

Semidefinite programming relaxations through quadratic reformulation for box-constrained polynomial optimization problems

Sourour Elloumi, Amélie Lambert, Arnaud Lazare

▶ **To cite this version:**

Sourour Elloumi, Amélie Lambert, Arnaud Lazare. Semidefinite programming relaxations through quadratic reformulation for box-constrained polynomial optimization problems. 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Apr 2019, Paris, France. pp.1498-1503, 10.1109/CoDIT.2019.8820690 . hal-02455410

HAL Id: hal-02455410
https://hal.science/hal-02455410

Submitted on 26 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Semidefinite programming relaxations through quadratic reformulation for box-constrained polynomial optimization problems

Sourour Elloumi[1], Amélie Lambert[2] and Arnaud Lazare[1]*†

Abstract

In this paper we introduce new semidefinite programming relaxations to box-constrained polynomial optimization programs (P). For this, we first reformulate (P) into a quadratic program. More precisely, we recursively reduce the degree of (P) to two by substituting the product of two variables by a new one. We obtain a quadratically constrained quadratic program. We build a first immediate SDP relaxation in the dimension of the total number of variables. We then strengthen the SDP relaxation by use of valid constraints that follow from the quadratization. We finally show the tightness of our relaxations through several experiments on box polynomial instances.

1. Introduction

In this paper, we consider the box-constrained polynomial optimization problem that can be stated as follows:

$$(P)\begin{cases} \min f(x) = \sum_{p=1}^{m} c_p \prod_{i \in \mathscr{M}_p} x_i \\ \text{s.t.} \\ x_i \in [0,1],\ i \in I \end{cases}$$

where $I = \{1,..,n\}$, f is an $n-$variable polynomial of degree d and m is the number of monomials. For a monomial p, $\mathscr{M}_p \subset I$ is a multiset containing the indexes of the variables involved in p. It follows that $d = \max_p |\mathscr{M}_p|$. Here, we suppose that each variable x_i is in the interval $[0,1]$, but variables in any $[\ell_i, u_i]$ interval can be transformed into $[0,1]$ by a simple variable change.

(P) is a general model that allows to formulate many important problems in optimization. It is an $\mathscr{NP}$-hard problem [11] in general and very difficult to solve in practice. Classical methods that are able to solve (P) to optimality are branch-and-bound algorithms based on convex relaxations. The most classical relaxation consists in the complete linearization of (P), but quadratic convex relaxation can also be used. For instance, the well known $\alpha-$branch-and-bound [2] computes convex underestimators of nonlinear functions by perturbing the diagonal of the Hessian matrix of the objective function. Several implementations of these algorithms are available, see for instance `Baron` [17], `Antigone` [16], `SCIP` [1] or `Couenne` [5]. In [8, 15], the authors use separable or convex underestimator to approximate a given polynomial. Another approach proposed in [4] handles binary unconstrained polynomial problems. It consists in building a quadratic equivalent formulation to (P) where the objective function is still non-convex. The obtained reformulation is then solved by the solver `Cplex` [13], which is possible since they consider problems with only binary variables. The idea of quadratization is also used for continuous variables in [9] in order to deduce linear programming relaxations. Several other approaches are based on semidefinite programming. Among these, we cite the exact method in [14] that can find an optimal solution thanks to a hierarchy of SDP relaxations and refer the reader to seminal surveys such as [3].

In this paper, we propose two families of SDP relaxation of (P). To this end, we first reformulate (P) into an equivalent quadratic problem (QP_ε). As in [4], we call it a *quadratization*. For an instance of (P), several quadratizations are possible and we formalize valid quadratizations in our context. We further build a compact semidefinite relaxation of (QP_ε) that follows immediately from the quadratization used. Then, we strengthen this compact relaxation. For this, we in-

*[1] UMA-ENSTA, 828 Boulevard des Maréchaux, 91120 Palaiseau, France {elloumi,lazare}@ensta-paristech.fr

†[2] CEDRIC-Cnam, 292 rue saint Martin, F-75141 Paris Cedex 03, France amelie.lambert@cnam.fr

troduce new valid constraints, which come from the quadratization step. We add these constraints to the compact SDP relaxation. Obviously, the bound associated to this last relaxation is stronger, but unfortunately harder to solve. To solve it efficiently, we develop a subgradient algorithm within a Lagrangian duality framework following the ideas of [6].

