AAD Aflevering 2

Andreas Hammer (qgk998), Anders Munkvad (tqs326), Johan Topp (fpg662)

December 2023

29.1-5

$$\begin{aligned} \text{maximize } & 2x_1 - 6x_3 \\ \text{subject to } & x_1 + x_2 - x_3 \leq 7 \\ & 3x_1 - x_2 \geq 8 \\ & -x_1 + 2x_2 + 3x_3 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{maximize } & 2x_1 - 6x_3 \\ \text{subject to } & x_1 + x_2 - x_3 \leq 7 \\ & -3x_1 + x_2 \leq -8 \\ & x_1 - 2x_2 - 3x_3 \leq 0 \end{aligned}$$

$$\begin{aligned} &\text{maximize } 2x_1 - 6x_3 \\ &\text{subject to } x_1 + x_2 - x_3 && \leq 7 \\ &3x_1 - x_2 && \geq 8 \\ &-x_1 + 2x_2 + 3x_3 && \geq 0 \end{aligned}$$

$$\begin{aligned} &\text{maximize } 2x_1 - 6x_3 \\ &\text{subject to } x_1 + x_2 - x_3 && \leq 7 \\ &-3x_1 + x_2 && \leq -8 \\ &x_1 - 2x_2 - 3x_3 && \leq 0 \end{aligned}$$

$$\begin{aligned} z &= 2x_1 - 6x_3 \\ x_4 &= 7 - x_1 - x_2 + x_3 \\ x_5 &= -8 + 3x_1 - x_2 \\ x_6 &= -x_1 + 2x_2 + 3x_3 \end{aligned}$$

Non-basic are $\{x_1, x_2, x_3\}$ and basic are $\{x_4, x_5, x_6\}$

29.2-6

$$\begin{aligned}
\text{maximize, } & \sum_{l\in L}\sum_{r\in R}(l,r) \\
\text{subject to } \forall r \in R, & \sum_{l\in L}(l,r)|(l,r)\in E \leq 1 \\
\forall l \in L, & \sum_{r\in R}(l,r)|(l,r)\in E \leq 1 \\
\forall l \in L, r \in R, & \ (l,r)|(l,r)\notin E = 0 \\
\forall l \in L, r \in R, & \ (l,r) \geq 0
\end{aligned}$$

29.3-5

Standard form:

$$\begin{aligned}
z &= 18x_1 + 12,5x_2 \\
x_3 &= 20 - x_1 - x_2 \\
x_4 &= 12 - x_1 \\
x_5 &= 16 - x_2
\end{aligned}$$

Basic solution is feasible, since all variables are zero or greater and there is a basic vector with positive coefficient so it is possible to maximize further.
Choose x_1 to isolate, capped by the 3 to $x_1 = 12.$ equation and replace in the other equations:

$$\begin{aligned}
z &= 18(12 - x_4) + 12,5x_2 &&= 216 + 12,5x_2 - 18x_4 \\
x_1 &= 12 - x_4 \\
x_3 &= 20 - (12 - x_4) - x_2 &&= 8 - x_2 + x_4 \\
x_5 &= 16 - x_2
\end{aligned}$$

Still feasible and positive basic variable, finding boundry of x_2, capped by equation 3 to $x_2 = 8$ isolating and replacing in the other equations:

$$\begin{aligned}
z &= 18(12 - x_4) + 12,5(8 - x_5) &&= 316 - 5,5x_4 - 12,5x_3 \\
x_1 &= 12 - x_4 \\
x_2 &= 8 + x_4 - x_3 \\
x_5 &= 16 - (8 + x_4 - x_3) &&= 8 - x_4 + x_3
\end{aligned}$$

No basic variables are in the equation with positive variables, we stop. The solution is $(12, 8, 0, 0, 8)$

