Individual Assignment 1

David Östling
dostl@kth.se
DD2440

December 11, 2022

Problem description

Given a randomized approximation algorithm A for minimum vertex cover. The authors of A claim that if you execute their code repeatedly on the same graph you will get a 1.5-approximation of a minimum vertex cover with $Pr[A_s \leq 1.5m] \geq 3/4$ *(probability of the algorithm A returning a 1.5 approximation of a minimum vertex cover where A_s is the size of the vertex cover returned by A and m is the size of a minimum vertex cover).* The code runs in O($|V|^2$) worst-case time, where |V| is the number of vertices in the graph, and that it *always returns a vertex cover of some size.* It can be assumed that the code works as described by the authors and that it can be freely used.

Improve the success rate to at least 0.999 while still using only O($|V|^2$) time by designing an O(|V|2) time algorithm that returns a vertex cover of size s 1.5m, where m is the size of a minimum vertex cover, with probability at least 0.999.

Solution

We begin by understanding how when running the code multiple times, nothing in one run of A will affect any upcoming run. Thus, we can assume that each execution of A runs *independently.*

As a result, a suitable strategy would be to execute the code a specific amount of times n until the bound probability of not returning a 1.5 approximation of a minimum vertex cover is $\leq 1/10^3$. This is due to the fact that we want to reach a success rate of at least 0,999 meaning a bound probability of not returning a 1.5 approximation of a minimum vertex cover of at most:

$$1 \text{ - } 0{,}999 = 1/10^3$$

When the overall probability *(the product of all probabilites)* of all executions of A not returning a 1.5 approximation of a minimimum vertex cover result in $\leq 1/10^3$ we will see an improvement of the desired grade. To ensure this probibalistic improvement in particular, we will need to execute the given algorithm *A at least 5 times.* We then store the outputs of each execution of A and return the minimum of them.

Pseudocode

Algorithm 1 Minimum vertex cover - probability improvement

$A \leftarrow$ a randomized approximation minimum vertex cover algorithm with time complexity O(|V|^2).
outputList $\leftarrow$ an empty list which is going to contain the outputs of A
for $i = 1, 2, \ldots, (5)$ **do**
    outputList.append(**call** function A)
return min(outputList)

Time Complexity

The time complexity of the algorithm A will remain the same as we do not modify the code; O(|V|^2) *where V is the number of vertices in the graph*. However, it is important to note that we do run the algorithm for a total of five times. Even though this might be the case, the number of times the algorithm A is executed does not depend on V. Because of this, the code will run a constant amount of times and the time complexity will remain bounded by O(|V|^2) since:

$$5 * \mathrm{O}(|\mathrm{V}|^2) = O(|V|^2)$$

(where V is the number of vertices in the graph)

Correctness

Lemma 1: $Pr[A_s > 1.5m] \leq 1/4$

Proof: Since the $Pr[A_s \leq 1.5m] \leq 3/4$ we can understand what the probability of the algorithm A not returning a 1.5 approximation of a minimum vertex cover (or in other words; $Pr[A_s > 1.5m]$) will be for each independent executon of A. This is done as below:

$$Pr[A_s > 1.5m] \leq 1 - (3/4) = 1/4.$$

(where A_s is the size of the vertex cover returned by A and m is the size of a minimum vertex cover)

Thus, meaning that $Pr[A_s > 1.5m] \leq 1/4$. □

Lemma 2: $(1/4)^5 \leq 1/10^3$

Proof: As of *Lemma 1*, a single execution of A has $Pr[A_s > 1.5m] \leq 1/4$ meaning for every run of A the probability will rise exponentially as below:

$$(1/4)^n \textit{ where n is the amount of code executions of A}$$

We also know that the bound probability of failure can be at most $\leq 1/10^3$. This is due to the fact that we want to reach a success rate of at least 0,999 meaning a bound probability of all executions of A not returning a 1.5 approximation of a minimum vertex cover of at most:

$$1 - 0{,}999 = 1/10^3$$

As a result of the above, to prove the lemma we will now need to check when $(1/4)^n \leq 1/10^3$ meaning:

$$(1/4)^n = 1/10^3 => log(1/4) * n = log(1/1000) => n = log(1/1000)/log(1/4) => n = 4.982892142 \approxeq 5$$

Hence, we now know that the algorithm A needs to be executed a minimum of 5 times before it reaches a probability score of at least 0,999 (or in other words $(1/4)^5 \leq 1/10^3$). $\square$