Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Compact mixed-integer programming formulations in quadratic optimization

Benjamin Beach · Robert Hildebrand · Joey Huchette

Received: date / Accepted: date

Abstract We present a technique for producing valid dual bounds for nonconvex quadratic optimization problems. The approach leverages an elegant piecewise linear approximation for univariate quadratic functions due to Yarotsky [57], formulating this (simple) approximation using mixed-integer programming (MIP). Notably, the number of constraints, binary variables, and auxiliary continuous variables used in this formulation grows logarithmically in the approximation error. Combining this with a diagonal perturbation technique to convert a nonseparable quadratic function into a separable one, we present a mixed-integer convex quadratic relaxation for nonconvex quadratic optimization problems. We study the strength (or *sharpness*) of our formulation and the tightness of its approximation. Further, we show that our formulation represents feasible points via a Gray code. We close with computational results on problems with quadratic objectives and/or constraints, showing that our proposed method i) across the board outperforms existing MIP relaxations from the literature, and ii) on hard instances produces better bounds than exact solvers within a fixed time budget.

Keywords Quadratic optimization · Nonconvex optimization · Mixed-integer programming · Gray Code

This work was supported by AFOSR (grant FA9550-21-0107) and ONR (Grant N00014-20-1-2156). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research or the Air Force Office of Scientific Research.

Benjamin Beach · Robert Hildebrand
Grado Department of Industrial and Systems Engineering, Virginia Tech
E-mail: {bben6,rhil}@vt.edu

Joey Huchette
Department of Computational and Applied Mathematics, Rice University
E-mail: joehuchette@rice.edu

1 Introduction

We are interested in methods to solve optimization problems with quadratic objectives and/or constraints. Consider the following generic problem with a quadratic objective:

$$\min_{x \in X} \quad h(x) := x'Qx + c \cdot x, \tag{1}$$

where $X \subseteq \mathbb{R}^n$ is some nonempty feasible region described by side constraints. When the quadratic objective matrix Q is not positive semidefinite, this is a difficult nonconvex optimization problem. We will focus on techniques to (approximately) reformulate nonconvex quadratic functions like the objective of (1).

Quadratic optimization problems naturally arise in a number of important applications across science and engineering (see [30,45] and references therein). In the presence of nonconvexity, such problems are in general very difficult to solve from both a practical and theoretical perspective [44]. As a result, there has been a steady stream of research developing new algorithmic techniques to solve quadratic optimization problems, and variants thereof (see [12] for a survey).

Our approach to approximately solving problems of the form (1) will be to reformulate the objective of (1) using mixed-integer programming (MIP). Given some diagonal matrix D, we can equivalently write (1) as

$$\min_{x \in X} \quad h^D(x, y) := x'(Q + D)x + c \cdot x - Dy \tag{2a}$$

$$\text{s.t.} \quad y_i = x_i^2 \qquad i \in [\![n]\!], \tag{2b}$$

where $[\![n]\!] := \{1, \ldots, n\}$. If D is chosen such that $Q+D$ is positive semidefinite, the quadratic objective will be convex, meaning that all the nonconvexity of this problem has been isolated in the univariate quadratic equations $y_i = x_i^2$. This technique is sometimes called "diagonal perturbation" [22].

In this work, we present a compact, tight MIP formulation for the graph of a univariate quadratic term: $\left\{ (x, y) \;\middle|\; l \leq x \leq u,\ y = x^2 \right\}$. We derive our formulation by adapting an elegant result of Yarotsky [57], who shows that there exists a simple neural network function that approximates $y = x^2$ exponentially well (in terms of the size of the network) over the unit interval. The resulting neural network can be interpreted as a function $F_L : \mathbb{R} \to \mathbb{R}$ that is build compositionally from a number of simple piecewise linear functions. There is a long and rich strain of research on MIP formulations for piecewise linear functions that serve as approximations for more complex nonlinear functions [18,19,36,39,40,43,51], with recent work focusing particularly on modeling neural networks [3,11,48,49,50,4].

We show that this approximation for univariate quadratic terms leads to a *relaxation* for optimization problems with quadratic objectives and/or constraints, meaning that it provides valid dual bounds for the true quadratic problem. We will show that our proposed formulation is *sharp*, meaning that its LP relaxation projects to the convex hull of all feasible points. Further,

we show that the formulation is in fact *hereditarily sharp*, meaning that this sharpness property holds throughout the branch and bound tree. The key to reaching this result is connecting the binary reformulation to the *reflected Gray code*, a well-studied binary sequence in electrical engineering.

1.1 Literature review

Our approach hews most closely to that of Dong and Luo [23] and Saxena et al. [47]. The diagonal perturbation approach we follow have been applied throughout the years in a number of settings; for example, nonconvex quadratic optimization (with or without integer variables) [8,9,25,31,55], more general nonlinear [28,29] optimization with binary variables, and general nonlinear optimization [1,2,5].

A string of recent work on optimization methods for nonconvex quadratic problems has focused on methods for relaxing bilinear terms using piecewise McCormick envelopes [13,14,15,41,42,16]. These piecewise envelopes can be formulated using mixed-integer programming in multiple ways, typically resulting in either a linear- or logarithmic-sized MIP formulation. Moreover, this piecewise relaxation can be refined dynamically to produce a tighter relaxation in a region of interest without resulting in an unduly large MIP formulation [13, 42]. In a similar vein, a paper of Galli and Letchford [32] presents a binarization heuristic for "box QP" problems, leveraging a structural result of Hansen et al. [35], and compares classical convexification techniques [26,33,34] within the heuristic.

An interesting recent paper of Xia et al. [56] reformulates optimization problems with quadratic objectives and linear constraints into MIPs via the KKT conditions. The approach outperforms commercial solvers on certain classes of instances; however, it does not seem to perform favorably on boxQP problems, and in general requires the careful computation of "big-M" coefficients which may lead to loose LP relaxations.

1.2 Outline

In Section 2 we describe our MIP approximation for $y = x^2$. In Section 3 we prove some properties of Gray codes that will be useful for proving the results in Section 4. In Section 4, we show that our formulation is strong (i.e. sharp), and establish the connection between our MIP approximation and the reflected Gray code. In Section 5, we show how to derive some facets of the full convex hull of our MIP approximation, with connections to the parity polytope. In Section 6, we present a relaxation version of our MIP approximation, derive the total area of the relaxation, and compare against the relaxation of Dong and Luo [23]. Finally, in Section 7, we numerically compare our relaxation with other competing methods, including other relaxations such as CDA [23] and NMDT [14], as well as state-of-the-art solvers with quadratic support like Gurobi, CPLEX, and BARON.

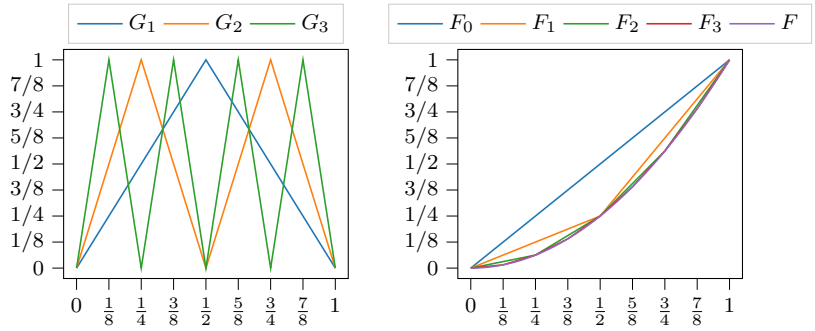


Fig. 1: *Left:* The intermediary sawtooth functions $G_i = 2^{2i}(F_{i-1} - F_i)$. *Right:* The approximation for $F(x) = x^2$ of Yarotsky by functions F_i [57, Figure 2].

2 A piecewise-linear approximation for univariate quadratic terms

In this section, we present our mixed-integer programming relaxation for (1). We start by describing the construction of Yarotsky, which is a piecewise linear neural network approximation for the univariate quadratic function $F(x) = x^2$. We then formulate the graph of this piecewise-linear function using mixed-integer programming, and use it to build a tight under-approximation for the quadratic optimization problem (1).

For ease of notation, for any integers $i \leq j$, we define $[\![i, j]\!] := \{i, i + 1, \ldots, j\}$, and for integers $i \geq 1$ we define $[\![i]\!] := \{1, 2, \ldots, i\}$.

2.1 The construction of Yarotsky

For fixed $L \in \mathbb{N}$, we wish to model the function $F_L(x)$, defined as the piecewise linear interpolant to $y = x^2$ on the interval $[0, 1]$ at $2^L + 1$ uniformly spaced breakpoints:

$$F_L(x) = \tfrac{2i-1}{N}(x - \tfrac{i}{N}) + \tfrac{i^2}{N^2} \quad \text{if } x \in [\tfrac{i-1}{N}, \tfrac{i}{N}] \text{ for some } i \in [\![2^L]\!]. \tag{3}$$

Define the *sawtooth functions* $G_i \colon [0, 1] \to [0, 1]$ as $G_i = 2^i(F_{i-1} - F_i)$. Yarotsky [57] shows that G_i can be defined recursively as

$$G_0(x) = x, \tag{4a}$$

$$G_i(x) = \begin{cases} 2G_{i-1}(x) & G_{i-1}(x) < 1/2 \\ 2(1 - G_{i-1}(x)) & G_{i-1}(x) \geq 1/2 \end{cases} \quad i \in [\![L]\!], \tag{4b}$$

and, furthermore, that

$$F_L(x) = x - \sum_{i=1}^{L} 2^{-2i} G_i(x). \tag{5}$$

Yarotsky further shows that $F(x)$ approximates x^2 to a pointwise error of $|x^2 - F_L(x)| \leq 2^{-2L-2}$ [57, Proposition 2].[1] We include an illustration of G_L and F_L for different values of L in Figure 1(b). Crucially, we will later make use of the fact that $F_L(x) \geq F(x)$ for each $0 \leq x \leq 1$, i.e. F_L is an overestimator for F.

2.2 A MIP formulation for F_L

We now turn our attention to constructing a mixed-integer programming formulation for F_L. As (5) tells us that F_L depends linearly on the sawtooth functions G_i, we turn our attention to formulating the piecewise-linear equations (4) using MIP.

For the remainder of the section we will use g_i as decision variables in our optimization formulation corresponding to the output of the i-th sawtooth function G_i. Therefore, $g_0 = x$, and for each of the other sawtooth functions G_i for $i \in [\![L]\!]$, we introduce a binary decision variable α_i. Given some input x, these binary variables serve to indicate which piece of the sawtooth the input lies on:

$$\alpha_i = 0 \implies (g_i = 2g_{i-1}) \wedge (0 \leq g_{i-1} \leq 1/2) \tag{6a}$$
$$\alpha_i = 1 \implies (g_i = 2(1 - g_{i-1})) \wedge (1/2 \leq g_{i-1} \leq 1) \tag{6b}$$

Define the set $S_i := \{ (g_{i-1}, g_i, \alpha_i) \in [0,1] \times [0,1] \times \{0,1\} \mid (6) \}$ for each $i \in [\![L]\!]$. It is not difficult to see that a convex hull formulation for S_i is given by

$$2(\alpha_i - g_{i-1}) \leq g_i \leq 2(1 - g_{i-1}), \tag{7a}$$
$$2(g_{i-1} - \alpha_i) \leq g_i \leq 2g_{i-1}. \tag{7b}$$
$$(g_{i-1}, g_i, \alpha_i) \in [0,1] \times [0,1] \times \{0,1\}. \tag{7c}$$

Chaining these formulations together for each i, we construct a MIP formulation for $\mathcal{G}_L := \{ (x, y) \in [0,1] \times [0,1] \mid y = F_L(x) \}$, the graph of the neural network approximation function F_L.

Proposition 1 *Fix some $L \in \mathbb{N}$. A MIP formulation for $(x, y) \in \mathcal{G}_L$ is*

$$\begin{aligned} &g_0 = x \\ &(g_{i-1}, g_i, \alpha_i) \in S_i \qquad i \in [\![L]\!] \\ &y = x - \textstyle\sum_{i=1}^{L} 2^{-2i} g_i. \end{aligned} \tag{8}$$

We emphasize that this formulation is extremely compact: it requires only $\mathcal{O}(L)$ binary variables, auxiliary continuous variables, and constraints. As noted in Section 2.1, F_L approximates F to within $\mathcal{O}(2^{-L})$ pointwise, which

[1] Furthermore, Yarotsky [57] observes that it is straightforward to represent each of the sawtooth functions as a composition of the standard ReLU activation function $\sigma(x) = \max\{0, x\}$. For example, $G_1(x) = 2\sigma(x) - 4\sigma(x - \frac{1}{2}) + 2\sigma(x - 1)$. In this way, F_L can be written as a neural network with a very particular choice of architecture and weight values.

implies that the size of our formulation scales logarithmically in the desired accuracy.

It is a straightforward extension of Proposition 1 to consider more general interval domains $x \in [l, u]$ on the inputs. In particular, introducing two auxiliary variables $\tilde{x}, \tilde{y} \in [0, 1]$, we formulate $\tilde{y} = F(\tilde{x}) = \tilde{x}^2$ using (8), and then map them to the (x, y) variables via the linear transformation

$$x = l + (u - l)\tilde{x}, \quad y = l^2 + 2l(u - l)\tilde{x} + (u - l)^2\tilde{y}.$$

2.3 Tying it all together

We are now prepared to construct our mixed-integer programming approximation for (1). For the objective of (1), compute a nonnegative diagonal matrix D such that $Q + D$ is positive semidefinite.[2] Then, for a given L, the approximation for (1) is:

$$\min_{x \in X, y} \quad h^D(x, y) \equiv x'(Q + D)x + c \cdot x - Dy \tag{9a}$$

$$\text{s.t.} \quad (x_i, y_i) \in \mathcal{G}_L \qquad i \in [\![n]\!]. \tag{9b}$$

Using the formulation (8) for the constraint (9b), this yields a mixed-integer convex quadratic reformulation of the problem (ignoring the potential structure of X). This formulation requires at most nL binary variables and $\mathcal{O}(nL)$ auxiliary continuous variables and linear constraints. Furthermore, recall that we may set $L = \mathcal{O}(\log(1/\varepsilon))$ to attain an approximation of accuracy ε for the equations (2b).

Consider any $\hat{x} \in X$, along with any $\hat{y}$ such that $(\hat{x}, \hat{y})$ satisfies (9b). Since F_L overestimates F, for each $i \in [\![n]\!]$ we have $\hat{x}_i^2 \leq \hat{y}_i$. Therefore, $h^D(\hat{x}, \hat{y}) \leq h(\hat{x})$. Since there always will exist such a $\hat{y}$ for any $\hat{x} \in X$, (9) offers a valid dual bound on the optimal cost of (1).

Note that this approach can readily be adapted to handle quadratic constraints. In particular, this transformation will offer a *relaxation* of the quadratically constrained problem. Note that the error bound derived above is with respect to the quadratic constraint that is being relaxed. It may not translate into an error bound on the objective value of the optimization problem, a known phenomena in the global optimization literature [20].

3 Gray Codes and Binary Representation

In this section, we introduce the reflected Gray code, and prove some of its useful properties.

[2] This can be accomplished in a number of ways: for example, by computing the minimum eigenvalue of D, or by solving a semidefinite programming problem [23].

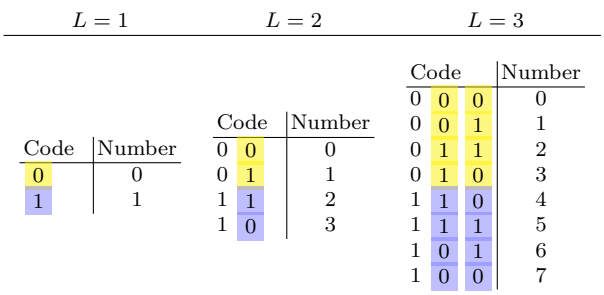


Fig. 2: Building the reflected Gray code. The reflected Gray code with $L+1$ bits is build from the reflected Gray code on L bits by appending 0s in front of it, then reflecting the Gray code sequence, and then appending 1s in front of it.

For the remainder of this work, we will work with two notions of expressing integers as vectors in $\{0,1\}^*$. First, we consider the standard binarization with L bits. That is, for an integer $i \in [\![0, 2^L - 1]\!]$ we define $\boldsymbol{\beta}^i \in \{0,1\}^L$ such that

$$i = \sum_{j=1}^{L} 2^{L-j} \beta_j^i. \tag{10}$$

Next, we define the *reflected Gray code* sequence, which is a sequence of binary representations of integers that is extremely well-studied in electrical engineering and engineering [46]. Notably, each adjacent pair in the sequence differs in exactly one bit. As presented in [27] and references therein, the L-bit reflected Gray code $\boldsymbol{\alpha}^i \in \{0,1\}^L$ representing the integer i can be described by the recursion

$$\alpha_1^i = \beta_1^i \tag{11}$$

$$\alpha_j^i := \beta_j^i \oplus \beta_{j-1}^i \qquad \text{for all } j = 2, \ldots, L, \tag{12}$$

where we use $\oplus$ to denote addition modulo 2. By inverting the relation, we obtain the formula

$$\beta_j^i = \alpha_1^i \oplus \alpha_2^i \oplus \cdots \oplus \alpha_j^i \qquad \text{for } j = 1, \ldots, L. \tag{13}$$

In this way, flipping any α_j bit implies that we flip all less significant bits β_k for $k \geq j$. See Figure 2 for an illustration of how to build the reflected Gray code, which we will henceforth refer to as 'the Gray code'.

One key property of any Gray code is that successive integer representations differ by only 1 bit, i.e.,

$$\|\boldsymbol{\alpha}^i - \boldsymbol{\alpha}^{i+1}\|_1 = 1. \tag{14}$$

That is, only one bit changes between adjacent binary vectors in the sequence. We show a similar property holds if we restrict the set of integers we work with

by fixing some of the bits in the Gray code vector. To help prove this property, we note the following well-known property of the reflected Gray code in this work.

Lemma 1 *For each $i \in [\![0, 2^L - 1]\!]$, let $\tilde{\boldsymbol{\alpha}}^i$ be the L-bit Gray code for i, and let $\boldsymbol{\alpha}^j$ be the $L+1$-bit Gray code for some $i \in [\![0, 2^{L+1} - 1]\!]$.*

1. *If $j \in [\![0, 2^L - 1]\!]$, then $\boldsymbol{\alpha}^j = [0, \tilde{\boldsymbol{\alpha}}^j]$.*
2. *If $j \in [\![2^L, 2^{L+1} - 1]\!]$, then $\boldsymbol{\alpha}^j = [1, \tilde{\boldsymbol{\alpha}}^i]$, where $i = 2^{L+1} - j - 1$.*

Proof First, for each $i \in [\![0, 2^L - 1]\!]$, let $\tilde{\boldsymbol{\beta}}^i$ be the corresponding L-bit binarization. Similarly, for each $j \in [\![0, 2^{L+1} - 1]\!]$, and let $\boldsymbol{\beta}^j$ be the corresponding $L+1$-bit binarization. Then, by (11) have that $\alpha_1^j = \beta_1^j = 0$ and $\boldsymbol{\beta}^i = [0, \tilde{\boldsymbol{\beta}}^i]$. Applying (11) recursively, this yields $\boldsymbol{\alpha}^i = [0, \tilde{\boldsymbol{\alpha}}^i]$, as desired.

Now, let $i \in [\![2^L, 2^{L+1} - 1]\!]$, and let $\tilde{i} = 2^{L+1} - i - 1$. Since $\tilde{i} \in [\![0, 2^L - 1]\!]$ we have as before that $\boldsymbol{\alpha}^{\tilde{i}} = [0, \tilde{\boldsymbol{\alpha}}^{\tilde{i}}]$. We wish to show that $\boldsymbol{\alpha}^i = [1, \tilde{\boldsymbol{\alpha}}^{\tilde{i}}]$. Now note that

$$\begin{aligned}
\tilde{i} &= 2^{L+1} - i - 1 = 2^{L+1} - \sum_{j=1}^{L+1} 2^{L+1-j}\beta_j - 1 \\
&= 2^{L+1} - 1 - \sum_{j=1}^{L+1} 2^{L+1-j} + \sum_{j=1}^{L+1} 2^{L+1-j}(1 - \beta_j) \\
&= \sum_{j=1}^{L+1} 2^{L+1-j}(1 - \beta_j)
\end{aligned}$$

That is, in the binarization for $\tilde{i}$, we have $\boldsymbol{\beta}^{\tilde{i}} = \boldsymbol{\beta}^i \oplus [1, \ldots, 1]$, so that every bit has been flipped. Observing (13), we see that this can be induced by enforcing $\alpha_1^{\tilde{i}} = 1 - \alpha_1^i$, with all other $\alpha_j^{\tilde{i}} = \alpha_j^i$: flipping the first α-bit induces a flip in all β-bits. Thus, we obtain that $\boldsymbol{\alpha}^i = [1, \tilde{\boldsymbol{\alpha}}^{\tilde{i}}]$, as desired.

