

Solving zero-one Linear Programming problem as Quadratic Programming problem

P.J.P.S. Dobbelsteyn

July 29, 2005

Abstract

The Maximum Clique problem can be formulated as a zero-one Linear Programming problem. Another mathematical programming formulation is presented: a Quadratic Programming problem formulation. The idea is to weaken the zero-one constraints and add a penalty in the *objective function* which has to ensure the global maxima are zero-one.

A proof is provided that the global maxima of both mathematical programming problems correspond. Furthermore, a series of experiments is conducted, with the conclusion that the Quadratic Programming problem solver, `quadprog`, does not solve the Maximum Clique problem efficient. On low density graphs `quadprog` performs very badly, providing fractional solutions. On high density graphs on the other hand `quadprog` performs very well, but the zero-one Linear Programming problem solver, `bintprog`, solves the problem just as fast.

Altogether the suggested Quadratic Programming formulation is not a good alternative for the zero-one Linear Programming formulation, because the expected computation time gain is absent and the degree of correctly solved problems is low. **Keywords.** Maximum Clique problem, Quadratic Programming, Zero-one Linear Programming.

1 Introduction

Zero-one Linear Programming is an often used technique for solving mathematical problems. This topic is a vast and highly complex field of study, about which an enormous literature exists. A good introduction in zero-one Linear Programming is found in Hromkovic [3] and references therein. Zero-one Linear Programming is shown to be \mathcal{NP} -complete, see Garey and Johnson [2].

1.1 The problem of interest

The fundamental problem of zero-one Linear Programming is to find the minimum or maximum of a linear *objective function* over a polyhedron, that is, subject to *linear constraints*. There are many standard forms in which zero-one Linear Programming problems can be written. For combinatorial applications, one common form is

$$\begin{aligned} & \text{maximize} && \mathbf{c} \mathbf{x} \\ & \text{subject to} && A \mathbf{x} \leq \mathbf{b} \\ & && \mathbf{x} \in \{0, 1\}^n \end{aligned} \quad (1)$$

where A is an $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, and each variable in \mathbf{x} is either 0 or 1.

Such a zero-one Linear Programming problem is hard to solve due to the zero-one constraints. The idea is to reformulate this zero-one Linear Programming problem as a Quadratic Programming problem, in which the zero-one constraints are replaced by weaker constraints and a penalty

$$\begin{aligned} & \text{maximize} && \mathbf{c} \mathbf{x} - K \mathbf{x}(\mathbf{1} - \mathbf{x}) \\ & \text{subject to} && A \mathbf{x} \leq \mathbf{b} \\ & && \mathbf{x} \in [0, 1]^n \end{aligned} \quad (2)$$

where K is a constant and $\mathbf{1}$ stands for the one vector of dimension n . The weaker constraints tell the variables in \mathbf{x} not to take on only the values 0 or 1, but also values between 0 and 1. The penalty has to ensure that the variables are all 0 or 1 in the optimal solution of the problem.

1.2 Problem statement

The goal of this thesis is to show whether or not the suggested Quadratic Programming problem in (2) is a good alternative for the zero-one Linear Programming problem in (4). First, the maxima of the Quadratic Programming problem in (2) have to be shown to correspond with the maxima of the zero-one Linear Programming problem in (4). Second, an empirical comparison on computation time of both mathematical programming problems has to be conducted.

It is not within the timespan of a Bachelor thesis to show these goals for all zero-one Linear Programming problems. Therefore, this thesis limits it self to the zero-one Linear Programming problem for the un-weighted Maximum Clique problem in graphs.

2 Maximum Clique problem

Let $G = (V, E)$ be an arbitrary undirected graph. $V = \{v_1, v_2, \dots, v_n\}$ is the vertex set of G and $E \subseteq V \times V$ is the edge set of G . The complement of $G = (V, E)$ is the graph $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j \text{ and } (v_i, v_j) \notin E\}$. For a subset $S \subseteq V$, let $G(S)$ denote the subgraph induced by S .

A graph $G = (V, E)$ is *complete* if its vertices are pairwise adjacent, i.e. $\forall v_i, v_j \in V, (v_i, v_j) \in E$. A *clique* C is a subset of V such that the induced graph $G(C)$ is complete. The Maximum Clique problem is to find a clique of maximum cardinality in a graph G .

