Programming Exercise 2 Advanced Algorithms
Semidefinite Relaxation of MAXCUT

Deadline: Sunday January 22, 2017, 23:50

Expected results. For this exercise we expect two products from you: an implementation of the algorithm, including the requested conversions and tests, and a report on the test results of this algorithm. Both should be submitted in one file through CPM. For this exercise, it is allowed to work in groups of two students.

Your implementation should take the following into consideration:

- It is allowed to make use of (both standard and algorithmic) libraries. You should indicate which ones you use.
- Your code should meet the requirements for neat programming from earlier courses.

Your report should meet the following constraints:

- It should be at most three pages A4 (excluding an optional front page, and space used by diagrams or figures), and in PDF.
- It should be written in clear, readable, and correct English.
- It should contain clearly readable graphs of the performance of the different algorithms (use for example gnuplot).
- It should contain your name(s), student number(s), the exercise number, and
- for each question a clear explanation about how you obtained the answer.

Submit your implementation and report through CPM before the deadline has passed. The most recent file submitted before the deadline will be graded.

Assessment and feedback. We will check both your report as well as your code on

- correctness,
- clarity,
- quality, and
- completeness with respect to the requirements above.

Your grade will be given through CPM and the feedback on your answers will be written on your report, which you can retrieve from the assistants.

Weighted MAXCUT. In this assignment you are going to determine upper and lower bounds on the optimal value of MAXCUT instances of varying sizes using the SDP-approach. The upper bounds are derived from the semidefinite relaxation, and the lower bounds from the Goemans-Williamson approximation algorithm.

To create your own weighted graphs a graph generator has been made available (GraphGenerator.java, see Blackboard). It takes four arguments:

- Number of nodes
- Edge probability (between 0 and 1)
- Maximum weight of the edges (edges are assigned a weight between [1..maxweight]
- An id, to create a unique filename in case you want to generate more graphs with the same settings

The output is a file containing the number of nodes and a list of edges in the form node1 node2 weight.

The assignment. You will solve a semidefinite relaxation of MAXCUT using an SDP solver. The SDP relaxation should be run in Matlab (which can be obtained from the student software on Blackboard) using the Yalmip shell:
`https://yalmip.github.io/`
and an SDP solver:
`https://yalmip.github.io/allsolvers/`
You can use the SEDUMI solver, but you are free to experiment with other solvers. Please indicate in your report which solver you use. You can use the standard timing used by Matlab to determine the performance. An example Matlab file (sdptest.m) is available to see how to run the SDP solver (see Blackboard). Do not forget to put Yalmip and your SDP solver in the Matlab path (File $\rightarrow$ Set Path).

Generate "enough" graphs with edge probability p either 0.5 or 1. Vary also with the maximum weight W (maxweight). Convert the MAXCUT problem into its SDP relaxation according the notes of Lecture 12 of the course. Run the SDP's and give the upper bounds for the created graphs. Use the generated matrices by the SDP solver to give lower bounds to the same samples. Try several numbers of trials T of hyperplane roundings to generate these lower bounds.

Estimate the polynomial runtime by gathering enough data for various p, W, T and number of nodes up to 70. (Of course you are allowed to take higher numbers, but for this assignment it is not needed to let the solver run more than 5 minutes per instance.)

Give in your report at least the following:

1. The intuition behind the SDP algorithm for giving both lower- and upper bounds.
2. Your expectations before running the algorithm on all test problems.
3. Various plots to visualize the results of your tests.
4. An estimation of the runtime of this algorithm and an estimation of its accuracy. Of course your plots should support your estimations.