Stat4DS / Homework 01

(Part A)

Pierpaolo Brutti

Due Sunday, November 22, 2020, 23:59 PM on Moodle

General Instructions

I expect you to upload your solutions on Moodle as a **single running** `R Markdown` file (`.rmd`) + its `html` output, named with your surnames.

You will give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Your responses must be supported by both textual explanations and the code you generate to produce your results. *Just examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands and functions.*

R Markdown Test

To be sure that everything is working fine, start `RStudio` and create an empty project called `HW1`. Now open a new `R Markdown` file (`File > New File > R Markdown...`); set the output to `HTML mode`, press `OK` and then click on `Knit HTML`. This should produce a web page with the knitting procedure executing the default code blocks. You can now start editing this file to produce your homework submission.

Please Notice

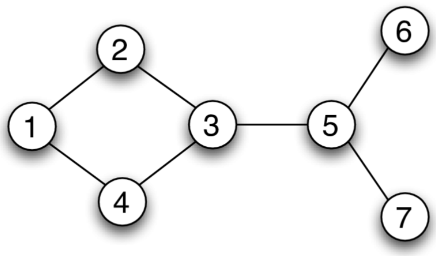
- For more info on `R Markdown`, check the support webpage that explains the main steps and ingredients: R Markdown from RStudio.
- For more info on how to write math formulas in LaTex: Wikibooks.
- Remember our **policy on collaboration**: *Collaboration on homework assignments with fellow students is* ***encouraged****. However, such collaboration should be clearly acknowledged, by listing the names of the students with whom you have had discussions concerning your solution. You may* ***not****, however, share written work or code after discussing a problem with others. The solutions should be written by* ***you****.*

Exercise: Randomize this...

1. Background

The **Max-Cut** problem is a foundational problem in combinatorial optimization and approximation algorithms that goes as follows. Let $G = (V, E)$ be an undirected graph where V are the vertices, and E the set of (undirected) edges, that is, unordered pairs of vertices.

For example the graph below has $V = \{1, 2, 3, 4, 5, 6, 7\}$ and $E = \big\{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{4, 3\}, \{3, 5\}, \{5, 6\}, \{5, 7\}\big\}$.



Now, for any vertex set $U \subseteq V$, define the **cut** determined by U as

$$\delta(U) = \big\{\{u, v\} \in E \text{ such that } u \in U \text{ and } v \notin U\big\}.$$

The Max-Cut problem is then to solve:

$$\max\big\{\mathrm{card}(\delta(U)) \text{ for } U \subseteq V\big\},$$

where $\mathrm{card}(A)$ denotes the cardinality of the set A.

This is notoriously an NP-hard problem hence we cannot hope to solve it exactly for even moderately sized graphs.

Instead, we will be content to design a randomized algorithm able to find a *large enough* cut with high probability.

We'll see that the algorithm is so simple it's almost incredible it works! Yeah... but in which probabilistic sense "it works"?

So, let $\textsc{opt} = \textsc{opt}(G)$ be the size of the maximum cut we are chasing. Our goal is to find an algorithm for which there is a factor $\alpha > 0$ independent on the graph G such that the set U it builds is guaranteed to have

$$\mathrm{card}(\delta(U)) \geqslant \alpha \times \textsc{opt}.$$

Since the algorithm will be randomized, we may want this property to hold with probability close to 1, for example. In this exercise we will content ourselves with something weaker than this.

As a matter of fact, there are many algorithms out there with $\alpha = 1/2$. Here's one that appeared in an old, 1967 paper (in Hungarian!) by the mythical Paul Erdos.

Randomized Max-Cut Algorithm

Create U as a random subset of V; that is, for each vertex $v \in V$, flip a fair coin: if Heads, add v to U otherwise do not.

Note that the algorithm does not even look at the edges of the graph G! Despite this myopic behaviour, it can be shown this algorithm outputs a cut whose expected size is large, more specifically we have:

Performance Analysis

Let U be the set chosen by this (randomized) algorithm.
Then the expected size of the cut set determined by U is at least $\textsc{opt}/2$:

$$\mathbb{E}\big(\mathrm{card}(\delta(U))\big) \geqslant \frac{\mathrm{OPT}}{2}.$$

$\rightsquigarrow$ Your job $\leftsquigarrow$

1. If you haven't already, take a look at basic tools to deal with graphs in R such as the `igraph`, `ggraph` packages, and maybe also the `maxcut()` function within the `sdpt3r` package.
2. Setup a suitable simulation study in R to double-check the above-mentioned performance of the *Randomized Max-Cut Algorithm*. More specifically, you should
   - Pick a specific (small) graph G (even a random graph).
   - Get its true $\textsc{opt}(G)$ or, at the very least, a quite good approximation to $\textsc{opt}(G)$.
   - Run the *Randomized Max-Cut Algorithm* a large number M of times.
   - Evaluate the average cut-size over these M simulations and compare it with the theoretical bound $\textsc{opt}(G)/2$.
   - Finally change the graph size to see if there is an impact on the performance.

   Please Notice: comment A LOT your results, complement your analysis with plots, tables and numerical summaries. Show the graphs you are dealing with, also highlighting the structure of the optimal cut.