Practical applications of the Maximum Clique problem include project selection, classification theory, fault tolerance, coding theory, computer vision, economics, information retrieval, signal transmission theory, aligning DNA and protein sequences, and other specific problems. These applications were reviewed in Pardalos et al. [7] and references therein. The Maximum Clique problem is shown to be \mathcal{NP} -complete, see Karp [4].

2.1 Zero-one Linear Programming problem

The Maximum Clique problem can be formulated as a zero-one Linear Programming problem, see Lovász and Plummer [5]:

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^n x_i \\ & \text{subject to} && x_i + x_j \leq 1, \forall (v_i, v_j) \in \bar{E} \\ & && x_i \in \{0, 1\}, i = 1, \dots, n \end{aligned} \quad (3)$$

In this formulation, if $x_i = 1$, then $v_i \in C$, and if $x_i = 0$, then $v_i \notin C$, where C is the maximum clique.

The constraints $x_i + x_j \leq 1, \forall (v_i, v_j) \in \bar{E}$ in (3) eliminate the possibilities that two vertices are in a clique together when these two vertices are not adjacent.

The zero-one constraints in (3) ensure that vertices are either in a clique or not in a clique.

The zero-one Linear Programming formulation in (3) is proven to solve the Maximum Clique problem, see [1].

2.1.1 Example

Consider the Maximum Clique problem in the following graph:

To formulate this Maximum Clique problem as a zero-one Linear Programming problem, one needs the complement graph of G :

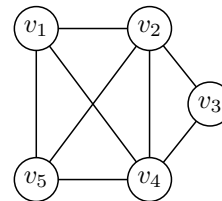


Figure 1: $G = (V, E)$

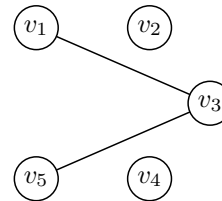


Figure 2: $\bar{G} = (V, \bar{E})$

For each edge $(v_i, v_j) \in \bar{E}$ the constraint $x_i + x_j \leq 1$ is added to the zero-one Linear Programming formulation. The complete zero-one Linear Programming formulation for this example is the following:

$$\begin{aligned} & \text{maximize} && x_1 + x_2 + x_3 + x_4 + x_5 \\ & \text{subject to} && x_1 + x_3 \leq 1 \\ & && x_3 + x_5 \leq 1 \\ & && x_i \in \{0, 1\}, i = 1, \dots, 5 \end{aligned} \quad (4)$$

The unique solution of this problem is the vector $x = (x_1, x_2, x_3, x_4, x_5) = (1, 1, 0, 1, 1)$. This solution corresponds with the maximum clique shown in the following graph:

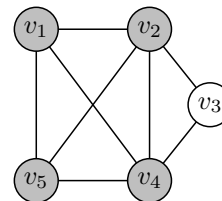


Figure 3: Maximum clique

2.2 Quadratic Programming problem

Now one knows how to formulate the Maximum Clique problem as a zero-one Linear Programming problem, the suggested Quadratic Programming formulation (2) can be applied to the zero-one Linear Programming problem in (3):

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^n x_i - K \cdot \sum_{i=1}^n x_i(1 - x_i) \\ & \text{subject to} && x_i + x_j \leq 1, \forall (v_i, v_j) \in \bar{E} \\ & && x_i \in [0, 1], i = 1, \dots, n. \end{aligned} \quad (5)$$

Recall the goal of this thesis in the Problem Statement, Section 1.2: to show whether or not this Quadratic Programming formulation in (5) is a good alternative for the zero-one Linear Programming formulation in (3). To show this, the global maxima of both mathematical programming problems have to correspond.

2.2.1 Global maxima

To show that the global maxima of both the Quadratic Programming problem and the zero-one Linear Programming problem correspond, it is sufficient to prove that the global maxima of the Quadratic Programming problem are zero-one. When the global maxima of the Quadratic Programming problem are all zero-one, the global maxima have to be a feasible solution for the zero-one Linear Programming problem. Even so, the global maxima of the zero-one Linear Programming problem have to be a feasible solution for the Quadratic Programming problem. Now it is clear to see that the global maxima of both mathematical programming problems correspond.

A value for K in (5) has to be found, in order to show that the global maxima of both mathematical programming problems indeed correspond. For example, $K = 1$ does not penalize the Quadratic Programming problem sufficiently. This is shown by an example:

Let $G = (V, E)$ be a graph with 100 vertices and 20 edges with a maximum clique size of 5.