The outline of our paper is the following. In Section 2, we focus on the quadratization step. In Section 3 and 4, we present our compact semidefinite relaxation and its strengthened version. In Section 5, we present experiments where we compare the quality of the lower bound obtained by the branch-and-bound of the solver `SCIP` [1] after one hour of CPU time with our two relaxations. We make our tests on polynomial instances inspired from [8]. Section 6 draws some conclusions and axes for future research.

2. Quadratic reformulation of (P)

In this section, we present how we build equivalent quadratic formulations to (P). The basic idea is to reduce the degree of f to 2. For this, in each monomial of degree 3 or greater, we simply recursively replace each product of two variables by an additional variable.

More formally, we define the set of indices of the additional variables $J = \{n+1,..,N\}$, where N is the total number of initial and additional variables. We also define the subsets ε_i for the initial or additional variable i as follows:

Definition 1. *For all $i \in I \cup J$, we define ε_i as the set of indices of the variables whose product is equal to x_i:*

- *If $i \in I$, i.e. x_i is an initial variable, then we set $\varepsilon_i = \{i\}$*
- *If $i \in J$, i.e. x_i is an additional variable, then there exist $(j,k) \in (I \cup J)^2$ such that x_i replaces $x_j x_k$ and we set $\varepsilon_i = \varepsilon_j \cup \varepsilon_k$*

□

Using these sets, we define a *valid* quadratization as a reformulation with N variables where any monomial of degree at least 3 is replaced by the product of two variables.

Definition 2. *The sets $J = \{n+1,..,N\}$ and $\{\varepsilon_i,\ i \in I \cup J\}$ define a valid quadratization with N variables if, for any monomial p of degree greater than or equal to 3 (i.e. $|\mathscr{M}_p| \geq 3$), there exist $(j,k) \in (I \cup J)^2$ such that $\mathscr{M}_p = \varepsilon_j \cup \varepsilon_k$ and $\prod_{i \in \mathscr{M}_p} x_i = x_j x_k$. Then the monomial p is replaced by a quadratic term.* □

With this definition of a quadratization, we reformulate (P) as a non-convex quadratically constrained quadratic program (QP_ε) with N variables:

$$(QP_\varepsilon) \begin{cases} \min g(x) = \sum\limits_{\substack{|\mathscr{M}_p| \geq 3 \\ \mathscr{M}_p = \varepsilon_j \cup \varepsilon_k}} c_p x_j x_k + \sum\limits_{|\mathscr{M}_p| \leq 2} c_p \prod\limits_{i \in \mathscr{M}_p} x_i & \\ \text{s.t.} & \\ x_i = x_j x_k,\ (i,j,k) \in J \times (I \cup J)^2 :\ \varepsilon_i = \varepsilon_j \cup \varepsilon_k & (1) \\ x_i \in [0,1],\ i \in I \cup J & (2) \end{cases}$$

By construction, problems (P) and (QP_ε) are equivalent in the sense that, from any solution of one problem, one can deduce a solution for the other problem with the same objective function value. Let us observe that (QP_ε) is parameterized by the quadratization defined by sets ε. Indeed, several valid quadratizations can be applied to (P), each of them leading to different sets ε_i.

For the sake of simplicity, $g(x)$ can be rewritten as $g(x) = \langle Q, xx^T \rangle + c^T x$, where $Q \in S_N$ (the set of $N \times N$ real symmetric matrices), and $c \in \mathbb{R}^N$.

