CS 6150: HW 6 – Optimization formulations, review

Submission date: Tuesday, December 14, 2021, 11:59 PM

This assignment has 5 questions, for a total of 50 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Understanding relax-and-round	16	
The power of two choices	10	
Optimal packaging	10	
Non-negativity in Markov	4	
Distributed independent set	10	
Total:	50	

Question 1: Understanding relax-and-round ..... **[16]**

In this question, you will implement the relax and round paradigm using the example of the Set Cover (or hiring) problem. Suppose we have n people (i.e., sets) and m skills that we wish to cover. Let us create an instance with $n = m = 500$. Let $d = 25$ be the size of each skill set.

(a) **[3]** Write code that generates a random instance of set cover, where for each person i, the skill set S_i is a random subset of the m skills, with $|S_i| = d$.

Solution.

Here is example code written with Python 3.9.5.

```
from random import Random

rand = Random(12345)  # initialize with a seed
N = 500               # number of people; people are labeled from 0 to N-1
M = 500               # number of skills; skills are labeled from 0 to M-1
D = 25                # size of each skill set
S = [rand.sample(range(M), D) for _ in range(N)]  # S[i] is a skill set for person i
```

Rubric:

- 1 point for code submission
- 1 point for code readability
- 1 point for code correctness

(b) **[3]** Write down the integer linear program using variables x_i that indicate if person i is chosen/hired (abstractly, as we did in class). Then write down the linear programming relaxation. Which one has the lower optimum objective value?

Solution.

ILP formulation:

- Variables: $x_1, \ldots, x_n \in \{0, 1\}$
- Objective: Minimize $\sum_{i=1}^{n} x_i$.
- Constraints: $\sum_{i:j \in S_i} x_i \geq 1$ for every skill $1 \leq j \leq m$.[1]

LP relaxation:

- Variables: $x_1, \ldots, x_n \in \mathbb{R}$
- Objective: Minimize $\sum_{i=1}^{n} x_i$.
- Constraints: $\sum_{i:j \in S_i} x_i \geq 1$ for every skill $1 \leq j \leq m$,
  and $0 \leq x_i \leq 1$ for every person $1 \leq i \leq n$.

The LP relaxation (potentially) has the lower optimum objective value because LP's feasible region includes and is larger than that of ILP.

Rubric:

- 1 point for ILP formulation
- 1 point for LP relaxation
- 1 point for stating that LP has the lower optimum

(c) **[6]** For the instance you created in part (a), solve the linear program from part (b) using an LP solver of your choice, and output the fractional solution.

Solution.

Here is an example using Python and Gurobi Optimizer 9.1.2.

[1] The notation $i : j \in S_i$ reads as iterating over a set of person i such that skill j is in S_i, that is, iterating over the people who have skill j.

```python
import gurobipy as gp
from gurobipy import GRB

with gp.Env(empty=True) as env:
    env.start()
    with gp.Model(env=env) as m:
        # Create variables.
        xs = m.addVars(N, lb=0, ub=1, vtype=GRB.CONTINUOUS)

        # Set objective.
        obj = gp.LinExpr([(1, xs[i]) for i in range(N)])
        m.setObjective(obj, GRB.MINIMIZE)

        # Add constraints.
        for j in range(M):
            constr = gp.LinExpr([(1, xs[i]) for i in range(N) if j in S[i]])
            m.addConstr(constr >= 1, f'c{j}')

        # Optimize model.
        m.optimize()

        # Obtain solution.
        cert = [xs[i].x for i in range(N)]
```

Obtained objective: 21.984

Fractional solution: $[0, 0, 0.064, 0.134, 0, 0, 0, 0.075, 0.195, 0, \ldots \text{(snip)}]$

Rubric:

- 2 points for submitting output
- 1 point for validity: there should be 500 variables
- 1 point for validity: each variable must be between 0 and 1, inclusive
- 1 point for validity: there should be a fractional variable (with high probability)
- 1 point for validity: objective should be around 21 and 22 (with high probability)

(d) [**4**] Round the fractional solution using randomized rounding, i.e., hire person i with probability $\min(1, tx_i)$. Try $t = 1, 2, 4, 8$, and in each case, report the (a) total number of people hired, and (b) number of "uncovered" skills (i.e., skills for which none of the people possessing the skill were hired).

