ROB 599: Sequential Decision Making in Robotics, Fall 2014
HW #2: Optimization and Informative Path Planning
Due: 2/13/15 (10am)

Questions

1. Performance Guarantees (15 pts)

a. You are given the problem of placing a limited number of sensors (less than or equal to a K) to maximize a submodular objective function [1]. **What is the performance guarantee relative to optimal for the greedy solution (i.e., selecting the best location for sensor one, placing the sensor, selecting the best location for the second sensor given the first is already placed, etc.)? Explain what this means in your own words (1-2 sentences).**

b. Assume you must generate a trajectory (with path constraints) to maximize a submodular objective function [2]. The vehicle can only take K samples along the trajectory. You devise the following algorithm: greedily select the best locations (as above) and connect them using an approximation to TSP. Assume that the optimal solution maximizes the total quantity F(P)/C(P), where F(P) is the submodular function evaluation for path P, and C(P) is the cost of trajectory P. **What, if any, is the performance guarantee for this algorithm? Can you give an example where this approach performs poorly?**

2. Optimization (15 pts)
Given the following linear program:
Maximize

$$x + y$$

Subject to:

$$\frac{x}{2} + y \leq 2$$
$$6x + 2y \leq 8$$
$$x, y \geq 0$$

i. What is the optimal solution of this LP? **Find by hand by sketching the feasible region and determining the intersection of the active constraints. You may use a linear programming toolbox (like Matlab's linprog function) to check. Provide your sketch of the feasible region in the writeup.**

ii. Now let's constrain the values of x and y to integers to make this an integer program. Use your sketch to find the optimal solution in this case. **What is integrality gap (the difference between the integer solution and the non-integer solution)? Explain (1-2 sentences) the importance of the integrality gap in terms of approximating integer programs by removing the integrality constraints.**

Programming Assignment (Do not start at last minute!) (70 points)

E-mail your code for this section to the TA Matt Rueben (ruebenm@onid.oregonstate.edu). To avoid mixup, e-mail with the following subject line and archive file name:
Subject: 15-780 Homework 2 Submission
Archive name: (yourID)-hw2.zip
Where (yourID) is your onid ID (e.g., ruebenm-hw2.zip).

The zip file "Homework2.zip" contains helper functions and data for this homework. Download it in addition to this word document. You may use any programming language you wish to complete the homework. Helper functions are provided in the zip file if you choose to use Matlab.

This assignment will use provided information in the maps.mat files. This matlab data file contains data on three configurations of Gaussian sources of information (*map1*, *map2, and mapBig*). These can be fed into the findInformation.m function to determine the amount of information gathered at any location. Assume the environment bounds go from (1,1) to (4,4) for *map1* and *map2* and (1,1) to (10,10) for *mapBig*.

The goal is to generate a trajectory starting in the top left corner (1,1) and ending in the bottom right corner (4,4) or (10,10) depending on the map that maximizes the amount of information gathered from the environment. The trajectory samples the environment (gathers information) only once per step, so subsequent waypoints should have distance exactly 1 from the previous waypoint. You may also allow the robot to remain still and take a sample at its current location for a "path penalty" of 1.

The function evaluatePath.m takes in an ordered set of waypoints and a vector of information sources and returns the information gathered along that trajectory. Your goal is to maximize this information for a given budget. We will use budgets of 6 and 8 (number of waypoints excluding the start). You may visualize your path and the information map using the plotPath.m function.

Step 1 (30 points): This assignment deals with a discrete version of the problem. The robot can only select waypoints that are on the grid (e.g., (1.5,1.5) is not a valid waypoint). Keep in mind that subsequent waypoints should always be distance 1 or 0 (if remaining stationary) from each other.

i. Implement a greedy solver for the discrete problem that can only move in the 4 cardinal directions (N,W,S,E) or remain still. The robot should look one step ahead and move to the location that gains the maximal information. However, the robot must still reach the bottom right corner (4,4) before the budget expires. **How do you ensure it reaches the goal within the budget? Show us an example trajectory on *map1* for budget 6. Provide an information quality table for both budgets (6 and 8) on *map1* and *map2* using your greedy algorithm.**

ii. Implement the branch and bound solver in [2] to find an optimal solution to the discrete problem. Note that in our case the reward is modular (i.e., it does not go down with increasing samples). **What bounding strategy did you use? How large of a budget can you run it on for *map1* and *map2*? How does the solution quality compare to the greedy solver? Show us an example trajectory for budget 8 on *map2*. Add the information attained by your branch and bound solver to those you recorded for the greedy algorithm on both maps.**

Step 2 (20 points):

i. Develop your own algorithm that works on *mapBig*. In this problem, you must start at (1,1) and also end at (1,1) (i.e., your path is a cycle). **Describe your algorithm. Report your highest attained information quality for budgets 20 and 30 on *mapBig*. Also report the time it took to find this solution. You should terminate your algorithm after 15 minutes of run time (i.e., solutions that take more than 15 minutes to find are considered invalid).** Your information values will be compared to others in the class. The winner with the highest information quality value within the budget will be announced in class and given extra credit. May the best algorithm designer win!

Discussion (20 points): Address the following questions.
1) Imagine an algorithm that operates in continuous space for this problem. What would be the advantage of the discrete variant versus the continuous variant? In what situations would using a discrete algorithm be more appropriate and vice versa?
2) When designing your algorithm in Step 2(ii), what design criteria did you use to determine what to implement? Compare and contrast how your algorithms meet these design criteria relative to algorithms taught in this course (use at least two examples from the course lectures or reading group discussions)?

References
[1] Andreas Krause, Carlos Guestrin, "Submodularity and its Applications in Optimized Information Gathering," In ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 4, 2011.
[2] Jonathan Binney and Gaurav S. Sukhatme. "Branch and Bound for Informative Path Planning". In IEEE International Conference on Robotics and Automation (ICRA), 2012.