Simulated Annealing & Dynamic Programming
Due Oct 24, midnight

PROBLEM #1: Dynamic Programming: Pathway Enumeration
Pathway enumeration in a directed graph is the problem of enumerating all pathways in a graph from every source to every sink. Depending on the graph topology, we expect the runtime of any algorithm to be exponential. Despite the exponential nature of the problem, there is great benefit in understanding how to write an algorithm that performs this task, while attempting to improve the runtime as much as possible.

P1W1. Calvin thinks that it is possible to use the cycle finding algorithm (Russell’s algorithm, or Johnson’s algorithm) to accomplish this task. Argue for or against this suggestion. If you are not supportive, explain your answer. If you are supportive, provide a modification for Johnson’s algorithm to backup your claim.

Ehsan states that you can use dynamic programming to solve this problem. As a matter of fact, he says that this looks so much like matrix multiplication, and using matrix multiplication to use all shortest paths.

P1W2. Study using shortest path algorithm using matrix multiplication. A write up is available here: http://serverbob.3x.ro/IA/DDU0156.html. MIT’s open course ware provides also an explanation (http://videolectures.net/mit6046jf05_demaine_lec19/ -- go to the time 23:36, where: Erik Demaine starts explaining it).

Recall the key steps in dynamic programming:
1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution in a bottom-up fashion.
4. Construct an optimal solution from computed information.

For the shortest path problem, identify and explain the above four components.

P1W3. Explain how you can adopt matrix multiplication/shortest path algorithm to enumerate all pathways. In particular, identify and explain the above four components of dynamic programming. Write pseudo-code summary of your algorithm.

PROBLEM #2: Simulated Annealing: Parameter Estimation
One of the most difficult problems is “reverse engineering” of biology. In particular, given a cell with a particular behavior, how can we come up with a mathematical model that we can use to predict the behavior of a cell under various conditions (without doing the experimental work)?

So the situation is as follows: You are given a set of time-series measurement data for several metabolites in a cell. You are given a “vague” description of the system using a particular format called the “S-System”, and you are asked to find the missing parameters that will help you tune the S-system to
generate data similar to the measured data. Better estimation of the parameters results to smaller differences between the calculated and measured data.

You will explore in this problem using simulated annealing in solving the parameter estimation problem. Give careful thought how the choices of initial temperature, number of iterations per temperature step, cooling schedule, cost function, and neighborhood structure will impact the quality of your final solution.

Grading is based on: (1) quality of solution, and (2) demonstrating that you have made several good decisions while implementing the algorithm.

There are several papers published on this topic - ones that use simulated annealing and others that use other methods. Please feel free to browse the literature and use any tricks that will enhance your solution. However, you have to implement a flavor of simulated annealing. Below is a short tutorial, followed by the problem details.


An S-system has the following format:

$$\dot{X}_i = \alpha_i \prod_{j=1}^{M} X_j^{q_{ij}} - \beta_i \prod_{j=1}^{M} X_j^{h_{ij}}, i = 1, 2, \ldots, M.$$  

“Here, $X_i$ represents the concentration of metabolite $i$, $\alpha_i$ and $\beta_i$ are non-negative rate constants, and $q_{ij}$ and $h_{ij}$ are real-valued kinetic orders for the production and degradation term, respectively”. An example is provided.

“Example of network topology mapping onto kinetic orders in an S-system [17]. The exponents in the equations directly correspond to effects of metabolites on processes (flux arrows) in the pathway diagram. As an example, the flux out of $X_3$ is affected by $X_3$ as substrate and by $X_4$ as activator. Both variables appear in the corresponding term with their respective kinetic orders. The gray-scale in the $g$ and $h$ matrices reflects the magnitudes of the exponents in the production and degradation terms of the S-system, respectively, with higher values shown in darker hues.”
Goodness of fit (from Wikipedia)
The goodness of fits between experimental and computational data were defined, following a common modeling approach, as the Root Mean Square Deviation (RMSD) between all simulated and observed time points (from 0 to k):

$$RMSD = \sum_{t=0}^{t=k} (sim_t - measured_t)^2$$

RMSD fits are commonly used in modeling, because in the approximation of a Gaussian distribution of experimental measurement errors, these fits are usually equivalent to maximum likelihood estimation (MLE) of the parameters.

