Solving unconstrained 0-1 polynomial programs through quadratic convex reformulation

Sourour Elloumi, Amélie Lambert, Arnaud Lazare

Solving unconstrained 0-1 polynomial programs through quadratic convex reformulation

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

1 UMA-ENSTA, 828 Boulevard des Maréchaux, 91120 Palaiseau, France
{sourour.elloumi,arnaud.lazare}@ensta-paristech.fr
2 CEDRIC-Cnam, 292 rue saint Martin, F-75141 Paris Cedex 03, France
amelie.lambert@cnam.fr

Abstract We propose an exact solution approach for the problem (P) of minimizing an unconstrained binary polynomial optimization problem. We call `PQCR` (`Polynomial Quadratic Convex Reformulation`) this three-phase method. The first phase consists in reformulating (P) into a quadratic program (QP). To that end, we recursively reduce the degree of (P) to two, by use of the standard substitution of the product of two variables by a new one. We then obtain a linearly constrained binary quadratic program. In the second phase, we rewrite the objective function of (QP) into an equivalent and parameterized quadratic function using the identity $x_i^2 = x_i$ and other valid quadratic equalities that we introduce from the reformulation of phase 1. Then, we focus on finding the best parameters to get a quadratic convex program which continuous relaxation's optimal value is maximized. For this, we build a new semi-definite relaxation (SDP) of (QP). Then, we prove that the standard linearization inequalities, used for the quadratization step, are redundant in presence of the new quadratic equalities. Next, we deduce our optimal parameters from the dual optimal solution of (SDP). The third phase consists in solving (QP^*), the optimally reformulated problem, with a standard solver. In particular, at each node of the branch-and-bound, the solver computes the optimal value of a continuous quadratic convex program. We present computational results where we compare `PQCR` with other convexification methods, and with the solver `Baron` [42]. We evaluate our method on instances of the image restoration problem [17] and the low auto-correlation binary sequence problem [7] from `minlplib` [35]. For this last problem, 33 instances among the 45 were unsolved in `minlplib`. We solve to optimality 6 of them, and for the 27 others we improve primal and/or dual bounds.

Key words: Unconstrained binary polynomial programming, Global optimization, Semi-definite programming, Quadratic convex reformulation, Experiments

1 Introduction

In this paper, we are interested in solving the unconstrained binary polynomial optimization problem that can be stated as follows:

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

where $I = \{1, .., n\}$, $f(x)$ is an $n-$variable polynomial of degree d and m is the number of monomials. For a monomial p, $\mathcal{M}_p$ is the subset of I containing the indexes of the variables involved in p. It follows that $d = \max_p |\mathcal{M}_p|$.

Unconstrained binary polynomial optimization is a general model that allows to formulate many important problems in optimization. The special case where the polynomial objective function of (P) is a quadratic function (i.e. $d = 2$) has been widely studied. In this case, (P) has many applications, including those from financial analysis [31], cluster analysis [39], computer aided design [27] or machine scheduling [40]. Moreover, many graph combinatorial optimization problems such as determining maximum cliques, maximum cuts, maximum vertex packing or maximum independent sets can be formulated as quadratic optimization problems [5,13,37]. In the cubic case (i.e. $d = 3$), the important class of satisfiability problems known as 3-SAT, can be formulated as (P) [26]. In the case where $d \geq 3$, there also exists many applications including, for example: the construction of binary sequences with low aperiodic correlation [7] that is one of the most challenging problems in signal design theory, or the image restoration problem in computer vision [17].

Problem (P) is $\mathcal{NP}$-hard [20]. Practical difficulties come from the non-convexity of f and the integrality of its variables. During the last decade, several algorithms that can handle (P) were introduced. In particular, methods were designed to solve the more general class of mixed-integer nonlinear programs. These methods are branch-and-bound algorithms based on a convex relaxation of (P). More precisely, in a first step a convex relaxation is designed and then a branch-and-bound is performed based on this relaxation. The most classical relaxation consists in the complete linearization of (P), but quadratic convex relaxations can also be used. For instance, the well known $\alpha-$branch-and-bound [2] computes convex under-estimators 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` [42], `Antigone` [36], `SCIP` [1] or `Couenne` [6].

In the case where the objective function is a polynomial, but the variables are continuous, Lasserre proposes in [29] an algorithm based on a hierarchy of semi-definite relaxations of (P). The idea is, at each rank of the hierarchy, to successively tighten semi-definite relaxations of (P) in order to reach its optimal solution value. It is also proven in [29] that this hierarchy converges in a finite number of iterations to the optimal solution of the considered problem. Further, this work has been extended to hierarchies of second order conic programs [3,21,28], and of sparse doubly non-negative relaxation [25]. Although these algorithms were not originally tailored for binary programming, they can

handle (P) by considering the quadratic constraint $x_i^2 = x_i$. Methods devoted to the binary polynomial case were also proposed. In [14,30], the authors use separable or convex under-estimators to approximate a given polynomial. Other methods based on linear reformulations can be found in [17,18,43], in which linear equivalent formulations to (P) are proposed and then improved. In [15], the authors focus on a polyhedral description of the linearization of a binary polynomial program. Finally, the work in [4] considers *quadratizations* with a minimal number of additional variables.

In this paper, we focus on finding equivalent quadratic convex formulations of (P). Quadratic convex reformulation methods [8,9] were introduced for the specific case where $d = 2$. The idea of these approaches is to build tight equivalent reformulations to (P) that have a convex objective function. This equivalent problem can be built using the dual solution of a semi-definite relaxation of (P), and further solved by a branch-and-bound algorithm based on quadratic convex relaxation. Here, we consider the more general case where $d \geq 3$, and we propose to compute an equivalent convex formulation to (P). Hence, we present an exact solution method for problem (P) that can be split in three phases. The first phase consists in building an equivalent formulation to (P) where both objective function and constraints are at most quadratic. For this, we need to add some auxiliary variables. We then obtain problem (QP) that has a quadratic objective function and linear inequalities.

Then in the second phase, we focus on the convexification of the obtained problem. As illustrated in the experiments of Section 4, the original `QCR` and `MIQCR` methods are not able to handle (QP). Indeed, `QCR` leads to a reformulation with a weak bound, and in method `MIQCR` the semi-definite program that we need to solve is too large. This is why, in this paper we introduce a tailored convexification phase. The idea is to apply convex quadratic reformulation to any quadratization of (P). For this, we need null quadratic functions on the domain of (QP) so as to perturb the Hessian matrix of the new quadratic objective function. One of these null functions comes from the classic binary identity, $x_i^2 = x_i$. One contribution of this paper is the introduction of new null quadratic functions on the domain of (QP). This set of functions varies according to the quadratization used in phase 1. Adding these functions to the new objective function, we get a family of convex equivalent formulations to (QP) that depend on some parameters. We then want to choose these parameters such that the continuous relaxation bound of the convexified problem is maximized. We show that they can be computed thanks to a semi-definite program. Finally, the last phase consists in solving the convexified problem using general-purpose optimization software.

Our experiments show that `PQCR` is able to solve to global optimality 6 unsolved instances of the low auto-correlation binary sequence problem and improves lower and/or upper bounds of 27 of the 45 instances available at the `minlplib` website, in comparison to the other available solvers.

The outline of the paper is the following. In Section 2, we define and present our quadratizations of (P). In Section 3, we introduce our family of convex

reformulations and we prove how we compute the best parameters. Then, in Section 4, we present our computational results on polynomial instances of degree 4 coming from the literature and we discuss different possible quadratizations of (P). Section 5 draws a conclusion.

