PREPRINT
DECOMPOSITION-BASED REFORMULATION OF NONSEPARABLE QUADRATIC EXPRESSIONS IN CONVEX MINLP

Joakim Blomqvist
Department of Mathematics
Åbo Akademi University
Turku, Finland
joakim.blomqvist@abo.fi

Erik Tamm
Department of Mathematics
KTH Royal Institute of Technology
Stockholm, Sweden
etamm@kth.se

Jan Kronqvist
Department of Mathematics
KTH Royal Institute of Technology
Stockholm, Sweden
jankr@kth.se

Andreas Lundell
Department of Information Technology
Åbo Akademi University
Vaasa, Finland
andreas.lundell@abo.fi

ABSTRACT

In this paper, we present a reformulation technique for convex mixed-integer nonlinear programming (MINLP) problems with nonseparable quadratic terms. For each convex non-diagonal matrix that defines quadratic expressions in the problem, we show that an eigenvalue or LDLT decomposition can be performed to transform the quadratic expressions into convex additively separable constraints. The reformulated problem type is more suitable for a polyhedral outer approximation (POA)-based MINLP solver such as the Supporting Hyperplane Optimization Toolkit (SHOT). Results from computational experiments on MINLPLib instances, QPLib instances, and generated test instances indicate enhanced performance with these reformulations on convex problems with dense quadratic structures when SHOT utilizes MIP subsolvers without quadratic support, such as Cbc and HiGHS.

Keywords Convex mixed-integer nonlinear programming (MINLP) · Nonseparable quadratic functions · Lifting Reformulations · Eigendecomposition · LDLT Decomposition · Supporting Hyperplane Optimization Toolkit (SHOT)

1 Introduction

The SHOT solver was initially released as an open-source solver for convex mixed-integer nonlinear programming (MINLP) problems [21]. Later, it was extended to support certain classes of nonconvex problems [20]. For convex minimization problems, the solver uses a *primal-dual* search strategy: upper bounds for the objective value are obtained through primal heuristics, and lower bounds are obtained by minimizing the objective over a polyhedral outer approximation (POA), which is iteratively refined by the extended supporting hyperplane (ESH) algorithm [16] or the extended cutting plane (ECP) algorithm [30].

It is well-known that a convex nonlinear constraint becomes increasingly more difficult to outer approximate by a polyhedron with an increasing number of nonlinear variables in the constraint [14]. If the nonlinear functions are separable, then it is possible to obtain significantly tighter polyhedral outer approximations by first performing a lifted reformulation [14, 17, 27]. These reformulations disaggregate a separable nonlinear constraint into multiple constraints

with a single nonlinear term as follows:

$$\sum_{i=1}^{n} g_i(x_i) \leq 0 \implies \begin{cases} \sum_{i=1}^{n} \mu_{g,i} \leq 0 \\ g_i(x_i) \leq \mu_{g,i}, \quad i = 1, \ldots, n, \end{cases} \tag{1}$$

where $\mu_{g,i} \in \mathbb{R}$ are auxiliary variables. Remarkable computational improvements were reported in [14, 17], with reductions in both the number of iterations and the solution times by orders of magnitude. Note that a similar lifted reformulation can be applied to a separable convex nonlinear objective function, where each term in the objective is replaced by an auxiliary variable and then moved into separate constraints as:

$$\text{minimize} \quad \sum_{i=1}^{n} f_i(x_i) \implies \begin{cases} \text{minimize} & \sum_{i=1}^{n} \mu_{f,i} \\ \text{subject to} & f_i(x_i) \leq \mu_{f,i}, \quad i = 1, \ldots, n, \end{cases} \tag{2}$$

where $\mu_{f,i} \in \mathbb{R}$ are auxiliary variables.

An obvious limitation of the reformulations in eqs (1) and (2) is that they are only applicable to additively separable functions. However, the framework does apply more generally. As described in [17], full separability is not necessary, but the advantages diminish with the number of variables linked together by nonlinear expressions. Already for a quadratic constraint with one off-diagonal element on each row of the Hessian, it is not possible to use the reformulation in eq. (1) to break up the constraint into simpler convex constraints. For simplicity, we will refer to quadratic functions with off-diagonal elements as *nonseparable*.

In this paper, we consider convex MINLP problems of the form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_1 \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{a}, \quad \mathbf{B}\mathbf{x} = \mathbf{b}, \\ & g_k(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_{k+1} \mathbf{x} \leq 0, \quad k \in K_g = \{1, 2, \ldots, m_g\}, \\ & h_k(\mathbf{x}) \leq 0, \quad k \in K_h = \{1, 2, \ldots, m_h\}, \\ & \underline{x}_i \leq x_i \leq \overline{x}_i \quad \forall i \in I = \{1, 2, \ldots, n\}, \\ & x_i \in \mathbb{R}, x_j \in \mathbb{Z} \quad \forall i \in I \setminus I_Z, \forall j \in I_Z, \end{aligned} \tag{3}$$

where I_Z denotes the index set corresponding to integer variables, $g_k(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_{k+1}\mathbf{x} \leq 0$ represents the nonlinear convex constraints with some quadratic structure, and $h_k(\mathbf{x}) \leq 0$ represents the general nonlinear convex constraints. Furthermore, it is assumed that all $\mathbf{Q}$ are $n \times n$ symmetric positive semidefinite matrices and $f(\mathbf{x})$, $g_k(\mathbf{x})$, and $h_k(\mathbf{x})$ are convex and at least once differentiable.

Continuing the discussion, when $\mathbf{Q}$ has only a few off-diagonal elements, meaning that the coupling between variables is limited, it could still be feasible to break up the original functions into simpler convex constraints. However, this does not result in full separability, as some variables remain coupled through the remaining off-diagonal elements. To address the limitation, we employ either eigendecomposition (EVD)- or LDLT decomposition-based reformulations to transform the problem into an equivalent problem in separable form. Note that, by equivalent, we mean that the problems share the same solutions in the $\mathbf{x}$-space and yield the same objective values for any given value for $\mathbf{x}$.

The transformation into separable form and the consecutive reformulations into constraints with a single quadratic term, by the reformulations in eqs (1) and (2), have been integrated into SHOT's automatic reformulation functionality. The reformulations are mainly intended for situations where only linear programming (LP)/mixed-integer linear programming (MILP) subproblems are allowed in SHOT's dual bounding strategy, *i.e.*, with subsolvers that do not directly support quadratic terms. When using mixed-integer quadratic programming (MIQP) or mixed-integer quadratically constrained quadratic programming (MIQCQP) subproblems, it can be more efficient to let the subsolver choose how to deal with the quadratic expressions.

Although SHOT is able to utilize commercial subsolvers, *e.g.*, Gurobi, a central motivation behind SHOT is to provide a competitive non-commercial solver framework. Currently, for problems containing convex nonseparable quadratic expressions, performance can degrade significantly as the supported non-commercial subsolvers in SHOT (HiGHS and Cbc) do not have specialized handling of quadratic expressions. This means that SHOT always solves MILP subproblems instead of MIQP or MIQCQP subproblems. Thus, the goal of this work is to improve the computational performance of SHOT when paired with Cbc and HiGHS. Accordingly, the focus is not on benchmarking the overall efficiency of SHOT, but rather on demonstrating and evaluating the impact of the proposed techniques when using non-commercial MILP subsolvers. It is worth noting that SCIP could in principle be used as an open-source subsolver within

SHOT and would provide native support for convex nonseparable quadratic expressions. However, this configuration is not currently available in SHOT (work in progress) and, more importantly, would make the motivation for the reformulation proposed here less relevant. The decomposition-based reformulation is specifically designed to transform nonseparable quadratic terms into a form that can be linearized within a POA-based framework more efficiently. While the reformulation's applicability is not necessarily limited to this MILP setting, we nonetheless restrict our attention to this case.

Note that a preliminary and partial version of this work, introducing the basic idea of using EVD-based reformulations within SHOT, was previously presented as an extended conference abstract [15]. The current paper extends these results with a theoretical background, an expanded reformulation technique, and computational experiments. The main contributions of this paper can be summarized as follows:

i) A decomposition-based reformulation framework (EVD- and LDLT-based) that transforms nonseparable convex quadratic terms into separable form, applicable to both objectives and constraints.

ii) An LDLT-based reformulation that, at the same auxiliary-variable and constraint count, typically offers sparser factorizations and improved numerical behavior compared to EVD.

iii) Implementation of both reformulations within SHOT's automatic reformulation framework.

iv) Computational experiments across 134 instances: 67 base instances, each in original and permuted form. These include 27 instances from MINLPLib [22], 7 from QPLib [9], and 33 newly generated instances.

In addition to the contributions listed above, this paper also includes strengthened cuts for the LDLT reformulation that exploit discreteness induced by binary variables. In particular, we extend the integer-based secant cuts [2] by propagating discreteness through the decomposition's linear equation system arising from the decomposition. This contribution is purely theoretical at the moment, as the cuts are not implemented in SHOT or evaluated computationally in this paper. We include this result as a theoretical extension of the reformulation framework rather than as one of the main computational contributions.

The remainder of this paper is organized as follows: similar and related work on lifted reformulations and decomposition approaches for quadratic problems are discussed in Section 2. In Section 3, the proposed EVD- and LDLT-based reformulations, highlighting their theoretical basis and practical implications, are introduced. This section also presents the theoretically strengthened cuts for the LDLT reformulation. Section 4 presents computational experiments that assess the performance of the reformulations when integrated into SHOT; this includes details on the implementation and test instances. Finally, Section 5 concludes with a discussion of key findings and directions for future research.

2 Background and Related Work

MINLP is a broad and challenging optimization field with applications in engineering, finance, and logistics. A common strategy in MINLP is the use of reformulations that rewrite nonlinear expressions into equivalent forms that are easier for solvers to handle. One of the most important subclasses of MINLP consists of problems involving quadratic terms; quadratic expressions often arise in industrial applications, *e.g.*, in portfolio optimization in finance. The presence of nonseparable quadratic terms can substantially increase problem complexity and computational effort.