3. A compact semidefinite programming relaxation

In this section, we build a semidefinite relaxation of (QP_ε). Classically, we linearize the products xx^T using a matrix variable $X \in S_N$ and we relax the equality $X = xx^T$ as $X - xx^T \succeq 0$. We obtain the following semidefinite program:

$$(SDP^0_\varepsilon) \begin{cases} \min < Q, X > + c^T x & \\ \text{s.t.} & \\ x_i = X_{jk},\ (i,j,k) \in J \times (I \cup J)^2 :\ \varepsilon_i = \varepsilon_j \cup \varepsilon_k & (3) \\ X_{ii} \leq x_i,\ i \in I \cup J & (4) \\ \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 & (5) \\ x \in \mathbb{R}^N,\ X \in S^N & (6) \end{cases}$$

where Constraints (3) and (4) correspond to the linearization of Constraints (1) and (2), respectively. By Schur's Lemma, Constraint (5) is equivalent to the relaxed constraint $X - xx^T \succeq 0$. It is important to note that the number $N - n$ of Constraints (3) depends on the quadratization ε used.

(SDP^0_ε) has a reasonable size of $\mathscr{O}(N^2)$ variables and $\mathscr{O}(N)$ constraints. In section 5, we evaluate the quality of the associated bound.

4. An improved semidefinite programming relaxation

In this section, we build an improved semidefinite relaxation of (QP_ε) by strengthening (SDP^0_ε). We start by introducing valid quadratic equalities and inequalities coming from the quadratization ε.

More formally, for a quadratization characterized by ε, we introduce the following set of constraints that is valid when Constraints (1)-(2) are satisfied.

Lemma 1. *The following quadratic equalities and inequalities are valid over $\mathscr{F}_\varepsilon$:*

$$\begin{cases} x_i x_j \leq x_i,\ (i,j) \in J \times (I \cup J) :\ \varepsilon_j \subset \varepsilon_i & (7) \\ x_i x_j = x_k x_l,\ (i,j,k,l) \in (I \cup J)^4 :\ \varepsilon_i \sqcup \varepsilon_j = \varepsilon_k \sqcup \varepsilon_l & (8) \end{cases}$$

where $\sqcup$ is the disjoint union operator.

Proof. Constraints (7) trivially hold since $x_i \in [0,1]$. We then prove the validity of the Constraints (8). By definition we have:

$$\begin{aligned} x_i x_j &= \prod_{i' \in \varepsilon_i} x_{i'} \prod_{j' \in \varepsilon_j} x_{j'} \\ &= \prod_{i' \in \varepsilon_i \sqcup \varepsilon_j} x_{i'} \\ &= \prod_{k' \in \varepsilon_k \sqcup \varepsilon_l} x_{k'} \text{ since } \varepsilon_i \sqcup \varepsilon_j = \varepsilon_k \sqcup \varepsilon_l \\ &= x_k x_l \end{aligned}$$

□

In a sense, Constraints (8) can be viewed as constraints that break symmetries. Constraints (7) and (8) are not convex, but since they are quadratic, we can easily linearize them using the matrix variable X.

Adding these constraints to (SDP^0_ε), we obtain the following semidefinite relaxation:

$$(SDP^1_\varepsilon) \begin{cases} \min < Q, X > + c^T x \\ \text{s.t.} \\ \quad (3)(4)(5)(6) \\ \quad X_{ij} \leq x_i,\ (i,j) \in J \times (I \cup J) :\ \varepsilon_j \subset \varepsilon_i & (9) \\ \quad X_{ij} = X_{kl},\ (i,j,k,l) \in (I \cup J)^4 :\ \varepsilon_i \sqcup \varepsilon_j = \varepsilon_k \sqcup \varepsilon_l & (10) \end{cases}$$

(SDP^1_ε) is larger than (SDP^0_ε). Indeed, it still has $\mathscr{O}(N^2)$ variables, but there are $\mathscr{O}(N^4)$ constraints. Due to its significant number of constraints (SDP^1_ε) is harder to solve than (SDP^0_ε). In Section 5, we evaluate the quality of the associated bound.

Here again, the number $N(N-n)$ of Constraints (9) depends on the quadratization ε used. As for the number of Constraints (10), it depends also on the structure of the instance and can be up to N^4.

We end this section with an illustration of our two relaxations on a small instance.