Lemma 2 *Let $J \subseteq [\![L]\!]$ and $\overline{\boldsymbol{\alpha}} \in \{0,1\}^J$. Let $X = \{x \in [\![0, 2^L - 1]\!] : \boldsymbol{\alpha}_J^x = \bar{\boldsymbol{\alpha}}\}$. We will write X as $X = \{x_1, \ldots, x_t\}$, ordered such that $x_j < x_{j+1}$. Let $I = [\![L]\!] \setminus J$. Then $\boldsymbol{\alpha}_I^{x_j}$ is a reflected Gray code for the indices j over X. That is, for any $j \in 1, \ldots, t$, we have*

$$\|\boldsymbol{\alpha}_I{}^{x_j} - \boldsymbol{\alpha}_I{}^{x_{j+1}}\|_1 = 1. \tag{15}$$

Furthermore, if $|I| \geq 1$, there exists a $\boldsymbol{\gamma} \in \{0,1\}^{|I|}$ such that, for all $j \in [\![0, t]\!]$, we have

$$\boldsymbol{\alpha}_I^{x_j} \oplus \boldsymbol{\gamma} = \boldsymbol{\alpha}^j_{[\![L-|I|+1, L]\!]}. \tag{16}$$

That is, the modified Gray code after fixing some bits is the original reflected Gray code on $|I|$ bits, with some bits flipped.

Proof We will prove this by induction on L. To enable the use of Lemma 1, we will also prove that, if $|I| \geq 1$, then for all $j \in [\![0, t]\!]$, $i = t - j$, we have

$$x^j = 2^L - x^i - 1. \tag{17}$$

Base case: $L = 1$

For $L = 1$, the possibilities are trivial, as there is only one bit. If we do not fix the bit, then we have $\alpha^j = [j]$ for each $j \in \{0, 1\}$; this sequence of two vectors is trivially a Gray code sequence. This yields the original reflected Gray code for $L = 1$, and so $\boldsymbol{\gamma} = [0]$. Finally, we have $t = 1$, and $x_j = j$, yielding, for $i = 1 - j$, $x_j = 1 - x_i = 2^1 - x^i - 1$, as required.

On the other hand, if we do fix the bit, then there are no pairs of consecutive bits, and (15) holds by default. In this case, we have $|I| = 0$, so that the other results do not apply.

Inductive step:

Let $L = k$, and suppose the desired properties hold for $L = k-1$. Let J, $\overline{\boldsymbol{\alpha}}$ be a choice of fixed bits for $L = k$. First, note that if $|J| = L$, then all bits are fixed and the (15) holds by default, while the others do not apply, as $|I| = 0$.

Next, suppose $I = \{1\}$, so that the newly added bit is the first unfixed bit. Then we have $t = 1$, and $\boldsymbol{\alpha}^{x_j} = [j, \bar{\boldsymbol{\alpha}}]$, with $\boldsymbol{\alpha}_I^{x_j} = [j]$. Thus, defining $\boldsymbol{\gamma} = [0]$, we have that (15) and (16) hold trivially. Finally, by Lemma 1 and the uniqueness of the L-bit Gray code for j, we have that $x_0 = 2^L - x_1 - 1$ and $x_1 = 2^L - x_0 - 1$, as required.

Otherwise, suppose $|J| \leq L - 1$, with $I \neq \{1\}$. Then there are three cases: $1 \notin J$, or $1 \in J$ and either $\bar{\alpha}_1 = 1$ or $\bar{\alpha}_1 = 0$. Regardless of this choice, the corresponding choices $\tilde{J}$ and $\tilde{\boldsymbol{\alpha}}$ for $L = k - 1$ can be attained by defining $\tilde{J} = J \setminus \{1\}$, and defining $\tilde{I}$ and $\tilde{\boldsymbol{\alpha}}$ accordingly. Consider the corresponding sequence $\tilde{X}$ for $L = k-1$. Then, by the inductive hypothesis, the desired results hold for $\tilde{X}$, and as $|\tilde{I}| \geq 1$, we can define a corresponding vector $\tilde{\boldsymbol{\gamma}} \in \{0, 1\}^{|\tilde{I}|}$ so that (16) holds.

Case 1: $1 \in J$ with $\bar{\alpha}_1 = 0$. In this case, define $\tilde{J} = J \setminus \{1\}$ and $\tilde{\alpha} = \bar{\alpha}_{[\![2,L]\!]}$, and define $\tilde{X}$ and $\tilde{\boldsymbol{\gamma}}$ accordingly, noting that $|X| = |\tilde{X}| = t = 2^{k-|J|-1} - 1$. Let $j \in [\![t]\!]$. Then we have by Lemma 1 that $\boldsymbol{\alpha}^{x_j} = [0, \tilde{\boldsymbol{\alpha}}^{\tilde{x}_j}]$, so that $\boldsymbol{\alpha}_I{}^{x_j} = \tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_j}$. Thus, (15) to (17) hold by the induction hypothesis, with $\boldsymbol{\gamma} = \tilde{\boldsymbol{\gamma}}$.

Case 2: $1 \in J$ with $\bar{\alpha}_1 = 1$. In this case, define $\tilde{J} = J \setminus \{1\}$ and $\tilde{\alpha} = \bar{\alpha}_{[\![2,L]\!]}$, and define $\tilde{X}$ accordingly, noting that $|X| = |\tilde{X}| = t$. Let $j \in [\![t]\!]$. Then, since $t - j = 2^{k-|J|-1} - j - 1$, we have by Lemma 1 that $\boldsymbol{\alpha}^{x_j} = [1, \tilde{\boldsymbol{\alpha}}^{\tilde{x}_{t-j}}]$, so that $\boldsymbol{\alpha}_I{}^{x_j} = \tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_{t-j}}$. That is, the sequence of $\boldsymbol{\alpha}_I^{x_j}$'s the sequence of $\tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_j}$'s, but in reversed order. Thus, (15) and (17) hold by the induction hypothesis, as reversing the order of a sequence has no impact on results for consecutive or centrally reflected terms. Furthermore, by Lemma 1 and the induction hypothesis, we have that reversing the order of the sequence corresponds with flipping only the first bit $\alpha_1^{x_j}$, so that we can define $\boldsymbol{\gamma} = \tilde{\boldsymbol{\gamma}} \oplus [1, 0, \ldots, 0]$ to obtain (16).

Case 3: $1 \notin J$, so that the first bit is unfixed. In this case, define $\tilde{J} = J$ and $\tilde{\boldsymbol{\alpha}} = \bar{\boldsymbol{\alpha}}$, and define $\tilde{X}$ accordingly. Then $\tilde{t} = |\tilde{X}| = 2^{k-|J|-1}$. Let $j \in [\![0, t]\!]$

and let $i = t - j = 2^{k-|J|} - j - 1$, so that $j = 2^{k-|J|} - i - 1$ Then, by Lemma 1, we can construct X as follows:

1. If $j \in [\![0, 2^{k-|J|-1} - 1]\!]$, then $\boldsymbol{\alpha}^{x_j} = [0, \tilde{\boldsymbol{\alpha}}^{\tilde{x}_j}]$
2. If $j \in [\![2^{k-|J|-1}, 2^{k-|J|} - 1]\!]$, then $\boldsymbol{\alpha}^{x_j} = [1, \tilde{\boldsymbol{\alpha}}^{\tilde{x}_i}]$.

This yields (17) immediately. Further, define $\boldsymbol{\gamma} = [0, \tilde{\boldsymbol{\gamma}}]$. Then for $j \in [\![0, 2^{k-|J|-1} - 1]\!]$, we have

$$\boldsymbol{\alpha}_I^{x_j} \oplus \boldsymbol{\gamma} = [0, \tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_j}] \oplus [0, \tilde{\boldsymbol{\gamma}}] = [0, \tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_j} \oplus \tilde{\boldsymbol{\gamma}}] = [0, \boldsymbol{\alpha}^j_{[\![L-|I|+2, L]\!]}] = \boldsymbol{\alpha}^j_{[\![L-(|I|)+1, L]\!]},$$

yielding (16). For $j \in [\![2^{k-|J|-1}, 2^{k-|J|-1} - 1]\!]$, defining $i = t - j = 2^{k-|J|} - j - 1 \in [\![\tilde{t} + 1, t]\!]$, we have by Lemma 1 that

$$\boldsymbol{\alpha}_I^{x_j} \oplus \boldsymbol{\gamma} = [1, \tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_i}] \oplus [0, \tilde{\boldsymbol{\gamma}}] = [1, \tilde{\boldsymbol{\alpha}}_I^{\tilde{x}_i} \oplus \tilde{\boldsymbol{\gamma}}] = [1, \boldsymbol{\alpha}^i_{[\![L-|I|+2, L]\!]}] = \boldsymbol{\alpha}^j_{[\![L-(|I|)+1, L]\!]},$$

again yielding (16).

Now, (15) trivially holds for all $j \neq 2^{k-|J|-1} - 1$ as in cases 1 and 2, since the first bits of $\boldsymbol{\alpha}^{x_j}$ and $\boldsymbol{\alpha}^{x_{j+1}}$ match, and exactly one other bit differs by the induction hypothesis. Otherwise, if $j = 2^{k-|J|-1} - 1$, then $i = 2^{k-|J|} - (2^{k-|J|-1} - 1) - 1 = 2^{k-|j|} = j + 1$, and thus x_j and x_{j+1} differ by exactly the first bit α_1 as indicated above. Thus, (15) holds. From these cases, (15) to (17) hold by induction.

Next we show one more property of the code that occurs when extending the Gray code by 1 bit.

Proposition 2 *For an integer $i \in [\![0, 1, \ldots, 2^L - 1]\!]$, let $\boldsymbol{\alpha}^i$ and $\boldsymbol{\beta}^i$ be the L-bit Gray code and binary representations of i. Let $\tilde{\boldsymbol{\alpha}}^{2i}$ and $\tilde{\boldsymbol{\alpha}}^{2i+1}$ be the $(L+1)$-bit Gray codes of the integers $2i$ and $2i+1$.*

Then

$$\tilde{\boldsymbol{\alpha}}^{2i} = \left[\alpha_1^i, \ldots, \alpha_L^i, \beta_L^i\right] \quad \textit{and} \quad \tilde{\boldsymbol{\alpha}}^{2i+1} = \left[\alpha_1^i, \ldots, \alpha_L^i, 1 - \beta_L^i\right].$$

Proof First, note that

$$\tilde{\boldsymbol{\beta}}^{2i} = [\beta_1, \ldots, \beta_L, 0] \textit{ and } \tilde{\boldsymbol{\beta}}^{2i+1} = [\beta_1, \ldots, \beta_L, 1].$$

Then

$$\tilde{\alpha}^{2i}_{L+1} = \tilde{\beta}^{2i}_{L+1} \oplus \tilde{\beta}^{2i}_L = 0 \oplus \beta^i_L = \beta^i_L \text{ and } \tilde{\alpha}^{2i+1}_{L+1} = \tilde{\beta}^{2i}_{L+1} \oplus \tilde{\beta}^{2i}_L = 1 \oplus \beta^i_L = 1 - \beta^i_L. \tag{18}$$

Furthermore, $\tilde{\beta}^{2i}_j = \beta^{2i+1}_j = \beta^i$ for all $j = 1, \ldots, L$, by definition, we have that $\tilde{\alpha}^{2i}_j = \tilde{\alpha}^{2i+1}_j = \alpha^i_j$ for all $j = 1, \ldots, L$.

4 Formulation Strength

The *strength* of a MIP formulation is a commonly used metric to assess its potential computational performance. We will work with the three following notions of strength.

Definition 1 Consider a set $U \subseteq \mathbb{R}^n$. For a formulation $P^{\text{IP}} = \{(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}) \in P^{\text{LP}} : \boldsymbol{z} \in \mathbb{Z}^L\}$, where $P^{\text{LP}} \subseteq \mathbb{R}^{n+d+L}$ and $\text{proj}_{\boldsymbol{u}}(P^{\text{IP}}) = U$, we say the the formulation P^{LP} is

- *sharp* if $\text{proj}_{\boldsymbol{u}}(P^{\text{LP}}) = \text{conv}(U)$.
- *hereditarily sharp* if, for all $I \subseteq [\![L]\!]$ and $\bar{\boldsymbol{z}}_I \in \mathbb{Z}^{|I|}$, we have

$$\text{proj}_{\boldsymbol{u}}(P^{\text{LP}}|_{\bar{\boldsymbol{z}}_I = \boldsymbol{z}_I}) = \text{conv}(\{\boldsymbol{u} \in U : (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}) \in P^{\text{LP}}\}|_{\bar{\boldsymbol{z}}_I = \boldsymbol{z}_I}).$$

- *ideal* if $\text{proj}_{\boldsymbol{z}}(\text{ext}(P^{\text{LP}})) \subseteq \mathbb{Z}^L$.

These definitions closely follow those in [37], except we define hereditary sharpness explicitly in terms of the current branch.

In this section, we explore the strength of our MIP formulation (8). We also draw an interesting connection between how our formulation represents feasible points through a Gray code: in essence, feasible points are represented in the formulation by their L most significant digits in a binary expansion.

For our particular problem, define

$$\begin{aligned} P^{\text{LP}} &:= \{(x, y, \boldsymbol{\alpha}, \boldsymbol{g}) \in [0,1] \times \mathbb{R}^+ \times [0,1]^L \times [0,1]^{L+1} : (8)\} \\ P^{\text{IP}} &:= \{(x, y, \boldsymbol{\alpha}, \boldsymbol{g}) \in [0,1] \times \mathbb{R}^+ \times \{0,1\}^L \times [0,1]^{L+1} : (8)\} \end{aligned} \tag{19}$$

and

$$\begin{aligned} Q^{\text{LP}} &:= \text{proj}_{x,y}(P^{\text{LP}}) \\ Q^{\text{IP}} &:= \text{proj}_{x,y}(P^{\text{IP}}). \end{aligned} \tag{20}$$

First, we show that Q^{IP} is, unfortunately, not ideal.

Example 1 The formulation P^{IP} approximating $y = x^2$ is not ideal.

Proof Consider $L = 2$, $x = 0.25$, $\boldsymbol{\alpha} = [\frac{1}{2}, 1]$, $\boldsymbol{g} = [\frac{1}{2}, 1]$, and $y = 0.25 - 2^{-2}(\frac{1}{2}) - 2^{-4}(0.25) = \frac{11}{64}$. This point is chosen to maximize g_2 along a facet $g_2 \leq 2 \cdot (2_x - \alpha_1)$ of the convex hull. It is an extreme point because it is incident with six facets: $g_2 \leq 1$, $\alpha_2 \leq 1$, $g_2 \leq 2g_1$, $g_1 \leq 2x$, $\alpha_1 - x \leq g_2$, and $y = x - 2^{-2}g_1 + 2^{-4}g_2$. Thus, P^{LP} has a fractional extreme point, and so is not ideal.

Next, we will show that Q^{IP} is sharp. To assist deriving this result, we define the generic sawtooth function $G\colon\ \mathbb{R} \to \mathbb{R}$ as

$$g_i = G(g_{i-1}) := \begin{cases} 2g_{i-1} & g_{i-1} < \frac{1}{2}, \\ 2(1 - g_{i-1}) & g_{i-1} \geq \frac{1}{2}. \end{cases} \tag{21}$$

Theorem 1 *The formulation P^{IP} is sharp for Q^{IP}.*

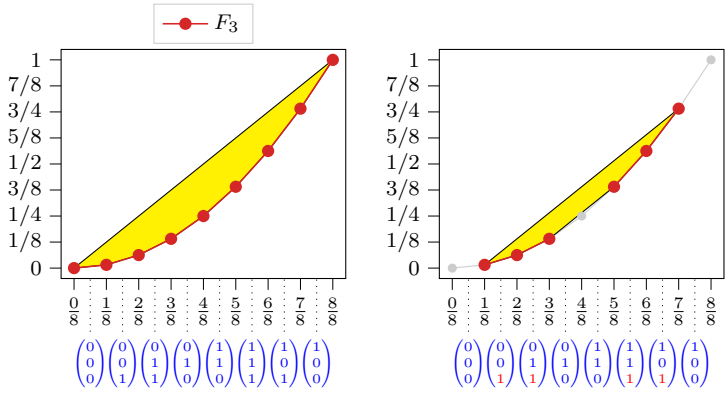


Fig. 3: *Left:* The piecewise linear approximation Q^{IP} in red and the linear relaxation Q^{LP} in yellow filled in. The formulation is sharp because Q^{LP} is the convex hull of Q^{IP}. The vectors $\boldsymbol{\alpha} \in \{0,1\}^3$ below are the corresponding binary α variables for any x value on that interval $(\frac{i}{8}, \frac{i+1}{8})$. *Right:* The branch $Q^{\text{IP}}|_{\alpha_3=1}$ in red and the linear relaxation $Q^{\text{LP}}|_{\alpha_3=1}$ in yellow. This demonstrates hereditary sharpness since it holds that $Q^{\text{LP}}|_{\alpha_3=1}$ is the convex hull of $Q^{\text{IP}}|_{\alpha_3=1}$.

Proof For sharpness, we wish to show that $Q^{\text{LP}} = \text{conv}(Q^{\text{IP}})$. Clearly $Q^{\text{LP}} \supseteq \text{conv}(Q^{\text{IP}})$ from validity of our formulation; therefore, we focus on showing that $Q^{\text{LP}} \subseteq \text{conv}(Q^{\text{IP}})$. We start by fixing some $\bar{x} \in [0,1]$. The result then follows if we can show that the "slice" of Q^{LP} at $\bar{x}$, $Q^{\text{LP}}|_{x=\bar{x}} \coloneqq \{\, y \mid (\bar{x}, y) \in Q^{\text{LP}} \,\}$, is contained in $\text{proj}_x(\text{conv}(Q^{\text{IP}}))$. Since $Q^{\text{LP}}|_{x=\bar{x}} \subset [0,1]$ and is convex, it suffices to show that both its maximum value and minimum value are contained in $\text{conv}(Q^{\text{IP}})$.

First, let $y^* = \max\{y \in Q^{\text{LP}}|_{x=\bar{x}}\}$. From Q^{LP}, we know that $y = x - \sum_{i=1}^{L} 2^{-2i} g_i \leq x$ since $g_i \geq 0$ for all i. Hence, $y^* \leq \bar{x}$. Next, we observe that, as $(0,0), (1,1) \in Q^{\text{IP}}$, convexity in turn implies that $(\bar{x}, \bar{x}) \in \text{conv}(Q^{\text{IP}}) \subseteq Q^{\text{LP}}$. Therefore, $y^* = \bar{x}$ by maximality, and so $(\bar{x}, y^*) \in \text{conv}(Q^{\text{IP}})$.

Next, let $y^* = \min\{y \in Q^{\text{LP}}|_{x=\bar{x}}\}$. From definition, it follows that there is some $\boldsymbol{g}^*$ and $\boldsymbol{\alpha}^*$ such that $(\bar{x}, y^*, \boldsymbol{g}^*, \boldsymbol{\alpha}^*) \in P^{\text{LP}}$. Define G as in (21). If $g_i^* = G(g_{i-1}^*)$ for each $i \in [\![L]\!]$, then we immediately conclude that there exists some $\tilde{\boldsymbol{\alpha}} \in \{0,1\}^L$ such that $(\bar{x}, y^*, \boldsymbol{g}^*, \tilde{\boldsymbol{\alpha}}) \in P^I$. This, in turn, implies that $(\hat{x}, y^*) \in Q^{\text{IP}}$.