The *objective function* value of the QP should not be higher than 5. When $K = 1$ the penalty term is not high enough, because the feasible solution $x_i = 0.5 \forall i$ has an *objective function* value of $100 \times 0.5 - 1 \times (100 \times 0.5^2) = 25$.

Therefore, K has to be high enough such that the *objective function* value cannot be higher than the maximum clique size. Although the maximum clique size is not known in advance, when proven that the global maxima of the Quadratic Programming problem correspond with the global maxima of the zero-one Linear Programming problem, it is obvious that for that K the *objective function* value is not higher than the maximum clique size.

The following proof shows that for $K \geq 2$ the global maxima of both mathematical programming problems correspond. The core of the proof is a construction that transforms a feasible fractional solution \mathbf{x} into a feasible zero-one solution. The proof consists of three steps.

Proof. Step 1. All variables $x_i < \frac{1}{2}$ in \mathbf{x} can be set to 0.

The new acquired solution is still feasible, because the constraints concerning these variables still apply: suppose such a variable x_i is in a constraint with another variable x_j , namely $x_i + x_j \leq 1$. The constraint will be $0 + x_j \leq 1$, which will always be satisfied.

The *objective function* value increases, because the old contribution of x_i to the *objective function* value is lower than the new contribution. The contribution $x_i - Kx_i(1-x_i)$ to the *objective function* value for $0 < x_i < \frac{1}{2}$ is lower than for $x_i = 0$. Mathematical proof:

$$\begin{aligned} x_i - Kx_i(1-x_i) &< && 0 \\ x_i &< && Kx_i(1-x_i) \\ 1 &< && K(1-x_i) \\ 1-K &< && -Kx_i \\ K-1 &> && Kx_i \end{aligned}$$

which is true for all $K \geq 2$.

Step 2. All variables $x_i = \frac{1}{2}$ in \mathbf{x} can be set to 0 except for one, this one can be set to 1.

The new acquired solution is still feasible, because the constraints $x_i + x_j \leq 1$, where $x_j \leq \frac{1}{2}$ still apply. All $x_j < \frac{1}{2}$ are set to 0 in Step 1, and all $x_j = \frac{1}{2}$ are set to 0 also, except for one. This one can be set to 1 and the constraints still apply, because $0 + 1 \leq 1$ is evident.

The *objective function* value increases, because the old contribution of x_i to the *objective function* value is lower than the new contribution. The contribution $x_i - Kx_i(1-x_i)$ to the *objective function* value for $x_i = \frac{1}{2}$ is lower than for $x_i = 0$. Mathematical proof:

$$\begin{aligned} \frac{1}{2} - K\frac{1}{2}\left(1 - \frac{1}{2}\right) &< && 0 \\ \frac{1}{2} - \frac{1}{4}K &< && 0 \end{aligned}$$

which is true for all $K \geq 2$.

Step 3. All variables $x_i > \frac{1}{2}$ in \mathbf{x} can be set to 1.

The new acquired solution is still feasible, because the constraints $x_i + x_j \leq 1$, where $x_j > \frac{1}{2}$ still apply. All $x_j < \frac{1}{2}$ are set to 0 in Step 1, thus $1 + 0 \leq 1$, which is evident.

The *objective function* value increases, because the old contribution of x_i to the *objective function* value is lower than the new contribution. The contribution $x_i - Kx_i(1-x_i)$ to the *objective function* value for $x_i > \frac{1}{2}$ is lower than for $x_i = 1$. Mathematical proof:

$$\begin{aligned} x_i - Kx_i(1-x_i) &< && 1 \\ x_i &< && 1 + Kx_i(1-x_i) \\ x_i - 1 &< && Kx_i(1-x_i) \\ \frac{x_i - 1}{1-x_i} &< && Kx_i \end{aligned}$$

which is true for all $K \geq 2$, because $\frac{x_i-1}{1-x_i}$ is negative and $2x_i$ is positive for $x_i > \frac{1}{2}$. \square

The above proves that for $K \geq 2$ the global maxima of both mathematical programming problems correspond. The following proof shows that $K = 2$ is a lower bound for K .

Proof. Suppose $K < 2$. Then all graphs with more than $4/(2-K)$ vertices without edges is an example with a fractional optimum for the Quadratic Programming problem. \square

3 Experiments

Recall that the goal of this thesis is to show whether or not the suggested Quadratic Programming problem is a good alternative for the zero-one Linear Programming problem. Section 2.2.1 theoretically shows that the global maxima of both the Quadratic Programming problem and the zero-one Linear Programming problem correspond. The Quadratic Programming problem solver, explained in more detail in Section 3.3.2, can be solved easier than a zero-one Linear Programming problem, but the solver uses a heuristic. That means there is no guarantee a global maximum is reached. Therefore, an empirical comparison on computation time of both mathematical programming problems have to be conducted.