2 Phase 1: Quadratization 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 $\mathcal{E}_i$ for the initial or additional variable i as follows:

Definition 1. *For all $i \in I \cup J$, we define $\mathcal{E}_i$ 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 $\mathcal{E}_i = \{i\}$*
- *If $i \in J$, i.e. x_i is an additional variable, then there exist $(i_1, i_2) \in (I \cup J)^2$ such that x_i replaces $x_{i_1} x_{i_2}$ and we set $\mathcal{E}_i = \mathcal{E}_{i_1} \cup \mathcal{E}_{i_2}$*

□

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 $\{\mathcal{E}_i,\ i \in I \cup J\}$ define a valid quadratization with N variables if, for any monomial p of degree greater than or equal to 3 (i.e. $|\mathcal{M}_p| \geq 3$), there exist $(j,k) \in (I \cup J)^2$ such that $\mathcal{M}_p = \mathcal{E}_j \cup \mathcal{E}_k$ and $\prod_{i \in \mathcal{M}_p} x_i = x_j x_k$. 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 $(QCQP)$ with N variables.

$$(QCQP) \begin{cases} \min g(x) = \displaystyle\sum_{\substack{|\mathcal{M}_p| \geq 3 \\ \mathcal{M}_p = \mathcal{E}_j \cup \mathcal{E}_k}} c_p x_j x_k + \sum_{|\mathcal{M}_p| \leq 2} c_p \prod_{i \in \mathcal{M}_p} x_i \\ \text{s.t.} \\ \quad x_i = x_{i_1} x_{i_2} \quad \forall (i, i_1, i_2) \in J \times (I \cup J)^2 :\ \mathcal{E}_i = \mathcal{E}_{i_1} \cup \mathcal{E}_{i_2} \quad (1) \\ \quad x \in \{0,1\}^N \end{cases}$$

As the variables are binary, Constraints (1) are equivalent to the classical set of Fortet inequalities [18]:

$$(C^i_{i_1,i_2}) \begin{cases} x_i - x_{i_1} \leq 0, \\ x_i - x_{i_2} \leq 0, \\ -x_i + x_{i_1} + x_{i_2} \leq 1, \\ -x_i \leq 0, \end{cases}$$

We now define set $\mathcal{F}_{\mathcal{E}}$:

$$\mathcal{F}_{\mathcal{E}} = \{x \in \{0,1\}^N : C^i_{i_1,i_2} \text{ is satisfied } \forall (i,i_1,i_2) \in J \times (I \cup J)^2 : \ \mathcal{E}_i = \mathcal{E}_{i_1} \cup \mathcal{E}_{i_2}\}.$$

We denote by $M = 4(N-n)$ the number of constraints of $\mathcal{F}_{\mathcal{E}}$. We thus obtain the following linearly constrained quadratic formulation that is equivalent to (P) and has N variables and M constraints:

$$(QP) \begin{cases} \min g(x) \equiv x^T Q x + c^t x \\ \text{s.t.} \\ \quad x \in \mathcal{F}_{\mathcal{E}} \end{cases}$$

where $Q \in S_N$ (the set of $N \times N$ real symmetric matrices), and $c \in \mathbb{R}^N$.

In the following, we will focus on the solution of problem (QP) that is an equivalent formulation to (P). Let us observe that (QP), as well as $(QCQP)$, are parameterized by the quadratization defined by sets $\mathcal{E}$. Indeed, several valid quadratizations can be applied to (P), each of them leading to different sets $\mathcal{E}_i$.

Different valid quadratizations were introduced and compared from the size point of view in [4]. In our case the comparison criterion is the continuous relaxation bound value from which we present our experimental comparison in Section 4.

Example 1 *[Different valid quadratizations]*
Let us consider the following problem:

$$(Ex) \Big\{ \min_{x \in \{0,1\}^4} 2x_1 + 3x_2x_3 - 2x_2x_3x_4 - 3x_1x_2x_3x_4$$

For instance, we can build three different equivalent functions:

- $g_1(x) = 2x_1 + 3x_2x_3 - 2\underbrace{x_2x_4}_{x_5}x_3 - 3\underbrace{x_1x_4}_{x_6}\underbrace{x_2x_3}_{x_7}$
- $g_2(x) = 2x_1 + 3x_2x_3 - 2\underbrace{x_3x_4}_{x_5}x_2 - 3\underbrace{x_1x_2}_{x_6}\underbrace{x_3x_4}_{x_5}$
- $g_3(x) = 2x_1 + 3x_2x_3 - 2\underbrace{x_2x_4}_{x_5}x_3 - 3\underbrace{x_1x_2}_{x_6}\underbrace{x_3x_4}_{x_7}$

$$(QEx_1)\begin{cases}\min g_1(x)\\ s.t.\\ \quad (x_2,x_4,x_5)\in C^5_{2,4}\\ \quad (x_1,x_4,x_6)\in C^6_{1,4}\\ \quad (x_2,x_3,x_7)\in C^7_{2,3}\\ \quad x\in\{0,1\}^7\end{cases}\quad(QEx_2)\begin{cases}\min g_2(x)\\ s.t.\\ \quad (x_3,x_4,x_5)\in C^5_{3,4}\\ \quad (x_1,x_2,x_6)\in C^6_{1,2}\\ \quad x\in\{0,1\}^6\end{cases}\quad(QEx_3)\begin{cases}\min g_3(x)\\ s.t.\\ \quad (x_2,x_4,x_5)\in C^5_{2,4}\\ \quad (x_1,x_2,x_6)\in C^6_{1,2}\\ \quad (x_3,x_4,x_7)\in C^7_{3,4}\\ \quad x\in\{0,1\}^7\end{cases}$$

Here we obtain 3 different quadratizations of (Ex) *with different sets* $\mathcal{E}$*. They have different sizes:* (QEx_1) *and* (QEx_3) *have 7 variables and 12 constraints, while* (QEx_2) *has 6 variables and 8 constraints.*

□

We have reduced the degree of the polynomial program (P) by building an equivalent quadratic program to (P). However, the solution of (QP) still has two difficulties, the non-convexity of the objective function g and the integrality of the variables.

Some state-of-the-art solvers can solve (QP) to global optimality (e.g. `Cplex 12.7` [24]). Unfortunately, these solvers may not be enough efficient for solving dense instances of (P). Here, we propose to compute an equivalent quadratic convex formulation to (QP). There exist several convexification methods devoted to quadratic programming (see, for example [9,11,16,22,34]). These approaches can be directly applied to (QP). For instance, one can use the `QCR` method, described in [11], that consists in computing an equivalent convex formulation to (QP) using semi-definite programming. The convexification is obtained thanks to a non uniform perturbation of the diagonal of the Hessian matrix. The semi-definite relaxation used can be easily solved due to its reasonable size. However, the bound obtained by continuous relaxation of the reformulation is very weak. As a consequence, for the considered instances of Section 4, the branch-and-bound used to solve the reformulation failed as soon as $n \geq 20$. Another alternative is to apply the `MIQCR` method [9]. In this method, the perturbation is generalized to the whole Hessian matrix and hence is more refined than the previous one. This leads to a reformulation with a significantly sharper bound. Unfortunately, the semi-definite relaxation used in this approach is too large and its computation failed even with instances of (P) containing only 10 variables. In the next section, we present a new convexification that leads to sharper bounds than `QCR` but with a better tractability than `MIQCR`.

3 Phase 2: A quadratic convex reformulation of (QP)

In this section, we consider the problem of reformulating (QP) by an equivalent quadratic 0-1 program with a convex objective function. To do this, we define a new convex function which value is equal to the value of $g(x)$, but which Hessian matrix is positive semi-definite. More precisely, we first add to $g(x)$ a combination of four sets of functions that vanish on the feasible set $\mathcal{F}_{\mathcal{E}}$. For each

function we introduce a scalar parameter. Then we focus on computing the best parameters that lead to a convex function and that maximize the optimal value of the continuous relaxation of the obtained problem.

3.1 Valid quadratic equalities for (QP)

For a quadratization characterized by $\mathcal{E}$, we introduce null quadratic functions over the set $\mathcal{F}_{\mathcal{E}}$.

Lemma 1 *The following quadratic equalities characterize null functions over* $\mathcal{F}_{\mathcal{E}}$*:*

$$(S_{\mathcal{E}}) \begin{cases} x_i^2 - x_i = 0 & i \in I \cup J & (2) \\ x_i - x_i x_j = 0 & (i,j) \in J \times (I \cup J) : \ \mathcal{E}_j \subset \mathcal{E}_i & (3) \\ x_i - x_j x_k = 0 & (i,j,k) \in J \times (I \cup J)^2 : \ \mathcal{E}_i = \mathcal{E}_j \cup \mathcal{E}_k & (4) \\ x_i x_j - x_k x_l = 0 & (i,j,k,l) \in (I \cup J)^4 : \ \mathcal{E}_i \cup \mathcal{E}_j = \mathcal{E}_k \cup \mathcal{E}_l & (5) \end{cases}$$

Proof. Constraints (2) trivially hold since $x_i \in \{0,1\}$. Constraints (4) come from Definition 1. We then prove the validity of the Constraints (3) and (5).