Solution.

From this code, we got the following result.

```python
for t in [1, 2, 4, 8]:
    # Randomized rounding: get a list of hired people.
    rounded = [i for i, x in enumerate(cert) if rand.random() <= x * t]
    # Compute a set of covered skills.
    covered = set(j for i in rounded for j in S[i])
    print(f't={t}, #hired: {len(rounded)}, #uncovered: {M - len(covered)}')
```

- $t = 1$: number of people hired: 25, number of uncovered skills: 122
- $t = 2$: number of people hired: 34, number of uncovered skills: 80
- $t = 4$: number of people hired: 93, number of uncovered skills: 1
- $t = 8$: number of people hired: 163, number of uncovered skills: 0

Rubric:

- 2 points for showing output
- 1 point for validity: the number of people hired increases with larger t
- 1 point for validity: the number of uncovered skills decreases with larger t

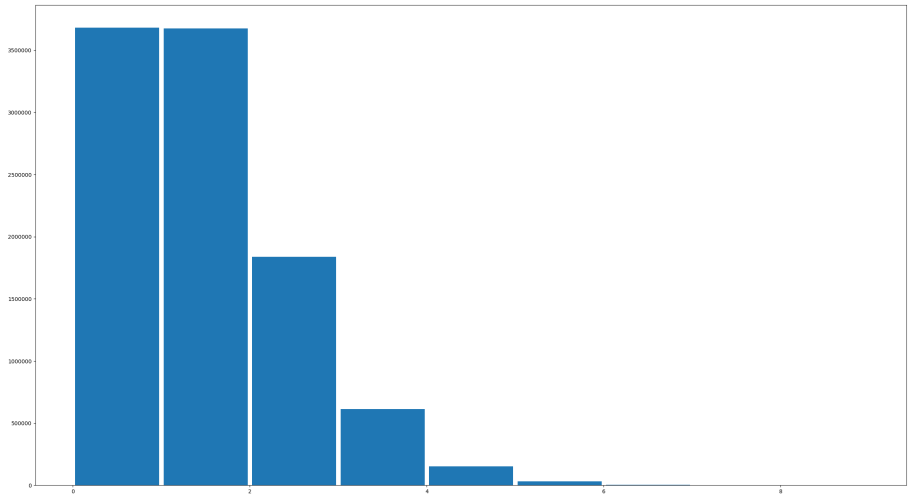
Question 2: The power of two choices . . . . . **[10]**

As I mentioned in class, one of the classic applications of random hashing is "on-the-fly" load balancing. In this problem, we will empirically see how "balanced" such an assignment is, and how to make it more balanced.

In what follows, set $N = 10^7$, i.e., 10 million. Suppose we have N servers, and N service requests arrive sequentially.

(a) [**4**] When a request arrives, suppose we generate a random index r between 1 and N and send the request to server r (and we do this independently for each request). Plot a histogram showing the distribution of the "loads" of the servers in the end. I.e., show how many servers have load 0, load 1, and so on. [The load is defined as the number of requests routed to that server.]

Solution.