Example 1

Let us consider the following polynomial optimization problem with $n = 4$ variables and $m = 7$ monomials.

$$(Ex) \begin{cases} \min f(x) = -0.21x_1 - 0.08x_2 - 0.58x_3 - 0.49x_1x_4 \\ +0.78x_1x_3x_4 - 0.54x_1x_2x_4 + 0.88x_2x_3x_4 \\ s.t. \\ \quad x \in [0,1]^4 \end{cases}$$

The optimal value of (Ex) is $-1,32$.

Applying the quadratization described in [10], we introduce 3 additionnal variables and 7 constraints. We obtain the following equivalent quadratic problem (Ex_ε) to (Ex) with 7 variables.

$$(Ex_\varepsilon) \begin{cases} \min g(x) = -0.21x_1 - 0.08x_2 - 0.58x_3 - 0.49x_1x_4 \\ +0.78x_5x_4 - 0.54x_6x_4 + 0.88x_7x_4 \\ s.t. \\ \quad x_1x_3 = x_5 \\ \quad x_1x_2 = x_6 \\ \quad x_2x_3 = x_7 \\ \quad x \in [0,1]^7 \end{cases}$$

We now build the semidefinite relaxation (Ex^0_ε) which contains 35 variables and 11 constraints. The optimal value of (Ex^0_ε) is $-1,37$.

$$(Ex^0_\varepsilon) \begin{cases} \min g(x,X) = -0.21x_1 - 0.08x_2 - 0.58x_3 - 0.49X_{1,4} \\ +0.78X_{5,4} - 0.54X_{6,4} + 0.88X_{7,4} \\ s.t. \\ \quad X_{ii} \leq x_i, \quad 1 \leq i \leq 7 \\ \quad X_{1,3} = x_5 \\ \quad X_{1,2} = x_6 \\ \quad X_{2,3} = x_7 \\ \quad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \\ \quad x \in \mathbb{R}^7,\ X \in S^7 \end{cases}$$

We strengthen this first relaxation by adding the two families of valid constraints (9) and (10) that we expand in the following. The resulting semidefinite relaxation (Ex^1_ε) has 35 variables and 23 constraints. The optimal value of (Ex^1_ε) is $-1,32$ which is also the optimal value of (Ex).

$$(Ex^1_\varepsilon)\begin{cases} \min g(x,X) \\ s.t. \\ \quad X_{ii} \le x_i, \quad 1 \le i \le 7 \\ \quad X_{1,3} = x_5 \\ \quad X_{1,2} = x_6 \\ \quad X_{2,3} = x_7 \\ \quad X_{1,5} \le x_5 \quad X_{3,5} \le x_5 \\ \quad X_{1,6} \le x_6 \quad X_{2,6} \le x_6 \\ \quad X_{2,7} \le x_7 \quad X_{3,7} \le x_7 \\ \quad X_{2,5} = X_{6,3} \quad X_{3,6} = X_{7,1} \\ \quad X_{2,5} = X_{7,1} \quad X_{6,5} = X_{6,3} \\ \quad X_{7,6} = X_{7,1} \quad X_{7,5} = X_{7,1} \\ \quad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \\ \quad x \in \mathbb{R}^7,\ X \in S^7 \end{cases}$$

Although the second relaxation gives the optimal value of the considered problem, the difference in the optimal value between the two relaxations is not much significant for this instance. It is also the case for most of the small-sized instances. But, as shown in the next section, the gap will markedly increase when considering medium and large-sized instances. □

5. Experimental results

In this section, we evaluate the bound obtained by our relaxations (SDP^0_ε) and (SDP^1_ε) on randomly generated instances of degree 4. The number of variables varies between 10 and 40 and the number of monomials from 10 to 400. These instances are quite sparse as their density, i.e. the ratio $\frac{m}{n}$, is between 1 and 10. The generation process is similar to [8], that is:

- the number of initial variables n is contained in $\{10,15,20,25,30,35,40\}$.
- the number of monomials m is a multiple of n.
- the coefficient of each monomial is uniformly generated in the interval $[-1,1]$.
- the variables composing each monomial are generated by randomly choosing an index within $\{0,1,...,n\}$. Each time the value 0 is generated, the degree of the monomial decreases by one. An index can only be chosen once in a monomial.

Concerning the choice of the quadratization for our experimental results, we used the one described in Algorithm 2 from [10].