Several approaches exist to handle nonseparable quadratic functions in mixed-integer optimization. In this section, we focus on some particular approaches that are closely related to the methods we propose. These approaches are discussed because they

i) address nonseparable quadratic terms or general convex constraints,

ii) provide general techniques that can be adapted to separable reformulations, or

iii) have a direct conceptual or practical connection to our work, such as lifting quadratic objectives or constraints, and supporting automated solver integration.

Although other methods have been developed for similar problems, we limit our discussion to the ones mentioned above as they offer the most immediate insights and transferable techniques relevant to our proposed framework.

2.1 Eigenvalue and Gauss Decomposition for Separable MIQP Reformulations

A similar approach to the one proposed in this paper is to transform a quadratic objective into a separable form by using matrix decompositions such as eigenvalue [6] or Gauss factorizations [23] of the Hessian matrix. These methods also introduce additional variables and constraints into the problem in a similar way to the method we propose.

In [23], the authors consider MIQP and reformulate the original problem by using Gauss decomposition into an equivalent problem with a separable objective function. The authors also present the standard technique of utilizing eigenvalue decomposition from [6], but suggest that using Gauss decomposition of the quadratic terms is preferable, as eigenvalues often are irrational and this may lead to the reformulated problem not being equivalent to the original one (rounding errors in solver implementation). In contrast to their work, which focuses only on quadratic objectives, our reformulations target both quadratic objectives and constraints.

The strategies from [6] and [23] come with advantages and disadvantages. In particular, they do not directly generalize to quadratic constraints, and although Gauss decomposition is exact, it can be computationally demanding and cumbersome to implement. The Gauss decomposition for a $n \times n$ matrix $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \mathbf{UDL}, \tag{4}$$

where $\mathbf{U}$ is an upper triangular unidiagonal matrix, $\mathbf{D}$ is a diagonal matrix and $\mathbf{L}$ is a lower triangular unidiagonal matrix. Obtaining this decomposition requires computing all three matrices. In contrast, the eigenvalue decomposition expresses $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{PDP}^\top, \tag{5}$$

where $\mathbf{P}$ is an orthogonal matrix and $\mathbf{D}$ is a diagonal matrix. Note that since $\mathbf{P}^\top$ is directly derived from $\mathbf{P}$, only two matrices need to be computed. From a computational perspective, reformulations that rely on fewer matrices should, in theory, lead to simpler implementations, reduce the computational effort, and thus make the quadratic decomposition more efficient. However, this relies heavily on the complexity of the decompositions. For example, for a general $n \times n$ matrix, Gauss decomposition requires approximately $2n^3/3$ floating-point operations, whereas computing the full eigenvalue decomposition requires approximately $9n^3$ [11, 7]. It is worth mentioning that the reformulation methods described in Section 3 only compute two different matrices.

The methods from [6] and [23] reformulate an MIQP problem into a discrete separable quadratic problem or a mixed-integer separable quadratic program, respectively. In [23], the reformulated problem is linearized by approximating each squared term by a set of K linear functions that correspond to the tangents of a hyperbola in K points. Our proposed method employs a similar lifting reformulation, applicable to both quadratic objectives and quadratic constraints, with the main difference being that the linearization is carried out via POA instead. For further details on the reformulation of nonseparable quadratic terms using eigenvalue and Gauss decompositions, we refer the reader to [23].

2.2 Lifted Polyhedral Relaxations

Another related class of methods for handling nonseparable expressions is based on a higher-dimensional, or lifted, polyhedral relaxation of quadratic constraints. In [28], the authors illustrate how lifted polyhedral relaxations and auxiliary variables can improve solver performance for problems with complex convex constraints; although they only consider mixed-integer conic quadratic programming, the underlying idea of utilizing lifted polyhedral relaxations and the resulting preprocessing capability is broadly applicable.

To describe their method briefly, the authors start by rewriting a conic constraint of the form

$$(\mathbf{x}, \mathbf{y}) \in \mathcal{CC} := \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+p} : ||\mathbf{Ax} + \mathbf{By} + \delta||_2 \leq \mathbf{ax} + \mathbf{by} + \delta_0\}, \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{r \times n}$, $\mathbf{B} \in \mathbb{R}^{r \times p}$, $\delta \in \mathbb{R}^r$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}^p$ and $\delta_0 \in \mathbb{R}$ for some $r \in \mathbb{Z}_+$, as

$$\begin{aligned} \mathcal{CC} = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+p} : \exists (z_0, \mathbf{z}) \in \mathbb{R}_+ \times \mathbb{R}^r \quad \text{s.t.} \quad & \mathbf{Ax} + \mathbf{By} + \delta = \mathbf{z}, \\ & \mathbf{ax} + \mathbf{by} + \delta_0 = z_0, \\ & (z_0, \mathbf{z}) \in \mathcal{L}^r\}. \end{aligned} \tag{7}$$

Here $\mathcal{L}^r$ is the $(r+1)$-dimensional Lorentz cone given by

$$\mathcal{L}^r := \{(z_0, \mathbf{z}) \in \mathbb{R}_+ \times \mathbb{R}^r : \sum_{k=1}^{r} z_k^2 \leq z_0. \tag{8}$$

The authors then rewrite $\mathcal{L}^r$ using $(r/2)$ three-dimensional Lorentz cones and one $(r/2+1)$-dimensional Lorentz cone

$$\begin{aligned} \mathcal{L}^r = \{(z_0, \mathbf{z}) \in \mathbb{R}_+ \times \mathbb{R}^r : \exists p \in \mathbb{R}^{r/2} \quad s.t. \quad & (\rho, z_o) \in \mathcal{L}^{r/2} \\ & (z_{2k-1}, z_{2k}, \rho_k) \in \mathcal{L}^2\} \end{aligned} \tag{9}$$

for $k = \{1, \ldots, r/2\}$, where they first assume that $r = 2^p$ for some $p \in \mathbb{Z}_+$ and group the variables $\mathbf{z}$ into $k = r/2$ pairs. A new variable ρ_k is introduced for each k-th pair. By applying the same recursive decomposition to the

$(r/2+1)$-dimensional Lorentz cone, they rewrite $\mathcal{L}^r$ with only $(r-2)$ 3-dimensional Lorentz cones and replace them with a polyhedral relaxation of $\mathcal{L}^2$. With this, a general r, and some careful selections of the parameters in the polyhedral relaxation, a polyhedral relaxation of $\mathcal{L}^r$ for a given tightness parameter ε can be obtained.

The approach in [28] can be seen as one of the precursors of what we refer to as *lift-and-separate* techniques used for handling general nonseparable convex constraints. Lift-and-separate techniques transform a problem into a higher-dimensional space (*i.e.*, lifts the problem) to isolate complex terms (*i.e.*, separates them) and handle them more easily, *cf.* [26], [18] and [29].

Other closely related techniques include the reformulation-linearization technique (RLT) [25, 24], lift-and-project cuts [3] and perspective reformulations [13, 8], which all rely on lifting the problem into a higher-dimensional space to obtain stronger relaxations. More specifically, RLT introduces auxiliary variables to represent nonlinear product terms and automatically generates valid inequalities, which are subsequently linearized to obtain a tighter relaxation; lift-and-project methods introduce auxiliary variables to construct a tighter formulation before projecting it back onto the original problem space, while perspective reformulations strengthen the formulation by replacing nonlinear expressions with their corresponding perspective functions. Note that some uses of these techniques may fall under the lift-and-separate category, depending on whether the resulting formulation is separable or not.

The quadratic decomposition and reformulation technique presented in Section 3 adopts a philosophy similar to [28]. By transforming nonseparable quadratic terms into an equivalent separable form via eigendecomposition or LDLT decomposition, we introduce auxiliary variables that lift the original problem into a higher-dimensional space. This lifting enables a POA-based solver to generate linearizations of individual quadratic terms more effectively, in much the same way that [28] use lifted representations to rewrite conic constraints.

2.3 Extended Conic Formulations

Another example of this lift-and-separate technique is the usage of extended formulations in a higher-dimensional space to strengthen the polyhedral approximation of general convex constraints in mixed-integer convex optimization. For instance, [19] demonstrates how extended formulations, automated tools, and mixed-integer disciplined convex programming (MIDCP) can be used for systematically converting complex convex constraints into forms that are easier to approximate and solve with MILP subsolvers. This paper shares conceptual overlap with ours, as both leverage decomposition and lifting ideas.

In [19], the authors use conic constraints to represent convex constraints and introduce auxiliary variables to construct extended formulations. This process leverages the algebraic modeling techniques of MIDCP, which provides a framework for expressing convex problems in a conic form. These problems can then be solved by using a conic solver, such as Pajarito. Specifically, Pajarito uses the second-order cone

$$\text{SOC}_n = \{(\mathbf{t}, \mathbf{x}) \in \mathbb{R}^n : ||\mathbf{x}|| \leq \mathbf{t}\}, \tag{10}$$

the cone of positive semidefinite matrices

$$\text{PSD}_n = \{\mathbf{A} \in \mathbb{R}^{n \times n} : \mathbf{A} = \mathbf{A}^\top, \;\; \mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0 \forall \mathbf{x} \in \mathbb{R}^n\}, \tag{11}$$

the exponential cone

$$\text{EXP}_n = \{(\mathbf{x}, \mathbf{z}, \mathbf{y}) \in \mathbb{R}^3 : \mathbf{y} \exp(\mathbf{x}/\mathbf{y}) \leq \mathbf{z}, \;\; \mathbf{y} > 0\}, \tag{12}$$

and the power cone

$$\text{POW}_\alpha = \{(\mathbf{x}, \mathbf{z}, \mathbf{y}) \in \mathbb{R}^3 : |\mathbf{z}| \leq \mathbf{x}^\alpha \mathbf{y}^{1-\alpha}, \;\; \mathbf{x} \geq 0, \;\; \mathbf{y} \geq 0\}, \tag{13}$$

to represent different convex constraints from the original mixed-integer convex problem. Pajarito then generates polyhedral approximations of these cones in the form of MILP subproblems, which are passed to a MILP subsolver and solved as in SHOT.