Otherwise, take $i \in [\![L]\!]$ as the largest index such that $g_i^* > G(g_{i-1}^*)$. Then, recursively define $\tilde{\boldsymbol{g}}$ such that $\tilde{g}_j = \begin{cases} g_j^* & j < i \\ G(\tilde{g}_{j-1}) & j \geq i \end{cases}$ for each $j \in \{0, \ldots, L\}$. Further, take $\tilde{y} = \bar{x} - \sum_{i=1}^{L} 2^{-2i} \tilde{g}_i$, with $\tilde{\alpha}_i = g_{i-1}$ for all i, inducing only lower bounds of 0 on all g_j. Then $(\tilde{y}, \bar{x}, \tilde{\boldsymbol{g}}, \tilde{\boldsymbol{\alpha}}) \in Q^{\text{LP}}$. We now show that $\tilde{y} < y^*$, contradicting the minimality of y^*.

Let $\varepsilon = \tilde{g}_i - g_i^*$. Note that $G\colon \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant 2. That is, for any $g, g' \in [0,1]$, we have that $|G(g) - G(g')| \leq 2|g - g'|$. Hence, since $|g_i^* - \tilde{g}_i| \leq \varepsilon$ we conclude inductively that $|g_{i+k}^* - \tilde{g}_{i+k}| \leq 2^k \varepsilon$ for each $k \in [\![L - i]\!]$. Thus, we have

$$\begin{aligned}
y^* - \tilde{y} &= \sum_{j=i}^{L} 2^{-2j}(\tilde{g}_j - g_j^*) \\
&\geq 2^{-2i}(\tilde{g}_i - g_i^*) + \sum_{j=i+1}^{L} 2^{-2j}|g_j^* - \tilde{g}_j| \\
&\geq 2^{-2i}\varepsilon - \textstyle\sum_{j=i+1}^{L} 2^{-2j} 2^{j-i}\varepsilon \\
&= 2^{-2i}\varepsilon(1 - 2^i \sum_{j=i+1}^{L} 2^{-j}) \\
&= 2^{-2i}\varepsilon(1 - 2^i(2^{-i} - 2^{-L})) \\
&= \varepsilon(2^{-i-l}) > 0.
\end{aligned}$$

Therefore, $\tilde{y} < y^*$, contradicting the minimality of y^*. From this, we conclude that no such i exists, completing the proof.

We now show the correspondence between our formulation and the Gray code. It is helpful to note that, in P^{IP}, we have $g_i = G(g_{i-1})$, where G is as defined in (21).

Theorem 2 (The MIP formulation follows a Gray code) *Take* $(x, \boldsymbol{g}, \boldsymbol{\alpha}) \in \text{proj}_{x,\boldsymbol{g},\boldsymbol{\alpha}}(P^{\text{IP}}|_{g_L \in (0,1)})$ *for some fixed value* $g_L \in (0,1)$. *Fix some* $j \in [\![0, L-1]\!]$, *and define* $i_j \coloneqq \lfloor 2^{L-j} g_j \rfloor$. *Then* $[\alpha_{j+1}, \ldots, \alpha_L]$ *is the* $L-j$*-bit Gray code* $\boldsymbol{\alpha}^{i_j}$ *for* i_j, *and* $g_j \in (\frac{i_j}{2^{L-j}}, \frac{i_j+1}{2^{L-j}})$.

Proof First, we note that, for each $j \leq L+1$, we have

$$g_j = \begin{cases} \frac{1}{2} g_{j+1} & \alpha_{j+1} = 0, \\ 1 - \frac{1}{2} g_{j+1} & \alpha_{j+1} = 1. \end{cases}$$

We will proceed by induction in a manner similar to the proof for Lemma 2.

Base Case: $j = L - 1$

In this case, if $\alpha_L = 0$, then $g_{L-1} = \frac{1}{2} g_L \in (0, \frac{1}{2})$, and so $i_{L-1} = \lfloor 2 g_{L-1} \rfloor = 0$, whose 1-bit Gray code is $[0] = [\alpha_L]$, as desired. On the other hand, if $\alpha_L = 1$, then $g_{L-1} = 1 - \frac{1}{2} g_L \in (\frac{1}{2}, 1)$, and so $i_{L-1} = \lfloor 2 g_{L-1} \rfloor = 1$, whose 1-bit Gray code is $[1] = [\alpha_L]$, as desired.

Inductive step: Let $j \in [\![0, L-2]\!]$ and suppose the statement holds $j+1$.

If $\alpha_{j+1} = 0$, then $g_j = \frac{1}{2} g_{j+1} \in \left(\frac{i_{j+1}}{2^{L-j}}, \frac{i_{j+1}+1}{2^{L-j}} \right)$ and $2^{L-j} g_j \in (i_{j+1}, i_{j+1} + 1)$. In this case, we have $i_j = \lfloor 2^{L-j} g_j \rfloor = i_{j+1} \in [\![0, 2^{L-j-1} - 1]\!]$. Thus, by Lemma 1, we have that the $L-j$-bit Gray code for i_j is given by $\boldsymbol{\alpha}^{i_j} = [0\ \tilde{\boldsymbol{\alpha}}^{i_{j+1}}]$, where $\tilde{\boldsymbol{\alpha}}^{i_j}$ is the $(L-j-1)$-bit Gray code for i_j, which is $[\alpha_{j+2} \ldots \alpha_L]$ by

the induction hypothesis. This yields $\boldsymbol{\alpha}^{i_j} = [0\ \alpha_{j+2} \dots \alpha_L] = [\alpha_{j+1} \dots \alpha_L]$ as desired. Further, observing the bounds we derived for g_j, we have $g_j \in (\frac{i_j}{2^{L-j}}, \frac{i_j+1}{2^{L-j}})$.

On the other hand, if $\alpha_{j+1} = 1$, then $g_j = 1 - \frac{1}{2}g_{j+1} \in (1 - \frac{i_{j+1}+1}{2^{L-j}}, 1 - \frac{i_{j+1}}{2^{L-j}})$ and $2^{L-j}g_j \in (2^{L-j} - i_{j+1} - 1, 2^{L-j} - i_{j+1})$. Thus, we have that $i_j = 2^{L-j} - i_{j+1} - 1 \in [\![2^{L-j-1}, 2^{L-j} - 1]\!]$. Thus, by Lemma 1, computing $\tilde{i}_j = 2^{L-j} - 1 - i_j = i_{j+1}$, we have that the $(L-j)$-bit Gray code for i_j is $\boldsymbol{\alpha}^{i_j} = [1\ \tilde{\boldsymbol{\alpha}}^{\tilde{i}_j}]$, where $\tilde{\boldsymbol{\alpha}}^{\tilde{i}_j}$ is the $(L-j-1)$-bit Gray code for $\tilde{i}_j$, which by the induction hypothesis is given as $[\alpha_{j+2} \dots \alpha_L]$. This yields $\boldsymbol{\alpha}^{i_j} = [1\ \alpha_{j+2} \dots \alpha_L] = [\alpha_{j+1} \dots \alpha_L]$ as desired. Further, observing the bounds we derived for g_j, we have $g_j \in (\frac{i_j}{2^{L-j}}, \frac{i_j+1}{2^{L-j}})$.

Thus, with these cases, the result holds by induction.

Note that, if $g_L \in \{0, 1\}$ (so that $x \in 2^{-L}\mathbb{Z} \cap (0, 1)$), the same Gray codes from Theorem 2 can be used as when $g_L \in (0, 1)$. The primary difference is that there two choices for this Gray code when $x \in 2^{-L}\mathbb{Z} \cap (0, 1)$. This dichotomy stems from the fact that we will obtain $g_j = \frac{1}{2}$ from some j, introducing ambiguity in the choice of α_j.

4.1 Hereditary Sharpness

In this section, we prove hereditary sharpness of the formulation P^{IP}.

Let L be a nonnegative integer, and define the sets $P^{\text{IP}}, P^{\text{LP}}$, Q^{IP}, and Q^{LP} as before. Suppose we fix some subset of the binary variables to $\boldsymbol{\alpha}_I = \bar{\boldsymbol{\alpha}}_I \in \{0, 1\}^{|I|}$ for some set $I \subseteq L$. Furthermore, define $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I} := P^{\text{IP}}|_{\boldsymbol{\alpha}_I = \bar{\boldsymbol{\alpha}}_I} := \{(x, y, \boldsymbol{g}, \boldsymbol{\alpha}) \in P^{\text{IP}} | \alpha_i = \bar{\alpha}_i \text{ for } i \in I\}$, and similarly for Q^{LP}, Q^{IP}, and P^{LP}. We then wish to show $Q^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I} = \text{conv}(Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$. We will use this notation for the remainder of this section. An example demonstrating the hereditary sharpness of the formulation is shown in Figure 3

To study this relationship in more detail, we in particular wish to study $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$. Let $(x, y, \boldsymbol{g}, \boldsymbol{\alpha}) \in P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$. For each $i \in I$, then g_{i-1} relates to g_i via the linear function

$$g_i = 2g_{i-1}(1 - \bar{\alpha}_i) + 2(1 - g_{i-1})\bar{\alpha}_i. \tag{22}$$

Alternatively, for each $i \in [\![L]\!] \backslash I$, the relationship can be written as

$$2|g_{i-1} - \alpha_i| \leq g_i \leq \min\{2g_{i-1}, 2(1 - g_{i-1})\}.$$

In this form, simply setting each $\alpha_i = g_{i-1}$ yields the least restrictive possible lower-bound of 0 on g_i in terms of g_{i-1}. Thus, making this choice, we find that $\text{proj}_{x,y,\boldsymbol{g}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I})$ can be expressed via the constraints

$$\begin{aligned}
y &= g_0 - \textstyle\sum_{i=1}^{L} 2^{-2i} g_i & & \\
g_0 &= x & & \\
g_i &\leq 2g_{i-1} & & i \in [\![L]\!], i \notin I \\
g_i &\leq 2(1 - g_{i-1}) & & i \in [\![L]\!], i \notin I \\
g_i &= 2g_{i-1}(1 - \bar{\alpha}_i) + 2(1 - g_{i-1})\bar{\alpha}_i & & i \in I \\
g_i &\in [0, 1] & & i \in [\![L]\!]
\end{aligned} \tag{23}$$

while, after combining the linear constraints with the new constraints (22) fixing variables α_i for $i \in I$, $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$ can be written as

$$\begin{aligned}
y &= g_0 - \textstyle\sum_{i=1}^{L} 2^{-2i} g_i \\
g_0 &= x \\
2(g_{i-1} - \alpha_i) \le g_i &\le 2g_{i-1} && i \in [\![L]\!], i \notin I \\
2(\alpha_i - g_{i-1}) \le g_i &\le 2(1 - g_{i-1}) && i \in [\![L]\!], i \notin I \\
g_i &= 2g_{i-1}(1 - \bar{\alpha}_i) + 2(1 - g_{i-1})\bar{\alpha}_i && i \in I \\
g_i &\in [0,1] && i \in [\![L]\!] \\
\alpha_i &\in \{0,1\} && i \in [\![L]\!], i \notin I
\end{aligned} \tag{24}$$

For convenience, for all $i \in I$, define

$$G_i(g_{i-1}, \alpha_i) := 2g_{i-1}(1 - \alpha_i) + 2(1 - g_{i-1})\alpha_i. \tag{25}$$

For $i \in [\![L]\!]$, define the shorthand $G_i(g_{i-1}) := G_i(g_{i-1}, \bar{\alpha}_i)$ if $i \in I$ and $G_i = G$ otherwise. Then by the construction of $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$, we have that $\boldsymbol{g} \in \text{proj}_{\boldsymbol{g}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$ if and only if for all $i \in [\![L]\!]$, we have $g_i = G_i(g_{i-1})$ and $g_i \in [0,1]$.

For all $i \in [\![0, L]\!]$, the below proposition explores how to compute the feasible region for g_i, $\text{proj}_{g_i}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) =: [a_i, b_i]$, while establishing that $\text{proj}_{g_i}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = \text{conv}(\text{proj}_{g_i}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}))$.

Lemma 3 (Bounds in Projection) *For all $i \in [\![0, L]\!]$ and $I \subseteq [\![L]\!]$, we have $\text{proj}_{g_i}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = \text{conv}(\text{proj}_{g_i}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) =: [a_i, b_i] \neq \emptyset$. Furthermore, $[a_L, b_L] = [0,1]$, and $[a_{i-1}, b_{i-1}]$ can be computed from $[a_i, b_i]$ as*

$$[a_{i-1}, b_{i-1}] = \begin{cases} [\frac{1}{2}a_i, \frac{1}{2}b_i] & \text{if } i \in I \text{ and } \bar{\alpha}_i = 0, \\ [1 - \frac{1}{2}b_i, 1 - \frac{1}{2}a_i] & \text{if } i \in I \text{ and } \bar{\alpha}_i = 1, \\ [\frac{1}{2}a_i, 1 - \frac{1}{2}a_i] & \text{if } i \notin I. \end{cases} \tag{26}$$

Note that in the last case $a_i \le \frac{1}{2}$ and $b_i \ge \frac{1}{2}$.

Proof We proceed by induction.

Base Case: $i = L$

In this case, Theorem 2 establishes that, even if $\hat{I} = [\![L]\!]$ with some corresponding $\hat{\boldsymbol{\alpha}}_{\hat{I}} \in \{0,1\}^L$, we have $[0,1] = \text{proj}_{g_i}(P^{\text{IP}}_{\hat{\boldsymbol{\alpha}}_{\hat{I}}})$, yielding

$$\begin{aligned}
[0,1] = \text{proj}_{g_L}(P^{\text{IP}}_{\hat{\boldsymbol{\alpha}}_{\hat{I}}}) &\subseteq \text{conv}(\text{proj}_{g_L}(P^{\text{IP}}_{\hat{\boldsymbol{\alpha}}_{\hat{I}}})) \\
&\subseteq \text{conv}(\text{proj}_{g_L}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) \subseteq \text{proj}_{g_L}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) \subseteq [0,1],
\end{aligned}$$

and so $\text{conv}(\text{proj}_{g_L}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) = \text{proj}_{g_L}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [0,1]$, as required.

Inductive step:

Let $i \in [\![L]\!]$, and suppose that $\text{conv}(\text{proj}_{g_i}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) = \text{proj}_{g_i}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [a_i, b_i]$. Then, observing (23) and (24), we find that there are three cases.

Case 1: If $i \in I$ and $\bar{\alpha}_i = 0$, then in both $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$ and $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$, we have $g_{i-1} = \frac{1}{2}g_i$, yielding

$$\text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) \subseteq \text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [\tfrac{1}{2}a_i, \tfrac{1}{2}b_i].$$

Furthermore, $a_i, b_i \in \text{proj}_{g_i}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$ yields $\frac{1}{2}a_i, \frac{1}{2}b_i \in \text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$. This implies $[\frac{1}{2}a_i, \frac{1}{2}b_i] \subseteq \text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}))$, yielding

$$\text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) = \text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [1 - \tfrac{1}{2}a_i, 1\tfrac{1}{2}b_i]$$

Note that $[a_i, b_i] \neq \emptyset \Rightarrow [\frac{1}{2}a_i, \frac{1}{2}b_i] \neq \emptyset$, and that $[a_i, b_i] \subseteq [0, 1] \Rightarrow [\frac{1}{2}a_i, \frac{1}{2}b_i] \subseteq [0, 1]$.

Case 2: If $i \in I$ and $\bar{\alpha}_i = 1$, then in both $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$ and $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$, we have $g_i = 1 - \frac{1}{2}g_i$, yielding

$$\text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) \subseteq \text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [1 - \tfrac{1}{2}b_i, 1 - \tfrac{1}{2}a_i].$$

Furthermore, $a_i, b_i \in \text{proj}_{g_i}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$ yields $1 - \frac{1}{2}b_i, 1 - \frac{1}{2}a_i \in \text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$. This implies $[1 - \frac{1}{2}b_i, 1 - \frac{1}{2}a_i] \subseteq \text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}))$, yielding

$$\text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) = \text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [1 - \tfrac{1}{2}b_i, 1 - \tfrac{1}{2}a_i].$$

Note that $[a_i, b_i] \neq \emptyset \Rightarrow [1 - \frac{1}{2}b_i, 1 - \frac{1}{2}a_i] \neq \emptyset$, and $[a_i, b_i] \subseteq [0, 1] \Rightarrow [1 - \frac{1}{2}b_i, 1 - \frac{1}{2}a_i] \subseteq [0, 1]$.

Case 3: If $i \notin I$, then $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$ and $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$ model different relations between g_i and g_{i-1}.

In $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$, we have

$$\begin{aligned} g_i &\leq 2g_{i-1} \\ g_i &\leq 2(1 - g_{i-1}), \end{aligned}$$

which can be written as

$$\begin{aligned} g_{i-1} &\geq \tfrac{1}{2}g_i \geq \tfrac{1}{2}a_i \\ g_{i-1} &\leq 1 - \tfrac{1}{2}g_i \leq 1 - \tfrac{1}{2}a_i. \end{aligned}$$

Further, as $a_i \in \text{proj}_{g_i}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I})$, we have that

$$\text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [\tfrac{1}{2}a_i, 1 - \tfrac{1}{2}a_i]),$$

where $a_i \in [0, 1]$ implies $\frac{1}{2}a_i \in [0, \frac{1}{2}]$, so that $[\frac{1}{2}a_i, 1 - \frac{1}{2}a_i] \neq \emptyset$. Thus,

$$\text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) \subseteq \text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [\tfrac{1}{2}a_i, 1 - \tfrac{1}{2}a_i].$$

To show $[\frac{1}{2}a_i, 1 - \frac{1}{2}a_i] \subseteq \text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$, we have only to show that the endpoints are in $\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$. To this end, let $g_i = a_i$. In $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$, we can choose either $\alpha_i = 0$ or $\alpha_i = 1$. If $\alpha_i = 0$, we have that $g_{i-1} = \frac{1}{2}g_i = \frac{1}{2}a_i$, while if $\alpha_i = 1$, we have $g_{i-1} = 1 - \frac{1}{2}g_i = 1 - \frac{1}{2}a_i$. Thus, we have $[\frac{1}{2}a_i, 1 - \frac{1}{2}a_i] \subseteq \text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}))$, yielding

$$\text{conv}(\text{proj}_{g_{i-1}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) = \text{proj}_{g_{i-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = [\tfrac{1}{2}a_i, 1 - \tfrac{1}{2}a_i] \neq \emptyset,$$

as desired.

Note that $\text{conv}(\text{proj}_x(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})) = \text{proj}_x(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I})$ is a direct corollary of the above lemma. This fact will be helpful during the proof of hereditary sharpness. Next, to prove hereditary sharpness, we want to show that if we fix some g_i to either a_i or b_i, then for each $j > i$, we have that g_j has only one feasible solution e_j in $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$, where $e_j \in \{a_i, b_i\}$.

Lemma 4 (Single solution at endpoints) *For all $i \in [\![0, L]\!]$ and $\hat{g}_i \in \{a_i, b_i\}$, we have for all $j \geq i$ that $\text{proj}_{g_j}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_i=\hat{g}_i}) = \{e_j\}$, where $e_j \in \{a_j, b_j\}$.*

Proof We will proceed by induction. Let $\hat{g}_i \in \{a_0, b_0\}$. Fortunately, the base case is trivial, since we have chosen $g_i \in \{a_i, b_i\}$.

Thus, for an induction, let $j \in [\![i+1, L]\!]$, and assume that $\text{proj}_{g_{j-1}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_i=\hat{g}_i}) = \{e_{j-1}\} \subset \{a_{j-1}, b_{j-1}\}$. Then there are three cases.

Case 1: If $j \in I$ and $\bar{\alpha}_j = 0$, then we have $g_j = 2g_{j-1}$, so that $\text{proj}_{g_j}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_i=\hat{g}_i}) = \{2e_{j-1}\} =: \{e_j\}$. Now, by Lemma 3, we have that $a_j = 2a_{j-1}$ and $b_j = 2b_{j-1}$, so that $e_{j-1} \in \{a_{j-1}, b_{j-1}\} \Rightarrow e_j \in \{a_j, b_j\}$.