Our experiments were carried out on a server with 2 CPU Intel Xeon each of them having 12 cores and 2 threads of 2.5 GHz and $4*16$ GB of RAM using a Linux operating system.

Used solvers

Relaxations (SDP^0_ε) and (SDP^1_ε) are hard to solve, and standard semidefinite programming solvers that implement interior point algorithms [7, 18] failed to solve it. Thus, following the ideas of [6] we develop a subgradient algorithm within a Lagrangian duality framework to solve it. For this, we use the solver `csdp` [7] together with the `Conic Bundle algorithm` [12].

We also use `SCIP 5.0.0` [1] with the default parameters for our comparisons.

Legend of Table 1

- n: number of variables in the polynomial formulation.
- N: number of variables after the quadratization.
- m: number of monomials.
- *BKN*: is the best known solution value at the end of the branch-and-cut of `SCIP` (1 hour).
- LB_{end}: is the lower bound obtained by the branch-and-cut of `SCIP`, after 1 hour.
- Gap_{end}: is the final optimality gap at the end of the branch-and-cut of `SCIP`, i.e. $Gap_{end} = \left|\frac{BKN - LB_{end}}{BKN}\right| * 100.$
- *LB*: is the lower bound obtained within 20 minutes of CPU time when solving (SDP^0_ε) and (SDP^1_ε).
- *Gap*: is the relative gap between *BKN* and *LB* for (SDP^0_ε) and (SDP^1_ε), i.e. $Gap = \left|\frac{BKN - LB}{BKN}\right| * 100.$

We present in Table 1 a comparison of the lower bounds obtained by (SDP^0_ε) and (SDP^1_ε) after 20 minutes of CPU time, with the final lower bound obtained by the solver `SCIP` after one hour of CPU time. As expected, we observe that the gap obtained with the relaxation (SDP^1_ε) is significantly smaller than the gap obtained with (SDP^0_ε). Indeed, out of the 70 considered instances, (SDP^0_ε) has an average gap of 64% whereas (SDP^1_ε) has an average gap of 33%. This shows the importance of the valid constraints added to (SDP^1_ε) that exploit both the quadratization used, and the structure of the instances. Comparing the gaps obtained by (SDP^1_ε) to those obtained by the solver `SCIP`, we observe that `SCIP` is not able to provide better lower bounds than our relaxation on 21 instances even after one hour of branch-and-bound. The average gap for `SCIP` is about 27% over all the instances. In fact, `SCIP` is able to close the gap within one hour of CPU time for the smaller instances, but its gap significantly increases on medium

sized instances and dense instances. On the contrary, the gap obtained with (SDP_ε^1) is more stable to the increase of n.

6. Conclusion and future work

We introduced two SDP relaxations based on a quadratization and we prove their practical interest. Future work consists in embedding these relaxations within a branch-and-bound or a branch-and-cut framework in order to compute an optimal solution. Another interesting aspect is to study the influence of the quadratization on the quality and the computational efficiency of the SDP relaxation. This is the topic of an ongoing work.

References

[1] T. Achterberg. Scip : solving constraint integer programs. *Mathematical Programming Computation*, (1):1–41, 2009.

[2] C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, αbb, for general twice-differentiable constrained nlpsi. theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.

[3] M.F. Anjos and J.B. Lasserre. Handbook of semidefinite, conic and polynomial optimization: Theory, algorithms, software and applications. *International Series in Operational Research and Management Science*, 166, 2012.

[4] M. Anthony, E. Boros, Y. Crama, and A. Gruber. Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming*, 162:115–144, 2017.

[5] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Br anching and bounds tightening techniques for non-convex minlp. *Optimization Methods and Software*, 4–5(24):597–634, 2009.

[6] A. Billionnet, S. Elloumi, A. Lambert, and A. Wiegele. Using a Conic Bundle method to accelerate both phases of a Quadratic Convex Reformulation. *INFORMS Journal on Computing*, 29(2):318–331, 2017.

[7] B. Borchers. CSDP, A C Library for Semidefinite Programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[8] C. Buchheim and C. D'Ambrosio. Monomial-wise optimal separable underestimators for mixed-integer polynomial optimization. *Journal of Global Optimization*, pages 1–28, 2016.

