HomeWork- 6

Collaboration - Niscal and Abhilash
Sources - Wikipedia, Prof.Bo Notes, Notes by Christopher Hudzik and Sarah Knoop

Problem 1:
Part a:

- Set each X_i unit to true/false independently.
- A clause Z_i is True if any of the 3 unit is True. It is false when all the units are False.
- Hence, the Probability that clause Z_i is false ⅛. Z_i is true = 1- Probability Z_i is False = 1-⅛=⅞.

Part b:

- Probability that coin comes up with heads is p.
- Probability that coin gives no head is 1-p
- Probability of no heads after tossing coin for k times is $(1-p)^k$

Part c:

- Probability of getting at least one head after tossing coin for k times is $1 - (1-p)^k$

Part d:

- Probability of getting at least one head after tossing coin for $1/p \ln(1/\delta)$ is $1 - (1-p)^{1/p \ln(1/\delta)}$
- From hint $1 - x \leq e^{-x}$ we can replace $1 - p \leq e^{-p}$
- $1 - e^{-(p/p)*\ln(1/\delta)} = 1 - e^{-\ln(1/\delta)} = 1 - e^{\ln(\delta)} = 1 - \delta$
- Probability of getting at least one head after tossing coin for $1/p \ln(1/\delta)$ is $1 - \delta$. Hence Proved.

Part e

- We can write the probability of returning a minimum cut if we run algorithm independently for $(n(n-1)/2)*\ln(1/\delta)$ times as: $1 - (1 - 2/(n*(n-1)))^{n(n-1)/2 * \ln(1/\delta)}$
- We know that $1-x<=e^{-x}$. Let $y = 2/(n*(n-1)$. Substituting above equation
- $1 - e^{-y/y * \ln(1/\delta)} = 1 - e^{-1*\ln(1/\delta)} = 1 - e^{\ln(\delta)} = 1 - \delta$
- Hence proved.

Problem 2

Part a

- Suppose we have a graph G = (V, E), we define cut as partition of vertices in V into U1 and U2 such that for an edge e ε E is in the cut its one endpoint lies in U1 and other in U2.
- Max-Cut of a G = (V, E) is a cut C such that |C| is maximised over all cuts of G.

Randomized approximation algorithm:

- Finding the optimal solution is NP-Hard. We use an approximation that gives us cut C. We give a ∝-competitive approximation $|C|/|C^*| >= \propto$. I.e $|C|>=\propto.|C^*|$
- Easiest random algorithm is we assign each vertex v ε V independently with equal probability to either U1 or U2.
- For any edge $e_{(u,v)}$ ε E P[e ε C] = ½.
- We define a random variable $X_e = 1$ when $e_{(u,v)}$ ε C or $X_e = 0$ otherwise.
- We can write $|C| = \sum_{e \, \varepsilon \, E} X_e$
- Hence $E[|C|] = \sum_{e \, \varepsilon \, E} E[X_e] = \sum_{e \, \varepsilon \, E} P[e \, \varepsilon \, C] = |E|/2 >= |C^*|/2$
- Therefore, $|C|>=0.5*|C^*|$. Hence proved.

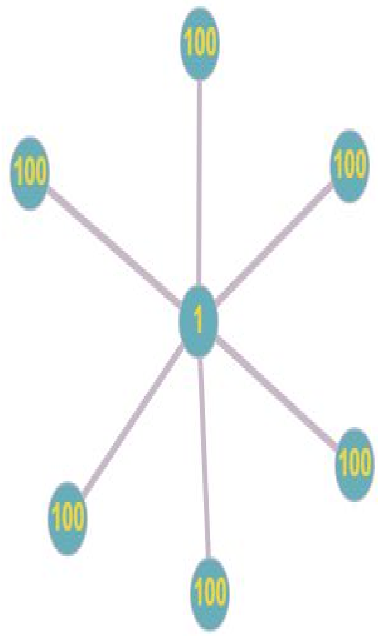
Part b

- If each edge has a weight associated with it. We can tweak the above algorithm a bit and achieve the same approximation ratio.
- We change the value of random variable as follows, $X_e = w_e$ when $e_{(u,v)}$ ε C or $X_e = 0$ otherwise.
- We can write $|C| = \sum_{e\,\varepsilon\,E} X_e = \sum_{e\,\varepsilon\,E} w_e$
- Thus $E[|C|] = E[\sum_{e\,\varepsilon\,E} w_e] = \sum_{e\,\varepsilon\,E} w_e\, P[e\ \varepsilon\ C]$
  $= ½ \sum_{e\,\varepsilon\,E} w_e = ½\ W >= ½\ |C^*|$ as total weight of all edges is the upper bound.
- Hence, $|C| >= ½\ |C^*|$ i.e $|C|>=0.5|C^*|$

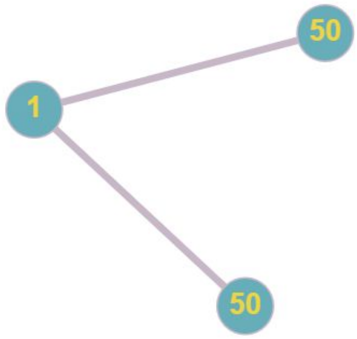
Part c

- For a graph $G(V,E)$ and $V = \{1,2,3\ldots,n\}$ we define $E_i = \{(k,i) \in E : k < i\}$
- Initially $V1 = \{1\}$ and $V2 = \{\phi\}$ for each i from 2 to n we add i to either V1 or V2 so that the number of edges that are on the cut are maximized.
- Let C be the cut obtained by the algorithm. The disjoint sets $E_1,E_2,\ldots,E_n$ partition E. So $|E| = \Sigma_i E_i$
- For each $i \in V$, let $E_i' = E_i \cap C$. As E_i sets are disjoint E_i' are disjoint too.
- $|C| = \Sigma_i E_i'$ and $E_i' >= 0.5*E_i$
- $|C|>=0.5*|E|$. $|C^*|<=|E|$ so $|C|>=0.5*|C^*|$