3.1 Data

Maximum Clique problems in graphs mainly vary in size, but also the densities of graphs plays a roll. As shown in Section 2.1 constraints are added for each two different vertices (v_i, v_j) that are not adjacent. When the density of a graph is low, thus the density of the complement graph is high, there are many constraints. When the density of a graph is high, thus the density of the complement graph is low, there are little constraints.

The data used in this thesis is assembled as follows: for each size of the graphs between 5 and 50 there are four graphs in our data. Each graph of equal size has different density. The densities are 20%, 40%, 60% and 80%. For each size of the graphs between 50 and 100 there are six graphs in our data, again each with different density, namely 20%, 35%, 50%, 65%, 80% and 90%. The choice to have more diversity in density on larger graphs is based on the idea that in larger graphs the steps may be to big with only four different densities.

3.2 Hardware

The experiments are conducted on a personal computer with a AMD64 3500+ processor with 1024MB DDRAM memory.

3.3 Software

Needless to say, a mathematical programming problem will not solve itself, it needs a solver. There are many zero-one Linear Programming problem solvers and Quadratic Programming problem solvers. This thesis uses the `bintprog` and `quadprog` solvers in the optimization toolbox in Matlab [6].

3.3.1 Zero-one Linear Programming problem solver

The solver used on zero-one Linear Programming problems is `bintprog`. `Bintprog` uses a Linear Programming-based branch-and-bound algorithm to solve zero-one Linear Programming problems. The algorithm searches for an optimal solution to the zero-one Linear Programming problem by solving a series of Linear Programming-relaxations problems in which the zero-one requirement on the variables is replaced by the weaker constraint $0 \leq x \leq 1$. The algorithm

- searches for a zero-one feasible solution,
- updates the best zero-one feasible point found so far as the search tree grows,
- verifies that no better zero-one feasible solution is possible by solving series of Linear Programming problems.

3.3.2 Quadratic Programming problem solver

The solver used on Quadratic Programming problems is `quadprog`. The `quadprog` algorithm is a subspace trust-region method based on the interior-reflective Newton method described in [?]. Each iteration involves the approximate solution of a linear system using the method of preconditioned conjugate gradients [?].

`Quadprog` uses a heuristic for solving Quadratic Programming problems. Therefore, it is possible that the `quadprog` algorithm “gets stuck” in a local maximum. Therefore, `quadprog` does not guarantee a global maximum.

3.4 Results

The first objective of this thesis is to show that the global maxima of both the Quadratic Programming problem and the zero-one Linear Programming problem correspond. This is done in Section 2.2.1. The second objective of this thesis is to conduct an empirical comparison on computation time of both these mathematical programming problems.

This thesis has conducted experiments on the data described in Section 3.1 and used value $K = 2$, which is proven in Section 2.2.1 to be the lower bound for K , such that the global maxima of both the Quadratic Programming problem and the zero-one Linear Programming problem correspond. The results are shown in Table 1 and Table 2.

Graph				(0,1)-LP	QP	
Graph	Edges	Density	Max Clique	Time (s)	Time (s)	Solved
5	2	0.20	2	0.797	0.266	X
5	4	0.40	3	0.031	0.031	√
5	6	0.60	4	0.016	0.031	√
5	8	0.80	4	0.016	0.109	√
10	9	0.20	3	0.063	0.031	X
10	18	0.40	3	0.063	61.266	X
10	27	0.60	4	0.016	60.625	X
10	36	0.80	6	0.016	0.047	√
15	21	0.20	4	0.250	68.469	X
15	42	0.40	5	0.094	0.063	X
15	63	0.60	8	0.031	0.031	√
15	84	0.80	10	0.016	0.031	√
20	38	0.20	4	1.391	77.109	X
20	76	0.40	6	0.375	0.063	X
20	114	0.60	9	0.156	0.047	X
20	152	0.80	13	0.016	0.016	√
25	60	0.20	3	7.859	93.406	X
25	120	0.40	8	2.031	85.422	X
25	180	0.60	10	0.344	90.391	X
25	240	0.80	14	0.047	0.063	√
30	87	0.20	7	21.781	102.203	X
30	174	0.40	10	7.891	94.906	X
30	261	0.60	13	2.016	85.953	X
30	348	0.80	19	0.047	0.047	√
35	119	0.20	6	64.625	160.422	X
35	238	0.40	8	31.453	119.109	X
35	357	0.60	15	3.984	100.406	X
35	476	0.80	20	0.063	0.063	√
40	156	0.20	5	226.344	212.031	X
40	312	0.40	10	72.063	172.281	X
40	468	0.60	15	10.219	125.516	X
40	624	0.80	24	0.109	0.125	√
45	198	0.20	7	491.844	284.813	X
45	396	0.40	11	164.563	202.297	X
45	594	0.60	17	35.938	152.172	X
45	792	0.80	27	0.125	0.219	√