– *Constraints (3):* we have $x_i = \prod_{i' \in \mathcal{E}_i} x_{i'}$ and $x_j = \prod_{j' \in \mathcal{E}_j} x_{j'}$, then:

$$\begin{aligned} x_i x_j &= \prod_{i' \in \mathcal{E}_i} x_{i'} \prod_{j' \in \mathcal{E}_j} x_{j'} \\ &= \prod_{j' \in \mathcal{E}_j} x_{j'}^2 \prod_{i' \in \mathcal{E}_i \backslash \mathcal{E}_j} x_{i'} \text{ since } \mathcal{E}_j \subset \mathcal{E}_i \\ &= \prod_{i' \in \mathcal{E}_i} x_{i'} \text{ since } x_{j'}^2 = x_{j'} \text{ and } \mathcal{E}_j \cup (\mathcal{E}_i \backslash \mathcal{E}_j) = \mathcal{E}_i \\ &= x_i \end{aligned}$$

– *Constraints (5):* by definition we have:

$$\begin{aligned} x_i x_j &= \prod_{i' \in \mathcal{E}_i} x'_i \prod_{j' \in \mathcal{E}_j} x_{j'} \\ &= \prod_{i' \in \mathcal{E}_i \cup \mathcal{E}_j} x_{i'} \prod_{j' \in \mathcal{E}_i \cap \mathcal{E}_j} x_{j'} \\ &= \prod_{i' \in (\mathcal{E}_i \cup \mathcal{E}_j) \backslash (\mathcal{E}_i \cap \mathcal{E}_j)} x_{i'} \prod_{j' \in \mathcal{E}_i \cap \mathcal{E}_j} x_{j'}^2 \\ &= \prod_{i' \in \mathcal{E}_i \cup \mathcal{E}_j} x_{i'} \text{ since } x_{j'}^2 = x_{j'} \text{ and } (\mathcal{E}_i \cup \mathcal{E}_j) \backslash (\mathcal{E}_i \cap \mathcal{E}_j) \cup (\mathcal{E}_i \cap \mathcal{E}_j) = (\mathcal{E}_i \cup \mathcal{E}_j) \\ &= \prod_{k' \in \mathcal{E}_k \cup \mathcal{E}_l} x_{k'} \text{ since } \mathcal{E}_i \cup \mathcal{E}_j = \mathcal{E}_k \cup \mathcal{E}_l \\ &= x_k x_l \end{aligned}$$

□

3.2 An equivalent quadratic convex reformulation to (QP)

We now compute a quadratic convex reformulation of (QP) and thus of (P). For this, we add to the objective function g the null quadratic forms in (2)–(5). For each of them, we associate a real scalar parameter: α_i for Constraints (2), β_{ij} for Constraints (3), δ_{ijk} for Constraints (4), and λ_{ijkl} for Constraints (5). We get the following parameterized function:

$$g_{\alpha,\beta,\delta,\lambda}(x) = g(x) + \sum_{i\in I\cup J} \alpha_i(x_i^2 - x_i) + \sum_{\substack{(i,j)\in J\times(I\cup J)\\ \mathcal{E}_j\subset\mathcal{E}_i}} \beta_{ij}(x_i - x_i x_j)$$
$$+ \sum_{\substack{(i,j,k)\in J\times(I\cup J)^2\\ \mathcal{E}_i=\mathcal{E}_j\cup\mathcal{E}_k}} \delta_{ijk}(x_i - x_j x_k) + \sum_{\substack{(i,j,k,l)\in(I\cup J)^4\\ \mathcal{E}_i\cup\mathcal{E}_j=\mathcal{E}_k\cup\mathcal{E}_l}} \lambda_{ijkl}(x_i x_j - x_k x_l)$$

Obviously $g_{\alpha,\beta,\delta,\lambda}(x)$ has the same value as $g(x)$ for any $x \in \mathcal{F}_\mathcal{E}$. Moreover, there exist vector parameters α, β, δ and λ such that $g_{\alpha,\beta,\delta,\lambda}$ is a convex function. Take for instance, α equals to the opposite of the smallest eigenvalue of Q, and $\beta = \delta = \lambda = 0$.

By replacing g by the new function, we obtain the following family of quadratic convex equivalent formulation to (QP):

$$(QP_{\alpha,\beta,\delta,\lambda}) \begin{cases} \min g_{\alpha,\beta,\delta,\lambda}(x) \equiv x^T Q_{\alpha,\beta,\delta,\lambda} x + c^T_{\alpha,\beta,\delta,\lambda} x \\ \text{s.t.} \\ \quad x \in \mathcal{F}_\mathcal{E} \end{cases}$$

where $Q_{\alpha,\beta,\delta,\lambda} \in S_N$ is the Hessian matrix of $g_{\alpha,\beta,\delta,\lambda}(x)$, and $c_{\alpha,\beta,\delta,\lambda} \in \mathbb{R}^N$ is the vector of linear coefficients of $g_{\alpha,\beta,\delta,\lambda}(x)$.

In order to use $(QP_{\alpha,\beta,\delta,\lambda})$ within a branch-and-bound procedure, we are interested by parameters (α, β, δ, λ) such that $g_{\alpha,\beta,\delta,\lambda}$ is a convex function. Moreover, in order to have a good behavior of the branch-and-bound algorithm, we want to find parameters that give the tightest continuous relaxation bound. More formally, we want to solve the following optimization problem:

$$(CP): \max_{\substack{\alpha\in\mathbb{R}^N,\beta\in\mathbb{R}^{T_1},\delta\in\mathbb{R}^{T_2},\lambda\in\mathbb{R}^{T_3}\\ Q_{\alpha,\beta,\delta,\lambda}\succeq 0}} \left\{ \min_{x\in\overline{\mathcal{F}}_\mathcal{E}} g_{\alpha,\beta,\delta,\lambda}(x) \right\}$$

where T_1, T_2 and T_3 are the number of Constraints (3), (4), and (5), respectively, and $\overline{\mathcal{F}}_\mathcal{E}$ is the set $\mathcal{F}_\mathcal{E}$ where the integrality constraints are relaxed, i.e. $x \in [0,1]^N$.

In the rest of the paper we will focus on solving (CP). For this, we build a compact semi-definite relaxation that uses our new valid equalities and prove that its optimal dual variables provide an optimal solution to (CP).

3.3 Computing an optimal solution to (CP)

The following theorem shows that problem (CP) is equivalent to the dual of a semi-definite relaxation of (QP).

Theorem 1. *The optimal value of (CP) is equal to the optimal value of the following semi-definite program (SDP):*

$$(SDP)\begin{cases} \min < Q, X > + c^T x & \\ s.t. & \\ \quad X_{ii} - x_i = 0 & i \in I \cup J \quad (6)\\ \quad -X_{ij} + x_i = 0 & (i,j) \in J \times (I \cup J) : \ \mathcal{E}_j \subset \mathcal{E}_i \quad (7)\\ \quad -X_{jk} + x_i = 0 & (i,j,k) \in J \times (I \cup J)^2 : \ \mathcal{E}_i = \mathcal{E}_j \cup \mathcal{E}_k \quad (8)\\ \quad X_{ij} - X_{kl} = 0 & (i,j,k,l) \in (I \cup J)^4 : \ \mathcal{E}_i \cup \mathcal{E}_j = \mathcal{E}_k \cup \mathcal{E}_l \quad (9)\\ \quad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 & \quad (10)\\ \quad x \in \mathbb{R}^N, \ X \in S^N & \quad (11) \end{cases}$$

The optimal values $(\alpha^*, \beta^*, \delta^*, \lambda^*)$ of problem (CP) are given by the optimal values of the dual variables associated with constraints (6)–(9) respectively.

Proof. For simplicity, we rewrite $\mathcal{F}_\mathcal{E}$ as follows: $\mathcal{F}_\mathcal{E} = \{x \in \{0,1\}^N : Ax \leq b\}$ where A is a $M \times N$-matrix, $b \in \mathbb{R}^M$, and we introduce $T = N + T_1 + T_2 + T_3$ the number of Constraints (2)–(5) respectively.

We start by observing that $x \in [0,1]^N$ is equivalent to $x^2 \leq x$, thus, (CP) is equivalent to $(Q1)$:

$$(Q1) : \max_{\substack{\alpha \in \mathbb{R}^N, \beta \in \mathbb{R}^{T_1}, \delta \in \mathbb{R}^{T_2}, \lambda \in \mathbb{R}^{T_3} \\ Q_{\alpha,\beta,\delta,\lambda} \succeq 0}} \left\{ \min_{x \in \mathbb{R}^N, \ x^2 \leq x, \ Ax \leq b} g_{\alpha,\beta,\delta,\lambda}(x) \right\}$$

$(Q1)$ is a convex optimization problem over a convex set. If we consider the solution $\tilde{x}_i = 0.5 \ \forall i \in I$ and $\tilde{x}_i = \tilde{x}_j \tilde{x}_k \ \forall (i,j,k) \in J \times (I \cup J)^2$, $\mathcal{E}_i = \mathcal{E}_j \cup \mathcal{E}_k$, the obtained $\tilde{x}$ is an interior point and the Slater's conditions are satisfied for the minimization sub-problem. Then, by Lagrangian duality, we have $(Q1)$ equivalent to $(Q2)$:

$$(Q2) : \max_{\substack{\alpha \in \mathbb{R}^N, \beta \in \mathbb{R}^{T_1}, \delta \in \mathbb{R}^{T_2}, \lambda \in \mathbb{R}^{T_3}, \omega \in \mathbb{R}^N_+, \gamma \in \mathbb{R}^M_+ \\ Q_{\alpha,\beta,\delta,\lambda} \succeq 0}} \left\{ \min_{x \in \mathbb{R}^N} g_{\alpha,\beta,\delta,\lambda}(x) + \omega^T(x^2 - x) + \gamma^T(Ax - b) \right\}$$

