Advanced Algorithms: Programming Assignment 3

Olivier Maas (4606159)
Johanna Korte (4697928)

January 21, 2018

Introduction

For our second programming assignment we were asked to implement a semidefinite relaxation of the MAXCUT problem using an SDP solver, and solve it on various test graphs. To this end, we employed the YALMIP package [2] implementing the SeDuMi-solver [1], and compared their results with those yielded by the Goemans-Williamson 0.87856-approximation algorithm [3] for the MAXCUT problem.

Given that the SDP-solver numerically (and optimally!) solves a more 'loose' version of any given problem, we took its solution to any problem to be the upper bound on the potential quality of any optimal cut for that problem.
Meanwhile, the Goemans-Williamson algorithm utilizes a 'rounding' step which is guaranteed to be repeated only a small number of times; the cut yielded by this step in any of its rounds, may not meet the algorithm's approximation guarantee; but the cut yielded by the algorithm as a whole, does.

Implementation

Our specific experiment settings were as follows.
We generated graphs based on three pertinent factors: The amount of nodes; the probability of there being an edge between any two nodes; and the maximum weight such an edge might have. The amount of nodes we let vary from 5 up to 75, in steps of 5. The probability of there being an edge between any two nodes was either 0.5 or 1. The maximum weights tested were: 1, 5, 10, 50, 100. For each possible combination of these features, we generated four graphs.

In our implementation of the Goemans-Williamson algorithm, we would always 'draw' at least five random vectors to calculate a cut with, regardless of how long it took to find one that fulfilled the algorithm's approximation guarantee. In practice, across our 600 test cases, we never once encountered that more than five random draws were needed.
For each random vector that was drawn, we calculated the corresponding cut and its weight. Across these five 'trials' that we ran for each test case, we took the minimum such cut weight that we would find, and took this as a lower bound.

Expectations

Prior to running the experiments, we had a number of expectations:

1. Since we coded an approximation algorithm, we expected a polynomial runtime.

2. We expected the runtime in the case of $p = 1$ to be higher than in the case of $p = 0.5$, owing to the increase in the amount of edges that should be checked; but this would be a polynomial increase, not an exponential one.

3. We had no expectations regarding the accuracy of the Goemans-Williamson algorithm, besides that the results should fulfill the approximation guarantee (which trivially follows from the way we coded the algorithm).

Results

The following sections show the obtained results. Figure 1 displays the runtime of the algorithm for different graph sizes. Figure 3 shows the resulting weights—namely the lower bound; the weight achieved by the Goemans-Williamson algorithm; and the super-optimal SDP-weight—for varying graph sizes and edge weights.

