

CS 6150: HW5 – Randomized algorithms, optimization

Submission date: Monday, Nov 29, 2021 (11:59 PM)

points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Trade-offs in sampling	12	
Satisfying ordering constraints	10	
Writing constraints	6	
LP Basics	10	
Identifying Corners	6	
Checking feasibility vs optimization	6	
Total:	50	

Instructions. For all problems in which you are asked to develop an algorithm, write down the pseudocode, along with a rough argument for correctness and an analysis of the running time (unless specified otherwise). Failure to do this may result in a penalty. If you are unsure how much detail to provide, please contact the instructors on Piazza.

Question 1: Trade-offs in sampling [12]

For this problem, you need to run some basic experiments and write down the results you obtained. You **do not need to submit your code**, but if you prefer, you may add a publicly accessible link to the code (e.g., on github).

Suppose we have a population of size 10 million, and suppose 52% of them vote for A and 48% of them vote for B .

- (a) [4] Randomly pick samples of size (a) 20, (b) 100, (c) 400, and evaluate the probability that A is majority even in your sample (by running the experiment say 100 times and taking the count of times A is the majority in your sample). Write down the values you observe for these probabilities in the cases (a-c).

Code link: <https://gist.github.com/mozartfish/b5e8a2dba17ec1ba9c8fb19a51b4fe5c>

a) sample size 20 - Percent A: 43%

b) sample size 100 - Percent A: 63%

c) sample size 400 - Percent A: 81%

- (b) [4] What is the size of the sample you need for the probability above to become 0.9? (Your answer can be within 20% of the ‘best’ value.)

Sample size needed to achieve a probability of 0.9 with 100 trials: 900 elements

- (c) [4] Suppose the population is more biased —55% of them vote for A and 45% of them vote for B — and re-solve part (b).

Sample size needed to achieve a probability of 0.9 with 100 trials: 900 elements

Note: This code was written with python and uses python’s random number generator with `randint()`. According to the docs <https://docs.python.org/3/library/random.html> Python uses the Mersenne twister to generate random numbers so if a particular experiment is run enough times a specified target probability can be achieved.

Question 2: Satisfying ordering constraints [10]

This problem is meant to illustrate a rather cool paradigm for solving problems, which is picking a *random solution*, and understanding how well it does. Indeed, there are many problems (“3-SAT”, a version of boolean satisfiability, being the chief of them) where doing better than a random solution is actually NP-hard!

Suppose we have n elements, labeled $1, 2, \dots, n$. Our goal is to find an ordering of these elements that satisfies as many “constraints” as possible, of the kind described below. We are given m constraints, where each constraint is given by a triple (a, b, c) , where $a, b, c \in \{1, 2, \dots, n\}$. The constraint is said to be *satisfied* if in the ordering we output, a does **not** lie “between” b and c (but the order between b and c doesn’t matter). For example, if $n = 4$ and we consider the ordering 2431, then the constraint $(1, 4, 3)$ is satisfied, but $(3, 1, 2)$ is not.

Given the constraints, the goal is to find an ordering that satisfies as many constraints as possible (for simplicity, assume in what follows that m is a multiple of 3). For large m, n , this problem becomes very difficult.

- (a) [6] As a baseline, let us consider a *uniformly random* ordering. What is the expected number of constraints that are satisfied by this ordering? [Hint: define appropriate random variables whose sum is the quantity of interest, and apply the linearity of expectation.]
- (b) [4] Let X be the random variable which is the number of constraints satisfied by a random ordering, and let E denote its expectation (which we computed in part (a)). Now, Markov's inequality tells us, for example, that $\Pr[X \geq 2E] \leq 1/2$. But it does not directly imply that $\Pr[X \geq E]$ is "large enough" (which we need if we want to say that generating a few random orderings and picking the best one leads to many constraints being satisfied with high probability). Use the definition of X above to conclude that $\Pr[X \geq E] \geq 1/m$. [Hint: X is a non-negative integer!]

Question 3: Writing constraints..... [6]

We will see how writing down constraints can be tricky when formulating problems as optimization.

Recall the traveling salesman problem (TSP): we are given a directed graph $G = (V, E)$ with edge lengths $w(e)$. The goal is to find a *directed cycle* that (a) visits every vertex exactly once, and (b) minimizes the total length (sum of lengths of the edges of the cycle).

