Advanced Algorithms

Programming Exercise 2

Alenka Bavdaz - 4615093
Sharif Jacobino - 4204557

January 2017

1 Introduction

This report is written to be submitted with the second programming exercise for the course Advanced Algorithms, as part of the Computer Science Masters programme at TU Delft. The assignment is to solve a semi-definite relaxation of MAXCUT using an SDP solver and derive lower and upper bounds on the optimal solution.

2 The SDP algorithm

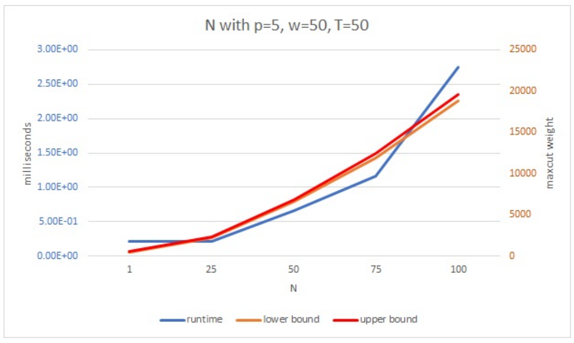
The Sedumi SDP solver is used to solve the semidefinite relaxation of the MAXCUT problem. The objective of the SDP is to find the maximum weight and so the optimal cut in the instance. By being a relaxation of the maximisation problem MAXCUT, the result of the solver would be an upper bound on the optimal MAXCUT weight. For the lower bound, we perform the randomised rounding part of the Goemans-Williamson algorithm for several trials. This gives us a set of approximate results at least 0.878 times the optimal weight (as a result of the Goemans-Williamson approximation guarantee). By taking the largest of these results we get the tightest lower bound on the optimal weight we can derive from the random results.

3 Expectations

A trivial expectation we had is that larger instances would take longer to solve. Thus by extension we also expected instances with an edge probability of 1 to take longer to solve than instances with an edge probability of 0.5. This is because we expect an instance with an edge probability of 0.5 to have half as many edges, and thus be smaller than an instances with an edge probability of 1 (a complete graph). We expect the maximum weight to be positively correlated with the optimal cut weight and thus the lower and upper bound. The effect of the maximum weight on the runtime however is not clear in advance. While we expect longer runtimes with more trials, we also expect the bounds to get tighter with more trials.

4 Results

We decided to plot each attribute three times (for small, medium and large instance size) with the other attributes being static, so that we can see the effect of each attribute on the runtime.



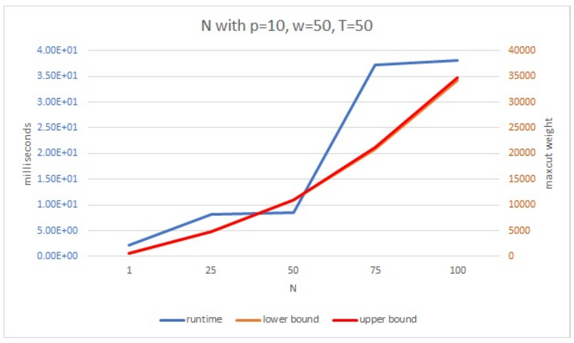


Figure 1: Variable N

Figure 1 shows that more nodes result in longer runtimes and higher optimal cut weights, as was expected.