Table 1: Experiments 1/2

Graph				(0,1)-LP	QP	
Graph	Edges	Density	Max Clique	Time (s)	Time (s)	Solved
50	245	0.20	6	889.125	357.813	X
50	490	0.40	13	393.734	283.516	X
50	735	0.60	18	88.672	220.797	X
50	980	0.80	28	0.188	0.125	√
60	354	0.20	9	4047.016	536.438	X
60	619	0.35	13	2104.625	489.844	X
60	885	0.50	19	777.438	449.547	X
60	1150	0.65	25	88.672	327.625	X
60	1416	0.80	35	0.328	0.281	√
60	1681	0.95	48	0.047	0.125	√
70	483	0.20	3	7321.188	740.906	X
70	845	0.35	15	6524.391	726.438	X
70	1207	0.50	23	2334.547	549.578	X
70	1569	0.65	29	344.141	532.859	X
70	1932	0.80	37	0.531	0.922	√
70	2294	0.95	59	0.047	0.188	√
80	632	0.20	3	7258.078	994.656	X
80	1106	0.35	-	7266.750	963.734	X
80	1580	0.50	28	5157.469	738.484	X
80	2054	0.65	36	547.578	722.438	X
80	2528	0.80	45	1.016	0.672	√
80	3002	0.95	65	0.078	0.313	√
90	801	0.20	3	7500.781	1310.984	X
90	1401	0.35	10	9002.922	1128.031	X
90	2002	0.50	-	7377.438	999.094	X
90	2603	0.65	35	3470.703	877.438	X
90	3204	0.80	51	1.797	2.063	√
90	3804	0.95	73	0.141	0.391	√
100	990	0.20	-	8188.938	1726.172	X
100	1732	0.35	-	7975.984	1848.109	X
100	2475	0.50	-	7838.000	1281.781	X
100	3217	0.65	40	7202.219	1143.234	X
100	3960	0.80	58	2.219	1.906	√
100	4702	0.95	78	0.219	0.625	√

Table 2: Experiments 2/2

The “-” in both tables means that the solver could not reach the maximum because of too little memory.

The “X” in both tables means that the solver did not come up with the correct answer. An “X” only occurs at `quadprog` and means in all cases that the `quadprog` algorithm lead to the fractional solution $\mathbf{x} = (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$. It could have been any other feasible solution, but it seems that this fractional solution is an attracting local maximum. More about local maxima can be read in Section 3.4.2.

3.4.1 Conclusions

The results in Table 1 and Table 2 show that when the density of a graph is low, the computation time is higher than when the density of a graph is high. This can be explained by the number of constraints in the programming problem, which depend on the density of the graph, see Section 3.1. When there are few constraints the problem can be solved in less time than when there are a lot of constraints.

The outcome of the experiments were not astonishing. `Quadprog` did not do a better job than `bintprog`, because `quadprog` did not solve graphs with low density correctly, whereas `bintprog` did. Even on graphs with high density, where `quadprog` did solve the problem, `bintprog` solved the same problem just as fast. The latter is a motive to test whether or not the graphs with high density are easy to solve, that is, they have an easy constructed maximal clique as maximum clique. More about these experiments can be read in Section 3.4.3.

3.4.2 Local maxima

The QP solver often “gets stuck” at a local maximum. In all experiments where the QP did not solve the Maximum Clique problem, the QP solver halted at the same local maximum: $x = (x_1, x_2, \dots, x_n) = (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$.

This feasible solution x is the local maximum the QP solver “gets stuck” in, when in this group of non integer variables a subgroup D can be found with the following properties:

1. the subgroup D contains an odd number of variables (more than one),
2. there exists a cycle along the variables of D , not in the graph itself, but in the complement of the graph.