Beyond the mathematical transformation, a key advantage of the approach described in [19] is the automated decomposition and lifting strategy incorporated into a solver workflow. In this paper, we adapt this idea specifically for convex quadratic terms, rather than providing a strategy that works for general MINLP problems as in Pajarito. Unlike manual reformulations, this automation allows SHOT to preprocess and linearize complex quadratic constraints without requiring user intervention, which is essential for solver efficiency. Furthermore, the lifted representation can expose sparsity patterns or structural decompositions, which can then be exploited by different functionalities in the solver to further reduce the computational effort. Note, however, that SHOT also supports other reformulations for nonquadratic separable convex functions [21].

In summary, the presented overview highlights the value of decomposition, higher-dimensional lifting, extended formulations, and automated solver integration for addressing nonseparable quadratic terms and strengthening polyhedral

approximations for convex MINLP problems. The mentioned existing approaches often target only quadratic objectives and do not apply to constraints or depend on specific conic representations that require efficient conic solvers; this limits their practical applicability when implemented in a general convex MINLP solver. Our work differs by avoiding the dependence on special subsolvers and providing simple reformulation techniques directly applicable to all quadratic terms. As a result, these methods are particularly suitable to be implemented in a POA-based MINLP solver.

3 Proposed Reformulations

Here, we briefly describe how the EVD- and LDLT-based reformulations are used to obtain a convex MINLP problem with separable constraints. Suppose we have a convex constraint of the form:

$$g(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} \leq 0, \tag{14}$$

where $\mathbf{Q}$ is an $n \times n$ symmetric positive semidefinite matrix. We make the following assumptions on constraint (14):

1. $\mathbf{Q}$ has several off-diagonal elements, making a direct disaggregation of the constraint into convex constraints difficult/impossible.
2. The function g is affine or nonlinear in only a few variables, *i.e.*, the difficulty in disaggregating the constraint comes from the quadratic term.

Then, we separate the quadratic term by introducing an auxiliary variable $\alpha \in \mathbb{R}$ to write the constraint as:

$$\begin{aligned} g(\mathbf{x}) + \alpha &\leq 0, \\ \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} &\leq \alpha. \end{aligned} \tag{15}$$

Note that the same approach applies to objectives, as shown in eq. (2). However, for simplicity, we will focus on the constraint case from now on. The next subsections present two strategies to handle the quadratic term in this form. Both approaches yield separable constraints that can be efficiently incorporated into a convex MINLP problem.

3.1 Eigendecomposition-Based Reformulation

In order to handle the quadratic term, we can perform an EVD to factorize the matrix $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{V}\mathbf{D}\mathbf{V}^\top, \tag{16}$$

where $\mathbf{D}$ is a diagonal matrix, with the eigenvalues of $\mathbf{Q}$ on the diagonal, and the columns of $\mathbf{V}$ contain the eigenvectors. To transform the constraint into a separable form, we introduce the auxiliary variables $\mathbf{y} \in \mathbb{R}^n$ defined as

$$\mathbf{y} = \mathbf{V}^\top \mathbf{x}. \tag{17}$$

Since $\mathbf{x}$ is bounded, the bounds on $\mathbf{y}$ can be directly derived from the linear relation in eq. (17) by, *e.g.*, solving an optimality-based bound tightening problem or by utilizing interval arithmetic. Continuing the discussion, by introducing the auxiliary variables $\mathbf{y}$, we can represent the quadratic term in constraint (14) in an equivalent separable form by the constraints

$$\begin{aligned} g(\mathbf{x}) + \alpha &\leq 0, \\ \frac{1}{2}\mathbf{y}^\top \mathbf{D}\mathbf{y} &\leq \alpha, \\ \mathbf{V}^\top \mathbf{x} &= \mathbf{y}. \end{aligned} \tag{18}$$

For any variable combination of $(\mathbf{x}, \mathbf{y}, \alpha)$ that satisfies the constraint in eq. (18), the $\mathbf{x}$-variables will also satisfy constraint (14), as

$$\frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} = \frac{1}{2}\mathbf{x}^\top \mathbf{V}\mathbf{D}\mathbf{V}^\top \mathbf{x} = \frac{1}{2}(\mathbf{V}^\top \mathbf{x})^\top \mathbf{D}(\mathbf{V}^\top \mathbf{x}) = \frac{1}{2}\mathbf{y}^\top \mathbf{D}\mathbf{y}.$$

Since $\mathbf{D}$ is a diagonal matrix, we can now disaggregate the quadratic constraint according to the reformulation in eq. (1). The combined reformulation introduces $2n + 1$ continuous auxiliary variables, where the bounds for μ can be further tightened from those of $\mathbf{x}$, $\mathbf{y}$, and α, and results in $2n + 2$ constraints. The same reformulation procedure can be performed independently on multiple constraints (and the objective function) of the convex MINLP problem. The reformulation procedure can result in significantly tighter polyhedral outer approximations and has the potential to greatly reduce the number of iterations needed to solve problems. Next, we illustrate the EVD-based approach to reformulate a quadratic constraint.

Example 1. *Illustration of the EVD-based reformulation. Consider the quadratic constraint*

$$g(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} = x_4 + 2x_1 + \frac{1}{2}\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq 0, \tag{19}$$

where $(x_1, x_2, x_3) \in \{0,1\}^3$ *and* $-5 \leq x_4 \leq 0$.

First, we perform an eigendecomposition to factorize the matrix $\mathbf{Q}$*:*

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{\sqrt{2}} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2+\sqrt{2} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2-\sqrt{2} \end{bmatrix} \begin{bmatrix} -\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \end{bmatrix}.$$

Next, we introduce the auxiliary variables $\mathbf{y}$*, defined as in eq.* (17)*. Then the quadratic constraint* (19) *can be reformulated in an equivalent separable form by the constraints*

$$\begin{aligned} x_4 + 2x_1 + \alpha &\leq 0, \\ \frac{1}{2}\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 2+\sqrt{2} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2-\sqrt{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} &\leq \alpha, \\ \begin{bmatrix} -\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} &= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \end{aligned} \tag{20}$$

Now, each quadratic term is separable, which allows for disaggregating the quadratic constraint according to the reformulation in eq. (1)*. The combined reformulation will introduce* 7 *continuous auxiliary variables and* 8 *constraints. To concretize, eq.* (20) *can be further reformulated into*

$$\begin{aligned} x_4 + 2x_1 + \alpha &\leq 0, \\ \mu_1 + \mu_2 + \mu_3 &\leq \alpha, \\ (1 + \frac{\sqrt{2}}{2})y_1^2 &\leq \mu_1, \\ y_2^2 &\leq \mu_2, \\ (1 - \frac{\sqrt{2}}{2})y_3^2 &\leq \mu_3, \\ \begin{bmatrix} -\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} &= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \end{aligned} \tag{21}$$

where $0 \leq \alpha \leq 5$, $-\frac{1}{2} \leq y_1 \leq \frac{\sqrt{2}+1}{2}$, $0 \leq y_2 \leq \sqrt{2}$, $-\frac{\sqrt{2}+1}{2} \leq y_3 \leq \frac{1}{2}$, $0 \leq \mu_1 \leq 5$, $0 \leq \mu_2 \leq 5$ *and* $0 \leq \mu_3 \leq 5$.

The EVD reformulation can introduce numerical challenges in solving the linear relaxation, as the matrix $\mathbf{V}$ is usually dense with a relatively large range of coefficients. Any similar decomposition of $\mathbf{Q}$ is theoretically valid, and a sparse $\mathbf{V}$ should be computationally favorable. Furthermore, since the eigenvalues of $\mathbf{Q}$ are often irrational, the reformulated problem may not be perfectly equivalent to the original one due to rounding and finite-precision errors.

3.2 LDLT Decomposition-Based Reformulation

Building on the observation in the previous section, we will now explore an LDLT (also known as LDL) decomposition-based reformulation, which can result in a sparser factorization and should therefore offer better computational properties.

The LDLT decomposition factorizes the matrix $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{L}\mathbf{D}\mathbf{L}^\top, \tag{22}$$

where $\mathbf{L}$ is a lower unit triangular matrix and $\mathbf{D}$ is a diagonal matrix with non-negative entries. Applying the same strategy as in the EVD, we can rewrite the quadratic term in constraint (14) in an equivalent separable form

$$\begin{aligned} g(\mathbf{x}) + \alpha &\leq 0, \\ \frac{1}{2}\mathbf{y}^\top \mathbf{D}\mathbf{y} &\leq \alpha, \\ \mathbf{L}^\top \mathbf{x} &= \mathbf{y}, \end{aligned} \tag{23}$$

where the bounds on $\mathbf{y}$ follow from the bounds on $\mathbf{x}$ and the linear relation in eq. (23), in a similar manner as mentioned before.

Just like in the EVD reformulation, the same procedure can be performed independently on multiple constraints (and the objective function) of the convex MINLP problem. The main advantage of the LDLT reformulation is that the lower unit triangular matrix $\mathbf{L}$ introduces sparsity, reducing the number of nonzero coefficients in the linear constraints. The reformulation based on the LDLT decomposition introduces $2n+1$ continuous auxiliary variables and $2n+2$ constraints, the same as for the EVD. However, due to the typically sparser structure of $\mathbf{L}$ compared to $\mathbf{V}$, the resulting linear relaxations are often easier to solve and can lead to faster convergence in convex MINLP solvers. Another important aspect is that the computational cost of LDLT decomposition is about half the cost of a generic Gauss decomposition, *i.e.*, $n^3/3$ [11]. Next, we illustrate the LDLT decomposition-based approach on the same constraint as in Example 1.