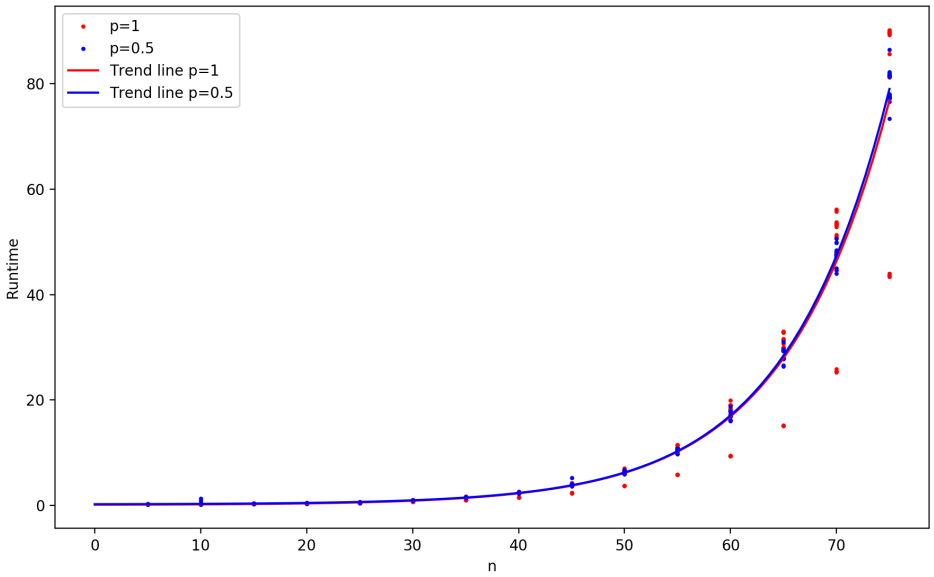

Figure 1: Scatter plot with trend lines showing runtime at different graph sizes, for edge probability 0.5 and 1.0. Runtime is in seconds.

Estimation of runtime

The trend lines in figure 1 give an indication of an exponential runtime. This is verified by figure 2, in which we have performed a numeric fit of the data, yielding the following exponential function (where runtime in seconds is y and amount of nodes is x):

$$y = 0.1034 \cdot e^{0.0841x}$$

Note that we also tried polynomial fits (dimensions ranging from 1 to 3); however, this exponential fit seemed the best one. This is surprising; we had expected a polynomial runtime! Fortunately, the exponential factor is very low.

Estimation of accuracy

Figure 1 shows another startling fact: namely, that contrary to our expectations, the value of p mattered little for the achieved runtime. This implies that the algorithm's runtime is heavily

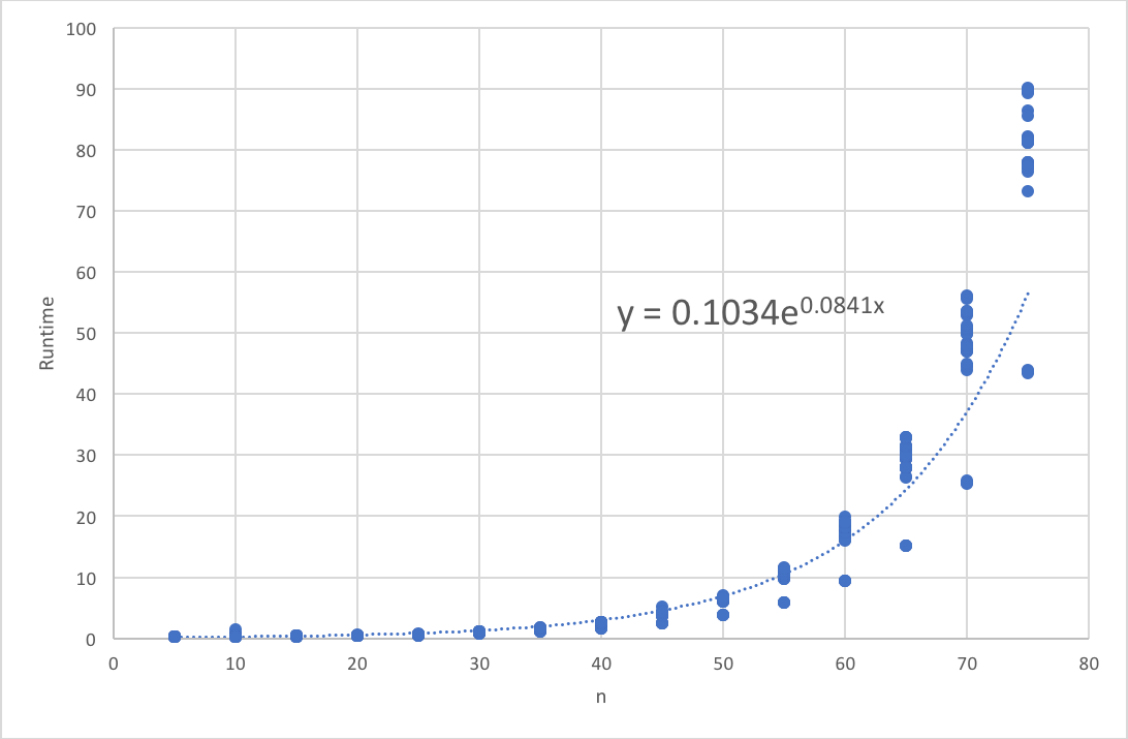

Figure 2: Runtime in seconds displayed as a scatterplot at different graph sizes, including exponential fit

dominated by the SDP-calculation, which, unlike the rest of the Goemans-Williamson algorithm, does not rely so directly on the amount of edges in the given graphs.