Case 2: If $j \in I$ and $\bar{\alpha}_j = 1$, then we have $g_j = 2(1 - g_{j-1})$, so that $\text{proj}_{g_j}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_j=\hat{g}_j}) = \{2(1 - e_{j-1})\} =: \{e_j\}$. Now, by Lemma 3, we have that $a_j = 2(1 - b_{j-1})$ and $b_j = 2(1 - a_{j-1})$, so that $e_{j-1} \in \{a_{j-1}, b_{j-1}\} \Rightarrow e_j \in \{a_j, b_j\}$.

Case 3: If $j \notin I$, then we have by Lemma 3 $a_j = 2a_{j-1} = 2(1 - b_{j-1})$, with $a_j \leq \frac{1}{2}$ and $b_j \geq \frac{1}{2}$. Further, we have in $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$ that

$$\begin{aligned} g_j &\leq 2g_{j-1} = 2e_{j-1}, \\ g_j &\leq 2(1 - g_{j-1}) = 2(1 - e_{j-1}). \end{aligned}$$

Now, if $e_{j-1} = a_{j-1}$, then $2e_{j-1} = 2a_{j-1} \leq 1$, while $2(1-e_{j-1}) = 2(1-a_{j-1}) \geq 1$. Thus, these bounds consolidate to $a_j \leq g_j \leq 2a_{j-1} = a_j$, which implies $g_j = a_j$.

On the other hand, if $e_{j-1} = b_{j-1}$, then $2e_{j-1} = 2b_{j-1} \geq 1$, while $2(1 - e_{j-1}) = 2(1 - b_{j-1}) \leq 1$. Thus, these bounds consolidate to $a_j \leq g_j \leq 2(1 - b_{j-1}) = a_j$, which implies $g_j = a_j$. Thus, in this case, we have $\text{proj}_{g_j}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_i=\hat{g}_i}) = \{a_j\} =: \{e_j\}$. This completes the proof.

Now, computing all a_i and b_i via Lemma 3, we obtain the following form for $\text{proj}_{x,y,\boldsymbol{g}}\{P^{\text{LP}}|_{I,\bar{\boldsymbol{\alpha}}_I}\}$:

$$\begin{aligned} y &= g_0 - \textstyle\sum_{i=1}^{L} 2^{-2i} g_i & & \\ g_0 &= x & & \\ g_i &\leq 2g_{i-1} & & i \in [\![L]\!] \backslash I \\ g_i &\leq 2(1 - g_{i-1}) & & i \in [\![L]\!] \backslash I \\ g_i &= 2g_{i-1}(1 - \bar{\alpha}_i) + 2(1 - g_{i-1})\bar{\alpha}_i & & i \in I \\ g_i &\in [a_i, b_i] & & i \in [\![L]\!]. \end{aligned} \tag{27}$$

Now, note that, for each $i \in [\![L]\!]$, g_i has negative coefficient in first equation of (27). Note also that this is the only constraint in (27) involving y. Thus, if we are to minimize over y, then we implicitly maximize g_i, and so g_i will tend

towards its upper bound. Furthermore, it turns out that, in any y-minimal or y-maximal solution in $P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I}$ given some fixed value of x, each g_i can be explicitly computed from g_{i-1} using only the constraints from (27) directly connecting g_i and g_{i-1}. We refer to this as the *greedy* solution property described in Lemma 5 below, and it holds due to the rapid decay of coefficients of g_i's.

For each $i \in [\![L]\!] \setminus I$, let b_i be as in Lemma 3, and define

$$u_i(g_{i-1}) \coloneqq \min\{b_i, 2g_{i-1}, 2(1-g_{i-1})\}. \tag{28}$$

Lemma 5 *Let a_i, b_i as in Lemma 3, and let $\hat{x} \in [a_0, b_0]$. Define*

$$(y^*, \boldsymbol{g}^*) = \arg\min\{y \,:\, (y, \boldsymbol{g}) \in \mathrm{proj}_{y,\boldsymbol{g}}(P^{\mathrm{LP}}_{I,\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}})\}, \tag{29a}$$

$$(y^\star, \boldsymbol{g}^\star) = \arg\max\{y \,:\, (y, \boldsymbol{g}) \in \mathrm{proj}_{y,\boldsymbol{g}}(P^{\mathrm{LP}}_{I,\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}})\}. \tag{29b}$$

Then, for $i \notin I$, we have

$$g_i^* = u_i(g_{i-1})$$
$$g_i^\star = a_i.$$

That is, one of the upper bounds is tight for each g_i when maximizing y, while the domain lower bound is tight for g_i when minimizing y.

Proof Let $\hat{x} \in [a_0, b_0]$. Then $P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}} \neq \emptyset$ by Lemma 3.

We begin by proving that $g_i^* = u_i(g_{i-1}^*)$ for all $i \notin I$. Suppose for a contradiction that, for some subset $j \in J \subseteq [\![L]\!] \setminus I$ and $\varepsilon_j > 0$, we have $g_j^* = u_j(g_{j-1}^*) - \varepsilon_j$. Let i be maximal in J, so that $g_j^* = u_j(g_{j-1}^*)$ for all $j > i$. Then we have $i \notin I$, since otherwise we have $g_i^* = 2g_{i-1}(1-\bar{\alpha}_i) + 2(1-g_{i-1}) = u_j(g_{j-1}^*)$ from (27).

Let $\tilde{g}_j = u_j(g_{j-1})$ for all $j \geq i$. for convenience, define

$$\tilde{\boldsymbol{g}} = (g_0^*, \ldots g_{i-1}^*, \tilde{g}_i, \ldots, \tilde{g}_L).$$

To show that this choice is feasible, i.e., $\tilde{\boldsymbol{g}} \in \mathrm{proj}_{\boldsymbol{g}}(P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I})$, we make an inductive observation. First, note that $\tilde{g}_{i-1} = g_{i-1}^* \in [a_{i-1}, b_{i-1}]$ by Lemma 3, with $\tilde{\boldsymbol{g}}_{[\![0,i-1]\!]} \in \mathrm{proj}_{\boldsymbol{g}_{[\![0,i-1]\!]}}(P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I})$. Furthermore, for any j, $\tilde{\boldsymbol{g}}_{[\![0,j-1]\!]} \in \mathrm{proj}_{\boldsymbol{g}_{[\![0,j-1]\!]}}(P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I})$ implies that there exists some $g_j \in \mathrm{proj}_{g_j}(P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{\boldsymbol{g}_{[\![0,j-1]\!]}=\tilde{\boldsymbol{g}}_{[\![0,j-1]\!]}}) \subseteq [a_j, b_j]$, and $\tilde{g}_j$ is the largest such value by construction, yielding $\tilde{g}_j \in [a_j, b_j]$ and $\tilde{g}_{[\![0,j]\!]} \in \mathrm{proj}_{\boldsymbol{g}_{[\![0,j]\!]}}(P^{\mathrm{LP}}_{\bar{\boldsymbol{\alpha}}_I})$.

Now, we have by definition that $\tilde{g}_i - g_i^* = \varepsilon_i$. Furthermore, observe that for all $j > i$, $u_j(g_{j-1})$ is Lipschitz continuous with Lipschitz constant 2, yielding

$$|\tilde{g}_j - g_j^*| = |u_j(\tilde{g}_{j-1}) - u_j(g_{j-1}^*)| \leq 2|\tilde{g}_{j-1} - g_{j-1}^*|$$

Applying this recursively yields, for all $j > i$,

$$|\tilde{g}_j - g_j^*| \leq 2^{j-i}\varepsilon.$$

Note that this implies that, for $j > i$, we have $\tilde{g}_j - g^*_j \geq -2^{j-i}\varepsilon$. Then we have

$$\begin{aligned} y^* - \tilde{y} &= 2^{-2i}(\tilde{g}_i - g^*_i) + \textstyle\sum_{j=i+1}^{L} 2^{-2j}(\tilde{g}_j - g^*_j) \\ &\geq 2^{-2i}\varepsilon + \textstyle\sum_{j=i+1}^{L} 2^{-2j}(-2^{j-i}\varepsilon) \\ &= 2^{-i}\varepsilon \left(2^{-i} - \textstyle\sum_{j=i+1}^{L} 2^{-j}\right) \\ &= 2^{-i}\varepsilon(2^{-i} - (2^{-i} - 2^{-L})) \\ &= 2^{-(i+L)}\varepsilon. \end{aligned} \tag{30}$$

However, this is a contradiction: it implies $\tilde{y} < y^*$, but y^* was optimal! Thus, all g_i must take on their upper-bounds given g_{i-1}.

Next, we prove that $g^\star_i = a_i$ for all $i \notin I$. The idea behind the proof is identical, with a notable simplifying difference: there is only one (constant) lower-bound a_i on each g_i for $i \notin I$. Thus, when enforcing that each $g^\star_j = a_i$ for $j > i$, $j \notin I$ with $g_i = a_i + \varepsilon$, the shift from $g^\star_i$ to $\tilde{g}_i$ effects no change in g_j, $j > i$. That is, if $i + 1 \notin I$, $\tilde{g}_{i+1} - g^\star_{i+1} = 0$, yielding $\tilde{g}_j - g^\star_j = 0$ for $j \geq i + 1$, so that (31) simplifies to

$$\tilde{y} - y^\star = 2^{-2i}(g^*_i - \tilde{g}_i) = 2^{-2i}\varepsilon, \tag{31}$$

implying $\tilde{y} > y^\star$, a contradiction. On the other hand, if $i + 1 \in I$, let $k = \min\{j \in L - I : j \geq i + 2\}$. Then, for each $j \in [\![i + 1, \ldots, k - 1]\!]$, we have $|\tilde{g}_i - g^\star_i| = 2^{j-i}\varepsilon$. Furthermore, as $k \notin I$, we have $\tilde{g}_k = a_k = g^\star_k$, so that $\tilde{g}_k = g^\star_k$ for all $j > k$, yielding

$$\begin{aligned} \tilde{y} - y^\star &= 2^{-2i}(g^\star_i - \tilde{g}_i) + \textstyle\sum_{j=i+1}^{k-1} 2^{-2j}(\tilde{g}_j - g^*_j) \\ &\geq 2^{-2i}\varepsilon + \textstyle\sum_{j=i+1}^{k-1} 2^{-2j}(-2^{j-i}\varepsilon) \\ &= 2^{-i}\varepsilon \left(2^{-i} - \textstyle\sum_{j=i+1}^{k-1} 2^{-j}\right) \\ &= 2^{-i}\varepsilon(2^{-i} - (2^{-i} - 2^{-(k-1)})) \\ &= 2^{-(i+k-1)}, \varepsilon \end{aligned} \tag{32}$$

again implying $\tilde{y} > y^\star$, a contradiction.

Observation 1 Since each u_i, defined in (28), is a continuous piecewise linear function, Lemma 5 recursively implies that each g^*_i is continuous in x, and is a function of g^*_{i-1}.

As a corollary to Lemma 4 and Lemma 5, we have that, for the optimal solution to the minimization problem in Lemma 5, we have that $g_j = b_j < G(g_{j-1})$ for exactly one value of j if $\hat{x}$ is not feasible in $P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$.

Corollary 1 *For all $\hat{x} \in \text{proj}_x(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) \setminus \text{proj}_x(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$, define $(y^*, \boldsymbol{g}^*)$ as in (29a). Then we have for exactly one $j \in [\![L]\!] \setminus I$ that $g^*_j = b_j < G(g^*_{i-1})$.*

Furthermore, let $x^1, x^2 \in \text{proj}_x P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$ be such that for all $\lambda \in (0, 1)$, we have $\hat{x} := \lambda x^1 + (1 - \lambda)x^2 \notin \text{proj}_x P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I} = \emptyset$. Then the uniquely determined j discussed above is the same for all $\hat{x} \in (x^1, x^2)$ where (x^1, x^2) denotes the open interval between x^1 and x^2.

Proof Consider the first $j \in [\![L]\!] \setminus I$ for which $g_j^* = b_j$. Such a j exists; otherwise, Lemma 5 would give $g_i^* = G_i(g_{i-1})$ for all $i \in [\![L]\!]$, a contradiction on the choice of $\hat{x}$ since there is a corresponding feasible choice of $\boldsymbol{\alpha}^* \in \{0,1\}^L$. Then, by Lemma 4, the set $\text{proj}_{\boldsymbol{g}_{[\![j+1,L]\!]}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_j=g_j^*})$ consists of only a single point, for which each $g_i \in \{a_i, b_i\}$ for $i \geq j+1$. Furthermore, by Lemma 4, this point is also in $\text{proj}_{\boldsymbol{g}_{[\![j+1,L]\!]}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}|_{g_j=g_j^*})$ so that $g_i^* = G_i(g_{j-1}^*)$ for all $i \geq j+1$. Further, if $g_j^* = G(g_{j-1}^*) = b_j$, then we would obtain $\boldsymbol{g}^* \in \text{proj}_{\boldsymbol{g}} P^{\text{IP}}_{x=\hat{x}}$, a contradiction on the choice of $\hat{x}$. Thus, as $g_j^* \leq G(g_{j-1}^*)$ by Lemma 5, we must have that $g_j^* = b_j < G(g_{j-1}^*)$.

Now, let $x^1, x^2 \in \text{proj}_x P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$ be such that $(x^1, x^2) \cap \text{proj}_x P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I} = \emptyset$. Suppose that there is some $\hat{x} \in (x^1, x^2)$ such that, for all sufficiently small $\varepsilon > 0$, we have that the value j^1 in for $x - \varepsilon$ is different from the value j^2 for $x + \varepsilon$. In this case, since the g_j^* is continuous in x and g_{j-1}^*, we have at $\hat{x}$ that both $g_{j_1}^* = b_{j_1} = G_{j_1}(g_{j_1-1}^*)$ and $g_{j_2}^* = b_{j_2} = G_{j_2}(g_{j_2}^*)$. Furthermore, for all other $i \in [\![L]\!] \setminus I$, we have by continuity that $g_i^* = G_i(g_{i-1}^*)$, yielding $g_i^* = G_i(g_{i-1}^*)$ for all i. This yields $\boldsymbol{g}^* \in \text{proj}_{\boldsymbol{g}} P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}$, a contradiction on the choice of $\hat{x}$.

With this greedy solution property and the following corollary, we are ready to prove the hereditary sharpness of P^{IP} as a formulation for Q^{IP}. It is helpful to note that, for the minimizer solutions in Lemma 5, $g_i^* < b_i$ implies $g_i^* = G_i(g_{i-1})$. Furthermore, if $\boldsymbol{g} \in \text{proj}_{\boldsymbol{g}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I})$ and $g_i^* = G_i(g_{i-1})$ for all $i \in [\![L]\!] \setminus I$, then we have $\boldsymbol{g} \in \text{proj}_{\boldsymbol{g}}(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$.

Theorem 3 *Q^{IP} is hereditarily sharp.*

Proof Let a_0, b_0 as in Lemma 3, so $[a_0, b_0] = \text{proj}_x(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}) = \text{conv}(\text{proj}_x(P^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}))$.

Let $\hat{x} \in [a_0, b_0]$. We need to show that $Q^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I} = \text{conv}(Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$. Since both sets are compact convex sets in $\mathbb{R}^2$, it suffices to show that $Q^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}} = \text{conv}(Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})|_{x=\hat{x}}$. In this vein, define we define the upper and lower bounds in y for each set

$$[y_l^{\text{LP}}(\hat{x}), y_u^{\text{LP}}(\hat{x})] = \left[\min\{y : (x,y) \in Q^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}}\}, \max\{y : (x,y) \in Q^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}}\}\right],$$
$$[y_l^{\text{IP}}(\hat{x}), y_u^{\text{IP}}(\hat{x})] = \left[\min\{y : (x,y) \in \text{conv}(Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})|_{x=\hat{x}}\}, \max\{y : (x,y) \in \text{conv}(Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})|_{x=\hat{x}}\}\right].$$

We will show that the upper and lower bounds coincide.

Lower bounds We begin by showing that $y_l^{\text{IP}}(x) = y_l^{\text{LP}}(x)$. Since $\hat{x} \in \text{proj}_x(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I})$, we have two cases. If $\hat{x} \in \text{proj}_x(Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I})$, then due to sharpness by Theorem 1, we have

$$\begin{aligned}
\min\{y : y \in Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}}\} &\geq \min\{y : y \in Q^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}}\} \\
&\geq \min\{y : y \in Q^{\text{LP}}|_{x=\hat{x}}\} \\
&= \min\{y : y \in Q^{\text{IP}}|_{x=\hat{x}}\} \\
&= \min\{y : y \in Q^{\text{IP}}_{\bar{\boldsymbol{\alpha}}_I}|_{x=\hat{x}}\},
\end{aligned}$$

and so the result holds. In order, the four relations hold by: relaxation; relaxation; sharpness; and the fact that $P^{\text{IP}}|_{x=\hat{x}}$ consists of a single point for feasible $\hat{x}$, so that the restriction does not change the optimal solution.

Otherwise, $\hat{x} \notin \text{proj}_x(Q^{\text{IP}}_{\tilde{\boldsymbol{\alpha}}_I})$. Let

$$(y^*, \boldsymbol{g}^*) := \arg\min\{y : (y, \boldsymbol{g}) \in \text{proj}_{y,\boldsymbol{g}}(P^{\text{LP}}_{\tilde{\boldsymbol{\alpha}}_I}|_{x=\hat{x}})\}$$

and let $x^1 < \hat{x}$ and $x^2 > \hat{x}$ be the closest lower and upper bounds to $\hat{x}$ in $\text{proj}_x(Q^{\text{IP}}_{\tilde{\boldsymbol{\alpha}}_I})$ (such points exist by Lemma 3). Let $(x^1, y^1, \boldsymbol{g}^1, \boldsymbol{\alpha}^1), (x^2, y^2, \boldsymbol{g}^2, \boldsymbol{\alpha}^2) \in P^{\text{IP}}_{\tilde{\boldsymbol{\alpha}}_I}$ be chosen such that $\boldsymbol{\alpha}^1 \in \text{proj}_{\boldsymbol{\alpha}} P^{\text{IP}}_{\tilde{\boldsymbol{\alpha}}_I}|_{x\in(x^1-2^{-L},x^1)}$ and $\boldsymbol{\alpha}^1 \in \text{proj}_{\boldsymbol{\alpha}} P^{\text{IP}}_{\tilde{\boldsymbol{\alpha}}_I}|_{x\in(x^2,x^2+2^{-L})}$. According to Theorem 2, $\boldsymbol{\alpha}^1$ and $\boldsymbol{\alpha}^2$ are the Gray codes for some integers i_j and i_{j+1}. By Lemma 2, we know that there exists exactly one index $k \in [\![L]\!]$ such that $\alpha^1_i = \alpha^2_i$ for all $i \neq k$ and $\alpha^1_k = 1 - \alpha^2_k$.

It follows that $\boldsymbol{g}^1$ and $\boldsymbol{g}^2$ satisfy the equations

$$g_i = G_i(g_{i-1}, \alpha^1_i) = 2g_{i-1}(1 - \alpha^1_i) + 2(1 - g_{i-1})\alpha^1_i \qquad i \in [\![L]\!] \setminus k. \tag{33}$$

Furthermore, by Lemma 5, for the y-minimal solution at all three x-values, we have that all g_i, $i \in [\![L]\!] \setminus I$ take on $u_i(g_{i-1}) = \min\{2g_{i-1}, 2(1-g_{i-1}), b_i\}$. This function is linear if $i \in I$; otherwise, it is the minimum of three functions: two linear, and one constant.

Choose $\lambda \in [0, 1]$ such that $\hat{x} = \lambda x^1 + (1 - \lambda)x^2$ and define $\hat{y}$ and $\hat{\boldsymbol{g}}$ by $(\hat{x}, \hat{y}, \hat{\boldsymbol{g}}) = \lambda(x^1, y^1, \boldsymbol{g}^1) + (1 - \lambda)(x^2, y^2, \boldsymbol{g}^2)$. By convexity of $P^{\text{LP}}_{\tilde{\boldsymbol{\alpha}}_I}$, the point $(\hat{x}, \hat{y}, \hat{\boldsymbol{g}})$ is in $\text{proj}_{x,y,\boldsymbol{g}}(P^{\text{LP}}_{\tilde{\boldsymbol{\alpha}}_I})$. We want to show that $(\hat{x}, \hat{y}, \hat{\boldsymbol{g}}) = (\hat{x}, y^*, \boldsymbol{g}^*)$. To do so, by Lemma 5, we have only to show that all $\hat{g}_i = u_i(\hat{g}_{i-1})$.