29.4-1

In order to get the dual of the linear program we transpose the equations so what we want to minimize the $\sum_{i=1}^{m} b_i y_i$ subject to $\sum_{i=1}^{m} a_{i,j} y_i \geq c_j$ for $j = 1, 2, .., n$, and $y_i \geq 0$ for $i = 1, 2, ..., m$:

$$\begin{aligned}
min &: 20y_1 + 12y_2 + 16y_3 \\
s..t.\ & y_1 + y_2 \geq 18 \\
& y_1 + y_3 \geq 12,5
\end{aligned}$$

Randomized Alg 1

If we assume the best case scenario of the depth of the tree, namely that the pivot is always in the middle (or close to it) our tree would resemble a perfectly balanced binary search tree. The depth of a perfectly balanced search tree is $log_2(n)$ since every node has two children (expect the leaves). So the lower bound for $\mathbb{E}[d(x)]$ is $log_2(n)$.

If we assume the worst case scenario, namely that we always choose the pivot to be the largest (or smallest) element, the depth of the tree would become n, essentially resembling a list. So the upper bound for $\mathbb{E}[d(x)]$ is n.

Since the pivot is chosen uniformly at random, the best case nor the worst case will be chosen consistently, hence most selections of the pivot will result in a reasonably balanced tree (not perfectly balanced but not resembling a list either). Given this we can define $f(x)$ as:

$$f(x) = c \cdot log_2(n)$$

For some constant c that is used to resemble the fact that the tree is not perfectly balanced. Now we want to demonstrate that $\mathbb{E}[d(x)] = \theta(f(n))$.
We consider $\mathbb{E}[2^{d(x)}]$. We can then apply Jensen's inequality, that states (CLRS, 3rd edition, p. 302):

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$$

By applying Jensen's inequality to $2^{d(x)}$:

$$2^{\mathbb{E}[d(x)]} \leq \mathbb{E}[2^{d(x)}]$$

This effectively binds the expected depth of the tree by the exponential of the expected depth. We then take the log of both sides of the inequality, hence:

$$\mathbb{E}[d(x)] \leq log(\mathbb{E}[2^{d(x)}])$$

This provides an estimation of the expected depth of the tree, proving that the expected depth is $\mathbb{E}(d(x)) = \theta(f(n))$, where $f(n) = c \cdot log(n) = \theta(log(n))$.

Randomized Alg 2

Given by the lecture notes, we have that the probability for any given min-cut C is:

$$Pr[\text{output is C}] \geq \frac{2}{n(n-1)}$$

Since we want to find how many runs of randomized contraction we need to be 99% sure that we got a minimum cut, we look at the probability of failing, namely:

$$Pr[\text{output is not C}] \leq (1 - \frac{2}{n(n-1)})^{t\frac{n(n-1)}{2}}$$

Now since we want this probability to be less or equal to 1%:

$$(1 - \frac{2}{n(n-1)})^{t\frac{n(n-1)}{2}} \leq 0,01$$

We can rewrite the probability of not getting a min-cut to the following:

$$Pr[\text{output is not C}] = e^{-t}$$

or

$$Pr[\text{output is C}] \leq 1 - e^{-t}$$

Now we have the following inequality:

$$1 - e^{-t} \geq 0,99$$

We then isolate t:

$$t \geq ln(100)$$

In conclusion, the number of runs needs to be at least $ln(100)$ to achieve a 99% confidence level of finding the minimum cut. We have that $ln(100)$ is scaled up by $\frac{n(n-1)}{2}$, since it also depends on the size of the graph.