As can be seen in figure 3, the accuracy of the achieved cut's weight relative to the corresponding SDP-weight is roughly equal for all different sizes of n and maximum edge weight. The weights of the lowerbound and the resulting cut are very close together on all occasions, whereas the semidefinite programming returns a noticeably higher weight. This suggests that the algorithm often performs very close to its approximation guarantee, which is of some interest.

References

[1] Sedumi. https://yalmip.github.io/solver/sedumi/.

[2] Yalmip. https://yalmip.github.io/.

[3] Goemans, M., Williamson, D. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the Association for Computing Machinery, Vol. 42, No.6, November 1995, pp.1115-1145., 1995.

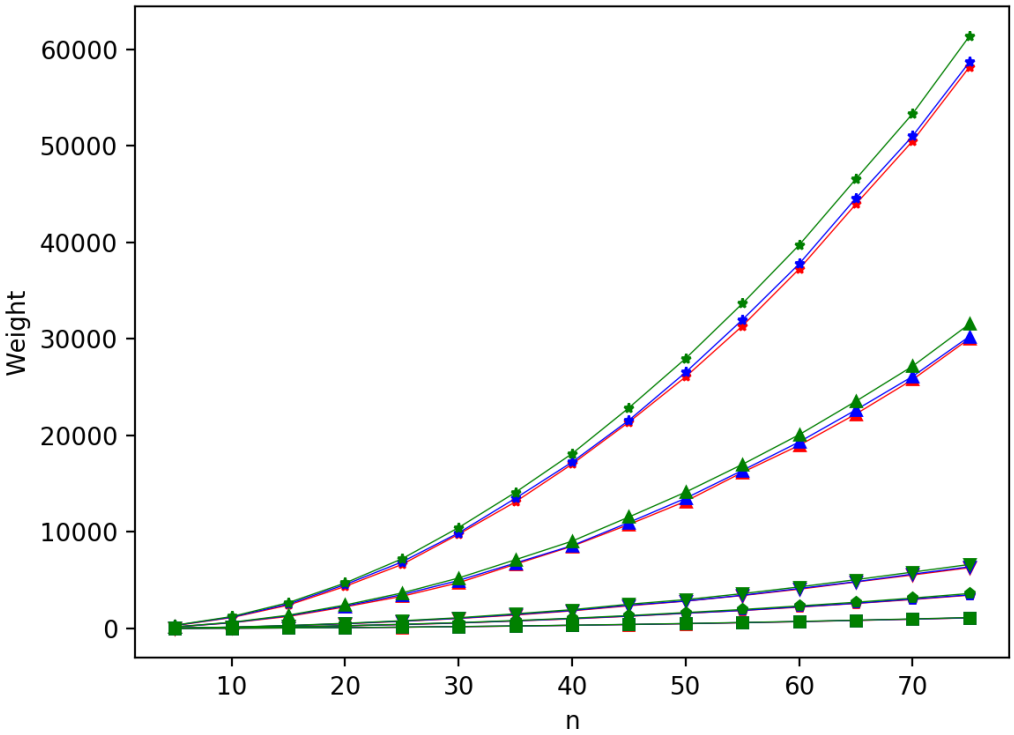

Figure 3: Lowerbound (red), weight of the resulting cut (blue), and sdp weight (green) for varying graph size. Star markers indicate a maximum edge weight (MEW) of 100, for upwards triangles MEW=50, downwards triangles indicate MEW=10, MEW=5 is represented by circles, and square marks represent MEW=1