By convexity, since $\boldsymbol{g}^1$ and $\boldsymbol{g}^2$ satisfy (33), it follows that $\hat{g}$ satisfies them as well. Hence, $\hat{g}_i = G_i(\hat{g}_{i-1}, \alpha^1_i)$ for all $i \neq k$. Furthermore, we have by induction that $g^*_i = \hat{g}_i$ for all $i \leq k - 1$: (base case) $g^*_0 = \hat{g}_0 = \hat{x}$; (inductive case) for $i \leq k - 1$, if $g^*_{i-1} = \hat{g}_{i-1}$, then by Lemma 5, we have

$$\hat{g}_i \leq \max\{g_i : (x, y, \boldsymbol{g}, \boldsymbol{\alpha}) \in P^{\text{LP}}_{\tilde{\boldsymbol{\alpha}}_I}|_{g_{i-1}=\hat{g}_{i-1}}\} = g^*_i = u_i(\hat{g}_{i-1}) \leq G_i(\hat{g}_{i-1}, \alpha^1_i) = \hat{g}_i,$$

so that $\hat{g}_i = g^*_i = G_i(\hat{g}_{i-1}, \alpha^1_i)$. Similarly, we have $\hat{g}_k \leq g^*_k$. We will show that $\hat{g}_k = b_k$.

To do so, we will first show that $g^*_k = b_k$. If this holds, then since $\hat{x}$ was arbitrary, and since g^*_k is continuous in $\hat{x}$ by Observation 1, we have that $g^1_k = g^2_k = b_k$. Since $\hat{g}_k$ is a convex combination of g^1_k and g^2_k, this yields $\hat{g}_k = b_k$.

To show $g^*_k = b_k$, we first note that, by Corollary 1, there exists some $j \in [\![L]\!] \setminus I$ such that, for all $\hat{x} \in (x^1, x^2)$, we have $g^*_j = b_j < G(g^*_{j-1})$, with $g^*_i < b_i$ for all $i < j$. Furthermore, since $g^*_i = G(g^*_{i-1})$ for $i \in [\![k-1]\!] \setminus I$, we must have that $j \geq k$. To establish that $j = k$, we will show that $g^*_k = b_k$ when $\lambda = \frac{1}{2}$, implying that $j > k$ is impossible, since then we would have $g^*_k < b_k$.

Now, define $\tilde{I} = [\![L]\!] \setminus \{k\}$, and define $\tilde{\alpha}_I$ so that $\tilde{\alpha}_i = \alpha^1_i$ for all $i \in \tilde{I}$. Define $\tilde{a}_i$ and $\tilde{b}_i$ as in Lemma 3 for $P^{\text{LP}}_{\tilde{\alpha}_{\tilde{I}}}$. Let

$$(\tilde{y}, \tilde{\boldsymbol{g}}) := \arg\min\{y : (y, \boldsymbol{g}) \in \text{proj}_{y,\boldsymbol{g}}(P^{\text{LP}}_{\tilde{\alpha}_{\tilde{I}}}|_{x=\hat{x}})\}.$$

Then by Lemma 5, we have $\tilde{g}_k = \min\{\tilde{b}_k, G(\tilde{g}_{k-1})\}$. However, $\tilde{g}_k = G(\tilde{g}_{k-1})$ would yield $\tilde{g} \in \text{proj}_{\boldsymbol{g}}(P^{\text{IP}}_{\bar{\alpha}_{\bar{I}}})$, a contradiction on the choice of $\hat{x}$. Thus, we have $\tilde{g}_k = \tilde{b}_k$. Since $\hat{x}$ was arbitrary and since $\tilde{g}_k$ is continuous in $\hat{x}$ by Observation 1, this implies that $g^1_k = g^2_k = \tilde{b}_k$. Now, this allows us to compute g^1_{k-1} and g^2_{k-1} given α^1_i, which will allow us to compute b^*_k for $\lambda = \frac{1}{2}$ via $g^*_{k-1} = \hat{g}_{k-1}$ and $g^*_k = u_k(g^*_{k-1})$.

To compute $\hat{g}_{k-1}$, there are two cases. Either $\alpha^1 = 0$, yielding $g^1_{k-1} = \frac{1}{2}\tilde{b}_k$ and $g^2_{k-1} = 1 - \frac{1}{2}\tilde{b}_k$, or $\alpha^1 = 1$, yielding $g^1_{k-1} = \frac{1}{2}(1 - \tilde{b}_k)$ and $g^2_{k-1} = \frac{1}{2}\tilde{b}_k$. In either case, we have $g^1_{k-1} + g^2_{k-1} = 1$, and so

$$g^*_{k-1} = \hat{g}_{k-1} = \tfrac{1}{2}(g^1_{k-1} + g^2_{k-1}) = \tfrac{1}{2},$$

yielding by Lemma 5 that

$$g^*_k = \min\{2g^*_{k-1}, 2(1 - g^*_{k-1}), b_k\} = \min\{1, 1, b_k\} = b_k,$$

as required. Thus, since $\hat{g}_k = b_k$ and $\hat{g}_i$ satisfies (33) for all $i \neq k$, it follows that $\hat{g}_i = u_i(\hat{g}_{i-1})$ for all $i \in [\![L]\!]$. Hence, by Lemma 5, we have $\boldsymbol{g}^* = \hat{\boldsymbol{g}}$.

Upper bounds For the upper bounds, note that $(x, y) \in Q^{\text{IP}}$ implies $y = F(x)$, where F is a convex function. Furthermore, by Lemma 4, we have that $y^{\text{IP}}_l(x) = y^{\text{IP}}_u(x) = y^{\text{LP}}_u(e_0) = y^{\text{LP}}_l(x)$ for $x \in \{a_0, b_0\}$ (as the extended-space solutions are unique in $P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}$ for $x \in \{a_0, b_0\}$). Thus, by the convexity of F, we have that $y^{\text{IP}}_u = y^{\text{LP}}_u$ if and only if for all $x \in [a_0, b_0]$, we have for some $\lambda \in [0, 1]$ that $(x, y^{\text{LP}}_u(x)) = \lambda(a_0, F(a_0)) + (1 - \lambda)(b_0, F(b_0))$. Alternatively, since $y_{\text{LP}_u}(x) = F(x)$ for $x \in [a_0, b_0]$, it suffices to show that $y^{\text{LP}}_u(a_0) = F(a_0)$, $y^{\text{LP}}_u(b_0) = F_{b_0}$, and that y^{LP}_u is a linear function of x.

To this end, consider any $x \in \text{proj}_x(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I})$, and consider $(y, \boldsymbol{g}) = \max_y\{(y, \boldsymbol{g}) \in \text{proj}_{y,\boldsymbol{g}}(P^{\text{LP}}_{\bar{\boldsymbol{\alpha}}_I}|_x)\}$. Then by Lemma 5, we have for all $i \in [\![L]\!]$, $i \notin 1$ that $g_i = a_i$, which is a constant, while for all $i \in I$ we have that $g_i = G(g_{i-1}, \bar{\alpha}_i)$. Thus, eliminating all $g'_i s$, we find that, y^* is defined as $y^* = x - t(x)$, where $t(x)$ is some linear function of x, and thus so is $y^*(x)$, as required.

4.2 A connection with existing MIP formulations

Interestingly, Gray codes also naturally appear in the "logarithmic" MIP formulations for general continuous univariate piecewise linear functions due to Vielma et al. [52,53]. Consider applying this existing formulation[3] to approximate the univariate quadratic term with the same $2^L + 1$ breakpoints as discussed in Section 3. The resulting MIP formulation uses L binary variables, which follow the same interpretation as the neural network formulation as discussed in Theorem 2. Moreover, it requires $\mathcal{O}(L)$ linear constraints (excluding

[3] In actuality, any Gray code, not just the reflected Gray code studied in this paper, yields a (potentially distinct) logarithmic formulation for a univariate function. Here, we mean the one constructed with the reflected Gray code, which is the most common choice regardless.

variable bounds), and is ideal, a stronger property than the sharpness shown in Theorem 3. However, it comes at the price of an additional $2^L + 1$ auxiliary continuous variables, and so is unlikely to be practical without a careful handling through, e.g., column generation. Therefore, our formulation sacrifices strength to reduce this to $\mathcal{O}(L)$ auxiliary continuous variables.

5 Convex hull characterization

We explore a facet characterization of the convex hull of our model. Such a characterization could be used to improve and branch & cut scheme when solving MIPs with our model.

Although (7) offers a convex hull formulation for a single "layer" in our construction, the composition over multiple layers ($L > 1$) in (8) will in general fail describe the integer hull. We characterize additional valid inequalities for the integer hull of (8) that are derived via a connection with the parity polytope.

We begin by rewriting the relationship between the variables associated with each layer with the quadratic recurrence relation

$$g_i = (1 - 2\alpha_i)(2g_{i-1} - 1) + 1 \quad i \in [\![L]\!]. \tag{34}$$

For convenience, let $h_i := 2g_i - 1$ and $a_i := (1 - 2\alpha_i)$ for each $i \in [\![L]\!]$. Then after some simple algebraic manipulation, (34) is equivalent to

$$h_i = 2a_i h_{i-1} + 1 \quad i \in [\![L]\!].$$

Expanding the recurrence relation, we have

$$h_L = 2^L \left(\prod_{i=1}^{L} a_i \right) \left(h_0 + \sum_{i=1}^{L-1} \frac{1}{2^i \prod_{j=1}^{i} a_j} \right) + 1.$$

Define $b_i := \prod_{j=1}^{i} a_j$ for each $i \in [\![L]\!]$. As each $a_j \in \{-1, +1\}$, each $b_i \in \{-1, +1\}$ as well, and so $b_i = 1/b_i$. Hence,

$$h_L = 2^L b_L \left(h_0 + \sum_{i=1}^{L-1} 2^{-i} b_i \right) + 1. \tag{35}$$

Multiplying both sides of (35) by $b_L \in \{-1, +1\}$ yields

$$h_L b_L = 2^L \left(h_0 + \sum_{i=1}^{L-1} 2^{-i} b_i \right) + b_L. \tag{36}$$

Combining this with the McCormick inequality $h_L + b_L - 1 \leq h_L b_L$ that is valid for the bilinear left-hand side of (36) (recall $h_L, b_L \in [-1, +1]$), we derive

the following valid inequalities:

$$h_L \leq 2^L \left(h_0 + \sum_{i=1}^{L-1} 2^{-i} b_i \right) + 1, \tag{37a}$$

$$h_L \leq -2^L \left(h_0 + \sum_{i=1}^{L-1} 2^{-i} b_i \right) + 1. \tag{37b}$$

which can be readily mapped back to the original space of variables.

Proposition 3 *The following inequalities are valid for* (8)*:*

$$g_L \leq 2^{L-1} \left(2g_0 - 1 + \sum_{i=1}^{L-1} 2^{-i} \prod_{j=1}^{i} (1 - 2\alpha_j) \right) + 1 \tag{38a}$$

$$g_L \leq -2^{L-1} \left(2g_0 - 1 + \sum_{i=1}^{L-1} 2^{-i} \prod_{j=1}^{i} (1 - 2\alpha_j) \right) + 1 \tag{38b}$$

If $L = 2$, then (37a) *and* (37b) *are both linear inequalities in g and α.*

Based on computational observations, for $L = 2$, these are exactly the nontrivial facet-defining linear inequalities for the integer hull of (8). For $L > 2$, we can produce a large class of valid inequalities by bounding the product variables b_i. In particular, bounds on these products can be derived from valid inequalities for the parity polytope.

5.1 Parity inequalities

The parity polytope is the convex hull of P_n^{even}, the set of all $\alpha \in \{0,1\}^n$ whose components sum to an even number. It has 2^{n-1} facets of the form

$$\sum_{i \in [\![n]\!] \setminus I} \alpha_i + \sum_{i \in I} (1 - \alpha_i) \geq 1 \qquad I \subseteq [\![n]\!] \text{ s.t. } |I| \text{ is odd.} \tag{39}$$

Define $\beta_i \in \{0,1\}$, such that $b_i = (1 - 2\beta_i)$. Then

$$1 = b_j^2 = (1 - 2\beta_j) \prod_{i=1}^{j} (1 - 2\alpha_i) = (-1)^{\beta_j + \sum_{i=1}^{j} \alpha_i}. \tag{40}$$

Hence, for $\alpha_i \in \{0,1\}$, the sum $\beta_j + \sum_{i=1}^{j} \alpha_i$ is even and therefore $(\beta_j, \alpha_1, \ldots, \alpha_j) \in P_{j+1}^{\text{even}}$. Therefore, we can apply (39) to derive valid inequalities for feasible solutions to (8) of the form

$$\beta_j + \sum_{i \in [\![j]\!] \setminus I} \alpha_i + \sum_{i \in I} (1 - \alpha_i) \geq 1 \qquad I \subseteq [\![j]\!] \text{ s.t. } |I| \text{ is odd,} \tag{41a}$$

$$(1 - \beta_j) + \sum_{i \in [\![j]\!] \setminus I} \alpha_i + \sum_{i \in I} (1 - \alpha_i) \geq 1 \qquad I \subseteq [\![j]\!] \text{ s.t. } |I| \text{ is even.} \tag{41b}$$

After recalling the definition $b_j = (1 - 2\beta_j)$ and rearranging, we are left with

$$b_j \leq 2\left(\sum_{i\in[\![j]\!]\setminus I} \alpha_i + \sum_{i\in I}(1-\alpha_i)\right) - 1 \qquad I \subseteq [\![j]\!] \text{ s.t. } |I| \text{ is odd}, \tag{42a}$$

$$-b_j \leq 2\left(\sum_{i\in[\![j]\!]\setminus I} \alpha_i + \sum_{i\in I}(1-\alpha_i)\right) - 1 \qquad I \subseteq [\![j]\!] \text{ s.t. } |I| \text{ is even}. \tag{42b}$$

Hence, combining these upper bounds on b_i with (37a) and the upper bounds on $-b_i$ with (37b) produces an exponential family of valid inequalities for feasible solutions to (8). We call these *parity inequalities*.

Example 2 For $L = 4$, we compute some of the nontrivial facet-defining inequalities of (8), written in terms of the original variables:

$$\begin{aligned}
g_4 \leq\ & 16g_0 - 14\alpha_1 + 6\alpha_2 + 2\alpha_3,\\
g_4 \leq\ & 16g_0 - 2\alpha_1 - 6\alpha_2 + 2\alpha_3,\\
g_4 \leq -& 16g_0 + 14\alpha_1 + 6\alpha_2 + 2\alpha_3 + 2,\\
g_4 \leq -& 16g_0 + 2\alpha_1 - 6\alpha_2 + 2\alpha_3 + 14.
\end{aligned}$$

Each of these inequalities is, in fact, a parity inequality, and can be constructed by a suitable combination of either (37a) with inequalities from (42a), or (37b) with inequalities from (42b).

For example, the facet $g_4 \leq 16g_0 - 14\alpha_1 + 6\alpha_2 + 2\alpha_3$ is equivalent to

$$h_4 \leq 16h_0 + 15 + 2(-14\alpha_1 + 6\alpha_2 + 2\alpha_3),$$

which can be produced as a conic combination of the inequality

$$h_4 \leq 2^4\left(h_0 + \frac{1}{2}b_1 + \frac{1}{4}b_2 + \frac{1}{8}b_3\right) + 1$$

from (37a) with the inequalities

$$\begin{aligned}
b_1 &\leq 2\big((1-\alpha_1)\big) - 1,\\
b_2 &\leq 2\big((1-\alpha_1) + \alpha_2\big) - 1,\\
b_3 &\leq 2\big((1-\alpha_1) + \alpha_2 + \alpha_3)\big) - 1
\end{aligned}$$

from the family (42a) corresponding to $I = \{1\}$ for all $j = 1, 2, 3$, respectively.

5.2 Separation over exponentially many parity inequalities

Since there may be exponentially many parity inequalities, we provide an algorithm to separate over them. In particular, given a point $(g, \alpha) \subseteq [0,1]^{L+1} \times [0,1]^L$, we can determine if it lies in the intersection of the parity inequalities by computing the inequalities that give smallest upper bounds for b_j for each $j \in [\![L]\!]$. To do so, for each b_j, we need to determine the set

$I_j \subseteq [\![j]\!]$ that minimizes the right-hand side of equation (42a) or (42b). This can be done, in fact, by optimizing over another parity polytope. That is, set $I_j := \{i \in [\![j]\!] : z_i^* = 1\}$, where

$$z^* \in \operatorname*{arg\,min}_{z \in \{0,1\}^j} \left\{ \sum_{i=1}^{j} z_i \alpha_i + (1 - z_i)(1 - \alpha_i) \;\middle|\; \sum_{i=1}^{j} (1 - z_i) \text{ is odd} \right\}.$$

Linear functions can be optimized over the parity polytope in polynomial time via a linear size extended formulation [38], or by writing an integer program with a single integer variable [7], or by a simple greedy-like algorithm. An analogous approach can be taken if the sum of z_i should be odd.

6 Area comparisons

The approximation presented above is an over approximation of $y = x^2$. This is sufficient for providing dual bounds due how the approximation is applied using the diagonal perturbation. However, our formulation can also be altered slightly to provide an under approximation of $y = x^2$, and in particular, creates a covering of the curve with a union of polytopes. We describe two relaxations that are comparable to that of Dong and Luo [23]. We then compare these models based on the combined area of the covering to see how these methods converge.

We construct our first relaxation, named `NN-R1`, from the constraints (7) and

$$y \leq x - \sum_{i=1}^{L} \tfrac{g_i}{2^{2i}} \tag{43a}$$

$$y \geq \left(x - \sum_{i=1}^{j} \frac{g_i}{2^{2i}} \right) - \frac{1}{2^{2j+2}} \qquad j \in [\![L-1]\!] \tag{43b}$$

$$y \geq 2x - 1 \tag{43c}$$

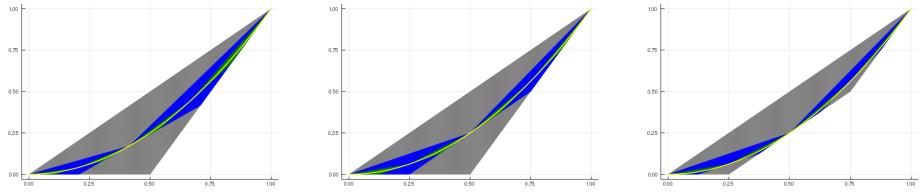
$$y \geq 0 \tag{43d}$$

We can form a tighter relaxation `NN-R2` by starting with `NN-R1`, then adding the cut (43b) with $j = L$:

$$y \geq \left(x - \sum_{i=1}^{j} \frac{g_i}{2^{2i}} \right) - \frac{1}{2^{2j+2}} \qquad j \in [\![L]\!]. \tag{44}$$

Tables 1 and 2 compare the volume of our relaxed method with the method of Dong and Luo [23] on the intervals $x \in [0, 1]$ and $x \in [-2, 1]$, respectively. As L increases, the volume of our relaxation consistently shrinks by a factor of 4, which is strictly greater by a fair margin to the improvement rate observed for the method of Dong and Luo. We can formalize our rate of improvement in the following proposition.

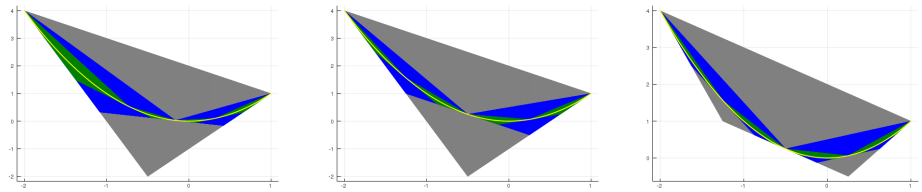
Method	$L=0$	$L=1$	$L=2$	$L=3$	$L=4$
`CDA`	0.25	0.0680 **(3.68)**	0.0177**(3.84)**	0.00448**(3.95)**	0.00112 **(3.994)**
`NN-R1`	0.25	0.0625**(4)**	0.0156**(4)**	0.00391**(4)**	0.000977**(4)**
`NN-R2`	0.188	0.0469**(4)**	0.0117**(4)**	0.00293**(4)**	0.000732**(4)**



`CDA` `NN-R1` `NN-R2`

Table 1: Area/ratio table $x \in [0,1]$. The Gray area is for $L=0$, the blue area is $L=1$, the green area (very small) is $L=2$, and the yellow curve is the curve $y=x^2$. The area for each method was approximated numerically using a Riemann sum.