[9] Evrim Dalkiran and Laleh Ghalami. On linear programming relaxations for solving polynomial programming problems. *Computers & Operations Research*, 99:67 – 77, 2018.

[10] Sourour Elloumi, Amélie Lambert, and Arnaud Lazare. Solving unconstrained 0-1 polynomial programs through quadratic convex reformulation. "https://hal.archives-ouvertes.fr/hal-01872996", September 2018.

[11] M.R. Garey and D.S. Johnson. Computers and Intractability: A guide to the theory of NP-Completness. *W.H. Freeman, San Francisco, CA*, 1979.

[12] C. Helmberg. *Conic Bundle v0.3.10*, 2011.

[13] IBM-ILOG. IBM ILOG CPLEX 12.7 Reference Manual. "`http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html`", 2017.

[14] J.B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[15] J.B. Lasserre and T.P. Thanh. Convex underestimators of polynomials. *Journal of Global Optimization*, pages 1–25, 2013.

[16] R. Misener and C.A. Floudas. Antigone: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

[17] N.V. Sahinidis and M. Tawarmalani. Baron 9.0.4: Global optimization of mixed-integer nonlinear programs. *User's Manual*, 2010.

[18] J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *OPTMS*, 11-12:625–653, 1999.

Instance				SCIP		SDP^0_ε		SDP^1_ε	
n	N	m	BKN	LB_{end}	Gap_{end}	LB	Gap	LB	Gap
10	17	10	-2,07	-2,07	0	-2,14	3,34	-2,13	3,19
10	26	20	-2,28	-2,28	0	-2,79	22,01	-2,68	17,20
10	25	30	-3,38	-3,38	0	-5,37	58,94	-4,74	40,38
10	34	40	-5,25	-5,25	0	-6,49	23,63	-5,76	9,70
10	36	50	-5,43	-5,43	0	-7,16	31,76	-6,31	16,03
10	38	60	-3,54	-3,54	0	-6,34	78,84	-4,65	31,29
10	46	70	-3,75	-3,75	0	-7,38	97,08	-4,88	30,20
10	39	80	-5,51	-5,51	0	-7,99	45,09	-5,97	8,46
10	44	90	-4,39	-4,39	0	-8,26	88,25	-5,33	21,53
10	50	100	-6,36	-6,36	0	-9,84	54,85	-6,73	5,82
15	29	15	-1,54	-1,54	0	-2,06	33,84	-1,98	28,90
15	33	30	-3,57	-3,57	0	-4,40	23,28	-3,87	8,43
15	59	45	-4,89	-4,89	0	-6,18	26,43	-5,58	14,04
15	64	60	-5,43	-5,43	0	-8,36	53,94	-7,09	30,64
15	75	75	-12,20	-12,20	0	-14,43	18,29	-12,80	4,88
15	77	90	-9,33	-9,33	0	-12,67	35,82	-10,44	11,93
15	85	105	-9,34	-9,34	0	-14,58	56,07	-11,17	19,53
15	85	120	-10,65	-10,65	0	-16,47	54,61	-12,05	13,13
15	87	135	-7,85	-7,85	0	-14,98	90,92	-10,56	34,57
15	98	150	-7,53	-9,31	23,65	-18,03	139,50	-11,13	47,78
20	27	20	-3,92	-3,92	0	-3,95	0,86	-3,98	1,46
20	73	40	-5,18	-5,18	0	-7,64	47,42	-6,80	31,23
20	95	60	-7,82	-7,82	0	-12,26	56,71	-10,57	35,15
20	101	80	-8,48	-8,48	0	-13,54	59,73	-10,73	26,63
20	116	100	-12,77	-12,77	0	-16,85	32,01	-14,47	13,35
20	115	120	-9,41	-10,64	13,11	-17,33	84,14	-13,43	42,76
20	138	140	-11,57	-12,54	8,44	-20,63	78,34	-15,27	32,04