Example 2. *Illustration of the LDLT decomposition-based reformulation. Consider the quadratic constraint*

$$g(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} = x_4 + 2x_1 + \frac{1}{2}\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq 0,$$

where $(x_1, x_2, x_3) \in \{0,1\}^3$ *and* $-5 \leq x_4 \leq 0$.

First, we perform an LDLT decomposition to factorize the matrix $\mathbf{Q}$*:*

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & \frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & \frac{4}{3} \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix}.$$

Next, we introduce the auxiliary variables $\mathbf{y}$*, defined as in eq.* (23)*. Then the quadratic constraint can be reformulated in an equivalent separable form by the constraints*

$$\begin{aligned} x_4 + 2x_1 + \alpha &\leq 0, \\ \frac{1}{2}\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & \frac{4}{3} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} &\leq \alpha, \\ \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} &= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \end{aligned} \tag{24}$$

Now, each quadratic term is separable, which allows for disaggregating the quadratic constraint according to the reformulation in eq. (1)*. The combined reformulation will introduce* 7 *continuous auxiliary variables and* 8 *constraints. To concretize, eq.* (24) *can be further reformulated into*

$$\begin{aligned} x_4 + 2x_1 + \alpha &\leq 0, \\ \mu_1 + \mu_2 + \mu_3 &\leq \alpha, \\ y_1^2 &\leq \mu_1, \\ \frac{3}{4}y_2^2 &\leq \mu_2, \\ \frac{2}{3}y_3^2 &\leq \mu_3, \\ \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} &= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \end{aligned} \tag{25}$$

where $0 \leq \alpha \leq 5$, $-\frac{1}{2} \leq y_1 \leq 1$, $0 \leq y_2 \leq \frac{5}{3}$, $0 \leq y_3 \leq 1$, $0 \leq \mu_1 \leq 5$, $0 \leq \mu_2 \leq 5$ *and* $0 \leq \mu_3 \leq 5$.

It can be noted that by using the LDLT decomposition, we obtain a sparser representation of the quadratic terms, potentially leading to more efficient linear relaxations. Compared to Example 1, we can also observe that the LDLT reformulation yields tighter bounds on the majority of the $\mathbf{y}$ *variables for this example, as illustrated in Table 1.*

Table 1: Comparison of variable bounds obtained from the EVD and LDLT reformulations.

	EVD		LDLT	
Variable	Exact	Approx.	Exact	Approx.
y_1	$[-\frac{1}{2}, \frac{\sqrt{2}+1}{2}]$	$[-0.5000, 1.2071]$	$[-\mathbf{\frac{1}{2}}, \mathbf{1}]$	$[\mathbf{-0.5000}, \mathbf{1.0000}]$
y_2	$[\mathbf{0}, \mathbf{\sqrt{2}}]$	$[\mathbf{0.0000}, \mathbf{1.4142}]$	$[0, \frac{5}{3}]$	$[0.0000, 1.6667]$
y_3	$[-\frac{\sqrt{2}+1}{2}, \frac{1}{2}]$	$[-1.2071, 0.5000]$	$[\mathbf{0}, \mathbf{1}]$	$[\mathbf{0.0000}, \mathbf{1.0000}]$

3.2.1 Strengthened Cuts in the LDLT decomposition

The LDLT decomposition-based reformulation can be further strengthened if binary variables are present. Assume for simplicity that $\mathbf{x}$ are all binary variables. If the problem contains integer or continuous variables, they can be reordered to place all binary variables first to get the required structure. The LDLT reformulation contains the constraint $\mathbf{y} = \mathbf{L}^\top \mathbf{x}$. Since $\mathbf{L}$ is a lower unit triangular matrix with elements $l_{i,j}$, the constraint can be written explicitly as

$$\begin{aligned} y_n &= x_n \\ y_{n-1} &= x_{n-1} + l_{n,n-1} x_n \\ &\vdots \\ y_1 &= x_1 + \sum_{k=2}^{n} l_{k,1} x_k. \end{aligned}$$

Since each x_k is binary, it is implied that each y_k can only achieve a discrete set of values. For example $y_n \in \{0, 1\}$ and $y_{n-1} \in \{0, 1, l_{n,n-1}, 1 + l_{n,n-1}\}$. For each y_k, we denote the discrete set of feasible values as Y_k. In theory, these sets can be explicitly calculated for all y_k, but if $\mathbf{L}$ is dense, this quickly becomes expensive (as enumerating all sets requires up to $2(2^n - 1)$ values in total). We can however, decide to only consider a few y_k such that it is possible to calculate Y_k, *e.g.*, those involving only a small number of binary variables (trivially starting with y_n). It is also possible to add the constraints derived below iteratively without explicitly calculating Y_k.

Consider now a single variable y_k. It is involved in the constraint $\mu_k \geq d_k y_k^2$ in the LDLT reformulation, as seen in eq. (25). Note that $d_k \geq 0$ since the matrix $\mathbf{Q}$ is positive semidefinite. The fact that $y_k \in Y_k$ can now be used to derive linear constraints strengthening this quadratic constraint. Let $a, b \in Y_k$ such that $Y_k \cap]a, b[= \emptyset$, where $]a, b[$ denotes the open interval (a, b). This simply means that $y_k \notin]a, b[$, as can be seen in Figure 1. Thus, the line connecting $(a, d_k a^2)$ and $(b, d_k b^2)$ will be a valid constraint, and we get

$$\mu_k \geq d_k(a + b)y_k - d_k ab. \tag{26}$$

The following example shows how the LDLT reformulation defined by eq. (25) can be strengthened using this technique.

Example 3. *The relationship between* (x_1, x_2, x_3) *and* (y_1, y_2, y_3) *is defined by the constraint*

$$\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}.$$

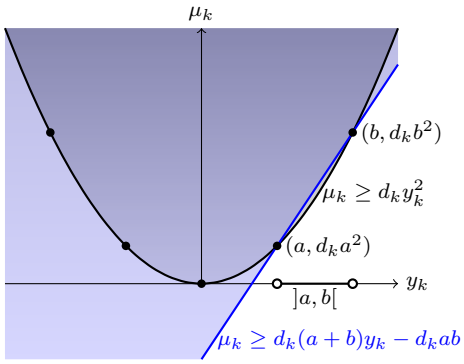


Figure 1: The quadratic constraint $\mu_k \geq d_k y_k^2$ is illustrated in gray, and the linear constraint obtained from two consecutive feasible values $a, b \in Y_k$ is shown in blue. Since $Y_k \cap]a, b[= \emptyset$, no feasible value of y_k lies in the open interval $]a, b[$, so the linear constraint $\mu_k \geq d_k(a+b)y_k - d_k ab$ defines a valid strengthening cut.

where $(x_1, x_2, x_3) \in \{0, 1\}^3$. *Thus, we get* $Y_1 = \{-\frac{1}{2}, 0, \frac{1}{2}, 1\}$, $Y_2 = \{0, \frac{2}{3}, 1, \frac{5}{3}\}$ *and* $Y_3 = \{0, 1\}$. *Using* (26)*, we can add the following inequalities to eq.* (25)*:*

$$\begin{aligned}
\mu_1 &\geq -\tfrac{1}{2} y_1 \\
\mu_1 &\geq \tfrac{1}{2} y_1 \\
\mu_1 &\geq \tfrac{3}{2} y_1 - \tfrac{1}{2} \\
\mu_2 &\geq \tfrac{1}{2} y_2 \\
\mu_2 &\geq \tfrac{5}{4} y_2 - \tfrac{1}{2} \\
\mu_2 &\geq 2 y_2 - \tfrac{5}{4} \\
\mu_3 &\geq \tfrac{2}{3} y_3.
\end{aligned}$$

Next, we briefly investigate how much the continuous relaxation can be improved by adding all the cuts from eq. (26).

Proposition 1. *Assume an LDLT reformulation is performed on a quadratic convex binary problem, and that all possible cuts from eq.* (26) *are computed and added. Let* Y *be the discrete lattice formed by combining all discrete values* $y_1, y_2, \ldots,$ *and,* y_n *can take, i.e.,* $Y = Y_1 \times Y_2 \times \cdots \times Y_n$*. Then, the cuts given by eq.* (26) *form the convex hull of* $\big\{(\mathbf{y}, \mu) \in Y \times \mathbb{R}^n_+ \mid d_i y_i^2 \leq \mu_i,\ i = 1, \ldots n\big\}$.

Proof. For each univariate function $d_i y_i^2$, the cuts from eq. (26) form the epigraph of the piecewise linear function with a breakpoint at each discrete value of $y_i \in Y_i$. Thus, the cuts form the convex hull of the set

$$\left\{(y_i, \mu_i) \in Y_i \times \mathbb{R}_+ \mid \sum_{i=1}^{n} d_i y_i^2 \leq \mu_i\right\}.$$

Since the convex hull is the set of all convex combinations, and convex combinations act componentwise over Cartesian products, we know that

$$\text{convhull}(X_1 \times \cdots \times X_n) = \text{convhull}(X_1) \times \cdots \times \text{convhull}(X_n).$$

Therefore, forming the convex hull of each set

$$\left\{(y_i, \mu_i) \in Y_i \times \mathbb{R}_+ \mid \sum_{i=1}^{n} d_i y_i^2 \leq \mu_i\right\}$$

independently, gives us the convex hull of

$$\big\{(\mathbf{y}, \mu) \in Y \times \mathbb{R}^n_+ \mid d_i y_i^2 \leq \mu_i, \quad i = 1, \ldots n\big\}.$$

□

An obvious follow-up question to the proposition is whether an LDLT reformulation together with the cuts in eq. (26) can form a tight continuous relaxation of some quadratic problems. The simplest case, at least to analyze, is an

unconstrained convex quadratic binary problem. Unfortunately, even if all possible cuts in eq. (26) are added, this does not generally yield a tight continuous relaxation. The main shortcoming originates from how the set of discrete points Y is constructed, and that not all the points in Y match a combination of binary variables $\mathbf{x}$. All possible assignments of binary variables $\mathbf{x}$ correspond to a point in Y, but not vice versa, as there can be additional points in Y. For a binary problem with n variables, Y contains $2^{n(n+1)/2}$ discrete points if there are no zeros in the L matrix.