Consider the following optimization formulation. We have variables $x_{uv} \in \{0, 1\}$, one for each edge (u, v) (remember that the graph is directed). The objective is to minimize $\sum_{(u,v) \in E} w(uv)x_{uv}$. The constraints are as follows: let $N_+(u)$ and $N_-(u)$ denote the out-neighbors and in-neighbors of u ; then we consider the constraints:

$$\begin{aligned} \forall u, \quad \sum_{v \in N_+(u)} x_{uv} &= 1, \\ \forall u, \quad \sum_{w \in N_-(u)} x_{wu} &= 1. \end{aligned}$$

(Intuitively, the constraints say that exactly one incoming edge and one outgoing edge must be chosen — as in a directed cycle.) Does an optimal solution to this optimization problem *always yield* an optimal solution to TSP? Provide a proof or a counterexample with a full explanation.

[Hint: consider a feasible solution to the optimization problem and focus on the edges with $x_{uv} = 1$. Do they have to form a *single* cycle?]

Question 4: LP Basics..... [10]

Consider the following two-variable linear program. The variables are $x, y \in \mathbb{R}$. And the constraints are:

$$\begin{aligned} 2x + y &\leq 10, \\ x &\geq 0, \\ y &\geq -1, \\ x + 3y &\leq 8. \end{aligned}$$

- (a) [4] Plot the feasible region for the linear program. (Diagram can be drawn approximately to scale + scanned).
See attached photo with assignment

- (b) [2] Suppose the objective function is to *maximize* $x + y$. Find the maximum value and the point at which this is attained.

The maximum point that satisfies the constraint is (4.4, 1.2)

- (c) [4] Say the maximum value found in part (b) is C . Then could you have concluded that $x + y \leq C$ by just “looking at” the equations? [*Hint*: does adding equations, possibly with constants, yield a bound?]

We can define 4 equations as the following:

- 1) $2x + y \leq 10$
- 2) $x \geq 0$
- 3) $y \geq -1$
- 4) $x + 3y \leq 8$

Operation 1: Multiply Equations 3 by 2

- 1) $2x + y \leq 10$
- 2) $x \geq 0$
- 3) $2y \geq -2$
- 4) $x + 3y \leq 8$

Operation 2: Subtract equation 2 from equation 1:

- 1) $x + y \leq 10$
- 3) $2y \geq -2$
- 4) $x + 3y \leq 8$

Operation 3: Add equations 1 and 3:

- 1) $x + 3y \leq 8$
- 4) $x + 3y \leq 8$

At this point we have shown that equations 1 and equations 4 are equivalent using a set of equation operations. The bound for this set of linear equations is equation 4.

Question 5: Identifying Corners..... [6]

Consider the following linear program, with variables $x_1, x_2, \dots, x_n \in \mathbb{R}$. Suppose that the constraints are:

$$\begin{aligned} 0 &\leq x_i \leq 1 \text{ for all } i, \\ x_1 + x_2 + \dots + x_n &\leq k, \end{aligned}$$

where k is some integer between 1 and n . Consider any point (a_1, a_2, \dots, a_n) where $a_i \in \{0, 1\}$, and exactly k of the $\{a_i\}$ are 1. Prove that any such point is (a) a feasible point for the LP, and (b) a corner point of the polytope defining the feasible set.

[*Hint*: To prove that a point is a corner point, we use the definition we mentioned in class: a feasible point x is a corner point if and only if for *all* nonzero vectors z , either $x + z$ or $x - z$ is infeasible.]

[*Remark*: It turns out that these are the only corner points of the polytope, and so it is an example of a polytope defined by $2n + 1$ constraints and having $\binom{n}{k}$ corner points.]

Question 6: Checking feasibility vs optimization [6]

Some of the algorithms for linear programming (e.g. simplex) start off with one of the corner points of the feasible set. This turns out to be tricky in general. In this problem, we will see that **in general**, finding one feasible point is as difficult as actually performing the optimization!

Consider the following linear program (in n variables x_1, \dots, x_n , represented by the vector x):

minimize $c^T x$ subject to

$$a_1^T x \geq b_1$$

$$a_2^T x \geq b_2$$

...

$$a_m^T x \geq b_m.$$

Suppose you know that the optimum value (i.e. the minimum of $c^T x$ over the feasible set) lies in the interval $[-M, M]$ for some real number M (this is typically possible in practice). Suppose also that you have an **oracle** that can take any linear program and say whether it is feasible or not. Prove that using $O(\log(M/\epsilon))$ calls to the oracle, one can determine the optimum value of the LP above up to an error of $\pm\epsilon$, for any given accuracy $\epsilon > 0$. [*Hint:* can you write a new LP that is feasible only if the LP above has optimum value $\leq z$, for some z ?]