Randomized Alg 3

1.2

In the case where a the graph contains a min-cut between A and B where $A \in L$ and $B \in R$ and the total number of notes $n = |L| + |R|$ and each of the sets $|L|, |R| = \frac{n}{2}$ then the odds of picking two vertices where both belong to L or R will be:

$$\frac{\binom{\frac{n}{2}}{2} + \binom{\frac{n}{2}}{2}}{\binom{n}{2}}$$

When doing this, the size of L or R will be reduced by 1.
Doing this $i = \frac{n}{4}$ times, then the size of each of the sets $L, R \geq \frac{n}{4}$ are there has only been selected from one of the two, then the chance of selecting two vertices where both belong to either L or R will be:

$$\frac{\binom{\frac{n}{4}}{2} + \binom{\frac{n}{2}}{2}}{\binom{\frac{3}{4}n}{2}} \geq \frac{\binom{\frac{n}{2}}{2} + \binom{\frac{n}{2}}{2}}{\binom{n}{2}}$$

If this probability from the $i = \frac{n}{4}$ step is used rather than each probability in between, which is bound to be lower, then the probability of having selected the right vertices through the first i steps will be at most

$$\left(\frac{\binom{\frac{n}{4}}{2} + \binom{\frac{n}{2}}{2}}{\binom{\frac{3}{4}n}{2}} \right)^{\frac{n}{4}}$$

which has an exponential run-time, if $c = \frac{1}{4}$ then the exponent would be cn. Continuing until $i = n - 3$ which would end up with only 2 vertices left, each step would be multiplied on, and each probability would be less than 1, therefore it would never become better than the equation above. Therefore the runtime will be exponential for this version of the of min-cut

1.3

We construct an algorithm that runs indefinitely until we find a solution. The Monte Carlo algortihm is denoted as A.

Algorithm 1 Las Vegas Algorithm

```
while solution not found do
    result ← A(X)
    is_correct ← validate(X)
    if is_correct then break
return result
```

In this pseudocode we can see each iteration is $O(T(n) + t(n))$, since we first find a candidate solution and then validate it. Now to get the expected running time we need to find the expected number of iterations. For the n'th iteration we have that the probability for success is the probability we didn't get the correct result $n-1$ times $\gamma(n)$ aka. $(1-\gamma(n))^{n-1}\gamma(n)$. We can then write the expected number of iterations as an infinite sum, $\sum_{i=1}^{\infty} i \cdot \gamma(n)(1-\gamma(n))^{i-1}$. This sum is well known to sum to $\frac{1}{\gamma(n)}$. We now have an algorithm with $\frac{1}{\gamma(n)}$ expected number of iterations each with $O(T(n) + t(n))$, so we get a worst-case expected runtime of $\frac{(T(n)+t(n))}{\gamma(n)}$.

1 Andreas Hammer

Linear Programming

- Max/Min objective value under constraints
- What is standard and slack form
- How to convert between standard and slack form (Example)
- Simplex
- Does simplex terminate?
- Dual problem proof

Randomized Algorithms

- Estimation vs. exact bounds
- Monte Carlo vs. Las Vegas
- RandQS expected runtime
- Min-cut probability of success (with repetition)
- converting between Monte Carlo and Las Vegas

2 Anders Munkvad

Linear Programming

- Objective function, constraints, forms of Linear Programming.
- Example (convert to standard form, then slack form and then Simplex).
- Simplex algorithm.
- Proof of duality (Don't know if the actual proof is needed or just the reasoning)

Randomized Algorithms

- Motivation as to why it sometimes makes sense to use Randomized Algorithms
- Monte Carlo and Las Vegas
- Example: RandQS (Runtime analysis)
- Example: Min-Cut algorithm

3 Johan

3.1 Linear Programming

- max/min problems, generic solution
- constraints
- + variables and basic variables
- Standard form
- Simplex
  - Feasible region
  - Optimizing
  - Termination
- Duality - optimal solution, bounded by min of dominant function

3.2 Randomized Algorithms

- Using randomization $->$ Estimated run time, simplicity
- Monte Carlo run time vs Las Vegas correctness
- QuickSort estimated run time vs $O(N^2)$
  - Number of comparisons
  - Returns correct result
  - Run time may not be the expected
- Min-Cut
  - Picking random edge
  - Might not return correct result
  - Run time is bounded.
- Conversion Las vegas to monte carlo and the other way around