If we consider Example 3, we have the point $(0, \frac{2}{3}, 0) \in Y$. Note that $y_3 = 0$ can only be obtained with $x_3 = 0$, and $y_2 = \frac{2}{3}, y_1 = 0$ can only be obtained with $x_2 = \frac{2}{3}$ and $x_1 = \frac{1}{3}$. Thus, the discrete point $(0, \frac{2}{3}, 0)$ corresponds to a fraction point in the $\mathbf{x}$-variables. Here, there are also points in Y that do not correspond to any point within $[0, 1]^3$, for example $(0, \frac{5}{3}, 0)$. These additional, non-obtainable points in Y do not make the LDLT reformulation incorrect. But these additional points can make the convex hull of

$$\left\{(\mathbf{y}, \mu) \in Y \times \mathbb{R}^n_+ \mid d_i y_i^2 \leq \mu_i, \quad i = 1, \dots n\right\}$$

a weaker relaxation of

$$\left\{(\mathbf{x}, \alpha) \in \{0, 1\}^n \mid \frac{1}{2}\mathbf{x}^T \mathbf{Q}\mathbf{x} \leq \alpha\right\}. \tag{27}$$

In fact, if we include additional points in Y and have these equally spread out, then the feasible set defined by the cuts in eq. (26) will approach the continuous relaxation of the set in eq. (27) as the number of points in Y approach infinity.

From preliminary results, we only observed a modest improvement in the continuous relaxation by adding all the cuts in eq. (26). To further strengthen the cuts would require dependencies among the $\mathbf{y}$ variables to be taken into consideration, *i.e.*, that some combinations of $\mathbf{y}$ variables are not obtainable. However, this complicates both the cuts and their generation. Therefore, we decided not to pursue more extensive numerical experiments for these cuts in this paper.

4 Computational Experiments

In this section, we present the computational experiments conducted to evaluate the decomposition and reformulation strategies. Section 4.1 introduces the test instances used for the experiments, and Section 4.2 reports and analyzes the computational results.

The goal of this work is to improve and evaluate the computational performance of SHOT with the open-source subsolvers Cbc and HiGHS as subsolvers, both with and without the proposed reformulations. This allows us to isolate the impact of the reformulation on solver performance. However, to provide a broader perspective on overall solver performance and an indication of the relative difficulty of the problems, SCIP, BARON, and Gurobi are included as reference points for comparison with SHOT. Of these, SCIP is an open-source MINLP solver, while BARON and Gurobi are commercial MINLP solvers. Note that, in the experiments, BARON and Gurobi are combined into Commercial Best and Commercial Worst, with the better and worse results of the two solvers reported for each instance and performance measure, respectively. This captures the range of commercial solver performance while avoiding an unnecessary distinction between the two commercial solvers.

All computational experiments were performed on a Linux machine with an Intel Core i9-10900X CPU at a base frequency of 3.70 GHz, featuring 10 cores and 20 threads. The amount of memory was 64 GB. Only one concurrent thread is used for all solvers and subsolvers to remove the impact of parallelism. A prerelease version of SHOT 2.0 (git hash 64b10f44) was used with the MIP subsolvers Cbc 2.10.10 and HiGHS 1.15.1 and the NLP subsolver IPOPT 3.14.19. To compare with other state-of-the-art solvers, SCIP 10.0.3, BARON 26.5.27, and Gurobi 13.0.2 were considered from GAMS 54.3.1.

The termination criteria for all solvers were $\text{GAP}_{\text{rel}} = 0.1\%$, $\text{GAP}_{\text{abs}} = 0$ and a time limit of 900 seconds. To analyze the results, Paver 2.0 [5] was used. Since not all solvers terminate immediately when the time limit is reached, a hard fail time limit of 920 seconds was used.

The EVD and LDLT reformulations are implemented using the built-in reformulation framework in SHOT. Eigen [12] is used in SHOT to determine the convexity of quadratic expressions, and these calculations are reused when performing the EVD reformulations. We also perform the LDLT matrix decomposition with Eigen, but this step always adds computational effort; for large matrices, this effort can be significant.

4.1 Test Instances

To evaluate the effectiveness of our decomposition-based reformulations, we conduct computational experiments on a diverse set of test instances.

The test set combines 27 benchmark problems from MINLPLib [22] that are classified as convex, contain dense quadratic objective functions and/or quadratic constraints, and have at least one binary or integer variable. The smallinv instances were not included, as this family of instances is solved easily by all solvers and would therefore contribute little information to the comparison. It is worth mentioning that relatively few problems in MINLPLib meet these criteria; therefore, 33 new instances were specifically generated to expand the test set, providing both a broader range of problem structures and additional instances that can be useful for the research community. Moreover, we also included seven instances from QPLib [9] that contain dense quadratic objective functions and/or quadratic constraints, to further increase the number of test problems. It should be noted that since QPLib and MINLPLib contain overlapping instances, only seven additional QPLib instances satisfying the stated criteria were included, ensuring that no instances were duplicated. Additionally, to account for performance variability, one permuted variant of each instance was generated by using CONVERT in GAMS with `PermuteEqus=1` and `PermuteVars=1`, resulting in a total of 134 instances in the final test set.

All test problems can be found in the GitHub repository `https://github.com/AAU-IT/quad-decomp-convex-minlp`, together with documentation on how the new instances were created. Furthermore, all generated instances are portfolio optimization problems with cardinality constraints, generated from historical stock data for n assets obtained via `yfinance` [1] (specifically, closing prices for n selected assets over different time periods) and modeled using `GAMSPy` [10]. Since the generated instances thus all belong to the same application class, they are reported as a separate category of 66 instances in the computational results. The MINLPLib and QPLib instances are instead combined into a single category of 68 instances, as the QPLib instances are too few to form a meaningful category on their own and both are established benchmark libraries with overlapping content. More detailed information on the new instances can be found in Appendix A, the test instances from MINLPLib in Appendix B, and the test instances from QPLib in Appendix C.

4.2 Results

In this subsection, we first analyze the impact of the proposed quadratic decomposition-based reformulations through performance profiles. We then compare SHOT, under all relevant configurations, against other state-of-the-art solvers to provide a broader perspective. The data supporting the findings are openly available in the GitHub repository `https://github.com/AAU-IT/quad-decomp-convex-minlp`. The source data used to obtain the Commercial Best and Commercial Worst results are not included in the repository but have been obtained and validated using Paver with the same methodology as the results for SHOT and SCIP.

4.2.1 Performance Profiles — Impact of Reformulations

Figure 2 presents performance profiles illustrating the number of instances solved to an objective gap $\leq 0.1\%$ as a function of solution time in seconds. In the figure, the following three configurations of SHOT are compared for both Cbc and HiGHS:

- **NR:** No quadratic reformulation (dotted),
- **EVD:** Eigendecomposition-based reformulation (dashed), and
- **LDLT:** LDLT decomposition-based reformulation (solid).

The results clearly demonstrate that enabling quadratic decomposition significantly improves SHOT's performance. For both Cbc and HiGHS, the reformulated versions dominate the no reformulation (NR) setting over almost the entire time horizon (note the comment above that the LDLT reformulation will always require more computational effort than the EVD). In particular, the number of instances solved within the time limit increases significantly when either EVD or LDLT is activated. This confirms that transforming nonseparable quadratic expressions into separable lifted formulations strengthens the polyhedral outer approximation and improves convergence.

Furthermore, it can be noted that the LDLT-based reformulation performs best overall on most instances when considering the termination criteria in these experiments. It solves the largest number of instances and generally reaches high-quality solutions faster than both NR and EVD. This is consistent with the theoretical discussion in Section 3, where sparsity, numerical stability, and potentially tighter bounds on the auxiliary variables were identified as possible advantages of LDLT.

Overall, the performance profiles demonstrate that the proposed reformulation strategies significantly enhanced SHOT's ability to handle convex MINLP problems with dense, nonseparable quadratic terms. The LDLT-based reformulation generally performs best, yet both decomposition strategies yield clear gains in both robustness and speed.