Due to Constraints (2), it holds that $(Q2)$ is equivalent to $(Q3)$:

$$(Q3) : \max_{\substack{\alpha \in \mathbb{R}^N, \beta \in \mathbb{R}^{T_1}, \delta \in \mathbb{R}^{T_2}, \lambda \in \mathbb{R}^{T_3}, \gamma \in \mathbb{R}^M_+ \\ Q_{\alpha,\beta,\delta,\lambda} \succeq 0}} \left\{ \min_{x \in \mathbb{R}^N} g_{\alpha,\beta,\delta,\lambda}(x) + \gamma^T(Ax - b) \right\}$$

It is well known that a necessary condition for the quadratic function $g_{\alpha,\beta,\delta,\lambda,\gamma}(x)+\gamma^T(Ax-b)$ to have a minimum not equal to $-\infty$ is that matrix $Q_{\alpha,\beta,\delta,\lambda}$ is positive semi-definite. Therefore $(Q3)$ is equivalent to $(Q4)$:

$$(Q4): \max_{\alpha\in\mathbb{R}^N,\beta\in\mathbb{R}^{T_1},\delta\in\mathbb{R}^{T_2},\lambda\in\mathbb{R}^{T_3},\gamma\in\mathbb{R}^M_+} \left\{ \min_{x\in\mathbb{R}^N} g_{\alpha,\beta,\delta,\lambda,\gamma}(x) + \gamma^T(Ax-b) \right\}$$

We know from [32] that $(Q4)$ is equivalent to problem (D):

$$(D)\begin{cases} \max t \\ \text{s.t.} \\ \begin{pmatrix} -\gamma^T b - t & \frac{1}{2}(c^T_{\alpha,\beta,\delta,\lambda}+\gamma^T A) \\ \frac{1}{2}(c_{\alpha,\beta,\delta,\lambda}+A^T\gamma) & Q_{\alpha,\beta,\delta,\lambda} \end{pmatrix} \succeq 0 \\ t\in\mathbb{R},\ \alpha\in\mathbb{R}^N,\ \beta\in\mathbb{R}^{T_1},\ \delta\in\mathbb{R}^{T_2},\ \lambda\in\mathbb{R}^{T_3},\ \gamma\in\mathbb{R}^M_+ \end{cases}$$

By semi-definite duality of program (D), and with $\alpha,\beta,\delta,\lambda$ the dual variables associated with Constraints (6)–(9) respectively, we get (SDP'):

$$(SDP')\begin{cases} \min < Q, X > + c^T x \\ \text{s.t.} \\ \quad (6)-(11) \\ \quad Ax \le b \end{cases}$$

We now prove that there is no duality gap between (D) and (SDP'), which holds since:

(i) The feasible domain of (SDP') is nonempty, as $(QP_{\alpha,\beta,\delta,\lambda})$ contains 0 as a feasible solution and (D) is bounded

(ii) (D) satisfies Slater's condition. It is sufficient to take β, δ and λ equal to 0, α large enough so that $Q_{\alpha,\beta,\delta,\lambda}\succeq 0$ holds, and t a large negative number that ensures the diagonal dominance of the first row and the first column of matrix $\begin{pmatrix} -\gamma^T b - t & \frac{1}{2}(c^T_{\alpha,\beta,\delta,\lambda}+\gamma^T A) \\ \frac{1}{2}(c_{\alpha,\beta,\delta,\lambda}+A^T\gamma) & Q_{\alpha,\beta,\delta,\lambda} \end{pmatrix}$.

From these equivalences, we know that we can build an optimal solution of (CP) from the optimal dual variables of (SDP'). However, constraints $Ax\le b$ are redundant in (SDP') and we thus prove in Lemma 2 that (SDP') and (SDP) are equivalent. As a consequence, an optimal solution to (CP) can be deduced from the optimal dual variables of (SDP).

Lemma 2 *Due to Constraints (6)–(8) and (10), inequalities $Ax\le b$ are redundant in (SDP').*

Proof. Recall that $Ax \leq b$ are the inequalities of $(C^i_{j,k})$, $\forall (i,j,k) \in J \times (I \cup J)^2 : \ \mathcal{E}_i = \mathcal{E}_j \cup \mathcal{E}_k$, i.e $x_i \geq 0$ (a), $x_i \leq x_j$ (b), $x_i \leq x_k$ (c), and $x_i \geq x_j + x_k - 1$ (d).

The basic idea used here is that, since matrix $\begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}$ is positive semi-definite, all its symmetric minors are non-negative.

- Constraint (a): $x_i \geq 0$. We consider the determinant $\begin{vmatrix} 1 & x_i \\ x_i & X_{ii} \end{vmatrix}$, which implies $X_{ii} - x_i^2 \geq 0$. By (6) we obtain $x_i - x_i^2 \geq 0$ and thus $x_i \geq 0$.
- Constraint (b): $x_i \leq x_j$. Considering the determinant of the symmetric minor $\begin{vmatrix} X_{jj} & X_{ji} \\ X_{ij} & X_{ii} \end{vmatrix}$ implies $X_{ii}X_{jj} - X_{ij}^2 \geq 0$. By (6) we have $x_j x_i - X_{ij}^2 \geq 0$ and by (7) we obtain $x_i x_j - x_i^2 \geq 0$. Either $x_i > 0$ and then we have $x_j - x_i \geq 0$, or $x_i = 0$ and the inequality comes from $x_j \geq 0$.
- Constraint (c): $x_i \leq x_k$. By symmetry, i.e. considering the determinant $\begin{vmatrix} X_{kk} & X_{ki} \\ X_{ik} & X_{ii} \end{vmatrix}$, the inequality holds.
- Constraint (d): $x_i \geq x_j + x_k - 1$. By definition (10) implies $z^T \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} z \geq 0$, $\forall z \in \mathbb{R}^{N+1}$. By taking $\bar{z} = (1, 0, .., 0, \underbrace{-1}_{j}, 0, .., 0, \underbrace{-1}_{k}, 0, .., 0, \underbrace{1}_{i}, 0, .., 0)$, we have:

$$\begin{aligned} 0 \leq \bar{z}^T \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \bar{z} &= (x_i + 1 - x_j - x_k) - (x_j - X_{jj} - X_{kj} + X_{ij}) \\ &\quad -(x_k - X_{kk} - X_{jk} + X_{ik}) + (x_i - X_{ji} - X_{ki} + X_{ii}) \\ &= (x_i + 1 - x_j - x_k) \text{ by (6), (7) and (8).} \end{aligned}$$

□

Let us state Corollary 1 that shows that from an optimal dual solution to (SDP) we can build an optimal solution to (CP).

Corollary 1. *We have $v(CP) = v(SDP)$ where $v(.)$ is the optimal value of problem $(.)$. Consequently, an optimal solution $(\alpha^*, \beta^*, \delta^*, \lambda^*)$ of (CP) corresponds to the optimal values of the dual variables associated with constraints (6)–(9) of (SDP) respectively.*

Proof. We have:

(i) $v(CP) = v(D)$
(ii) since there is no duality gap between (D) and (SDP'), we have $v(D) = v(SDP')$
(iii) by Lemma 2, we get $v(CP) = v(D) = v(SDP') = v(SDP)$

□

□

To sum up, we obtain (QP^*), the best equivalent convex formulation to (QP):

$$(QP^*) \begin{cases} \min g_{\alpha^*,\beta^*,\delta^*,\lambda^*}(x) \\ \text{s.t.} \\ \quad x \in \mathcal{F}_{\mathcal{E}} \end{cases}$$

From Theorem 1, we deduce the Algorithm 1 to solve (P).

Algorithm 1 `PQCR` an exact solution method for (P)

Step 1: Apply a quadratization $\mathcal{E}$ to (P) and thus generate sets $\mathcal{F}_{\mathcal{E}}$ and $\mathcal{S}_{\mathcal{E}}$.
Step 2: Solve (SDP), deduce optimal values α^*, β^*, δ^*, λ^*, and build (QP^*).
Step 3: Solve (QP^*) by a standard quadratic convex programming solver.

4 Numerical results

In this section, we evaluate `PQCR` on two applications. The first one is the *image restoration (vision)* problem [17], which results are presented in Section 4.1. The instances of this application are quite sparse with an average ratio $\frac{m}{n}$ of about 7. We choose to use these instances in order to compare `PQCR` with existing convexifications and in particular with methods `QCR` and `MIQCR` that are not able to handle larger and/or denser instances. Then, in Section 4.2, we present the results of the second application, the *low auto-correlation binary sequence (LABS)* problem [7] which instances are much denser (average ratio $\frac{m}{n}$ of about 212). These instances are available on the `minlplib` website [35], and are very hard to solve. For most of them, the optimal solution value is not known. For these experiments, we have chosen the quadratization described in Algorithm 2 for Step 1 of `PQCR`. This choice impacts the number of constraints within the sets $\mathcal{F}_{\mathcal{E}}$ and $\mathcal{S}_{\mathcal{E}}$, and the associated continuous relaxation bound value can thus vary. We further illustrate this variation on toy instances in Section 4.3.