Problem #2 Details
You will be given some code infrastructure to work with, and 3 Ordinary Differential Equation (ODE) test cases. Here are their details. The state equations are provided with the code. We also provide the measuredData associated with each example.

<table>
<thead>
<tr>
<th></th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of State Variables</td>
<td>4</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>11</td>
<td>24</td>
<td>49</td>
</tr>
<tr>
<td>Alpha and Beta Param</td>
<td>0-3</td>
<td>0-5</td>
<td>0-9</td>
</tr>
<tr>
<td>Value Range</td>
<td>[0 3]</td>
<td>[0 2]</td>
<td>[0 3]</td>
</tr>
<tr>
<td>g and h Param</td>
<td>4-10</td>
<td>6-23</td>
<td>10-48</td>
</tr>
<tr>
<td>Value Range</td>
<td>[-2 2]</td>
<td>[-5 5]</td>
<td>[-2 2]</td>
</tr>
</tbody>
</table>

P2W1. Pseudo-Code
Hand in pseudo code for simulated annealing (similar to what we did in class), for a minimization problem.

P2W2. Initial temperature
The initial temperature must be hot enough to allow free exchange of neighboring solutions and to make the final solution independent of the starting solution. In 2-3 sentences, explain how you chose your initial temperature.

P2W3. Initial solution
In 1-2 sentences, explain how you chose your initial solution.
P2W4. Cooling schedule
The criteria for a good cooling schedule are the following:
1) Keep rate of acceptance high at the beginning (exploration of the search space), reduce it towards the end (exploitation of the current solution’s neighborhood)
2) The system should get close to its stationary distribution (cost difference should stabilize when searching neighborhood) at the current temperature before temperature reduction
3) The temperature (and probability of accepting non-improving moves) should converge to zero

We thus can design the cooling schedule to either explore (1) a large number of iterations at few temperatures or (2) a small number of iterations at many temperatures.

Contrast the following two cooling schedules:
1) A geometric cooling schedule of the form, f1(t)=at where 0 < a <1 0.8-0.99 suggested for a.
2) Another schedule: f2(t)=t/(1+ b t) where b is small.

Assuming you are allowed a fixed number of evaluations to solve this particular problem, what would you recommend for the number of neighborhood evaluations (max iteration per step) as the temperature decreases for both f1 and f2? Which of the two cooling schedules would more efficiently lead to the best solution? Explain your answer.

P2P5. Implementation: Implement simulated annealing. We provided some skeleton code to organize your code through two classes (SA class, and TestCase class), and a short main. Your implementation files should be named SA.cpp and SA.h. Your SA.run function should be well-written and readable, like reading the pseudo code. Use supporting functions as needed.

Your SA algorithm should allow for both cooling schedules described in P2W4 that gets passed as an option.

P2E6. Test. Evaluate/refine your algorithm using the three given examples. Your goal is to minimize the error function. You can focus on only one cooling schedule, and just use the other cooling schedule for a comparison point.

P2P7. Generate Summary. Using your code, report the following results for each of the three test cases:
- one plot that shows temperature (y-axis) against (outer) iteration number,
- one plot that shows the Number of rejected moves (Y-axis) and Total Number of Evaluated Moves (Y-axis) against (outer) iteration number.
- one plot that shows the values on the Y-axis of the evaluation function every time a solution is updated.
- One Table the shows running each of the examples 5 times (it is random!), and reporting the final RMSD value for each of the 5 times and the average RMSD.
  [optional: run for 25 times, and report the results, instead of 5 times].

Misc
The assignment consists of a written part (designated by W), and a programming assignment (designated by P), as well as exercises (designated by E). There is no need to turn in the exercise. Turn in the programming assignments (.cpp and .h files – make sure they compile). Turn in a .doc or .pdf or .txt for your written assignments. Make sure that we can use your code to reproduce any of your results in your written assignments if appropriate.
To submit your code and your write-up, enter at the command prompt,

“provide comp150ga hw2 file1 file2 file3.. ”