Proof. Number the variables in D as x_1, x_2, \dots, x_{n+1} in a sequence in which the cycle can be walked. In the QP formulation the following scheme of constraints is present:

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_2 + x_3 &\leq 1 \\ &\vdots \\ x_{2n} + x_{2n+1} &\leq 1 \\ x_1 + x_{2n+1} &\leq 1 \end{aligned}$$

When one wants to “get out” of this non integer solution, some variables have to be raised and others have to be reduced. When a variable out D is incremented with ϵ , his two neighbors in the cycle have to decrement with at least ϵ to ensure that the solution stays feasible. Because the cykel contains an odd number of variables, one can conclude that more variables have to decrement then can increment with ϵ . But with such an adjustment of the variables (with small values of ϵ), the *objective function* value decreases. Apparently, this non integer solution is a local maximum. The value of K does not matter, because the value of ϵ is almost zero. \square

From this proof can be inferred that for graphs with low densities the chance of finding such a D as described above is bigger than in graphs with high densities. This explains why the QP solver often does not find a global maximum for graphs with low densities and does find a global maximum for graphs with high densities.

Avoid local maxima

There are methods that can guide the QP solver to avoid local maxima. For example providing a different initial solution could be used, but unfortunately there was not enough time to extensively test this on the data.

Trying other QP solvers did not work either, because all other QP solvers tried, could only deal with only non-negative hessian, and the hessian in the maximum clique problem is not non-negative.

3.4.3 Maximal clique

Table 1 and Table 2 show that graphs with high densities can be solved correctly by `quadprog`. In less than one second, an optimal solution is reached by `quadprog` as well as by `bintprog`. A possible explanation for this is that the optimal solution is easily constructed as a maximal clique. The maximal clique used for comparison with the optimal solution of `quadprog` is constructed as follows:

1. Choose the vertex with the highest number edges and make this vertex part of the maximal clique.
2. Then choose another vertex with second highest number of edges which is connected to all vertices in the maximal clique and make this vertex part of the maximal clique.

3. Continue step 2 until no more vertices can be added to the maximal clique.

Experiments show that for graphs up to a size of 50, the `quadprog` solutions are the same as the maximal cliques, but on bigger graphs this is not the case. The coincidence

4 Conclusions

The Maximum Clique problem can be formulated as a zero-one Linear Programming problem. This thesis showed that this zero-one Linear Programming problem could be formulated as a Quadratic Programming problem. This thesis shows that the global maxima of both mathematical programming problems correspond. However, because the Quadratic Programming problem solver, `quadprog`, uses a heuristic, there is no guarantee that the solution is optimal, because the method can end in a local maximum. When solving Maximum Clique problems in low density graphs, `quadprog` gives the fractional solution $\mathbf{x} = (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}, \frac{1}{2})$. There are methods which can be used to avoid this particular local maximum, but none were conducted in this thesis.

In the case that `quadprog` did solve the Maximum Clique problem correctly, the zero-one Linear Programming problem solver `bintprog` solved the same problem just as fast. Apparently is a maximum clique in high density graphs easy to find, because in most cases it corresponds with an easy to construct maximal clique.

Altogether the suggested Quadratic Programming formulation is not a good alternative for the zero-one Linear Programming formulation, because the expected computation time gain is absent and the degree of correctly solved problems is low. Perhaps, if more research is conducted in avoiding local maxima, `quadprog` would perform better on the Maximum Clique problem.

References

- [1] ? (2005). ? ? ?
- [2] Garey, M. R. and Johnson, David S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman.
- [3] Hromkovic, Juraj (1999). Stability of approximation algorithms for hard optimization problems. *SOFSEM '99: Proceedings of the 26th Conference on Current Trends in Theory and Practice of Informatics on Theory and Practice of Informatics*, pp. 29–47, Springer-Verlag, London, UK.
- [4] Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations* (eds. R. E. Miller and J. W. Thatcher), pp. 85–103.
- [5] Lovasz, L. and Plummer, M. D. (1986). *Matching Theory*. North-Holland Mathematics Studies : Annals of Discrete Mathematics. North-Holland. LOV 1 86:1, pp. 456-466.
- [6]
- [7] Pardalos, Panos M. and Xue, Jue (Apr 1994). The maximum clique problem. *Journal of Global Optimization (Historical Archive)*, Vol. 4, pp. 301–321.