The quadratization used in our experiments is presented in Algorithm 2.

Algorithm 2 Quadratization(f)

Require: A polynomial f of degree $d > 2$
Ensure: A quadratic function f' verifying $\forall x \in \{0,1\}^n,\ f'(x) = f(x)$

for each monomial p from 1 to m **do**
  Sort p by lexicographical order
  $deg \leftarrow deg(p)$
  while $deg > 2$ **do**
    $s \leftarrow \lfloor \frac{deg}{2} \rfloor$
    for l from 1 to s **do**
      Consider the l^{th} consecutive pair of variables $x_j x_k$
      Find x_i that represents the product $x_j x_k$
      if x_i does not exist **then**
        Create an additional variable x_i and $\mathcal{E}_i \leftarrow \mathcal{E}_j \cup \mathcal{E}_k$
      end if
      Replace $x_j x_k$ by x_i
    end for
    $deg \leftarrow \lceil \frac{deg}{2} \rceil$
  end while
end for

Example 1. Applying the quadratization of Algorithm 2 to the monomial $x_1x_2x_3x_4x_5$ we obtain the following monomial of degree 3 at the first iteration:

$$\underbrace{x_1x_2}_{x_6}\underbrace{x_3x_4}_{x_7}x_5$$

we then obtain a quadratic reformulation of the monomial at the second iteration using 3 additional variables:

$$\underbrace{x_6x_7}_{x_8}x_5$$

□

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. For all algorithms, we used the multi-threading version of `Cplex 12.7` with up to 48 threads.

In our experiments, we use three classes of solution algorithms:

i) The first class includes 3-phase algorithms that consist in a quadratization and a convexification followed by the solution of the equivalent convex problem with the solver `Cplex 12.7`: these methods are `PQCR` and `Q+QCR`. For both methods, the quadratization is implemented in `C`, and we used

the solver `csdp` to solve the semi-definite programs. For denser instances (Section 4.2), we used the solver `csdp` [12] together with the `Conic Bundle algorithm` [23] to solve the semi-definite program of `PQCR`, as described in [10]. Then, we used the `ampl` [19] interface of the solver `Cplex 12.7` [24] to solve the obtained quadratic convex problem with binary variables.

ii) The second class includes a 2-phase algorithm, called `Q+Cplex`, that consists in a quadratization followed by the direct submission to `Cplex 12.7`. Here again, the quadratization is implemented in `C`, and we used the `ampl` interface of the solver `Cplex 12.7`.

iii) The third class includes the direct submission to the general mixed-integer non-linear solver `Baron 17.4.1` [42]. Here, we used the `gams` interface of the solver `Baron 17.4.1`.

Parameters of the solvers

- `Cplex` : we let the default parameters, except the parameter `qptolin` that is set to `0` for methods `PQCR` and `Q+QCR`.
- `csdp` : Parameters `axtol, aytol` of `Csdp` are set to 10^{-3}.
- `Conic Bundle` : the precision is set to 10^{-3}. Parameter p (see [10]) is set to $0.2 * |\mathcal{F}_{\mathcal{E}}|$.

Legends of Tables 1-3

- *Name*: Name of the considered instance.
- n: number of variables in the polynomial formulation.
- m: number of monomials.
- BKN: is the optimal solution value or the best known solution value of the instance.
- N: number of variables after quadratization.
- *gap*: is the initial gap, i.e. the gap at the root node of the branch-and-bound, $gap = \left|\frac{BKN - LB_i}{BKN}\right| * 100$, where LB_i is the initial lower bound.
- *Solution*: best solution value found within the time limit.
- *tSdp*: CPU time in seconds for solving semi-definite programs in `PQCR` and `Q+QCR`. The time limit is set to 2400 seconds for the *vision* problem and 3 hours for the *LABS* problem. If the solver reaches the time limit, *tSdp* is labeled as "-".
- *tTotal*: total CPU time in seconds of the associated method. The time limit is set to 1 hour for the *vision* problem and 5 hours for the *LABS* problem. If an instance remains unsolved within the time limit, we put the *final gap*= $\left|\frac{BKN - LB_f}{BKN}\right| * 100$, where LB_f is the final lower bound.
- *Nodes*: number of nodes visited by the branch-and-bound algorithm.

4.1 The image restoration problem

The *vision* instances are inspired from the image restoration problem, which arises in computer vision. The goal is to reconstruct an original sharp base image from a blurred image. An image is a rectangle containing $n = l \times h$ pixels. This rectangle is modeled as a binary matrix of the same dimension. A complete description of these instances can be found in [17]. The problem is modeled by the minimization of a degree 4 polynomial of binary variables where each variable represents a pixel. The coefficients of the monomials are indicative of how likely a configuration is to appear on the sharp base image. The size of the considered instances are $l \times h = 10 \times 10$, 10×15, and 15×15, or in the polynomial formulation $n = 100, 150$ and 225, with a number of monomials of $m = 668$, 1033, and 1598 respectively. In our experiments, 15 instances of each size are considered obtaining a total of 45 instances. Observe that the 15 instances of the same size have identical monomials with different coefficients, because they represent different images with the same number of pixels.

We now focus on the comparison of several convexification methods after quadratization. Indeed, several ways are possible to solve the quadratic non-convex program (QP). For instance, the standard solver `Cplex` can directly handle it, or one can apply the `QCR` [11] or `MIQCR` [9] methods. We compare `PQCR` with these three approaches. We do not report the results for method `Q+MIQCR` since it was not able to start the computation due to the size of the considered instances. We also give the computational results coming from the direct submission of (P) to the solver `Baron 17.4.1`. Our observations for these instances are summed up in Table 1, where each line corresponds to one instance, where the i^{th} instance of $l \times h$ pixels is labeled `v.l.h i`.