20	139	160	-10,38	-13,28	27,97	-20,39	96,42	-13,91	34,04
20	143	180	-14,22	-19,82	39,35	-26,66	87,47	-19,09	34,27
20	157	200	-12,53	-16,37	30,64	-24,24	93,47	-16,44	31,19
25	51	25	-4,37	-4,37	0	-5,11	17,00	-4,99	14,35
25	79	50	-8,81	-8,81	0	-10,50	19,14	-9,77	10,92
25	118	75	-8,73	-8,73	0	-13,18	51,03	-11,50	31,82
25	147	100	-12,02	-12,02	0	-17,22	43,20	-14,15	17,73
25	165	125	-15,28	-17,92	17,28	-24,17	58,18	-19,62	28,42
25	165	150	-12,20	-14,16	16,04	-20,86	70,99	-16,01	31,22
25	197	175	-16,24	-22,98	41,50	-28,78	77,22	-22,24	36,96
25	188	200	-18,28	-28,22	54,36	-32,02	75,18	-24,62	34,67
25	217	225	-19,03	-28,81	51,38	-34,06	78,95	-25,45	33,72
25	213	250	-12,54	-25,50	103,33	-30,82	145,76	-21,13	68,49
30	43	30	-6,65	-6,65	0	-6,82	2,53	-6,73	1,09
30	103	60	-9,38	-9,38	0	-13,33	42,09	-12,19	29,93
30	150	90	-12,27	-12,27	0	-17,88	45,75	-16,09	31,14
30	181	120	-9,25	-13,04	40,96	-19,46	110,41	-15,86	71,47
30	216	150	-12,98	-16,35	25,99	-22,58	73,96	-18,35	41,40
30	237	180	-12,74	-22,31	75,11	-26,14	105,14	-20,69	62,37
30	260	210	-20,39	-31,54	54,68	-36,75	80,22	-28,47	39,62
30	266	240	-20,32	-35,89	76,60	-40,24	98,01	-30,04	47,81
30	274	270	-17,44	-35,58	104,04	-39,40	125,92	-28,23	61,87
30	292	300	-15,38	-38,68	151,52	-42,09	173,69	-28,30	84,00
35	59	35	-6,12	-6,12	0	-6,51	6,28	-6,49	6,05
35	130	70	-10,29	-10,29	0	-13,93	35,38	-13,33	29,52
35	182	105	-15,97	-15,97	0	-21,13	32,28	-19,28	20,73
35	220	140	-15,36	-20,68	34,64	-25,41	65,42	-22,21	44,62
35	252	175	-15,76	-22,67	43,83	-28,59	81,39	-23,83	51,21
35	275	210	-20,81	-27,07	30,09	-31,29	50,35	-26,71	28,38
35	304	245	-22,83	-36,02	57,80	-39,64	73,63	-32,09	40,55
35	344	280	-29,01	-47,36	63,26	-48,13	65,90	-39,39	35,77
35	347	315	-27,09	-46,81	72,80	-48,66	79,61	-38,42	41,84
35	369	350	-34,13	-55,09	61,41	-57,00	67,01	-45,27	32,63
40	85	40	-4,22	-4,22	0	-5,00	18,32	-4,86	15,04
40	139	80	-9,50	-9,50	0	-11,98	26,05	-11,61	22,15
40	215	120	-11,19	-13,28	18,70	-18,23	62,93	-16,38	46,34
40	272	160	-19,26	-22,46	16,59	-27,07	40,54	-23,98	24,50
40	300	200	-23,04	-34,49	49,69	-38,22	65,89	-33,09	43,61
40	327	240	-22,25	-37,66	69,25	-40,37	81,42	-34,44	54,80
40	375	280	-22,29	-38,30	71,81	-41,20	84,85	-33,80	51,62
40	397	320	-26,01	-51,03	96,21	-51,72	98,85	-44,38	70,63
40	436	360	-25,29	-54,27	114,60	-53,90	113,14	-47,40	87,43
40	454	400	-24,68	-62,84	154,61	-61,88	150,72	-53,97	118,68

Table 1: *Comparison of the bounds of SDP^0_ε and SDP^1_ε with upper and lower bounds obtained by the branch-and-cut of SCIP in one hour of CPU time. Each line represents one instance.*