Lab12-RandomizedAlgorithm

Exercises for Algorithms by Xiaofeng Gao, 2018 Spring Semester.

∗ If there is any problem, please contact TA Jiaxi Liu (jsxzljx@163.com).

∗

1. Let $G(V, E)$ be a weighted graph with the weight function $d : E \to \mathbb{N}$. There is no edge from one node to itself. We partition V into k subsets: $V_1, V_2, \dots, V_k$. We define the weight of a subset V_i as $w(V_i) = \sum_{u,v \in V_i} d(u, v)$. You should design a k-approximation algorithm to find the k subsets to maximize the sum of subset weight $\sum_{i=1}^{k} w(V_i)$ and prove the approximation ratio of your algorithm.

   Hint: local search is a good option to solve the problem.

   Solution:
   Considering the Max cut problem from the lectures we conclude the same approximation ratio of our algorithm for this weighted graph G(V,E). We define the weight of a subset V_i as $w(V_i) = \sum_{u,v \in V_i} d(u, v)$. Now to design the k-approximation algorithm to find the k subsets to maximize the sum of subsets weight $\sum_{i=1}^{k} w(V_i)$ the ratio of our our algorithm —
   we will assume that G=(V,E) has a weight function on the nodes, and we consider formulation $d : E \to \mathbb{N}$ The aim of this, Is to show that metaheuristic techniques can be successfully applied to the k-cardinality tree problem. In contrast to constructive methods, they contain some random steps to avoid a fixed sequence of steps.

Algorithm 1: Local Cut Max Algorithm for k-Approximation

```
INPUT: G (V,E)

OUTPUT: Local Optimal Cut $(V_1..V_K)$

$s = s_0 = (\emptyset, V)$

$N(V_1..V_k)$ includes all $(V_{1k}...V_{nk})$ for $n = 1... V$

Now for the sake of simplicity Consider $(V_1, V_2)$

if $V_n \in V_1$ then $V_{1k} = V_1 - V_k$, $V_{2k} = V_2 + V_k$

if $V_n \in V_2$ then $V_{1k} = V_1 + V_k$, $V_{2k} - V_k$

repeat

select $s' \in N(s)$

if $m(s) < m(s')$ then

$s = s'$;

end if
UNTIL: All solutions in $N(s)$ has been visited

return s
```

The approximation ratio of our algorithm can be considered to be that the By the assumption on V there exist d such that $v = v_1 ... v_k$ Therefore either $v = \frac{V(G)=V(G)_{wd}}{V'}$ or $V = V(G)_{w_{d+1}}$ assuming that in G there is no edge between a node $v \in V$ and a node. Then all paths in G starting from a node $v \in V$ to a node contain a node. Hence there are atleast two troughts in G.

References:
https://www.sciencedirect.com/science/article/pii/S0166218X02005486
https://en.wikipedia.org/wiki/Maximum_cut
Lecture 18-Approximation Slides

2. Review the set cover problem introduced in the class and consider the following variant. Given a set of n elements $U = \{e_1, \dots, e_n\}$, a covering requirement set $R = \{r_1, r_2, \dots, r_n\}$ in which $r_i \in Z^+$ (here Z^+ means it is a positive integer) is assigned for element e_i, a collection of subsets $\mathbf{S} = \{S_1, \dots, S_m\}$, and a cost function $\mathbf{S} \to Q+$, find a minimum cost sub-collection of $\mathbf{S}$ such that each element $e_i \in U$ is covered at least r_i (note that the cost of picking a set $S \in \mathbf{S}$ k times is $k \cdot c(S)$).

   Give a LP relaxation for this problem, and a randomized rounding of it to obtain an $O(\log n)$ approximation algorithm for this problem.

   Solution:
   We select S_i as the subset assuming that X_i denotes to the variable our objective will be:
   $min \sum_{i=1}^{m} X_i.c(S_i)$
   $\sigma_{i=1}^{m} X_i I(e_j \in S_i \leq r_j) \ forj = 1..n$
   $X_i >= 0$ for $i = 1..m$

Algorithm 2: Set Cover via LP-Rounding (Randomized)

```
**INPUT:** U with n items; S with m subsets, cost function c($S_i$)

**OUTPUT:** Subset $S^{'} \subset S$ such that $U_{e_i} \in S_k \in S^{'}\ e_i = U$

Find an optimal solution $X_s$ to the LP-relaxation

Pick a constant c such that $(\frac{1}{e})^{clogn} \geq \frac{1}{4n}$

for $r = i$ to c log n do

for all S$\in$ S do

Pick S into $S_i'$ with probability $x_s$;

end for

end for

return $C' = U_{i=1}^{clogn} S_i'$
```

Reference

Lecture 18 slides

3. Alice and Bob want to choose one of the numbers 1,2 and 3 with equal probabilities. All they have is a biased coin that comes up heads with probability p, where $0 < p < 1$. Here are their ideas:

   - **Alice's method:** Toss the coin twice. If the two flips are HH, output 1, if they are HT, output 2, if they are TH, output 3. If the flips are TT, repeat the experiment.
   - **Bob's method:** Toss the coin twice. Output the number of heads obtained, plus 1.

   Consider their methods and answer the following questions:

   - For which values of p, if any, does Alice's method produce the numbers 1,2 and 3 with equal probabilities? And what about Bob's method then?
   - What is the expected number of tosses used by Alice's method.

Solution
All three methods in this question involve a simple experiment that is repeated until a number is generated. Clearly the whole method will be unbiased if and only if the simple experiment is unbiased.
(a) Only for p = 1/2, since we need p^2 = p(1 p), i.e., p = 1 p. No values, since in each experiment the probability of getting 1 is $(1p)^2$, that of 2 is 2p(1 p) and that of 3 is p^2, but these values can never all be equal.
(b) Let X be the number of rounds required for Alice to succeed, and Y be the number of tosses. Then Y = 2X, and X is a geometric random variable with probability of success being 1 $(1p)^2$.
Hence $E[Y] = 2 *$ [1-$(1 - p)^2$] ^{1} =2/(2p-p^2) p = 1/2
<u>Reference:</u>
https://people.eecs.berkeley.edu/~luca/cs174/psets/pmidterm-sol.ps

This is the last homework and wish you a success in the coming final.