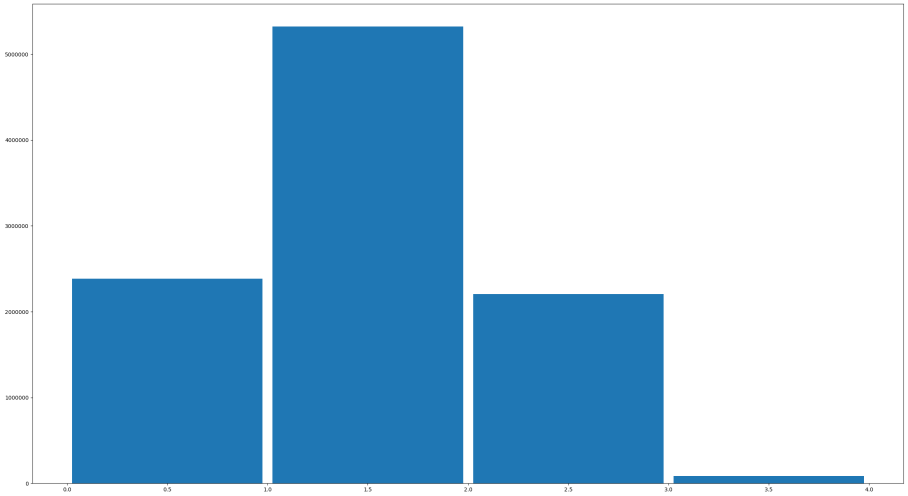


Rubric:

- 4 points. By using this load distribution strategy, we should observe a significant amount of servers with load = 0 (freq should be almost equal to that of servers with load = 1).

(b) [**6**] Now, suppose we do something slightly smarter: when a request arrives, we generate *two* random indices r_1 and r_2 between 1 and N, query to find the *current load* on the servers r_1 and r_2, and assign the request to the server with the *lesser* load (breaking ties arbitrarily). With this allocation, plot the histogram showing the load distribution, as above.

Solution.



Rubric:

- 6 points, compared to Q2.A, the #servers with 0 load should decrease. (should start looking like a normal distribution)

Question 3: Optimal packaging ..... **[10]**

With the holiday season around the corner, company Bezo wants to minimize the number of shipping boxes. Let us consider the one dimensional version of the problem: suppose a customer orders n items of lengths $a_1, a_2, \ldots, a_n$ respectively, and suppose $0 < a_i \leq 1$. The goal is to place them into boxes of length 1 such that the total **number of boxes** is minimized.

It turns out that this is a rather difficult problem. But now, suppose that there are only r *distinct* values that the lengths could take. In other words, suppose that there is some set $L = \{s_1, \ldots, s_r\}$ such that every $a_i \in L$. Let us think of r is as a small constant. Devise an algorithm that runs in time $O(n^r)$, and computes the optimal number of boxes.

[*Hint:* first find all the possible "configurations" that can fit in a single box. Then use dynamic programming.]

Solution.

Let $Y = \{y_1, y_2, \ldots, y_r\}$ be the count of items of each size, such that $\sum_{i=1}^{r} y_i = n$. Since $0 < s_i \leq 1$ for all i, we can include at most of $1/s_i$ of each item in a single box.

Let's set up a dynamic programming structure. Let $S[\{x_1, x_2, \ldots, x_r\}]$ denote the minimum number of boxes required to store $\{x_1, x_2, \ldots, x_r\}$ items of each size. We can say

$$S[\{x_1, x_2, \ldots, x_r\}]$$

Let, y_i denote the number of elements in $\{a_1, a_2, \ldots a_n\}$, whose length is s_i. Now, let S be the set of all possible configurations which can fit in single box of length 1. Let x_i be the number of elements of s_i in any one configuration in S. Any configuration in this set will satisfy following two constraints:

Max Box size constraint:

$$\sum_{i=1}^{r} x_i s_i \leq 1$$

Number of elements constraint:

$$0 \leq x_i \leq y_i \leq n$$

This immediately implies, $|S| = O(n^r)$, so in time $O(n^r)$ we can find the set of all "valid configurations" for one box. First, let's form the recursion needed for the solution. Let B denote the number of boxes, let m be the number of all possible configurations, and solution configuration s

$$OptPack(m, y, B, s) = \min_{i=1}^{n} OptPack(m-1, y-si, B+i, s, i)$$

Now such recursion can be optimally solved by dynamic programming algorithm techniques which avoid recomputation by utlising extra space to store computes subcases.

If we enumerate all the possible boxes that we can fill, this will be at most n^r unique ways of filling boxes, since at most n of each size can be placed in a box, and there are r unique sizes. Let each of these configurations, along with the count of each item stored, be recorded in a structure S, where $S[\{b_1, b_2, \ldots, b_r\}]$

Rubric:

- 5 points for a correct algorithm
- 3 points for explanation of runtime
- 2 points for argument of correctness

Question 4: Non-negativity in Markov ..... **[4]**

Markov's inequality states that for a non-negative random variable X, we have $\Pr[X > t \cdot \mathbb{E}[X]] \leq 1/t$, for any $t \geq 1$.