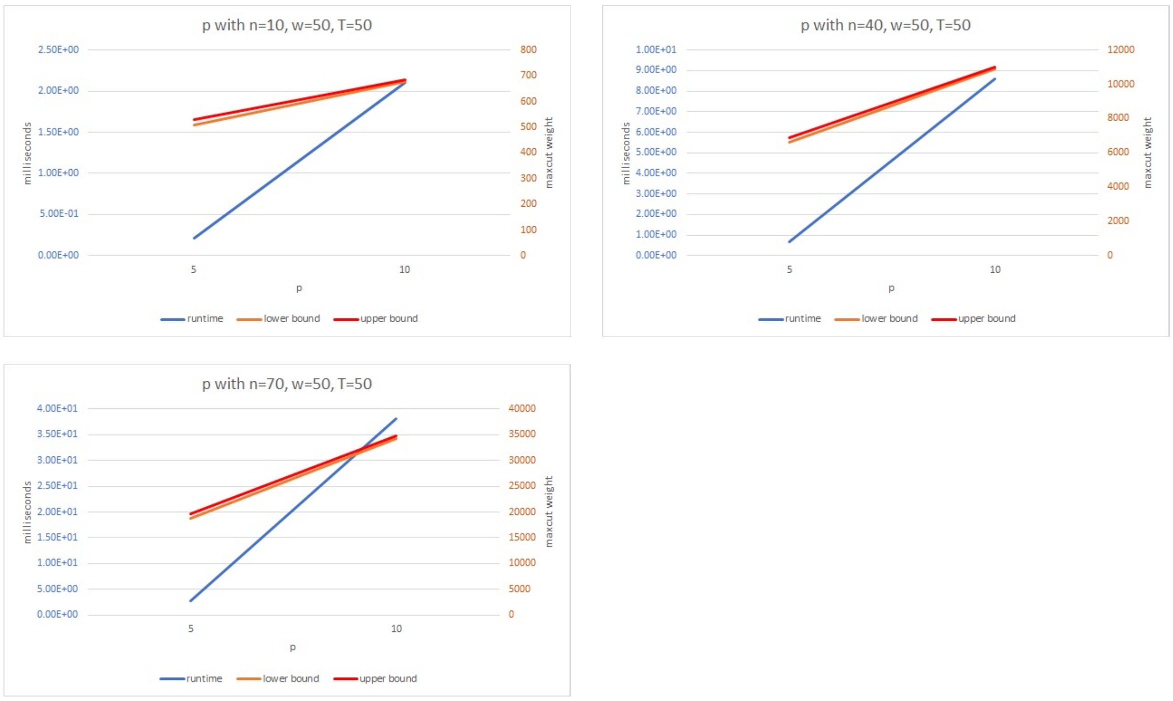


Figure 2: Variable p

Increasing the edge probability also results in longer runtimes but also higher optimal cut weights, both as a result of having more edges. This can be seen in figure 2.

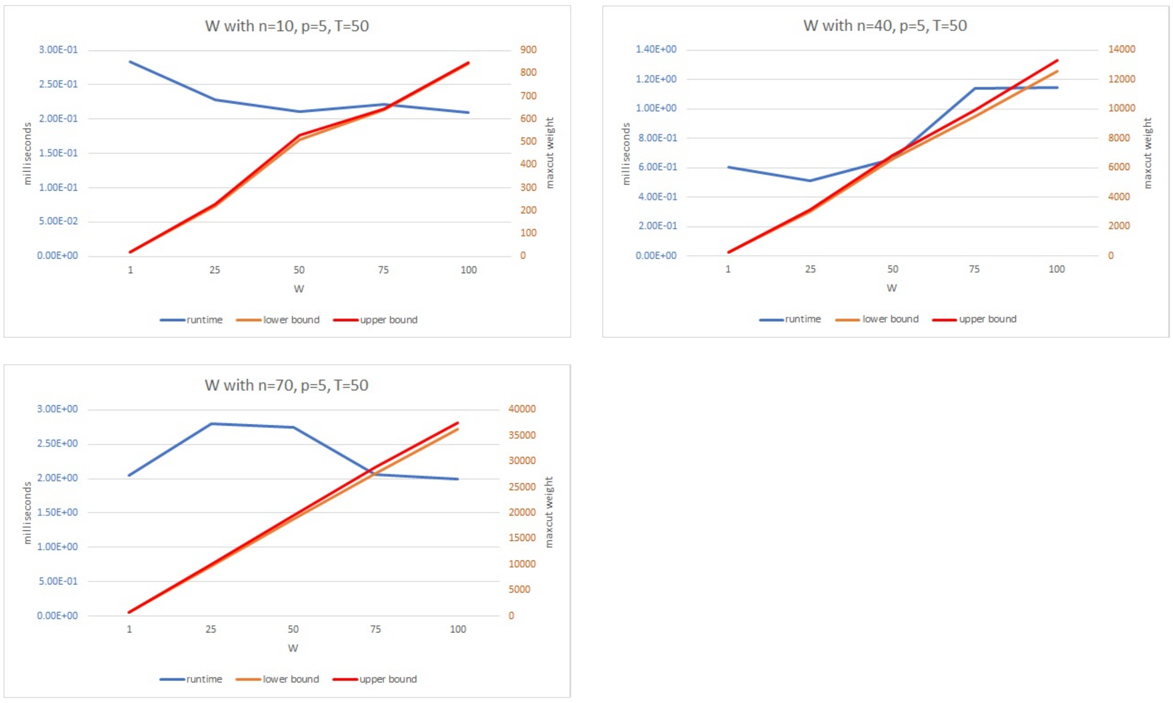


Figure 3: Variable W

In Figure 3 we clearly see that the maximum weight is positively correlated with the optimal cut weight. The maximum weight seems to have no clear effect on the runtime however.