Instance				PQCR			Q+QCR			Q+Cplex		Baron	
Name	n	m	N	*Gap*	*tSdp*	*tTotal*	*Gap*	*tSdp*	*tTotal*	*Gap*	*tTotal*	*Gap*	*tTotal*
v.10.10 1	100	668	352	**0,59**	66	68	396	7	(250 %)	1113	**2**	1098	15
v.10.10 2	100	668	352	**0,28**	64	66	536	8	(343 %)	1549	**2**	1529	10
v.10.10 3	100	668	352	**0,05**	65	67	973	8	(573 %)	3375	**1**	3332	6
v.10.10 4	100	668	352	**0,12**	63	65	957	8	(561 %)	3377	**1**	3334	6
v.10.10 5	100	668	352	**0,13**	65	66	1006	8	(585 %)	3568	**1**	3523	5
v.10.10 6	100	668	352	**0,11**	73	74	359	9	(229 %)	984	**2**	972	11
v.10.10 7	100	668	352	**0,02**	64	65	305	8	(194 %)	829	**2**	817	14
v.10.10 8	100	668	352	**1,37**	64	66	1376	8	(804 %)	4765	**1**	4705	7
v.10.10 9	100	668	352	**3,02**	65	67	1749	8	(1026 %)	6187	**1**	6110	4
v.10.10 10	100	668	352	**3,64**	66	68	1879	8	(1075 %)	6843	**1**	6757	4
v.10.10 11	100	668	352	**0,36**	70	72	489	8	(316 %)	1388	**2**	1370	35
v.10.10 12	100	668	352	**0,20**	70	72	361	9	(232 %)	997	**2**	984	23
v.10.10 13	100	668	352	**0,00**	60	61	709	8	(392 %)	2654	**1**	2620	2
v.10.10 14	100	668	352	**0,00**	60	61	546	8	(297 %)	2027	**1**	2001	2
v.10.10 15	100	668	352	**0,00**	118	119	541	8	(285 %)	2048	**1**	2022	1
v.10.15 1	150	1033	542	**0,31**	290	294	447	24	(351 %)	1245	**5**	1234	80
v.10.15 2	150	1033	542	**0,00**	285	287	367	24	(287 %)	999	**5**	990	36
v.10.15 3	150	1033	542	**0,05**	280	283	1027	27	(772 %)	3549	**3**	3520	6
v.10.15 4	150	1033	542	**0,33**	276	280	845	27	(640 %)	2840	**3**	2817	7
v.10.15 5	150	1033	542	**0,07**	269	271	799	27	(595 %)	2808	**3**	2785	4
v.10.15 6	150	1033	542	**0,55**	297	302	462	25	(366 %)	1277	**5**	1266	47
v.10.15 7	150	1033	542	**0,08**	288	291	360	26	(283 %)	981	**5**	972	38
v.10.15 8	150	1033	542	**0,79**	281	284	1792	26	(1356 %)	6202	**3**	6152	19
v.10.15 9	150	1033	542	**1,80**	283	286	1525	26	(1160 %)	5209	**3**	5167	10
v.10.15 10	150	1033	542	**1,38**	275	279	1510	25	(1124 %)	5500	**3**	5456	7
v.10.15 11	150	1033	542	**0,10**	283	286	391	25	(305 %)	1102	**5**	1092	41
v.10.15 12	150	1033	542	**0,60**	275	279	453	25	(355 %)	1269	**5**	1258	125
v.10.15 13	150	1033	542	**0,00**	254	256	634	27	(469 %)	2254	**3**	2236	4
v.10.15 14	150	1033	542	**0,04**	269	273	731	27	(547 %)	2590	**3**	2569	2
v.10.15 15	150	1033	542	**0,00**	258	259	576	28	(423 %)	2183	**2**	2165	2
v.15.15 1	225	1598	827	**0,12**	1234	1244	365	70	(320 %)	998	**9**	993	(95 %)
v.15.15 2	225	1598	827	**0,43**	1251	1265	482	83	(421 %)	1350	**9**	1343	(138 %)
v.15.15 3	225	1598	827	**0,10**	1167	1175	678	65	(582 %)	2326	**5**	2313	(83 %)
v.15.15 4	225	1598	827	**0,04**	1251	1256	877	65	(753 %)	2996	**5**	2980	(127 %)
v.15.15 5	225	1598	827	**0,03**	1167	1174	641	67	(546 %)	2252	**5**	2240	(76 %)
v.15.15 6	225	1598	827	**0,28**	1238	1249	403	64	(353 %)	1104	**10**	1098	(107 %)
v.15.15 7	225	1598	827	**0,36**	1237	1246	525	67	(463 %)	1455	**10**	1447	(144 %)
v.15.15 8	225	1598	827	**0,29**	1197	1205	1148	73	(979 %)	4104	**5**	4082	(137 %)
v.15.15 9	225	1598	827	**0,27**	1170	1176	1542	66	(1315 %)	5570	**5**	5541	(171 %)
v.15.15 10	225	1598	827	**0,31**	1173	1179	1194	67	(1020 %)	4380	**5**	4357	1154
v.15.15 11	225	1598	827	**0,27**	1224	1230	529	69	(462 %)	1528	**8**	1520	(144 %)
v.15.15 12	225	1598	827	**0,25**	1225	1235	461	68	(414 %)	1273	**12**	1266	(133 %)
v.15.15 13	225	1598	827	**0,00**	1124	1128	651	63	(551 %)	2398	**4**	2385	1239
v.15.15 14	225	1598	827	**0,02**	1171	1177	651	63	(553 %)	2359	**5**	2346	(79 %)
v.15.15 15	225	1598	827	**0,00**	1100	1103	609	65	(513 %)	2320	**4**	2308	263

Table 1: *Comparison of the maximum time on 4 solution methods for the vision instances - time limit one hour*

We start by comparing the convexification phase of our new algorithm with the original QCR and MIQCR methods. We observe that none of these convexifications are able to handle any considered instances: QCR because of the weakness of its initial gap, and MIQCR because of the size of the semidefinite problem considered for computing the best reformulation. These experiments confirm the interest of designing PQCR, an algorithm devoted to polynomial optimisation. Then, we can see that Q+Cplex dominates PQCR. However, one have to note that these instances are very sparse (average ratio $\frac{m}{n}$ of about 7). It is well known

that the standard linearization performs very well on sparse instances. Clearly, for these instances, the time spent on solving a large semidefinite program, even once, is not profitable in comparison to the efficiency of LP heuristic or cut methods implemented in `cplex 12.7`. Indeed, `Q+Cplex` solves all the considered instances at the root node of its branch-and-bound. Moreover, it is interesting to remark that 99% of the CPU time of `PQCR` is spent for solving (SDP), while the CPU time for solving (QP^*) is always smaller than 14 seconds. Finally, we compare `PQCR` with the direct submission to the solver `Baron`. We observe that `Baron` is faster than `PQCR` on the medium size instances ($n = 100$ or 150), but is not able to solve all the larger instances within the time limit. Indeed, for $n = 225$, it solves only 3 instances out of 15. On the contrary, `PQCR` seems quite stable to the increase of the size of the instances. Indeed, the initial gap remains stable (0.42% on average) while the total CPU time increases reasonably.

4.2 The *Low Auto-correlation Binary Sequence* problem

We consider the problem of binary sequences with low off-peak auto-correlations. More formally, let S be a sequence $S = (s_1, \ldots, s_n)$ with $s \in \{-1, 1\}^n$, and for a given $k = 0, \ldots, n-1$, we define the auto-correlations $C_k(S)$ of S:

$$C_k(S) = \sum_{i=1}^{n-k} s_i s_{i+k}$$

The problem is to find a sequence S of length n that minimizes $E(S)$, a degree 4 polynomial:

$$E(S) = \sum_{k=1}^{n-1} C_k^2(S)$$

This problem has numerous practical applications in communication engineering, or theoretical physics [7]. For our experiments, we consider truncated instances, i.e. sequences of length n where we compute low off-peak auto-correlation up to a certain distance $n_0 \leq n$, i.e. we consider the following function to minimize:

$$E_{n_0}(S) = \sum_{k=1}^{n_0-1} C_k^2(S)$$

In order to apply `PQCR`, which is initially tailored for $\{0, 1\}$ polynomial programs, we convert the variables from $\{-1, 1\}$ to $\{0, 1\}$ using the standard transformation $x = \frac{s+1}{2}$.

This problem admits a lot of symmetries. In particular the correlations C_k are identical for a sequence S and its complement. We exploited this symmetry by fixing to 0 the variable that appears the most. Each instance is labeled `b.n.`$\mathtt{n_0}$. These instances were introduced by [33] and can be found on the `minlplib` [35] or the `polip` [38] websites. We do not report the results for methods `Q+QCR` and `Q+MIQCR` since they have failed to solve all the considered instances. Two instances that are already quadratic (`b.20.03` and `b.25.03`) are solved by the

method `Q+Cplex` in 7 and 75 seconds respectively. However, this method was not able to solve the other instances within the time limit.

Instance			PQCR					Baron 17.4.1		
Name	n	m	N	*Gap*	*tSdp*	*tTotal*	*Nodes*	*Gap*	*tTotal*	*Nodes*
b.20.03	20	38	20	**0**	1	2	0	100	**1**	1
b.20.05	20	207	65	**23**	22	23	5886	1838	**2**	1
b.20.10	20	833	124	**8**	837	846	24183	2918	**125**	7
b.20.15	20	1494	164	**5**	1228	1242	9130	3202	**728**	9
b.25.03	25	48	25	**0**	1	2	0	100	**0**	1
b.25.06	25	407	105	**17**	461	469	163903	2307	**65**	27
b.25.13	25	1782	206	**4**	**1552**	1603	76828	3109	3750	75
b.25.19	25	3040	265	**4**	-	**13433**	224550	3356	14399	129
b.25.25	25	3677	289	**5**	-	**13395**	167423	3405	(12 %)	100
b.30.04	30	223	82	**23**	58	78	134635	1347	**7**	7
b.30.08	30	926	174	**10**	1940	**2040**	752765	2696	2778	237
b.30.15	30	2944	296	**5**	-	**13525**	438278	3221	(21 %)	103
b.30.23	30	5376	390	**11**	5953	**6865**	9337391	3450	(135 %)	8
b.30.30	30	6412	422	**4**	8500	**15352**	452460	3470	(161 %)	5
b.35.04	35	263	97	**19**	135	167	156085	1350	**32**	13
b.35.09	35	1381	234	**10**	2245	**4630**	8163651	2826	(29 %)	354
b.35.18	35	5002	419	**644**	-	**(12 %)**	4899872	3356	(133 %)	10
b.35.26	35	8347	530	**30**	-	**(5 %)**	5006407	3508	(229 %)	3
b.35.35	35	10252	579	**12**	-	**(11 %)**	134426	3499	(214 %)	3
b.40.05	40	447	145	**25**	430	**1630**	23459121	1856	3674	1021
b.40.10	40	2053	304	**9**	-	**(4 %)**	25480163	2953	(54 %)	147
b.40.20	40	7243	544	**9**	-	**(4 %)**	9783350	3405	(203 %)	3
b.40.30	40	12690	702	**360**	-	**(25 %)**	281134	3561	(274 %)	1
b.40.40	40	15384	762	**62**	-	**(44 %)**	57534	3536	(464 %)	1
b.45.05	45	507	165	**24**	1384	**(4 %)**	84159279	1854	**16609**	4727
b.45.11	45	2813	382	**9**	-	**(2 %)**	25114985	3018	(132 %)	33
b.45.23	45	10776	706	**21**	-	**(16 %)**	1225234	3470	(242 %)	2
b.45.34	45	18348	898	**137**	-	**(105 %)**	38513	3604	(375 %)	1
b.45.45	45	21993	969	**187**	-	**(153 %)**	25964	3559	(624 %)	1
b.50.06	50	882	230	**19**	1230	**(9 %)**	49490829	2321	(35 %)	1225
b.50.13	50	4457	506	**8**	-	**(5 %)**	12039566	3131	(192 %)	7
b.50.25	50	14412	866	**676**	-	**(247 %)**	684010	3511	(280 %)	1
b.50.38	50	25446	1118	**242**	-	**(163 %)**	309289	3646	(505 %)	1
b.50.50	50	30271	1202	**360**	-	**(305 %)**	49507	3541	(729 %)	1
b.55.06	55	977	255	**21**	-	**(11 %)**	23603952	2323	(54 %)	6
b.55.14	55	5790	607	**11**	-	**(7 %)**	7829649	3186	(373 %)	6
b.55.28	55	19897	1069	**174**	-	**(106 %)**	580827	3553	(646 %)	2
b.55.41	55	33318	1347	**330**	-	**(244 %)**	117912	3654	(639 %)	1
b.55.55	55	40402	1459	**547**	-	**(493 %)**	117027	3575	(705 %)	1
b.60.08	60	2036	384	**12**	-	**(9 %)**	24800852	2712	(175 %)	1
b.60.15	60	7294	716	**16**	-	**(14 %)**	4044387	3236	(404 %)	1
b.60.30	60	25230	1264	**256**	-	**(165 %)**	295197	3578	(471 %)	1
b.60.45	60	43689	1614	**547**	-	**(439 %)**	26955	704	(671 %)	1
b.60.60	60	52575	1742	**784**	-	**(704 %)**	23716	3604	(762 %)	1