The point of this exercise is to show that the non-negativity is important. Give an example of a random variable (that takes negative values), for which (a) $\mathbb{E}[X] = 1$, and (b) $\Pr[X > 5] \geq 0.9$.

Solution.

Consider a random variable X which takes the value 6 with probability $Pr(X = 6) = 0.9$, and the value -44 with $Pr(X = -44) = 0.1$. Then the expected value of X is:

$$\mathbb{E}[X] = 0.9(6) + 0.1(-44) = 1$$

We also have that $Pr(X > 5) \geq 0.9$, which satisfies the conditions.

Rubric:

- 2 points for finding a random variable X
- 1 point for showing $\mathbb{E}[X] = 1$
- 1 point for showing $Pr(X > 5) \geq 0.9$

Question 5: Distributed independent set ..... **[10]**

A fundamental problem in distributed algorithms (used in P2P networks, distributed coloring, etc.) is the following: we are given an undirected graph with n vertices where each vertex corresponds to an 'agent'. The goal is to find a large independent set (a set of vertices with no edges between them) in a distributed manner. It turns out that this problem has a nice solution when the degree of each vertex is $\leq d$, for some known parameter d.

Consider the following algorithm. (1) Every vertex becomes *active* with probability $\frac{1}{2d}$. (2) Every active vertex queries its neighbors, and if any vertex in the neighborhood is also active, it becomes inactive. (Step (2) is done in parallel; thus if i and j are neighbors and they were both activated in step (1), they both become inactive.) (3) The set of active vertices in the end is output as the independent set.

(a) **[2]** Let X be the random variable that is the number of vertices activated in step (1). Find $\mathbb{E}[X]$.

Solution.

Z_i be the random variable which equates to 1 if the ith vertex is activated, else 0.

By X's definition,

$$X = \sum_i Z_i$$

By Linearity of Expectation,

$$E(X) = E(\sum_i Z_i) = \sum_i E(Z_i)$$

$E(Z_i) = 1(1/2d) + 0(1 - 1/2d) = 1/2d$

Since, there are n vertexes,

$$E(X) = n/2d$$

Rubric:

- 2 points for evaluating E[X]

(b) **[3]** Let Y be the random variable that is the number of edges $\{i, j\}$ *both of whose end points* are activated in step (1). Find $\mathbb{E}[Y]$ (in terms of m, the total number of edges in the graph).

Solution.

Let L_k be the random variable which equates to 1 if the kth edge has both ends activated, else 0.

By Y's definition,

$$Y = \sum_i L_k$$

By Linearity of Expectation,

$$E(Y) = E(\sum_k L_k) = \sum_k E(L_k)$$

If both ends are activated (with Probability 1/2d),
$E(L_k) = 1(1/2d)(1/2d) + 0(1 - (1/2d)(1/2d)) = 1/4d^2$

$$E(Y) = m/4d^2$$

Rubric:

- 3 points for evaluating E[Y]

(c) **[5]** Prove that the size of the independent set output in (3) is at least $X - 2Y$, and thus show that the expectation of this quantity is $\geq n/4d$.

Solution.

Independent set represents the total number of active vertices that do not have an edge between them. X vertexes are activated, then Y edges deactivate their two endpoints. If none of the endpoints overlap, the highest number of deactivation is 2Y. The number of active endpoints is at least X-2Y, all of which are independent since vertexes which aren't independent are deactivated before.

Let R be the number of active points.

By Linearity of Expectation,

$$E(R) \geq E(X - 2Y) = n/2d - 2(m/4d^2) = n/2d - m/2d^2$$

Since, each edge has two vertexes, and the degree of each vertex is not larger than d,

$$m \leq nd/2$$

$$E(R) \geq n/2d - m/2d^2 \geq n/2d - nd/4d^2 = n/4d$$

$$E(R) \geq n/4d$$

Rubric:

- 2 point to prove that the size of the independent set output is aleast X-2Y
- 2 points to show that the expectation is $\geq$ n/4d.