Method	$L=0$	$L=1$	$L=2$	$L=3$	$L=4$
`CDA`	6.75	1.94**(3.47)**	0.659**(2.95)**	0.197**(3.34)**	0.0530**(3.72)**
`NN-R1`	6.75	1.69**(4)**	0.422**(4)**	0.105**(4)**	0.0264**(4)**
`NN-R2`	5.06	1.27**(4)**	0.316**(4)**	0.0791**(4)**	0.0198**(4)**



`CDA` `NN-R1` `NN-R2`

Table 2: Area/ratio table $x \in [-2,1]$. Area/ratio table $x \in [0,1]$. The Gray area is for $L=0$, the blue area is $L=1$, the green area (very small) is $L=2$, and the yellow curve is the curve $y=x^2$. The area for each method was approximated numerically using a Riemann sum. Interestingly, the `CDA` model converges at different rates depending on the domain, whereas our methods always converges at the same rate.

Proposition 4 *The volume of our approximation decreases by a factor of 4 with each subsequent layer (i.e. as L increases). Furthermore, the expected error at points x sampled uniformly at random from the input interval domain is proportional to the total volume.*

Proposition 4 relies on the characterization of `NN-R1` as the piecewise McCormick relaxation of $y=x^2$ at uniformly-spaced breakpoints. For one piece $[x_1, x_2]$, this relaxation consists of the tangent lines, or outer-approximation cuts, at x_1 and x_2 for the lower bound, and the secant line between x_1 and x_2 for the upper bound. We have already established that the upper bound, the `NN` approximation, is a piecewise interpolant to x^2 at the chosen break-

points, yielding the secant line on each interval $[x_1, x_2]$ between interpolation points, and thus the upper-bound of the piecewise McCormick approximation. In Lemma 6 below, we show that the lower bound of the `NN-R1` and `NN-R2` approximations give the piecewise McCormick lower bounds for 2^L and 2^{L+1} uniformly-spaced breakpoints, respectively.

Lemma 6 *Define T as the lower-bounding set in (x, y) for the relaxation `NN-R2`, $T = \text{proj}_{x,y}\{(x, y, \boldsymbol{g}, \boldsymbol{\alpha}) \in [0,1] \times \mathbb{R} \times [0,1]^L \times \{0,1\}^L : (7), (43c), (43d), \text{and } (44)\}$. Then T can be constructed via $2^{L+1}+1$ uniformly-spaced outer-approximation cuts for $y \geq x^2$ on the interval $[0,1]$, based on the tangent lines to x^2 at $x_i = i \cdot 2^{-(L+1)}$, $i = 0, \ldots, 2^{(L+1)}$. That is, letting $p_{\hat{x}}(x)$ be the tangent line to $y = x^2$ at the point $\hat{x}$,*

$$p_{\hat{x}}(x) = 2\hat{x}(x - \hat{x}) + \hat{x}^2 = \hat{x}(2x - \hat{x}), \tag{45}$$

T is equivalent to $\left\{(x, y) \in [0,1] \times \mathbb{R} : y \geq p_{\hat{x}}(x) \quad \forall i \in \{0, \ldots, L\},\ \hat{x} = i \cdot 2^{-(L+1)}\right\}$.

Proof To begin, we note that, as shown by Yarotsky [57], we have that for each $l = 0, \ldots, L$, $F_l(x)$ gives a piecewise-linear interpolant in $y = x^2$ at $2^l + 1$ uniformly-spaced points on $x \in [0,1]$. That is, for each $i \in \{0, 1, \ldots, 2^l\}$, we have that $F_l(i \cdot 2^{-l}) = (i \cdot 2^{-l})^2$.

For $x_1 < x_2$, Consider a single linear interpolant to $y = x^2$ on the interval $[x_1, x_2]$, given as

$$\begin{aligned} h(x) &= \frac{x_2^2 - x_1^2}{x_2 - x_1}(x - x_1) + x_1^2 \\ &= (x_1 + x_2)(x - x_1) + x_1^2. \end{aligned} \tag{46}$$

Since $h(x) - x^2$ is concave, the deviation $h(x) - x^2$ is maximized when $\frac{d}{dx}(h(x) - x^2) = x_1 + x_2 - 2x = 0$, yielding $x^* = \frac{x_1 + x_2}{2}$. At this point, we have

$$\begin{aligned} h(x^*) - (x^*)^2 &= (x_1 + x_2)\left(\frac{x_1 + x_2}{2} - x_1\right) + x_1^2 - (\frac{x_1 + x_2}{2})^2 \\ &= \tfrac{1}{2}(x_1 + x_2)^2 - \tfrac{1}{4}(x_1 + x_2)^2 - (x_1 + x_2)x_1 + x_1^2 \\ &= \tfrac{1}{4}(x_1 + x_2)^2 - x_1 x_2 \\ &= \tfrac{1}{4}x_1^2 + \tfrac{1}{2}x_1 x_2 - x_1 x_2 + \tfrac{1}{4}x_2^2 \\ &= \tfrac{1}{4}(x_2 - x_1)^2. \end{aligned} \tag{47}$$

Thus, we have that $(x^*)^2 = h(x^*) - \frac{1}{4}(x_2 - x_1)^2$. Since x^2 is a convex function, and since $h - \frac{1}{4}(x_2 - x_1)^2$ is tangent to x^2 at x^* (as both slopes are $x_2 - x_1$), this implies that for all $x \in [0,1]$, we have $(x^*)^2 \geq h(x) - \frac{1}{4}(x_2 - x_1)^2$.

Now, since $F_l(x)$ is a linear interpolant to $y = x^2$ on each interval $[i \cdot 2^{-l}, (i + 1) \cdot 2^{-l}]$, with interval width 2^{-l}, we have that the cuts $y \geq F_l(x) - \frac{1}{4}2^{-2l} = F_l(x) - 2^{-2l-2}$ are valid for $y = x^2$, and furthermore are tangent to x^2 at the point $x = (i + \frac{1}{2}) \cdot 2^{-l}$, the midpoints of all interpolants. Thus, we obtain outer-approximation cuts at $x = (i + \frac{1}{2}) \cdot 2^{-l}$ for each $i = 0, \ldots, 2^l - 1$.

We obtain T by applying these cuts for $l = 0, \ldots, L$, combined with the outer-approximation cuts $y \geq 0$ (tangent at $x = 0$) and $y \geq 2x - 1$ (tangent

at $x = 1$). We now show that this yields $2^{L+1} + 1$ uniformly-spaced outer-approximation cuts to $y = x^2$. This can be easily seen by expressing the outer-approximation points in binary. The points $x = (i + \frac{1}{2}) \cdot 2^{-l}$ for each $i = 0, \ldots, 2^l - 1$ can be characterized as exactly the numbers in $(0, 1)$ such that the $(l+1)$th binary decimal is a 1, and all later binary decimals are zero. Alternatively, it is the set of points

$$P_l := \left\{ x \in [0,1] : \exists\ \mathbf{a} \in \{0,1\}^l \text{ s.t. } x = 2^{-(l+1)} + \sum_{i=1}^{l} 2^{-i} a_i \right\}.$$

We then have that the set of outer-approximation points, $x \in \{0, 1\} \cup \bigcup_{l \in [\![0,L]\!]} P_l$, is the set of all binary decimals in the interval $[0, 1]$ for which the last 1 occurs no later than the $L + 1$st decimal place, which has cardinality $2^{L+1} + 1$. This forms the set of uniformly-spaced points $i \cdot 2^{-(L+1)}$, $i = 0, \ldots, 2^{L+1}$. Thus, we have that the set T consists of a set of $2^{L+1} + 1$ uniformly-spaced outer-approximation cuts to the function $y = x^2$, as required.

With Lemma 6, we establish that the `NN-R1` relaxation is equivalent to the piecewise McCormick relaxation of $y = x^2$ on the uniformly-spaced breakpoints $x_i = 2^{-L} i$, $i \in [\![0, 2^L]\!]$. We now establish the area of any given piece of the relaxation.

Lemma 7 *The area of the McCormick Relaxation of $y = x^2$ on the interval $[x_1, x_2]$ is $\frac{1}{4}(x_2 - x_1)^3$.*

Proof We wish to obtain a closed-form solution for the area of the resulting triangle region. To do so, we compute the vertices of this triangle, construct vectors between them, then compute the cross-product of these vectors.

The equations of the tangent lines are given by

$$y = 2x_1(x - x_1) + x_1^2 = 2x_1(x - \tfrac{x_1}{2}),$$
$$y = 2x_2(x - x_2) + x_2^2 = 2x_2(x - \tfrac{x_2}{2}).$$

We thus find that the intersection of these lines is given by the point $(\frac{x_1+x_2}{2}, x_1 x_2)$. Thus, the three vertices are given as $v_1 = (x_1, x_1^2)$, $v_2 = (\frac{x_1+x_2}{2}, x_1 x_2)$, and $v_3 = (x_2, x_2^2)$. From these vertices, we obtain two vectors

$$v_2 - v_1 = [\tfrac{1}{2}(x_2 - x_1), x_1(x_2 - x_1)] = (x_2 - x_1)[\tfrac{1}{2}, x_1],$$
$$v_3 - v_2 = [\tfrac{1}{2}(x_2 - x_1), x_2(x_2 - x_1)] = (x_2 - x_1)[\tfrac{1}{2}, x_2].$$

The area is given by the magnitude of the cross product as

$$A = \frac{1}{2}|(v_2 - v_1) \times (v_3 - v_2)| = \frac{1}{2}(x_2 - x_1)^2 |\tfrac{1}{2}x_2 - \tfrac{1}{2}x_1| = \frac{1}{4}(x_2 - x_1)^3.$$

As required.

With Lemma 7, we are now ready to show that the area of `NN-R1` is optimal among all piecewise McCormick relaxations with a fixed number of pieces.

Proposition 5 *The minimum possible area covering $y = x^2$ on $x \in [a, b]$ with a sequence of n McCormick Relaxations is $\frac{1}{4}(b-a)^3/n^2$, and is achieved via uniformly spaced breakpoints.*

Proof Consider a general piecewise McCormick relaxation of $y = x^2$ on the interval $[a, b]$ with consecutive breakpoints $a = x_0 \leq x_1 \leq \cdots \leq x_n = b$. For any segment of consecutive breakpoints x_i and x_{i+1}, the McCormick relaxation between those points is bounded by the secant line between (x_i, x_i^2) and (x_{i+1}, x_{i+1}^2), and the tangent lines to $y = x^2$ at x_i and x_{i+1}. For simplicity, let $i = 1$ for this discussion.

Thus, letting $y_i = x_i - x_{i-1},\ i = 1 \dots n$, the problem of choosing the area-optimal breakpoints $a \leq x_1, \leq x_2 \leq \cdots \leq x_{n-1} \leq x_n$ reduces to solving

$$\min_{y \in \mathbb{R}^n_+} \left\{ \frac{1}{4} \sum_{i=1}^{n} y_i^3 : \sum_{i=1}^{n} y_i = b - a \right\}. \tag{48}$$

It is then easy to show via the KKT conditions that, due to the convexity of $\sum_i y_i^3$ on positive support, all y_i must be equal, yielding the uniformly-spaced solution $y_i = \frac{b-a}{n}$. Thus, the choice of breakpoints induced by our algorithm is optimal. □

The `NN-R1` relaxation is exactly a union of McCormick Relaxations 2^n uniformly spaced breakpoints. Hence the total area is $\frac{1}{4}(b-a)^3 2^{-2n}$. We now show that adding the inequality (43b) with $j = L$ cuts off an extra fourth of the total area.

Proposition 6 *The area of the relaxation in (43) is $\frac{3}{16}(b-a)^3 2^{-2n}$.*

Proof We will compute the area removed by the addition of this cut on a general interval $[x_1, x_2]$, where $x_1 = 2^{-L} i$ for some $i \in [\![0, 2^L - 1]\!]$. As shown in Lemma 6, the cut (43b) with $j = L$ intersects the curve at $x = \frac{x_1 + x_2}{2}$. Now, the area removed by the addition of this cut is the area of the triangle formed by the intersections of the tangent lines at x_1, x_2, and the midpoint $x_3 := \frac{x_1+x_2}{2}$. These vertices, given as intersection points v_{ij} for tangent lines for x_i and x_j, are derived following the process in Lemma 7 as:

$$\begin{aligned}
v_{12} &= \left(\tfrac{x_1+x_2}{2}, x_1 x_2\right) \\
v_{13} &= \left(\tfrac{1}{2} \cdot \left(\tfrac{x_1+x_2}{2} + x_1\right), \tfrac{1}{2} x_1 (x_1 + x_2)\right) = \left(\tfrac{1}{4}(3x_1 + x_2), \tfrac{1}{2} x_1 (x_1 + x_2)\right) \\
v_{23} &= \left(\tfrac{1}{2} \cdot \left(\tfrac{x_1+x_2}{2} + x_2\right), \tfrac{1}{2} x_2 (x_1 + x_2)\right) = \left(\tfrac{1}{4}(x_1 + 3x_2), \tfrac{1}{2} x_2 (x_1 + x_2)\right)
\end{aligned}$$

From these points, we obtain vectors

$$\begin{aligned}
v_{12} - v_{13} &= \left(\tfrac{1}{4}(x_2 - x_1), \tfrac{1}{2} x_1 (x_2 - x_1)\right) = (x_2 - x_1)\left(\tfrac{1}{4}, \tfrac{1}{2} x_1\right) \\
v_{23} - v_{12} &= \left(\tfrac{1}{4}(x_2 - x_1), \tfrac{1}{2} x_2 (x_2 - x_1)\right) = (x_2 - x_1)\left(\tfrac{1}{4}, \tfrac{1}{2} x_2\right).
\end{aligned}$$

Finally, we obtain cut area

$$A_{cut} = \frac{1}{2} |(v_{12} - v_{13}) \times (v_{23} - v_{12})| = \frac{1}{2}(x_2 - x_1)^2 \cdot \frac{1}{8}(x_2 - x_1) = \frac{1}{16}(x_2 - x_1)^3.$$

The result easily follows.

7 A computational study

We study the efficacy of our MIP relaxation approach on a family of nonconvex quadratic optimization problems. We compare 9 methods:

1. `GRB`: The native method in Gurobi v9.1.1 for nonconvex quadratic problems.
2. `GRB-S`: The native method in Gurobi v9.1.1, applied to the diagonalized shift reformulation of (2).
3. `BRN`: Baron v21.1.13, using CPLEX v12.10 for the MIP/LP solver.
4. `BRN-S`: Baron v21.1.13, using CPLEX v12.10 for the MIP/LP solver, applied to the diagonalized shift reformulation of (2).
5. `CPLEX`: The native method in CPLEX v12.10 for nonconvex quadratic objectives.
6. `CDA`: The algorithm of Dong and Luo [23]. The number of layers L will correspond to the parameter ν appearing in their paper.
7. `NN`: The new formulation (8).
8. `NMDT`: The "normalized multi-parametric disaggregation technique" (NMDT) of Castro [14]. See Appendix A for a restatement in terms of the number of levels L.
9. `T-NMDT`: A tightened variant of NMDT also described in Appendix A.

We can cluster these methods into two families: five "native" methods (`GRB`, `GRB-S`, `BRN`, `BRN-S`, and `CPLEX`) that pass an exact representation of the quadratic problem to the solver, and four "relaxations" (`CDA`, `NN`, `NMDT`, and `T-NMDT`) which relax the quadratic problem using a MIP reformulation, which is then passed to an underlying solver. For each of these relaxations, we use Gurobi v9.1.1 as the underlying MIP solver. Note that `GRB`, `BRN`, and `CPLEX` are directly given (50), which is an optimization problem with linear constraints and a nonconvex quadratic objective. In contrast, `GRB-S` and `BRN-S` are given the diagonalized reformulation of (50) per (2), which is an optimization problem with a convex quadratic objective and a series of nonconvex quadratic constraints.[4]

Our objective in this computational study is to measure the quality of the dual bound provided by the different methods. To place the methods on an even footing on the primal side, as initialization we run the nonconvex quadratic optimization method in Gurobi v9.1.1 with "feasible solution emphasis" to produce a good starting feasible solution. We then inject this primal objective bound as a "cut-off" for each method.

We will consider 4 metrics, which will be applied with respect to a given family of instances:

– time: The shifted geometric mean of the solve time in seconds (shift is minimum solve time in the family).

[4] CPLEX does not support nonconvex quadratic constraints of this form, so we do not include a corresponding approach with the diagonal shift.

- gap: The shifted geometric mean of the final relative optimality gap $\frac{|\texttt{db}-\texttt{bpb}|}{|\texttt{bpb}|}$, where `db` is the dual bound provided by the method and `bpb` is the best observed primal solution for the instance across all methods. Shift is taken as $\max\{10^{-4}, \text{minimum gap observed in the family}\}$.
- BB: The number of instances in which the method either produced the best dual bound, or attained Gurobi's default optimality criteria of $\texttt{gap} < 10^{-4}$. Note that on a given instance, more than one method can potentially attain the best bound.
- TO: The number of instances in which the solver *times out* and terminates due to the time limit.

We note that even if the solver terminates within the time limit (with an "optimal" solver status), the optimality gap for `NN` or `CDA` as reported in Table 3 may be nonzero, due to the fact that these two methods serve as relaxations for the original boxQP problem.

We implement each model in the JuMP algebraic modeling language [24]. To compute the shift used by the four relaxations, `GRB-S`, and `BRN-S`, we use Mosek v9.2 to solve a semidefinite programming problem to produce the "tightest" diagonal matrix $D = \text{diag}(\delta)$ such that $Q+D$ is positive semidefinite as in Dong and Luo [23]:

$$\min_{\delta \in \mathbb{R}^n} \mathbf{e} \cdot \delta \quad \text{s.t.} \quad Q + \text{diag}(\delta) \succeq 0. \tag{49}$$

In Section 7.4 we study an alternative, simpler method for computing this shift and its computational implications. Note that this time to solve the SDP is not included in the solve time numbers, but is relatively small (on the order of a few seconds for the largest instances) and is computation that is shared by most of the approaches.

Each method is provided a time limit of 10 minutes. Computational experiments are performed on a machine with a 3.8 GHz CPU with 24 cores and 128 GB of RAM. Each solver is restricted to one thread, and all experiments for a given instance are run concurrently. Our code and the corresponding problem instances are publicly available at https://github.com/joehuchette/quadratic-relaxation-experiments.

7.1 Baseline comparison

We start by comparing our nine methods on 99 box constrained quadratic objective (*boxQP*) optimization problem instances as studied in Chen and Burer [17] and Dong and Luo [23]:

$$\min_{0 \le x \le 1} \quad x'Qx + c \cdot x. \tag{50}$$

Despite its simple constraint structure, this is a nonconvex optimization problem when Q is not positive semidefinite, and is difficult from both a theoretical and a practical perspective.

For this baseline study, we fix each of the relaxations to use $L = 3$ layers; we will revisit this selection in Section 7.3. We leave as future work an implementation that iteratively refines the approximation to guarantee a pre-specified approximation error, as done by Dong and Luo [23].