Table 2: *Results of* PQCR *and* Baron *for the 45 instances of the LABS problem. Time limit 5 hours. - means that the time limit of 3h on the SDP phase is reached.*

We present in Table 2 a detailed comparison of PQCR with the direct submission to baron 17.1.4. For these experiences the total time limit was set to 5 hours, and we limit the CPU time for solving (SDP) to 3 hours. Indeed, any feasible solution to the dual of (SDP) can be used to get a convex objective function in the equivalent formulation. Thus, if the CPU time in column $tSdp$ is smaller than three hours it means that (SDP) was solved to optimality. In the other case, we get a feasible dual solution and we can suppose that the initial gap of PQCR could be improved. For these instances PQCR is faster than baron since it solves 17 instances out of 45 within the time limit while baron solves only 13 instances. Here, baron 17.1.4 solves 2 instances that were stated as unsolved on minlplib. As expected, PQCR has an initial gap much smaller that baron (reduced by a factor 22 on average). We also observe that the number of nodes visited by PQCR during the branch-and-bound is significantly larger than the the number of nodes of baron (increased by a factor of about 40000 on average).

We present in Table 3 the values of the best solutions and of the final lower bounds obtained by PQCR within 5 hours of CPU time, and those available on the minlplib website. More precisely, we report in the column minlplib the best solution/final lower bound value obtained among the results of the solvers Antigone, Baron, Couenne, Lindo, and Scip. PQCR solves to optimality 6 unsolved instances (labeled as **). It also improves the best known solution values of 9 instances (labeled as $^{\#}$), and improves the final lower bound of all the unsolved instances (labeled as *). In this table, each line corresponds to one instance, and we only present results for instances that were stated as unsolved on minlplib. To illustrate these results, we plot in Figure 1, for each instance reported in Table 3, the *final gap* of PQCR and minlplib. Clearly, the final gap of PQCR is much smaller than the final gap of minlplib (reduced by a factor 3 on average).

A last remark concerns the CPU time necessary to solve (SDP). Indeed, this time represents on average 75% of the total CPU time. A natural improvement is to identify the set of "important equalities" in a preprocessing step in order to improve the behavior of the solution of (SDP). Obviously, this step should be dependent on the quadratization.

4.3 A short discussion on the impact of the chosen quadratization

In this section, we shortly explore the impact of the chosen quadratization on the tightness of the associated continuous relaxation bound. In Table 4, we report the continuous relaxation bound values obtained by convexification after applying the quadratization of Algorithm 2, and three quadratizations from [41], namely Pairwise Cover 1, 2 and 3 (PC1, PC2 and PC3). In Pairwise Cover 1, for each monomial of degree $d \geq 3$, the first two variables are linearized to obtain a monomial of degree $d-1$. The process is recursively reproduced until $d = 2$. Pairwise Covers 2 and 3 try to minimize the number of additional variables. In PC2, the authors compute the sub-monomials of any degree that appear the most among all the intersection of pairs of monomials. Then they linearize

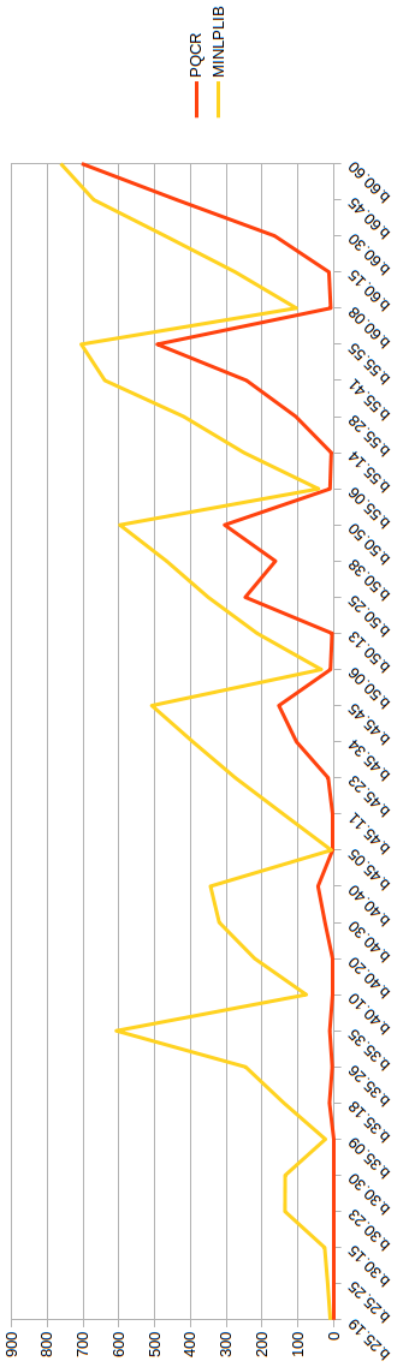

Fig. 1: *Comparison between the final gap of* `PQCR` *and the final gap computed with the best known solution and the best bound from* `minlplib` *for the unsolved auto-correlation instances*

these sub-monomials using the set $\mathcal{F}_\mathcal{E}$ and they repeat the process until the objective function is quadratic. PC3 linearizes in priority the pair of variables that occurs the most frequently in all the monomials. For instance, if we consider the quadratization of the following monomial of degree 4, $x_1x_2x_3x_4$, we will compute the most frequent pair of variables among the six possible products. If x_1x_2 is the most frequent, then the monomial will be quadratized using two variables, one for the reformulation of x_1x_2 and the other for x_3x_4.

We observe that the chosen quadratization impacts N, the number of variables of (QP). It also impacts the quality of the associated semidefinite bound, LB_i. Indeed, the more variables are added, the more the size of sets $\mathcal{F}_\mathcal{E}$ and $\mathcal{S}_\mathcal{E}$ increases. Clearly, some equalities of $\mathcal{S}_\mathcal{E}$ may be stronger than others. Interesting future research directions would be to identify, for a given quadratization, a set of "important" equalities in $\mathcal{S}_\mathcal{E}$, and to determine which quadratization used in `PQCR` leads to faster solution time and/or sharper initial lower bound.

5 Conclusion

We consider the general problem (P) of minimizing a polynomial function where the variables are binary. In this paper, we present `PQCR` a solution approach for (P). `PQCR` can be split in 3 phases. We called the first phase *quadratization*, where we rewrite (P) as an equivalent quadratic program (QP). For this we have to add new variables and linear constraints. We get a linearly constrained quadratic program that still has a non-convex objective function and binary variables. Moreover, even for small instances of (P), the existing convexification methods failed to solve the associate (QP). This is why, we present a family of tailored quadratic convex reformulations of (QP) that exploits its specific structure. For this, we introduce new valid quadratic equalities that vanish on the feasible domain of (QP). We use these equalities to build a family of equivalent quadratic convex formulations to (QP). Then, we focus on finding, within this family, the equivalent convex formulation that maximizes the continuous relaxation bound value. We show that we can compute this "best" convex reformulation using a new semidefinite relaxation of (QP). Finally, we solve our optimal reformulation with a standard solver.