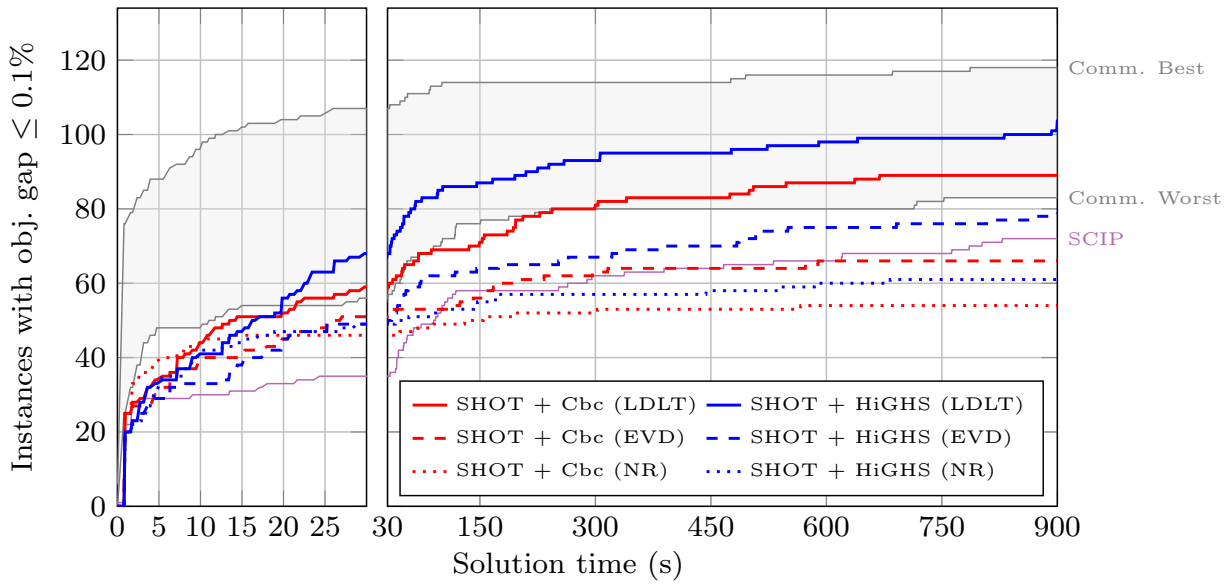


Figure 2: Number of benchmark instances solved with a relative objective gap $\leq 0.1\%$ as a function of solution time in seconds. SHOT with Cbc is shown in red and SHOT with HiGHS in blue. No reformulation (NR) is represented with dots, EVD with dashes, and LDLT with solid lines. Furthermore, SCIP is shown in violet, and the gray band indicates the range between Commercial Best and Commercial Worst.

4.2.2 Overall Solver Comparison

Figures 3, 4, and 5 present comparisons of solver performance measured by the number of instances with an objective gap $\leq 0.1\%$, $\leq 1\%$, and $\leq 10\%$, respectively, within the time limit, for all SHOT configurations as well as SCIP, Commercial Worst, and Commercial Best. It should be emphasized that these figures only compare the number of instances solved to a given gap within the time limit, not solution times. As shown in Figure 2, Commercial Best generally solves instances faster than the open-source options. The same holds for Commercial Worst, except for SHOT with the LDLT reformulation, which overtakes it as solution times increase. Thus, the reference solvers are used to indicate the difficulty of the instances and to show how much of the gap in the number of solved instances the reformulations can close, rather than to suggest that SHOT is competitive in terms of solution time in these experiments.

It can be noted that without any reformulations, SHOT solves noticeably fewer instances compared to Commercial Best, highlighting the computational difficulty introduced by dense quadratic structures when no dedicated quadratic functionality is available. This also directly reflects the central challenge addressed in this paper.

When the decomposition-based reformulations are enabled, SHOT's performance improves substantially. In particular, both reformulations lead to an increased number of solved instances for both Cbc and HiGHS. It should be noted, however, that the LDLT-based reformulation from Eigen is not uniformly applicable across all instances. In two of the test problems (specifically fac3, fac3_permuted), the LDLT factorization fails due to numerical issues associated with singular positive semidefinite matrices. In these cases, Eigen falls back to the EVD reformulation, and as a result, the reported LDLT performance includes two instances where EVD is used instead. In practice, this could probably be addressed through pivoted LDLT factorizations (*e.g.*, Bunch-Kaufman [4]), but these enhancements are not included in the current implementation and are left for future work.

Although Commercial Best achieves the best results overall in our experiments, SHOT with the LDLT-reformulation demonstrates that decomposition-based reformulations can substantially reduce the performance gap to the commercial options in terms of the number of solved instances. Across the gap levels presented in the figures, the results consistently show tighter bounds and improved solution quality, confirming the effectiveness of the proposed reformulations in practice. Overall, the results confirm that the proposed reformulations achieve the main objective of this work.

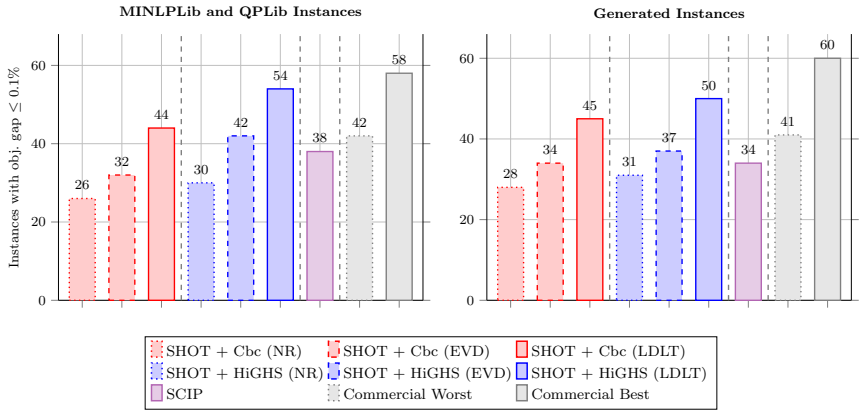


Figure 3: Comparison of solver performance measured by the number of instances with objective gap $\leq 0.1\%$.

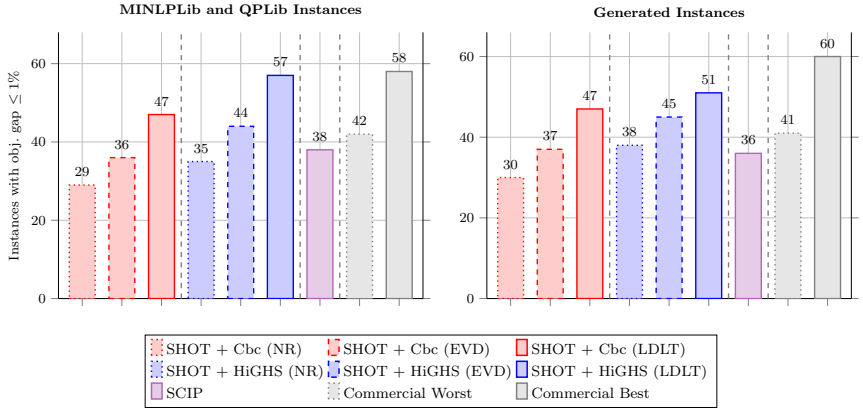


Figure 4: Comparison of solver performance measured by the number of instances with objective gap $\leq 1\%$.

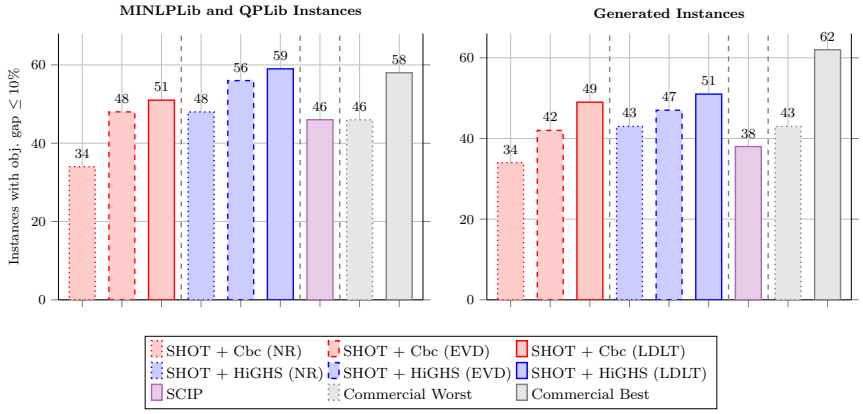


Figure 5: Comparison of solver performance measured by the number of instances with objective gap $\leq 10\%$.

5 Concluding Remarks and Future Work

In this paper, we demonstrate the impact of the decomposition-based reformulations and improve the computational performance of SHOT when using MIP subsolvers without specialized support for quadratic terms on convex problems with nonseparable quadratic expressions. The computational experiments show that the proposed reformulations substantially increase the number of instances solved by SHOT with open-source subsolvers (Cbc/HiGHS and Ipopt), bringing it closer to the performance of commercial solvers in this respect. While demonstrated within SHOT, the

reformulations themselves are not specific to it and are applicable to any POA-based convex MINLP solver restricted to MILP subsolvers. In the future, we will investigate pivoted LDLT factorizations to improve numerical robustness in cases where standard LDLT fails. In addition, a more rigorous scalability and error analysis will be performed to better characterize how the proposed reformulations behave with increasing problem size and structure complexity.

Funding

This work was supported by the Swedish Cultural Foundation in Finland, the Swedish Research Council [grant number 2022-03502], and Högskolestiftelsen i Österbotten.

Data availability

The test instances and the computational results for SHOT and SCIP are openly available at `https://github.com/AAU-IT/quad-decomp-convex-minlp`. The BARON and Gurobi results are not publicly available, as we were granted permission to publish them only in combined form as Commercial Best and Commercial Worst.

A Generated Test Instances

In this appendix, the mathematical models and the documentation on how the new instances were created are presented. The problems were created specifically to test the decomposition-based reformulation of nonseparable quadratic expressions. Hence, the problems are quite simple in nature and do not involve many complex constraints. The complexity of these instances primarily comes from the number of variables and constraints.

All of the newly generated instances, see Table 2, are considered portfolio optimization problems with cardinality constraints[1], generated from historical stock data for n assets obtained via `yfinance`. Specifically, closing prices for n selected assets were collected over different time periods and modeled using `GAMSPy` in a `Jupyter notebook`. Note that they are inspired by realistic applications but do not necessarily represent fully realistic models.

Some tickers in `yfinance` may no longer be available, as most of the instances were generated during the summer of 2025. In addition, `GAMSPy` has undergone subsequent updates, implying that certain parts of the code may require adjustments to remain functional under newer versions. As a result, we suggest that the notebooks should be interpreted primarily as blueprints documenting the methodological workflow rather than as exact reproductions of the computational steps used to generate the instances. For the exact models, we refer to the `.gms` files. All generated instances (both `.ipynb` and `.gms` files) can be found in their respective problem-type folders in the GitHub repository `https://github.com/AAU-IT/quad-decomp-convex-minlp`.

