heuristic to use? Show it mathematically. (Hint: When you want to optimize a function in general, how would you do it? Now, can you do it iteratively or use the definition there to define a way to show "how far" or "which direction" to go?) [10 points]

Cont.

3. Algorithms like simulated annealing or genetic algorithms are important algorithms but often overlooked. In fact, I initially forgot to write a question or two about those algorithms. So, I will make this slightly easy for you. [10 points]

   (a) Consider the Metropolis-Hasting algorithm, which is one of the crucial algorithms for Bayesian statistics and numerical integration,

Algorithm 1 Metropolis-Hastings Algorithm

```
Input: Target distribution $\pi(x)$, proposal distribution $q(x'|x)$, initial state $x_0$, number of iterations $N$
Initialize: Set $x \leftarrow x_0$
for $i = 1$ to $N$ do
    Sample $x' \sim q(x'|x)$
    Calculate acceptance ratio $\alpha = \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}$
    Sample $u \sim \mathcal{U}(0,1)$
    if $u < \min(1, \alpha)$ then
        Accept the new state: $x \leftarrow x'$
    else
        Reject the new state: $x$ remains unchanged
    end if
    Record the state $x$
end for
Output: Sequence of states $\{x_i\}_{i=1}^N$
```

   How is this similar or different from the simulated annealing? Explain it mathematically. (Hint: Both are Markovian processes.) [10 points]

   (b) We love Maze here, and I don't want your last question to be too hard. Write down a genetic algorithm for solving a maze and argue that your algorithm will perform better than a random walk in terms of the mathematical guarantee of the solution. [10 points]

Cont.

4. Oh, we still have one page of paper left. All right, the last question was a false alarm! Anyways, here are some relatively easy optimization problems. [10 points]

   (a) You can think of Particle Swarm Optimization (PSO) as a parallel version of hill-climbing agents that can communicate with each other. Let's say you are one of the particles; you might heard your friends say, "Hey, here is very deep; I will move closer." You can be egoistic and say, "Mine is even deeper; I will explore more here," or give your friend the benefit of the doubt and move closer to your friend. Design an algorithm like PSO to solve the maze. [10 points]

   (b) Optimize this process (in terms of the maximization of the expectation of this process with respect to time), which is the function that satisfies the stochastic differential equation (SDE)

   $$dX_t = 3\sin(t)dt + 2tdB_t$$

   where B_t is a standard Brownian motion and $X_0 = 0$. (Hint: This is just a normal ODE with some stochastic element; just the scaling (in the limit) for dB_t is $\sqrt{t}$ not t like in dt that is important to note, and the expectation is deterministic so... you might not even need to solve the whole SDE. Also, there might be multiple solutions of t.) [10 points]

The End.