We split these instances into three families: 63 "solved" instances on which each method terminates at optimality within the time limit, 18 "unsolved" instances on which each method terminates due to the time limit, and 18 "contested" instances on which some methods terminate and some do not. We present the computational results in Table 3, stratified by family. At a high level, we observe that `NN` attains the "best bound" on 87 of 99 instances. We now survey each family in more detail. Alternatively, we stratify the results based on the size of the instances in Appendix B.

family	method	time (sec)	gap	BB	TO
solved	`BRN`	0.51	0.00%	63/63	-
	`CPLEX`	0.68	0.00%	63/63	-
	`GRB`	0.37	0.00%	63/63	-
	`BRN-S`	0.96	0.00%	63/63	-
	`GRB-S`	0.63	0.00%	62/63	-
	`CDA`	1.24	0.08%	5/63	-
	`NN`	0.39	0.01%	26/63	-
	`NMDT`	0.67	0.02%	19/63	-
	`T-NMDT`	1.07	0.01%	24/63	-
contested	`BRN`	66.1	0.00%	12/18	6/18
	`CPLEX`	46.0	0.00%	17/18	1/18
	`GRB`	34.4	0.00%	15/18	3/18
	`BRN-S`	154.6	0.02%	10/18	8/18
	`GRB-S`	450.1	1.22%	2/18	16/18
	`CDA`	429.0	1.49%	0/18	15/18
	`NN`	273.0	0.24%	2/18	10/18
	`NMDT`	318.0	0.54%	1/18	11/18
	`T-NMDT`	446.5	0.83%	1/18	14/18
unsolved	`BRN`	-	11.48%	0/18	-
	`CPLEX`	-	15.67%	3/18	-
	`GRB`	-	30.73%	0/18	-
	`BRN-S`	-	11.86%	0/18	-
	`GRB-S`	-	5.21%	0/18	-
	`CDA`	-	5.33%	0/18	-
	`NN`	-	4.31%	15/18	-
	`NMDT`	-	4.59%	0/18	-
	`T-NMDT`	-	5.04%	0/18	-

Table 3: Baseline computational results on 99 boxQP instances.

Solved instances On the solved instances, all methods are able to terminate quickly–all in under two seconds, on average. The native methods are able

to meet the termination criteria on nearly all instances, while the relaxation methods lag behind. Our new `NN` method performs the best, attaining the termination gap criteria on roughly half of the instances, while `CDA` performs the worst, attaining it on only 5 of 63 instances. We stress that, for these experiments, L is set relatively low. In Section 7.3 we revisit this decision, and observe that this gap can be closed on these easy instances by increasing L at a nominal computational cost.

Contested instances On the contested instances the native solvers `BRN`, `CPLEX`, and `GRB` perform best, producing the best bound in a clear majority of the 18 instances. Interestingly, the shifted variants `BRN-S` and `GRB-S` perform worse, with Gurobi exhibiting a substantial degradation in performance as opposed to without the diagonal shift. In contrast, the relaxations time out on a majority of the instances. Taken together, we conclude that there is a transitional family of instances wherein the native solvers still excel, but which the more complex relaxations succumb to the curse of dimensionality.

Unsolved instances This family of instances tests the scenario where a method is given a fixed time budget and is asked to produce the best possible dual bound. On these 18 instances, `NN` is the clear winner, producing the best bounds on 15 and the lowest mean gap. The other relaxations come relatively close in terms of termination gap, but do not attain the best bound on any instance. The native `GRB` performs the worst of all methods in terms of gap closed, but applying the shift as in `GRB-S` helps tremendously, producing gaps than are much lower than the other native solver methods and close to what the relaxations are able to provide.

7.2 Varying the solver focus

Modern solvers such as Gurobi expose high-level parameters for configuring the search algorithm for different goals. In this subsection, we configure Gurobi to focus on the best objective bound (`MIPFocus=3`). We summarize our results in Table 4. The story on the "solved" and "contested" instances remains roughly the same as in Section 7.1. However, on the "unsolved" instances all methods perform better, with the largest improvement coming from the native Gurobi methods. Nonetheless, the `NN` method is still attaining the best bound on 10 of 18 instances, with the `GRB-S` method coming close in terms of gap closed, and is able to produce the best bound on the remaining 8 instances.

7.3 Varying the relaxation resolution

In the previous experiments, we fixed the number of layers for each relaxation at $L = 3$. In this subsection, we study how varying this parameter affects each relaxation, in terms of both solve time and gap closed. In particular, we

family	method	time (sec)	gap	BB	TO
solved	GRB	0.70	0.00%	63/63	-
	GRB-S	0.64	0.00%	63/63	-
	CDA	1.71	0.08%	5/63	-
	NN	0.48	0.01%	18/63	-
	NMDT	1.45	0.04%	17/63	-
	T-NMDT	1.13	0.01%	24/63	-
contested	GRB	49.3	0.00%	14/18	4/18
	GRB-S	458.9	0.26%	3/18	15/18
	CDA	470.2	1.31%	0/18	16/18
	NN	301.9	0.18%	4/18	10/18
	NMDT	457.8	0.83%	0/18	15/18
	T-NMDT	436.6	0.34%	1/18	14/18
unsolved	GRB	-	6.13%	0/18	-
	GRB-S	-	3.58%	8/18	-
	CDA	-	4.71%	0/18	-
	NN	-	3.34%	10/18	-
	NMDT	-	4.50%	0/18	-
	T-NMDT	-	4.05%	0/18	-

Table 4: Computational results with Gurobi configured with `MIPFocus=3`.

consider setting $L \in \{2, 4, 6, 8\}$ for each relaxation method, and experiment with the same set of 99 boxQP instances as before. We summarize the results in Table 5.

On the "solved" instances we observe that, unsurprisingly, increasing L allows us to reach the best bound criteria on far more instances. Moreover, we observe that this results in only a nominal increase in computational cost; all methods terminate with a mean solve time of seconds, even with the finest discretization. We observe that `NN` performs the best, in terms of mean solve time and "best bound" for each value of L considered. Morever, `NN` can attain the termination criteria on each instance with $L = 8$, which is not the case for any other method. These results indicate that, on easy instances, increasing the resolution is cheap, and can attain the same dual bound quality as the native solvers in roughly the same time. In contrast, on the "unsolved" instances we observe that increasing L results in *higher* gaps across the board. This is unsurprising–increasing L results in larger formulations, and on instances where the solver is already struggling this will quickly lead to performance degradation due to the "combinatorial explosion" effect. Moreover, even small values for L offer a nontrivial refinement in a branch-and-bound setting over a tight convex relaxation. This result suggests that, for instances known to be hard, a reasonable strategy would be to set L to a small value by default and then target finer discretizations on individual quadratic terms as-needed, through a dynamic refinement approach or otherwise.

family	method	L	time (sec)	gap	BB	TO
solved	CDA	2	0.39	0.79%	0/63	-
		4	1.64	0.01%	24/63	-
		6	2.84	0.00%	52/63	-
		8	3.74	0.00%	61/63	-
	NN	2	0.30	0.04%	8/63	-
		4	0.41	0.00%	41/63	-
		6	0.48	0.00%	58/63	-
		8	0.55	0.00%	63/63	-
	NMDT	2	0.41	0.12%	8/63	-
		4	0.84	0.01%	31/63	-
		6	1.17	0.00%	41/63	-
		8	1.41	0.00%	46/63	-
	T-NMDT	2	0.58	0.05%	8/63	-
		4	1.24	0.00%	37/63	-
		6	1.43	0.00%	53/63	-
		8	1.59	0.00%	62/63	-
contested	CDA	2	136.18	1.10%	2/13	0/13
		4	552.44	0.68%	0/13	10/13
		6	595.58	1.33%	0/13	12/13
		8	600.00	2.06%	0/13	13/13
	NN	2	206.03	0.16%	1/13	3/13
		4	268.56	0.04%	4/13	3/13
		6	293.47	0.07%	4/13	5/13
		8	319.46	0.08%	8/13	5/13
	NMDT	2	208.66	0.42%	0/13	3/13
		4	376.76	0.16%	2/13	5/13
		6	447.43	0.18%	4/13	7/13
		8	473.95	0.18%	4/13	7/13
	T-NMDT	2	344.76	0.33%	0/13	6/13
		4	511.43	0.36%	1/13	9/13
		6	505.40	0.24%	3/13	8/13
		8	534.02	0.26%	4/13	8/13
unsolved	CDA	2	-	3.98%	0/23	-
		4	-	4.72%	0/23	-
		6	-	5.02%	0/23	-
		8	-	5.29%	0/23	-
	NN	2	-	3.37%	23/23	-
		4	-	3.53%	0/23	-
		6	-	3.63%	0/23	-
		8	-	3.72%	0/23	-
	NMDT	2	-	3.55%	0/23	-
		4	-	3.92%	0/23	-
		6	-	4.05%	0/23	-
		8	-	4.16%	0/23	-
	T-NMDT	2	-	3.84%	0/23	-
		4	-	4.36%	0/23	-
		6	-	4.31%	0/23	-
		8	-	4.30%	0/23	-

Table 5: Computational results with varying discretization levels.

7.4 Varying the diagonal perturbation

We now turn our attention to how the diagonal shift that is used by the relaxations, `BRN-S`, and `GRB-S` is computed. As discussed above, we may solve the SDP (49) to compute a valid shift that is "tightest" under some reasonable objective measure. While this approach is reasonable for the boxQP instances studied here, it may not be practical for larger-scale instances due to the scalability of the SDP solver. Therefore, we compare this shift against a simpler "eigenvalue" shift $D = -\lambda_{\min} I$, where $I \in \mathbb{R}^{n \times n}$ is the identity matrix and $\lambda_{\min}$ is the smallest eigenvalue of Q. This minimum eigenvalue can be readily computed, and the resulting shift is conceptually similar to the convexification process used in αBB [6], for example.

We perform a similar experiment as in Section 7.1, focusing on comparing our two shifts head-to-head for each method that utilizes it. We summarize our results in Table 6. We observe that the tighter shift provided by the SDP (49) offers a substantial improvement over the eigenvalue shift across the board. On the "solved" instances, we observe an order of magnitude reduction in solve time for all methods, as well as a significant increase in best bound attainment for the relaxation methods. On the "contested" instances, we observe a similar degradation when using the shift. The difference is perhaps most striking for `GRB`: on the 22 instances, produces the best bound on 16 of 22 with the SDP shift, but with the eigenvalue shift only attains it on only one, and times out on remaining 21. Finally, for the "unsolved" instances we observe that the SDP shift provides 2-3x smaller gaps than the eigenvalue shift for each method.

7.5 A (simple) problem with quadratic constraints

In this section, we present a unique class of instances on which our model displays suprisingly strong performance compared to Gurobi. In this model, we minimize a 1-norm with respect to box constraints and an additional quadratic constraint stating that the 2-norm is greater than some bound. The specific model considered is

$$\begin{array}{ll} \min & \frac{100}{n} \sum_{i=1}^{n} |x_i - \varepsilon_i| \\ \text{s.t.} & x_i \in [-1, 1] \qquad\qquad i \in [\![n]\!] \\ & \sum_{i=1}^{n} x_i^2 \geq n - 0.5 \end{array} \tag{51}$$

where $\varepsilon_i = \text{rand}(-1, 1) \cdot 10^{-3}$, sorted in ascending order of $|\varepsilon_i|$. We note that this problem can be solved in closed form.

We compare against `GRB`, `GRB-S`, and `T-NMDT` for various values of n, with $L = 10$. The results are shown in Table 7 below.

The results indicate a strong performance advantage of `NN` above the competing methods shown. Note that the number of nodes for `GRB` and `GRB-S` are consistently close to 2^{n+1}, with computational times to match, while `T-NMDT` is even worse. On the other hand, while `NN` shows only moderate increases in

family	method	shift	time	gap	BB	TO
solved	GRB	eigen	1.81	0.00%	51/51	-
		sdp	0.19	0.00%	51/51	-
	CDA	eigen	3.57	0.14%	17/51	-
		sdp	0.47	0.07%	34/51	-
	NN	eigen	1.22	0.01%	14/51	-
		sdp	0.14	0.00%	16/51	-
	NMDT	eigen	1.73	0.04%	12/51	-
		sdp	0.26	0.01%	19/51	-
	T-NMDT	eigen	3.85	0.02%	19/51	-
		sdp	0.40	0.00%	24/51	-
contested	GRB	eigen	582.97	3.52%	1/22	21/22
		sdp	130.76	0.05%	16/22	6/22
	CDA	eigen	600.00	5.02%	0/22	22/22
		sdp	126.65	0.23%	0/22	5/22
	NN	eigen	562.82	2.03%	0/22	19/22
		sdp	43.24	0.02%	6/22	0/22
	NMDT	eigen	577.73	2.84%	0/22	20/22
		sdp	62.64	0.11%	1/22	2/22
	T-NMDT	eigen	594.63	3.72%	0/22	22/22
		sdp	116.27	0.05%	1/22	4/22
unsolved	GRB	eigen	-	8.64%	0/26	-
		sdp	-	4.17%	0/26	-
	CDA	eigen	-	9.22%	0/26	-
		sdp	-	4.23%	0/26	-
	NN	eigen	-	8.31%	0/26	-
		sdp	-	3.12%	26/26	-
	NMDT	eigen	-	8.47%	0/26	-
		sdp	-	3.42%	0/26	-
	T-NMDT	eigen	-	9.01%	0/26	-
		sdp	-	3.95%	0/26	-

Table 6: Computational results with two different diagonal shifts.

computational time, with a max of about 1.66s, with a far smaller node count to match.

Upon deeper investigation, we found that the primary computational advantage of the NN method stems from Gurobi's Gomory cuts. In fact, turning off presolve, heuristics, and all cuts except for Gomory cuts, the performance significantly improves over the baseline performance. For $n = 22$ the problem solves in 0.09s with only 191 nodes and 39 Gomory cuts. For $n = 250$, over an order of magnitude higher, the problem solves in 10.65s with only 13016 nodes and 378 Gomory cuts.

We expect that Gurobi is performing a spatial branching algorithm. However, the problem was constructed so that the feasible solutions are near cor-

n	method	time (sec)	gap	nodes
10	GRB	0.30	0.00%	2047
	GRB-S	0.08	0.00%	2047
	NN	0.10	0.00%	89
	T-NMDT	16.89	0.00%	45306
15	GRB	1.66	0.00%	66828
	GRB-S	1.22	0.00%	66828
	NN	0.72	0.00%	3477
	T-NMDT	318.95	0.00%	1494473
18	GRB	8.88	0.00%	528270
	GRB-S	8.08	0.00%	528210
	NN	1.20	0.00%	7757
	T-NMDT	(TO)	0.73%	3451026
20	GRB	37.15	0.00%	2099824
	GRB-S	35.49	0.00%	2099563
	NN	1.17	0.00%	9746
	T-NMDT	(TO)	1.04%	3976996
22	GRB	202.63	0.00%	8389900
	GRB-S	309.72	0.00%	8389887
	NN	1.66	0.00%	11226
	T-NMDT	(TO)	0.91%	2937766

Table 7: Computational results for a stylized quadratically constrained problem with $L = 10$.

ners of a hypercube, while at spatial branching relaxations, optimal solutions to the relaxations are close to the center. Moreover, this property is likely to hold in a spatial branch-and-bound algorithm, meaning that you will likely need to branch on all variables in order to identify the correct corner. This behavior would yield at least $O(2^n)$ spatial branching nodes, and give poor bounds throughout the branching process, as observed in the computational results. On the other hand, for the NN formulation provides an alternative branching structure that, when combined with Gomory cuts, provide excellent computational performance for this collection of instances.

7.6 More difficult problems with nonconvex quadratic constraints

To conclude our computational section, we study a "best nearest" variant of the boxQP problem that is in the spirit of the problem from Section 7.5. In more detail, for some fixed $\hat{\gamma} \in \mathbb{R}$ and $\hat{x} \in [0,1]^n$, we solve the problem

$$\begin{aligned} \min_{x} \quad & \|x - \hat{x}\|_1 \\ \text{s.t.} \quad & x'Qx + c \cdot x \leq 0.95\hat{\gamma} \\ & 0 \leq x \leq 1. \end{aligned}$$

Since each boxQP instance considered has negative objective cost, this will constrain the feasible region to those points which are "close" to optimal for the original boxQP instance. We construct 54 instances based on the the `basic` family of boxQP instances from Chen and Burer [17]. We set $\hat{x}$ as the vector of all 0.5s, and set $\hat{\gamma}$ to be the best primal cost on the underlying boxQP instance that is found by Gurobi after 10 minutes. We summarize the results in Table 8.

family	method	time (sec)	gap	BB	TO
solved	BRN	12.64	0.00%	8/8	-
	GRB	5.27	0.00%	8/8	-
	BRN-S	13.06	0.00%	8/8	-
	GRB-S	32.31	0.00%	8/8	-
	CDA	6.13	1.70%	0/8	-
	NN	4.50	0.06%	6/8	-
	NMDT	11.99	1.03%	0/8	-
	T-NMDT	22.66	0.07%	6/8	-
contested	BRN	176.89	0.00%	30/40	12/40
	GRB	66.53	0.04%	28/40	12/40
	BRN-S	173.21	0.01%	29/40	12/40
	GRB-S	566.15	10.07%	1/40	39/40
	CDA	412.17	7.12%	0/40	34/40
	NN	383.73	2.92%	5/40	33/40
	NMDT	481.24	6.10%	0/40	35/40
	T-NMDT	522.14	4.16%	4/40	36/40
unsolved	BRN	-	69.60%	0/6	-
	GRB	-	65.66%	0/6	-
	BRN-S	-	48.23%	0/6	-
	GRB-S	-	24.42%	0/6	-
	CDA	-	12.29%	0/6	-
	NN	-	8.98%	6/6	-
	NMDT	-	10.10%	0/6	-
	T-NMDT	-	10.31%	0/6	-

Table 8: Baseline computational results on 54 "best nearest" boxQP instances.

On the "solved" instances, we observe that our `NN` relaxation has the lowest mean solve time of all methods, and is able to prove optimality on 6 of 8 methods. We note that the optimality gaps for `CDA` and `NMDT` are nearly two orders of magnitude greater than what was observed on the baseline boxQP instances in Table 3. This is in keeping with the common knowledge in the global optimization (e.g. Dey and Gupte [20]) that tight relaxations for quadratic functions in the constraints do not necessarily lead to tight relaxations in the objective.

Similar to the baseline boxQP instances, we observe that the "contested" instances are a transient class where the native methods are able to terminate within the time limit with greater frequency than the relaxations, leading

to significantly smaller mean optimality gaps. On the hardest "unsolved" instances, we again see that our NN method produces the smallest optimality gap across all methods on each of the 6 instances, outperforming all other methods.

8 Conclusion

We present a simple MIP model for relaxing quadratic optimization problems that competes with robust commercial solvers in terms of solve time and bound quality. There are a number of ways that our method could be further improved. For example, we could follow the strategy of Dong and Luo [23] and implement an adaptive strategy that dynamically refines individual quadratic terms as-needed. Additionally, for boxQP instances we can potentially improve performance by leveraging the results of Hansen et al. [35], or applying existing cutting plane procedures [10]. Further, we could apply bound tightening on variables [32] or include a tail-end call to a nonlinear solver to produce an optimal primal feasible solution.

We also have performed preliminary analysis on a variant of this method to model higher-order monomials as opposed to quadratics. Fundamental results of Wei [54] show that our sawtooth functions form a basis for *any* continuous functions. Unfortunately, we have observed that a comparable approximation for x^3 seem to require a relatively large number of basis functions. We summarize our preliminary results in Appendix C. We believe it would be interesting future work to observe if this seeming obstruction is fundamental, or if compact methods for higher-order monomials can be derived through our approach.