Table 2: Summary of the generated test problems.

Name	Type	C	#Vars	#BinVars	#Cons
CCPOP5_20	MBQP	Convex	40	20	22
CCPOP5_50	MBQP	Convex	100	50	52
CCPOP5_100	MBQP	Convex	200	100	102
CCPOP5_186	MBQP	Convex	372	186	188
CCPOP5_219	MBQP	Convex	438	219	221
CCPOP5_300	MBQP	Convex	600	300	302
CCPOPwTCaD5_20	MBQCQP	Convex	40	20	23
CCPOPwTCaD5_50	MBQCQP	Convex	100	50	53
CCPOPwTCaD5_100	MBQCQP	Convex	200	100	103
CCPOPwTCaD5_300	MBQCQP	Convex	600	300	303
CCPOPwTCaRL5_20	MBQCQP	Convex	40	20	23
CCPOPwTCaRL5_50	MBQCQP	Convex	100	50	53
CCPOPwTCaRL5_165	MBQCQP	Convex	330	165	168
CCPOPwTCaRL5_308	MBQCQP	Convex	616	308	311
CCPOPwTCaRL5_538	MBQCQP	Convex	1076	538	541
RPOPwCaTC5_20	MBQCQP	Convex	42	20	25
RPOPwCaTC5_50	MBQCQP	Convex	102	50	55
RPOPwCaTC5_100	MBQCQP	Convex	202	100	105
RPOPwCaTC5_319	MBQCQP	Convex	640	319	324
CCPOPwTCRTEaD5_20	MBQCQP	Convex	40	20	25
CCPOPwTCRTEaD5_50	MBQCQP	Convex	100	50	55
SaCCDPOP5_3_17	MBNLP	Convex	50	25	53
SaCCDPOP5_3_61	MBNLP	Convex	132	66	135
SaCCDPOP5_3_115	MBNLP	Convex	240	120	243
SaCCDPOP5_3_326	MBNLP	Convex	662	331	665
MPCCPOP12_10	MBQP	Convex	2160	720	1585
MPCCPOP12_20	MBQP	Convex	4320	1440	3025
MPCCPOP12_30	MBQP	Convex	6480	2160	4465
MPCCPOP12_50	MBQP	Convex	10800	3600	7345
MPCCPOP_crypto8_1mo_1d	MBQP	Convex	481	240	541
MPCCPOP_crypto8_1mo_5d	MBQP	Convex	97	48	109
MPCCPOP_crypto8_6d_30m	MBQP	Convex	4081	2040	4591
MPCCPOP_crypto8_30d_90m	MBQP	Convex	7505	3752	8443

[1]Permuted variants are not listed in the table as they differ from the original instances only in the ordering of their variables and constraints. Furthermore, all objective functions have been scaled by a factor of 10^5 in their respective `.gms` files for numerical stability.

A.1 Mathematical Models and Documentation

1. Cardinality-Constrained Portfolio Optimization Problem (CCPOP)

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 && \text{(Budget constraint)} \\
& x_i \le y_i, \quad \forall i = 1,\ldots,n && \text{(Linking constraint)} \\
& \sum_{i=1}^{n} y_i \le 5 && \text{(Cardinality constraint)} \\
& x_i \ge 0, \quad \forall i = 1,\ldots,n && \text{(Non-negativity)} \\
& y_i \in \{0,1\}, \quad \forall i = 1,\ldots,n && \text{(Binary selection variables)}
\end{aligned}
$$

There are six different versions of this problem. In all of them, we select a maximum of 5 assets out of n assets. In the first version, $n = 20$; in the second, $n = 50$; in the third, $n = 100$; and in the fourth $n = 186$; in the fifth, $n = 219$; and in the last $n = 300$.

The portfolio optimization problem was generated using historical stock data for n assets downloaded with `yfinance`. Specifically, closing prices for n selected assets were collected over the period from January 1, 2022, to May 31, 2025.

2. Cardinality-Constrained Portfolio Optimization with Transaction Costs and Diversification (CCPOPwTCaD)

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j + \sum_{i=1}^{n} c_i (x_i - x_i^{prev})^2 \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 && \text{(Budget constraint)} \\
& x_i \le y_i, \quad \forall i = 1,\ldots,n && \text{(Linking constraint)} \\
& \sum_{i=1}^{n} y_i \le 5 && \text{(Cardinality constraint)} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}x_i x_j \le 1 && \text{(Diversification constraint)} \\
& x_i \ge 0, \quad \forall i = 1,\ldots,n && \text{(Non-negativity)} \\
& y_i \in \{0,1\}, \quad \forall i = 1,\ldots,n && \text{(Binary selection variables)}
\end{aligned}
$$

There are four different versions of this problem. In all of them, we select a maximum of 5 assets out of n assets. In the first version, $n = 20$; in the second, $n = 50$; in the third, $n = 100$; and in the last, $n = 300$.

This portfolio optimization problem was generated using historical stock data for n assets downloaded with `yfinance`. Specifically, closing prices for n assets were collected over the period from January 1, 2023, to May 31, 2024.

The transaction cost coefficients were simulated as small random values, and a diversification constraint was introduced using a random positive semidefinite matrix M. (Note: the diversification constraint used in this problem did not work that well in practice.)

3. Cardinality-Constrained Portfolio Optimization with Risk Limit and Transaction Costs (CCPOPwTCaRL)

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} \mu_i x_i - \sum_{i=1}^{n} c_i (x_i - x_i^{prev})^2 \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 && \text{(Budget constraint)} \\
& x_i \leq y_i, \quad \forall i = 1, \ldots, n && \text{(Linking constraint)} \\
& \sum_{i=1}^{n} y_i \leq 5 && \text{(Cardinality constraint)} \\
& \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij} x_i x_j \leq \sigma_{\text{max}}^2 && \text{(Risk constraint)} \\
& x_i \geq 0, \quad \forall i = 1, \ldots, n && \text{(Non-negativity)} \\
& y_i \in \{0, 1\}, \quad \forall i = 1, \ldots, n && \text{(Binary selection variables)}
\end{aligned}
$$

There are five different versions of this problem. In all of them, we want to select a maximum of 5 assets out of n assets. In the first version, $n = 20$; in the second, $n = 50$; in the third, $n = 165$; in the fourth, $n = 308$; and in the last, $n = 538$.

This problem was generated using historical stock data for n assets, downloaded with `yfinance`, (closing prices from January 1, 2022, to May 31, 2025), incorporating expected returns, a risk limit $\sigma_{\text{max}}^2 = 0.01$, and simulated transaction cost penalties based on deviation from a previous equally weighted portfolio.

4. Robust Portfolio Optimization with Cardinality and Transaction Costs (RPOPwCaTC)

$$
\begin{aligned}
\text{maximize} \quad & t - \sum_{i=1}^{n} c_i (x_i - x_i^{prev})^2 \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 && \text{(Budget)} \\
& \sum_{i=1}^{n} y_i \leq 5 && \text{(Cardinality)} \\
& x_i \leq y_i, \quad \forall i && \text{(Linking)} \\
& \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij} x_i x_j \leq 0.002 && \text{(Risk)} \\
& t + 0.005 \cdot z \leq \sum_{i=1}^{n} \mu_i x_i && \text{(Robust return UB)} \\
& \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j x_i x_j \leq z && \text{(SOC 1)} \\
& x_i \geq 0, \quad y_i \in \{0, 1\}, \quad \forall i && \text{(Domains)}
\end{aligned}
$$

There are four different versions of this problem. In all of them, we select a maximum of 5 assets out of n assets. In the first version, $n = 20$; in the second, $n = 50$; in the third, $n = 100$; and in the last, $n = 319$. Note that t is a variable representing the worst-case return, and z is used for an uncertainty bound.

This problem was generated using historical stock data for n assets (from January 1, 2022, to May 31, 2025) downloaded with `yfinance`, using simulated transaction cost coefficients and a robust optimization framework to hedge against return estimation uncertainty.

5. Portfolio Optimization with Risk, Tracking Error, Diversification, and Transaction Costs (CCPOPwTCRTEaD)

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j - \sum_{i=1}^{n} \mu_i x_i + \sum_{i=1}^{n} c_i (x_i - x_i^{prev})^2 \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 && \text{(Budget)} \\
& x_i \leq y_i, \quad \forall i && \text{(Linking)} \\
& \sum_{i=1}^{n} y_i \leq 5 && \text{(Cardinality)} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j \leq \tau = 0.05 && \text{(Risk)} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}(x_i - x_i^{bench})(x_j - x_j^{bench}) \leq \delta = 0.001 && \text{(Tracking Error)} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}x_i x_j \leq 1 && \text{(Diversification)} \\
& x_i \in [0,1], \quad y_i \in \{0,1\}, \quad \forall i && \text{(Domains)}
\end{aligned}
$$

The model was built using historical data for $n = 20, 50$ assets (from January 1, 2023, to May 31, 2024), downloaded via `yfinance`. Transaction cost parameters and a benchmark portfolio were simulated.