Problem 3
Part a

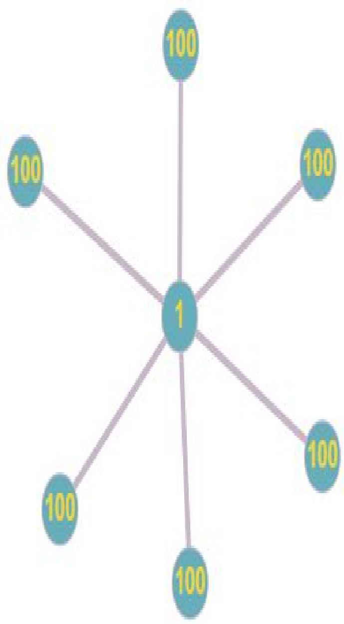


- For maximal matching M we choose an edge and add it to M and delete all the edges incident on either of the endpoints. Then S is set of all endpoints in M which forms a vertex cover.
- In above Graph, if we pick any edge we will have an edge(u,v) such that w_u= 1 and w_v = 100. Total weight of vertices in vertex cover S is 101.
- But optimal vertex cover set would only have center vertex with weight 1 as it covers all the edges. Hence, $|S|/|S^*|>=100/1$ i.e $|S|>=100*|S^*|$



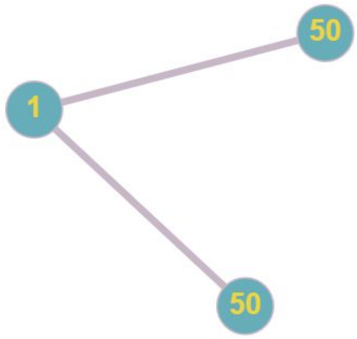
- In above graph irrespective of the edge we choose we get a $|S|>=50*|S^*|$ ratio.
- So we cannot get a constant approximation ratio. It depends on the weight of the vertices. Hence our maximal matching algorithm does not work in this case.

Part b



- For randomized algorithm,we pick an edge check if neither of its endpoints are in vertex cover S. If that's the case we pick one of its endpoints uniformly and add it to S.
- Initially S = {Φ}. Suppose we pick an edge e(u,v) from above graph such that w_u=1 and w_v=100. Edge e is not covered. Probability that we add vertex u = probability that we add vertex v = 0.5.
- Suppose we choose vertex v. S = S U {v}.
- We check next edge e(u,w) and neither of its vertices are in S. We may either choose u or w let's choose w. S = S U {w}
- In above case optimal algorithm would have chosen just the centre vertex with weight 1 but randomized algorithm can choose other vertices with equal probability.

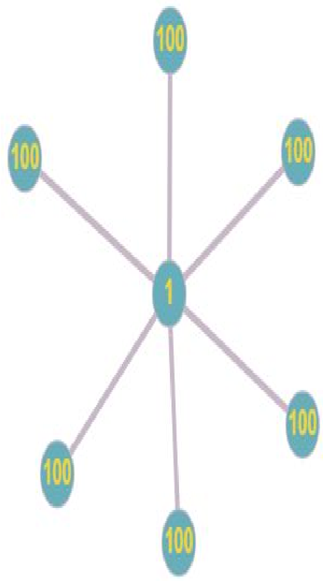
- Hence the approximation ratio $|S|/|S^*|$ can range from 1(when we select center vertex) to 600(when we keep selecting endpoints which are not the centre one)



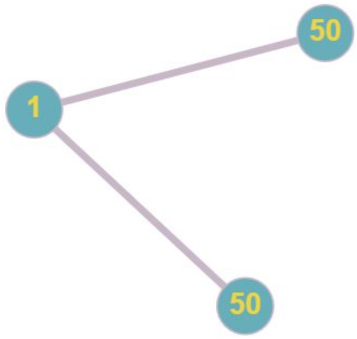
- Even for the above case approximation ratio ranges from 1 to 100 and is not a constant factor.
- We reach a similar conclusion as we did in part a that ratio is dependent on vertex weight and changes from example to example. Hence random algorithm does not not work either.

Part c

- For the randomized algorithm discussed in the class we check if edge e(u,v) is covered or not in S.
- If neither of edge e's endpoints are in S we add u to S with probability of $w_v/(w_u+w_v)$ and v to S with a probability of $w_u/(w_u+w_v)$
- The above algorithm is better than the algorithms in parts a and b.



- In the above case,consider edge e(u,v) where w_u=1 and w_v=100.
- We select vertex u with probability of 100/101 and vertex v by 1/101. If we add u to our vertex cover S we have covered atleast endpoint of all edges and in this case $|S|=|S^*|$



- Same for above example we select vertex with weight as 1 with probability 50/51 and rest with probability 1/51. Even in this case we get $|S|=|S^*|$
- $|S|=|S^*|$ is optimal case and may not happen that quite often but from Lemma 2 in class notes we have that $|S|<=2*|S^*|$