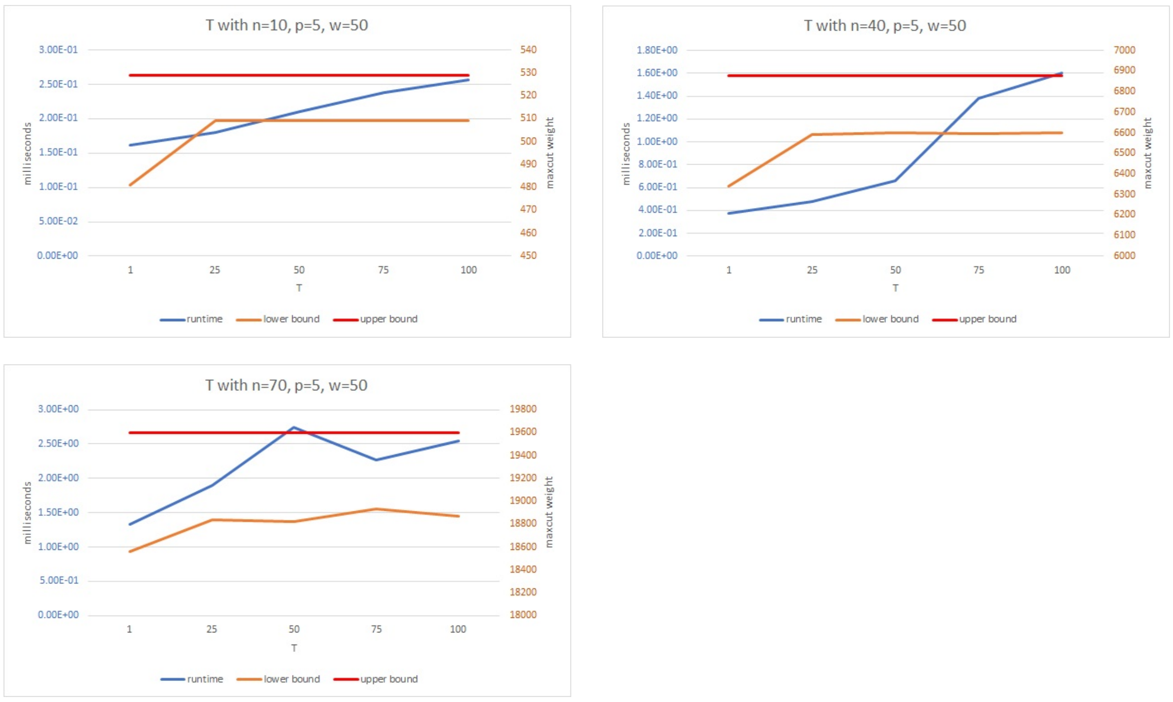


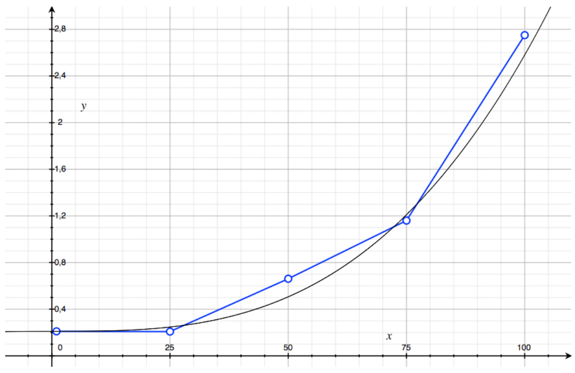
Figure 4: Variable T

In exchange for taking longer to run, the bounds get tighter with more trials, as shown in figure 4. Less trials are needed with less nodes to tighten the bounds.

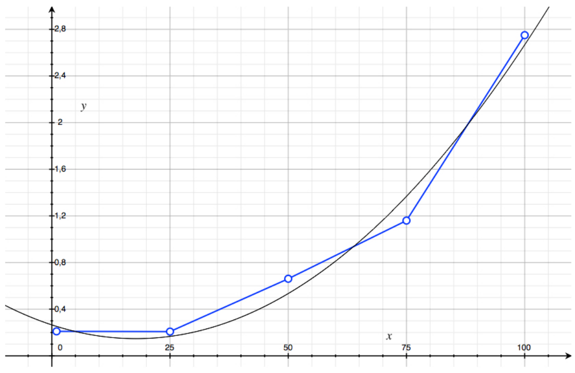
5 Run time estimation

The runtime of the algorithm is polynomially bounded in the size of the instance due to the SDP solver running in polynomial time and the polynomial amount of steps required to for the randomised rounding in the Goemas-Williamson algorithm.

We tried finding a curve that approximates the runtime curve shown in the first graph of figure 1. Graph a in figure 5 shows a simple 3rd degree polynomial that we constructed to approximate the runtime. Graph b in figure 5 shows a 2nd degree polynomial constructed by an interpolator. From these graphs we can conclude that the runtime of the algorithm is close to $O(n^2)$.



(a) $0.21 + (\frac{x}{75})^3$



(b) $0.2656 - 0.0133x + 0.00037286x^2$

Figure 5: Runtime approximation