Problem Description

Given a randomized approximation algorithm $A()$ which returns a 1.5-approximation of the Minimum Vertex Cover at least 3 out of 4 times when run on a graph. $A()$ has a runtime of $\mathcal{O}(|V|^2)$ where $|V|$ is the number of vertices in the graph. How can this algorithm be used to have a success rate of a least 0.999 while still using $\mathcal{O}(|V|^2)$ time.

Solution

Assuming that the return value of $A()$ is a list of the vertices that would compose the vertex cover, three out of four times the length of this list will be at most 1.5 times the length of a minimum vertex cover, while the remaining one out of four times the length can be of any length of a legitimate vertex cover. One simple approach to maximize that a correct candidate is found is by running $A()$ a multitude of time to get a number of different results and by some heuristic pick one of the obtained results that is most likely to be the 1.5-approximation of the vertex cover. The success of this is determined by two factors, firstly how many times $A()$ is ran to increase the likelihood that at least on of the results is the 1.5-approximation, and secondly, what heuristic we use to pick the best approximation out of all the different solutions generated by $A()$.

A simple approach is to store the shortest answer that $A()$ returns after running it multiple times since will always return a vertex cover. This way each time a solution for a graph is given the shortest and thus minimum solution found will be saved and returned at the end of the execution. This would manifest itself in the following pseudocode.

Algorithm 1 1.5 Minimum Vertex Cover

Input: G
Output: MVC

$MVC \leftarrow A(G)$
for 1 to $k-1$ **do**
  $TEMP \leftarrow A(G)$
  if $TEMP.length \leq MCV.length$ **then**
    $MVC \leftarrow TEMP$
  end if
end for

The number of loops in the for loop is currently represented by the variable k the value of k will be determined later.

Proof of Correctness

Let Q be the length of a minimal vertex cover for the graph G. The 1.5 Minimum Vertex Cover algorithm is expected to return a 1.5-approximation of the Minimum Vertex Cover (i.e. a vertex cover of length $1.5Q$) with a probability of at least 0.999. This can be proved by defining the random variable Y. Let for each run i in the *for* loop let $Y_i = 1$ **if** $A(G)_i.length \leq 1.5Q$ and $Y_i = 0$ **otherwise**. Since the algorithm picks the minimum result from all the calls to $A()$ if is sufficient to say that if there is a single result $A(G)_i$ that has a length less than or equal to $1.5Q$ the length of the vertex cover that is stored in MVC will also be at most $1.5Q$ long. This means that if a single Y_i is equal to 1 a vertex cover with a length less than or equal to $1.5Q$ is stored in MVC.

Let Y_{avg} be the average of all $Y_1...Y_k$, then the expected value of Y_{avg} will be $E[Y_{avg}] = 1\times\frac{3}{4}+0\times\frac{1}{4} = 0.75$. If the average of Y_{avg} is greater than 0, it means that there is at least a single $Y_i = 1$ and thus MCV is length $1.5Q$ or less. Thus the Chernoff bound can be used to calculate the probability of $Y_{avg} > 0$.

$$Pr[Y_{avg} < E[Y_{avg}] - \epsilon] \leq e^{-2k\epsilon^2}$$

Since we want to find the probability of Y_{avg} being greater than 0 we let $\epsilon = E[Y_{avg}] = 0.75$. To avoid any conflicts with limit of when $Y_{avg} = 0$ we give can instead find when $Y_{avg} > 0.05$ since $Y_{avg} > 0.05 > 0$. This implies that $\epsilon = 0.70$ yielding us the final equation.

$$Pr[Y_{avg} < 0.05] \leq e^{-2k\times 0.7^2}$$

For $k = 10$ this yields 0.0000555 So in this scenario the probability that $Y_{avg} < 0.05$ is very small, in other words is is very likely that $Y_{avg} > 0.05$ meaning that the probability of at least one $Y_i = 1$ is $Pr[\exists Y_i = 1] > 1 - 0.0000555 = 0.9999$. In summary, the algorithm has a probability of more than 0.999 that there will be a $Y_i = 1$ which implies that MVC is a vertex cover of length less than or equal to $1.5Q$ when $k = 10$. Thus MVC is a vertex cover of size $\leq 1.5m$ where m is the size of a minimum vertex cover with probability of at least 0.999.

Complexity of 1.5 Minimum Vertex Cover

The algorithm $A()$ will be run k times, once on line 1 and then $k-1$ times in the *for* loop, this is $\mathcal{O}(k|V|^2)$. On line 4 in the *for* loop the length of $TEMP$ and MCV is checked, if they are implemented as lists this operation takes at worst $|V|$ steps since the vertex cover can at worst include each vertex. Resulting in $\mathcal{O}(k|V|)$. The assignment on line 5 can be assumed to take constant time with some clever pointer arithmetic. In summary the Complexity of the algorithm as a whole is $\mathcal{O}(k|V|^2 + k|V|) = \mathcal{O}(|V|^2)$