We present computational results on two applications and compare our algorithm with other convexification methods and the general solver `Baron`. In particular, we show that for the *low auto-correlation binary sequence problem*, `PQCR` is able to improve the best known solution of 10 instances out of 45. A future research direction would be to characterize which quadratization best fit with our convexification phase from the continuous relaxation value point of view.

Acknowledgment

The authors are thankful to Elisabeth Rodriguez-Heck and Yves Crama for sharing the executable of their quadratization code in order to compute the Pairwise Covers 1, 2 and 3 on `LABS` instances.

Instance	PQCR (5h)		minlplib [35]	
Name	*Solution*	LB_f	*Solution*	LB_f
b.25.19**	-14644	-14644	-14644	-16108
b.25.25**	-10664	-10664	-10664	-12494
b.30.15**	-15744	-15744	-15744	-19780
b.30.23**	-30460	-30460	-30420	-72030
b.30.30**	-22888	-22888	-22888	-54014
b.35.09**	-5108	-5108	-5108	-6312
b.35.18*	-31144	-34964	-31160	-74586
b.35.26#*	-55288	-57789	-55184	-191466
b.35.35*	-41052	-45787	-41068	-290424
b.40.10#*	-8248	-8551	-8240	-14618
b.40.20#*	-50576	-52465	-50516	-162365
b.40.30#*	-94872	-118324	-94768	-398617
b.40.40*	-67528	-98031	-67964	-302028
b.45.05*	-1068	-1112	-1068	-1145
b.45.11#*	-12748	-13035	-12740	-30771
b.45.23#*	-85423	-98984	-85248	-320397
b.45.34*	-151352	-311627	-152368	-752427
b.45.45*	-111292	-285811	-112764	-685911
b.50.06*	-2160	-2363	-2160	-2921
b.50.13#*	-23791	-24975	-23772	-74768
b.50.25*	-124572	-433247	-124748	-562446
b.50.38*	-232344	-611906	-232496	-1318325
b.50.50*	-162640	-681105	-168216	-1173058
b.55.06*	-2400	-2659	-2400	-3439
b.55.14#*	-33272	-35698	-33168	-116748
b.55.28*	-189896	-392929	-190472	-989145
b.55.41*	-335388	-1160180	-337388	-2494477
b.55.55*	-233648	-1434663	-241912	-1947633
b.60.08*	-6792	-7388	-6792	-13915
b.60.15#*	-45232	-51467	-44896	-169767
b.60.30*	-259271	-692721	-261048	-1491016
b.60.45*	-475504	-2579935	-478528	-3687344
b.60.60*	-343400	-2816441	-350312	-3021077

Table 3: *Comparison of the best known solution and best lower bound values of PQCR and of the minlplib for the unsolved LABS instances. **: solved for the first time, #: best known solution improved, and *: best known lower bound improved*

		$PC1$		$PC2$		$PC3$		$Quad$	
	Opt	N	LB_i	N	LB_i	N	LB_i	N	LB_i
`b.20.05`	-416	64	-435	56	-439	40	-436	65	-435
`b.20.10`	-2936	123	-3052	135	-3115	93	-3068	124	-3051
`b.40.10`	-8248	303	-8590	315	-8659	262	-8745	304	-8589

Table 4: *Comparison of bounds and number of variables of* `PQCR` *after different quadratizations*

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 nlps—i. theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.
3. A. A. Ahmadi and A. Majumdar. Dsos and sdsos optimization: Lp and socp-based alternatives to sum of squares optimization. In *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, March 2014.
4. M. Anthony, E. Boros, Y. Crama, and A. Gruber. Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming*, 162:115–144, 2017.
5. B. Balasundaram and A.0. Prokopyev. On characterization of maximal independent sets via quadratic optimization. 19, 06 2011.
6. 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.
7. J. Bernasconi. Low autocorrelation binary sequences: statistical mechanics and configuration space analysis. *J. Physique*, 141(48):559–567, 1987.
8. A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1):55–68, 2007.
9. A. Billionnet, S. Elloumi, and A. Lambert. Exact quadratic convex reformulations of mixed-integer quadratically constrained problems. *Mathematical Programming*, 158(1):235–266, 2016.
10. 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.
11. A. Billionnet, S. Elloumi, and M. C. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Applied Mathematics*, 157(6):1185 – 1197, 2009. Reformulation Techniques and Mathematical Programming.
12. B. Borchers. CSDP, A C Library for Semidefinite Programming. *Optimization Methods and Software*, 11(1):613–623, 1999.
13. E. Boros, P.L. Hammer, and X. Sun. Network flows and minimization of quadratic pseudo-boolean functions. Technical Report TR: 1991-17, RUTCOR, 1991.
14. C. Buchheim and C. D'Ambrosio. Monomial-wise optimal separable underestimators for mixed-integer polynomial optimization. *Journal of Global Optimization*, pages 1–28, 2016.
15. C. Buchheim and G. Rinaldi. Efficient reduction of polynomial zero-one optimization to the quadratic case. *SIAM Journal on Optimization*, 18(4):1398–1413, 2007.
16. M.W. Carter. The indefinite zero-one quadratic problem. *Discrete Applied Mathematics*, pages 23–44, 1984.
17. Y. Crama and E. Rodriguez-Heck. A class of valid inequalities for multilinear 0-1 optimization problems. *Discrete Optimization*, pages 28–47, 2017.
18. R. Fortet. L'algèbre de Boole et ses Applications en Recherche Opérationnelle. *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, 4:5–36, 1959.
19. R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* The Scientific Press (now an imprint of Boyd & Fraser Publishing Co.), Danvers, MA, USA, 1993.

20. 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.
21. B. Ghaddar, J. C. Vera, and M. F. Anjos. A dynamic inequality generation scheme for polynomial programming. *Mathematical Programming*, 156(1):21–57, Mar 2016.
22. P.L. Hammer and A.A. Rubin. Some remarks on quadratic programming with 0-1 variables. *Revue Française d'Informatique et de Recherche Opérationnelle*, 4:67–79, 1970.
23. C. Helmberg. *Conic Bundle v0.3.10*, 2011.
24. 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.
25. N. Ito, S. Kim, and M. Kojima nad A.Takeda K.C. Toh. BBCPOP: A Sparse Doubly Nonnegative Relaxation of Polynomial Optimization Problems with Binary, Box and Complementarity Constraints. *ArXiv e-prints*, April 2018.
26. R.M. Karp. Reducibility among combinatorial problems. pages 85–103, 1972.
27. J. Krarup and P.M. Pruzan. Computer-aided layout design. pages 75–94, 1978.
28. X. Kuang, , B. Ghaddar, J . Naoum-Sawaya, and L.F. Zuluaga. Alternative SDP and SOCP Approximations for Polynomial Optimization. *ArXiv e-prints*, October 2015.
29. J.B. Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge University Press, Cambridge, 2015.
30. J.B. Lasserre and T.P. Thanh. Convex underestimators of polynomials. *Journal of Global Optimization*, pages 1–25, 2013.
31. J.D. Laughhunn. Quadratic binary programming with applications to capital budgeting problems. 18:454–461, 06 1970.
32. C. Lemarechal and F. Oustry. Semidefinite relaxations and lagrangian duality with application to combinatorial optimization. Technical report, RR-3710, INRIA Rhones-Alpes, 1999.
33. F. Liers, E. Marinari, U. Pagacz, F. Ricci-Tersenghi, and V. Schmitz. A non-disordered glassy model with a tunable interaction range. *Journal of Statistical Mechanics: Theory and Experiment*, page L05003, 2010.
34. R.D. McBride and J.S. Yormark. An implicit enumeration algorithm for quadratic integer programming. *Management Science*, 1980.
35. MINLPLIB. Library of mixed integer non linear programs. "`http://www.gamsworld.org/minlp/minlplib.htm`", 2012.
36. 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.
37. P.M Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, Apr 1994.
38. POLIP. Library for polynomially constrained mixed-integer programming. "`http://polip.zib.de/`", 2014.
39. M.R. Rao. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, 66(335):622–626, 1971.
40. J.M.W. Rhys. A selection problem of shared fixed costs and network flows. *Management Science*, 17(3):200–207, 1970.
41. E. Rodriguez-Heck. *Linear and quadratic reformulations of nonlinear optimization problems in binary variables*. PhD thesis, Université de Liège, 2018.
42. N.V. Sahinidis and M. Tawarmalani. Baron 9.0.4: Global optimization of mixed-integer nonlinear programs. *User's Manual*, 2010.

43. H.D. Sherali and C.H. Tuncbilek. A global optimization algorithm for polynomial programming using a reformulation-linearization technique. *Journal of Global Optimization*, 2:101–112, 1992.