6. Sector-Constrained Diversified Portfolio Optimization (SaCCDPOP)

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j - \lambda \sum_{i=1}^{n} \log(x_i + \varepsilon) \\
\text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 && \text{(Budget)} \\
& x_i \leq y_i, \quad \forall i && \text{(Linking)} \\
& \sum_{i=1}^{n} y_i \leq 5 && \text{(Cardinality)} \\
& \sum_{s=1}^{S} z_s \leq 3 && \text{(Sector limit)} \\
& y_i \leq \sum_{s=1}^{S} \text{sector_indicator}_{i,s} \cdot z_s && \forall i \quad \text{(Sector-asset link)} \\
& \text{sector_weight}_s = \sum_{i=1}^{n} \text{sector_indicator}_{i,s} \cdot x_i && \forall s \quad \text{(Aggregation)} \\
& \text{sector_weight}_s \leq \text{sector_max_exposure}_s \cdot z_s && \forall s \quad \text{(Exposure)} \\
& x_i \geq 0, \quad y_i, z_s \in \{0,1\} && \text{(Domains)}
\end{aligned}
$$

This problem was built using historical stock data downloaded with `yfinance` (January 1, 2023, to May 31 2024). There are four different versions. In all of them, we select a maximum of 5 assets out of n assets and 3 sectors. In the first version, $n = 17$; in the second, $n = 61$; in the third, $n = 115$; and in the last, $n = 326$.

7. Multi-Period Portfolio Optimization (MPCCPOP)

$$
\begin{aligned}
\text{minimize} \quad & \sum_{t \in T} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} Q_{t,ij} x_{t,i} x_{t,j} + \lambda \sum_{i=1}^{n} \delta_{t,i}^2 \right) \\
\text{subject to} \quad & \sum_{i=1}^{n} x_{t,i} = 1 && \forall t \in T \quad \text{(Budget)} \\
& \sum_{i=1}^{n} y_{t,i} \leq K && \forall t \in T \quad \text{(Cardinality)} \\
& x_{t,i} \leq y_{t,i} && \forall t \in T,\ \forall i \in I \quad \text{(Linking)} \\
& x_{t,i} \geq 0.01 \cdot y_{t,i} && \forall t \in T,\ \forall i \in I \quad \text{(Min investment)} \\
& \delta_{t,i} = x_{t,i} - x_{t-1,i} && \forall t > 1,\ \forall i \in I \quad \text{(Trade update)} \\
& x_{t,i} \geq 0,\ y_{t,i} \in \{0,1\},\ \delta_{t,i} \in \mathbb{R} && \forall t,\ i \quad \text{(Domains)}
\end{aligned}
$$

This model was generated using historical stock data from `yfinance` (June 21, 2019, to June 26 2025). There are four different versions of this problem: one with $n = 10$, one with $n = 20$, one with $n = 30$, and the final one with $n = 50$. Note that this problem is quite similar to the first problem, *i.e.*, CCPOP5. The only difference is the tickers and that we do multi-period optimization.

8. Multi-Period Crypto Portfolio Optimization Problem (MPCCPOP_crypto)

$$
\begin{aligned}
\text{minimize} \quad & \lambda \cdot \frac{\sum_t \sum_{i,j} Q_{t,ij} x_{t,i} x_{t,j}}{\sum |Q_{t,ij}| + \epsilon} - (1 - \lambda) \cdot \frac{\sum_t \sum_i \text{ER}_{t,i} \cdot x_{t,i}}{\sum |\text{ER}_{t,i}| + \epsilon} \\
\text{subject to} \quad & \sum_{i=1}^{n} x_{t,i} = 1 && \forall t \in T \quad \text{(Budget)} \\
& \sum_{i=1}^{n} y_{t,i} \leq 12 && \forall t \in T \quad \text{(Cardinality)} \\
& x_{t,i} \leq y_{t,i} && \forall t, i \quad \text{(Linking)} \\
& x_{t,i} \geq 0.01 \cdot y_{t,i} && \forall t, i \quad \text{(Min investment)} \\
& \delta_{t,i} = x_{t,i} - x_{t-1,i} && \forall t > 1,\ i \quad \text{(Trade dynamics)} \\
& x_{t,i} \geq 0,\ y_{t,i} \in \{0,1\},\ \delta_{t,i} \in \mathbb{R} && \forall t,\ i \quad \text{(Domains)}
\end{aligned}
$$

where $\lambda = 0.4$.

This problem was built using historical stock data downloaded with `yfinance`. There are four different types of this problem, all of which take data from 8 cryptocurrencies. Two of the instances take data over 1 month but use different time intervals. The first time interval is 1 day, and the second is 5 days. The last two instances take data over 6 days and 30 days, respectively, with 30 minutes and 90 minutes as the intervals. Note that ER is the expected return.

B Test Instances from MINLPLib

In this appendix, a list of the test instances from MINLPLib [22] can be found. All of these instances can also be found in the GitHub repository `https://github.com/AAU-IT/quad-decomp-convex-minlp`, along with one permuted variant of each instance.

alan, ball_mk4_05, ball_mk4_10, ball_mk4_15, color_lab2_4x0, color_lab6b_4x20, du-opt, du-opt5, fac3, meanvar-orl400_05_e_8, meanvarx, nvs10, nvs11, nvs12, nvs15, pedigree_ex1058, pedigree_ex485, pedigree_ex485_2, pedigree_sim2000, pedigree_sim400, pedigree_sp_top4_250, pedigree_sp_top4_300, pedigree_sp_top4_350tr, pedigree_sp_top5_200, pedigree_sp_top5_250, watercontamination0202r, watercontamination0303r.

C Test Instances from QPLib

In this appendix, a list of the test instances from QPLib [9] can be found. All of these instances can also be found in the GitHub repository `https://github.com/AAU-IT/quad-decomp-convex-minlp`, along with one permuted variant of each instance.

QPLIB_8845, QPLIB_8906, QPLIB_10006, QPLIB_10007, QPLIB_10050, QPLIB_10056, QPLIB_10069.

References

[1] R. Aroussi. yfinance. URL: https://github.com/ranaroussi/yfinance, n.d.

[2] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S.J. Maher, F. Matter, E. Mühmer, B. Müller, M.E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The SCIP Optimization Suite 8.0. *arXiv*, 2021.

[3] P. Bonami. Lift-and-Project Cuts for Mixed Integer Convex Programs. In *Integer Programming and Combinatorial Optimization*, pages 52–64, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[4] J.R. Bunch and L. Kaufman. Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems. *MATHEMATICS OF COMPUTATION*, 31(137):163–179, 1977.

[5] M. Bussieck, S. Dirkse, and S. Vigerske. PAVER 2.0: An open source environment for automated performance analysis of benchmarking data. *Journal of Global Optimization*, 59(2-3):259–275, 2014.

[6] M. Djerdjour. An enumerative algorithm framework for a class of nonlinear integer programming problems. *European Journal of Operational Research*, 101(1):104–121, 1997.

[7] Eigen. Eigen::SelfAdjointEigenSolver<MatrixType_> Class Template Reference. Eigen Documentation. URL: https://libeigen.gitlab.io/eigen/docs-nightly/classEigen_1_1SelfAdjointEigenSolver.html. Accessed: 2026-08-24.

[8] A. Frangioni and C. Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106:225–236, 2005.

[9] F. Furini, E. Traversi, P. Belotti, A. Frangioni, A. Gleixner, N. Gould, L. Liberti, A. Lodi, R. Misener, H. Mittelmann, N. Sahinidis, S. Vigerske, and A. Wiegele. QPLIB: A Library of Quadratic Programming Instances. *Mathematical Programming Computation*, 2018.

[10] GAMS Development Corporation. GAMSPy. URL: https://github.com/GAMS-dev/gamspy, n.d.

[11] G.H. Golub and C.F. van Loan. *Matrix Computations*. JHU Press, fourth edition, 2013.

[12] G. Guennebaud, B. Jacob, et al. Eigen: A c++ template library for linear algebra. URL: http://eigen.tuxfamily.org, 2021.

[13] O. Günlük and J. Linderoth. Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical Programming*, 124:182–205, 2010.

[14] H. Hijazi, P. Bonami, and A. Ouorou. An outer-inner approximation for separable mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 26(1):31–44, 2014.

[15] J. Kronqvist and A. Lundell. Eigendecomposition-based reformulations for convex MINLP in the SHOT solver. In *Journal of Global Optimization: Special Issue on Global Optimization: HUGO*, pages 107–110, 2022.

[16] J. Kronqvist, A. Lundell, and T. Westerlund. The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. *Journal of Global Optimization*, 64(2):249–272, 2016.

[17] J. Kronqvist, A. Lundell, and T. Westerlund. Reformulations for utilizing separability when solving convex MINLP problems. *Journal of Global Optimization*, 71:571–592, 2018.

[18] M.R. Kılınç, J. Linderoth, and J. Luedtke. Lift-and-project cuts for convex mixed integer nonlinear programs. *Mathematical Programming Computation*, 9:499–526, 2017.

[19] M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming*, 172:139–168, 2018.

[20] A. Lundell and J. Kronqvist. Polyhedral approximation strategies for nonconvex mixed-integer nonlinear programming in SHOT. *Journal of Global Optimization*, 82(4):863–896, 2022.

[21] A. Lundell, J. Kronqvist, and T. Westerlund. The supporting hyperplane optimization toolkit for convex MINLP. *Journal of Global Optimization*, 84:1–41, 2022.

[22] MINLPLib. MINLPLib: Mixed-integer nonlinear programming library. http://www.minlplib.org/. Downloaded 19-August-2025.

[23] D. Quadri and E. Soutil. Reformulation and solution approach for non-separable integer quadratic programs. *Journal of the Operational Research Society*, 66(8):1270–1280, 2015.

[24] H.D. Sherali and W. P. Adams. A Reformulation-Linearization Technique (RLT) for semi-infinite and convex programs under mixed 0-1 and general discrete restrictions. *Discrete Applied Mathematics*, 157(6):1319–1333, 2009.

[25] H.D. Sherali and W.P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[26] H.D. Sherali, Y. Lee, and Y. Kim. Partial convexification cuts for 0–1 mixed-integer programs. *European Journal of Operational Research*, 165(3):625–648, 2005.

[27] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[28] J. P. Vielma, S. Ahmed, and G. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008.

[29] S. Vigerske and A. Gleixner. SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software*, 33(3):563–593, 2018.

[30] T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19(1):131–136, 1995.