References

1. Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: A global optimization method, αbb, for general twice-differentiable constrained NLPs—II. Implementation and computational results. Computers and Chemical Engineering **22**(9), 1159–1179 (1998)
2. Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. Theoretical advances. Computers and Chemical Engineering **22**(9), 1137–1158 (1998)
3. Anderson, R., Huchette, J., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. In: A. Lodi, V. Nagarajan (eds.) Proceedings of the 20th Conference on Integer Programming and Combinatorial Optimization, pp. 27–42. Springer International Publishing, Cham (2019). https://arxiv.org/abs/1811.08359
4. Anderson, R., Huchette, J., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. In: A. Lodi, V. Nagarajan (eds.) Integer Programming and Combinatorial Optimization, pp. 27–42. Springer International Publishing, Cham (2019)
5. Androulakis, I., Maranas, C.D.: αBB: A global optimization method for general constrained nonconvex problems. Journal of Global Optimization **7**(4), 337–363 (1995)
6. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αbb: A global optimization method for general constrained nonconvex problems. Journal of Global Optimization **7**(4), 337–363 (1995)

7. Bader, J., Hildebrand, R., Weismantel, R., Zenklusen, R.: Mixed integer reformulations of integer programs and the affine tu-dimension of a matrix. Mathematical Programming **169**(2), 565–584 (2018)
8. Billionnet, A., Elloumi, S., Lambert, A.: Extending the QCR method to general mixed-integer programs. Mathematical Programming **131**(1-2), 381–401 (2012). DOI 10.1007/s10107-010-0381-7. URL https://doi.org/10.1007/s10107-010-0381-7
9. Billionnet, A., Elloumi, S., Lambert, A.: Exact quadratic convex reformulations of mixed-integer quadratically constrained problems. Mathematical Programming **158**(1), 235–266 (2016). DOI 10.1007/s10107-015-0921-2. URL https://doi.org/10.1007/s10107-015-0921-2
10. Bonami, P., Günlük, O., Linderoth, J.: Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods. Mathematical Programming Computation **10**(3), 333–382 (2018). DOI 10.1007/s12532-018-0133-x. URL https://doi.org/10.1007/s12532-018-0133-x
11. Bunel, R., Lu, J., Turkaslan, I., Torr, P.H., Kohli, P., Kumar, M.P.: Branch and bound for piecewise linear neural network verification (2019). https://arxiv.org/abs/1909.06588
12. Burer, S., Saxena, A.: The MILP road to MIQCP. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, pp. 373–405. Springer New York (2012)
13. Castillo, P.A.C., Castro, P.M., Mahalec, V.: Global optimization of MIQCPs with dynamic piecewise relaxations. Journal of Global Optimization **71**(4), 691–716 (2018). DOI 10.1007/s10898-018-0612-7. URL https://doi.org/10.1007/s10898-018-0612-7
14. Castro, P.M.: Normalized multiparametric disaggregation: an efficient relaxation for mixed-integer bilinear problems. Journal of Global Optimization **64**(4), 765–784 (2015)
15. Castro, P.M.: Tightening piecewise McCormick relaxations for bilinear problems. Computers & Chemical Engineering **72**, 300–311 (2015). DOI 10.1016/j.compchemeng.2014.03.025. URL https://doi.org/10.1016/j.compchemeng.2014.03.025
16. Castro, P.M., Liao, Q., Liang, Y.: Comparison of mixed-integer relaxations with linear and logarithmic partitioning schemes for quadratically constrained problems. Optimization and Engineering (2021). DOI 10.1007/s11081-021-09603-5. URL https://doi.org/10.1007/s11081-021-09603-5
17. Chen, J., Burer, S.: Globally solving nonconvex quadratic programming problems via completely positive programming. Mathematical Programming Computation **4**(1), 33–52 (2012)
18. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Management Science **49**(9), 1268–1273 (2003)
19. Dantzig, G.B.: On the significance of solving linear programming problems with some integer variables. Econometrica, Journal of the Econometric Society pp. 30–44 (1960)
20. Dey, S.S., Gupte, A.: Analysis of milp techniques for the pooling problem. Operations Research **63**(2), 412–427 (2015)
21. Dey, S.S., Kazachkov, A.M., Lodi, A., Mu, G.: Cutting plane generation through sparse principal component analysis. URL http://www.optimization-online.org/DB_HTML/2021/02/8259.html
22. Dong, H.: Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. SIAM Journal on Optimization **26**(3), 1962–1985 (2016)
23. Dong, H., Luo, Y.: Compact disjunctive approximations to nonconvex quadratically constrained programs (2018)
24. Dunning, I., Huchette, J., Lubin, M.: JuMP: A modeling language for mathematical optimization. SIAM Review **59**(2), 295–320 (2017)
25. Elloumi, S., Lambert, A.: Global solution of non-convex quadratically constrained quadratic programs. Optimization Methods and Software **34**(1), 98–114 (2019). DOI 10.1080/10556788.2017.1350675. URL https://doi.org/10.1080/10556788.2017.1350675
26. Fortet, R.: L'algebre de boole et ses applications en recherche operationnelle. Trabajos de Estadistica **11**(2), 111–118 (1960). DOI 10.1007/bf03006558. URL https://doi.org/10.1007/bf03006558

27. Foss, F.A.: The use of a reflected code in digital control systems. Transactions of the I.R.E. Professional Group on Electronic Computers **EC-3**(4), 1–6 (1954). DOI 10.1109/irepgelc.1954.6499244. URL https://doi.org/10.1109/irepgelc.1954.6499244
28. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex $0-1$ mixed integer programs. Math. Program., Ser. A **106**, 225–236 (2006)
29. Frangioni, A., Gentile, C.: SDP diagonalizations and perspective cuts for a class of nonseparable MIQP. Operations Research Letters **35**(2), 181–185 (2007)
30. Furini, F., Traversi, E., Belotti, P., Frangioni, A., Gleixner, A., Gould, N., Liberti, L., Lodi, A., Misener, R., Mittelmann, H., Sahinidis, N.V., Vigerske, S., Wiegele, A.: QPLIB: a library of quadratic programming instances. Mathematical Programming Computation **11**(2), 237–265 (2019)
31. Galli, L., Letchford, A.N.: A compact variant of the qcr method for quadratically constrained quadratic 0–1 programs. Optimization Letters **8**(4), 1213–1224 (2014). DOI 10.1007/s11590-013-0676-8. URL https://doi.org/10.1007/s11590-013-0676-8
32. Galli, L., Letchford, A.N.: A binarisation heuristic for non-convex quadratic programming with box constraints. Operations Research Letters **46**(5), 529–533 (2018). DOI 10.1016/j.orl.2018.08.005. URL https://doi.org/10.1016/j.orl.2018.08.005
33. Glover, F.: Improved linear integer programming formulations of nonlinear integer problems. Management Science **22**(4), 455–460 (1975). DOI 10.1287/mnsc.22.4.455. URL https://doi.org/10.1287/mnsc.22.4.455
34. Hammer, P., Ruben, A.: Some remarks on quadratic programming with 0-1 variables. Revue Francaise D Automatique Informatique Recherche Operationnelle **4**(3), 67–79 (1970)
35. Hansen, P., Jaumard, B., Ruiz, M., Xiong, J.: Global minimization of indefinite quadratic functions subject to box constraints. Naval Research Logistics (NRL) **40**(3), 373–392 (1993). DOI https://doi.org/10.1002/1520-6750(199304)40:3⟨373::AID-NAV3220400307⟩3.0.CO;2-A. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28199304%2940%3A3%3C373%3A%3AAID-NAV3220400307%3E3.0.CO%3B2-A
36. Huchette, J., Vielma, J.P.: Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. Operations Research (To appear). https://arxiv.org/abs/1708.00050
37. Huchette, J.A.: Advanced mixed-integer programming formulations : methodology, computation, and application. Ph.D. thesis, Massachusetts Institute of Technology (2018)
38. Kaibel, V., Pashkovich, K.: Constructing Extended Formulations from Reflection Relations, pp. 77–100. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
39. Lee, J., Wilson, D.: Polyhedral methods for piecewise-linear functions I: the lambda method. Discrete Applied Mathematics **108**, 269–285 (2001)
40. Magnanti, T.L., Stratila, D.: Separable concave optimization approximately equals piecewise linear optimization. In: D. Bienstock, G. Nemhauser (eds.) Lecture Notes in Computer Science, vol. 3064, pp. 234–243. Springer (2004)
41. Misener, R., Floudas, C.A.: Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. Mathematical Programming **136**(1), 155–182 (2012). DOI 10.1007/s10107-012-0555-6. URL https://doi.org/10.1007/s10107-012-0555-6
42. Nagarajan, H., Lu, M., Wang, S., Bent, R., Sundar, K.: An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. Journal of Global Optimization **74**, 639–675 (2019)
43. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. Operations Research Letters **27**, 1–5 (2000)
44. Pardalos, P., Vavasis, S.: Quadratic programming with one negative eigenvalue is NP-hard. Journal of Global Optimization **1**(1), 15–22 (1991)
45. Phan-huy-Hao, E.: Quadratically constrained quadratic programming: Some applications and a method for solution. Zeitschrift für Operations Research **26**(1), 105–119 (1982)
46. Savage, C.: A survey of combinatorial gray codes. SIAM Review **39**(4), 605–629 (1997)
47. Saxena, A., Bomani, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations. Mathematical Programming **130**, 359–413 (2011)

48. Serra, T., Ramalingam, S.: Empirical bounds on linear regions of deep rectifier networks (2018). https://arxiv.org/abs/1810.03370
49. Serra, T., Tjandraatmadja, C., Ramalingam, S.: Bounding and counting linear regions of deep neural networks. In: Thirty-fifth International Conference on Machine Learning (2018)
50. Tjeng, V., Xiao, K., Tedrake, R.: Verifying neural networks with mixed integer programming. In: International Conference on Learning Representations (2019)
51. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. Operations Research **58**(2), 303–315 (2010)
52. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. Operations Research **58**(2), 303–315 (2010). DOI 10.1287/opre.1090.0721. URL https://doi.org/10.1287/opre.1090.0721
53. Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Mathematical Programming **128**(1-2), 49–72 (2009). DOI 10.1007/s10107-009-0295-4. URL https://doi.org/10.1007/s10107-009-0295-4
54. Wei, Y.: Triangular function analysis. Computers & Mathematics with Applications **37**(6), 37–56 (1999). DOI 10.1016/s0898-1221(99)00075-9. URL https://doi.org/10.1016/s0898-1221(99)00075-9
55. Wiese, S.: A computational practicability study of MIQCQP reformulations. https://docs.mosek.com/whitepapers/miqcqp.pdf (2021). Accessed: 2021-02-22
56. Xia, W., Vera, J.C., Zuluaga, L.F.: Globally solving nonconvex quadratic programs via linear integer programming techniques. INFORMS Journal on Computing **32**(1), 40–56 (2020). DOI 10.1287/ijoc.2018.0883. URL https://doi.org/10.1287/ijoc.2018.0883
57. Yarotsky, D.: Error bounds for approximations with deep relu networks. Neural Networks **94**, 103 – 114 (2017)

A Normalized multi-parametric disaggregation technique

We present a standard approach to descretizing continuous variables for handling bilinear products in nonlinear models. This approach is perhaps the most straightforward way to convert bilinear problems to MILPs and has been referred to as *Normalized Multi-Parametric Disaggregation Technique* (NMDT) [14]. We adapt the bilinear approach here to a squaring a single variable.

Consider $x \in [0,1]$, and let L be a positive integer. We then use the representation

$$x = \sum_{i=1}^{p} 2^{-i}\beta_i + \Delta x \tag{52a}$$

$$\beta_i \in \{0,1\} \qquad i \in [\![L]\!] \tag{52b}$$

$$\Delta x \in [0, 2^{-L}], \tag{52c}$$

where L is the number of binary variables to use.

Multiplying (52a) by x, and substituting the representation into the $x\Delta x$ term, we obtain

$$y = x \cdot x \quad = \sum_{i=1}^{L} 2^{-i} x\beta_i + x\Delta x \quad = \sum_{i=1}^{L} 2^{-i} x\beta_i + \left(\sum_{i=1}^{L} 2^{-i}\beta_i + \Delta x\right)\Delta x$$
$$= \sum_{i=1}^{L} 2^{-i}(x+\Delta x)\beta_i + \Delta x^2$$

Now, using the fact that $x + \Delta x \in [0, 1 + 2^{-L}]$, first lift the model by adding variables u_i and Δu such that $u_i = (x + \Delta x)\beta_i$ and $\Delta u = \Delta x^2$, and then we relax these equations using McCormick Envelopes.

Given bounds $x \in [\underline{x}^{\min}, \underline{x}^{\max}]$ and $\beta \in [0,1]$, The McCormick envelope $\mathcal{M}(x,\beta)$ is defined as the following relaxation of $u = x\beta$

$$\mathcal{M}(x,\beta) = \left\{(x,\beta,y) \in [\underline{x}^{\min}, \underline{x}^{\max}] \times [0,1] \times \mathbb{R} \ : (55)\right\}. \tag{54}$$

$$\begin{aligned} \underline{x}^{\min} \cdot \beta \le u \le \underline{x}^{\max} \cdot \beta \\ x - \underline{x}^{\max} \cdot (1-\beta) \le u \le x - \underline{x}^{\min} \cdot (1-\beta) \end{aligned} \tag{55}$$

To approximate $u = x^2$ with $x \in [0, \underline{x}^{\max}]$, this becomes

$$\mathcal{M}(x) = \left\{(x,u) \in [0, \underline{x}^{\max}] \times \mathbb{R} \ : \ \begin{array}{c} u \ge 0 \\ \underline{x}^{\max}(2x - \underline{x}^{\max}) \le u \le \underline{x}^{\max} \cdot x \end{array}\right\}. \tag{56}$$

We present two ways to use this approach. The first is the most direct use of `NMDT`, as used in [14]. This model is

$$x = \sum_{i=1}^{L} 2^{-i}\beta_i + \Delta x \tag{57a}$$

$$y = \sum_{i=1}^{L} 2^{-i}u_i + \Delta u \tag{57b}$$

$$(x, \beta_i, u_i) \in \mathcal{M}(x, \beta_i) \qquad i \in [\![L]\!] \tag{57c}$$

$$(\Delta x, x, \Delta u) \in \mathcal{M}(\Delta x, x) \tag{57d}$$

$$\beta_i \in \{0,1\} \qquad i \in [\![L]\!] \tag{57e}$$

$$\Delta x \in [0, 2^{-L}] \tag{57f}$$

Here, the only error introduced in the relaxation is from $\Delta u = x\Delta x$, yielding a maximum error of 2^{-L-2}, again occurring when $\Delta x = 2^{-L-1}$.

Alternatively, we consider the expansion of the $x\Delta x$ term. We thus obtain the `T-NMDT` relaxation for $y = x^2$.

$$x = \sum_{i=1}^{L} 2^{-i}\beta_i + \Delta x \tag{58a}$$

$$y = \sum_{i=1}^{L} 2^{-i}u_i + \Delta u \tag{58b}$$

$$(x + \Delta x, \beta_i, u_i) \in \mathcal{M}(x + \Delta x, \beta_i) \qquad i \in [\![L]\!] \tag{58c}$$

$$(\Delta x, \Delta u) \in \mathcal{M}(\Delta x) \tag{58d}$$

$$\beta_i \in \{0,1\} \qquad i \in [\![L]\!] \tag{58e}$$

$$\Delta x \in [0, 2^{-L}] \tag{58f}$$

Since β_i is binary, $u_i = \beta_i(x + \Delta x)$ is represented exactly. Thus, the only possible error is introduced in the relaxation of $\Delta y = \Delta x^2$, which yields a maximum error of 2^{-2L-2}, occurring when $\Delta x = 2^{-L-1}$.

Now, the expected error of `T-NMDT` is the expected error from the relaxation of $\Delta y = \Delta x^2$. Modeling Δx as a uniform random variable within its bounds $[0, 2^{-L}]$, and noting that the only overestimator from (56) is $y \le 2^{-L}\Delta x$ we obtain expected overapproximation error

$$\begin{aligned} \mathbb{E}(2^{-L}\Delta x - \Delta x^2) &= \textstyle\int_0^{2^{-L}} 2^L(2^{-L}\Delta x - \Delta x^2)\mathrm{d}\Delta x \\ &= 2^L \textstyle\int_0^{2^{-L}} (2^{-L}\Delta x - \Delta x^2)\mathrm{d}\Delta x \\ &= 2^L(\tfrac{1}{6}(2^{-L})^3) \\ &= \tfrac{1}{6}2^{-2L}. \end{aligned} \tag{59}$$

Similarly, the expected underapproximation error can be computed as $\frac{1}{12}2^{-2L}$.

B Additional baseline computation summaries

In Table 9 we summarize the results of our baseline experiments stratified by the number of decision variables as in, e.g., Table 4 of Dey et al. [21].

family	method	time (sec)	gap	BB	TO
$n \in [20, 30]$	BRN	0.19	0.00%	18/18	0/33
	CPLEX	0.20	0.00%	18/18	0/18
	GRB	0.14	0.00%	18/18	0/18
	BRN-S	0.34	0.00%	18/18	0/18
	GRB-S	0.05	0.00%	18/18	0/18
	CDA	0.16	0.06%	2/18	0/18
	NN	0.05	0.00%	9/18	0/18
	NMDT	0.09	0.01%	7/18	0/18
	T-NMDT	0.11	0.00%	8/18	0/18
$n \in [40, 50]$	BRN	0.46	0.00%	33/33	0/33
	CPLEX	0.70	0.00%	33/33	0/33
	GRB	0.37	0.00%	33/33	0/33
	BRN-S	0.87	0.00%	33/33	0/33
	GRB-S	0.54	0.00%	33/33	0/33
	CDA	1.00	0.07%	3/33	0/33
	NN	0.36	0.00%	16/33	0/33
	NMDT	0.61	0.03%	11/33	0/33
	T-NMDT	0.98	0.01%	15/33	0/33
$n \in [60, 80]$	BRN	13.66	0.00%	15/21	6/21
	CPLEX	15.10	0.00%	19/21	3/21
	GRB	9.99	0.00%	16/21	5/21
	BRN-S	25.29	0.00%	15/21	6/21
	GRB-S	112.85	0.00%	12/21	8/21
	CDA	112.31	0.30%	0/21	7/21
	NN	37.26	0.04%	4/21	3/21
	NMDT	53.73	0.12%	2/21	4/21
	T-NMDT	110.42	0.07%	2/21	6/21
$n \in [90, 125]$	BRN	261.48	0.24%	9/27	18/27
	CPLEX	218.22	0.13%	13/27	16/27
	GRB	170.56	0.20%	11/27	16/27
	BRN-S	375.45	0.55%	7/27	20/27
	GRB-S	578.95	3.43%	1/27	26/27
	CDA	569.53	3.84%	0/27	26/27
	NN	533.97	2.35%	14/27	25/27
	NMDT	543.72	2.84%	0/27	25/27
	T-NMDT	563.94	3.39%	0/27	26/27

Table 9: Computational results with instances stratified based on number of variables.

C General representations with sawtooth bases

The premise our formulation is that the function $y = x^2$ can be arbitrarily closely approximated by a series of sawtooth functions. We discuss here if such approximations could conveniently apply to other polynomials.

In [54], the authors present a Fourier series-like method that leverages orthogonal triangular functions to derive a convergent class of L_2-optimal approximations for general functions on the interval $[-\pi, \pi]$. Define the periodic triangular functions

$$\begin{aligned} X(x) &= \begin{cases} \frac{\pi^2+2\pi x}{8} & -\pi < x + 2\pi k \leq 0, k \in \mathbb{Z} \\ \frac{\pi^2-2\pi x}{8} & 0 < x + 2\pi k \leq \pi, k \in \mathbb{Z} \end{cases} \\ Y(x) &= \begin{cases} \frac{\pi x}{4} & -\frac{\pi}{2} < x + 2\pi k \leq \frac{\pi}{2}, k \in \mathbb{Z} \\ \frac{\pi^2-\pi x}{4} & \frac{\pi}{2} < x + 2\pi k \leq \frac{3\pi}{2}, k \in \mathbb{Z} \end{cases} \end{aligned} \tag{60}$$

The authors then build their orthogonal basis functions using an orthogonal linear transformation of the basis

$$1, X(x), Y(x), X(2x), Y(2x), \ldots, X(nx), Y(nx).$$

However, as with Fourier series approximations, this method has the limitation that all approximating functions are equal at the endpoints of the interval, resulting in a poor approximation for functions at which the endpoints are not equal. Thus, to obtain good approximations for x^3 on $[-\pi, \pi]$, we first add the linear function $-\pi^2 x$ to enforce equality at the endpoints.

Then, applying this method to x^2 and $x^3 - \pi^2 x$ on the interval $x \in [-\pi, \pi]$, we obtain the following numbers for the (L_1-error). Note that almost all of the $Y(nx)$ functions are relevant for approximating $x^3 - \pi^2 x$ (and no $X(nx)$'s), while only a few $X(nx)$ functions (and no $Y(nx)'s$) are relevant for approximating x^2.

Function	$N = 2$	$N = 4$		$N = 8$		$N = 16$		$N = 32$	
x^2	0.994	0.249	**(4)**	0.0622	**(4)**	0.0155	**(4)**	0.0039	**(3.97)**
$x^3 - \pi^2 x$	7.23	2.07	**(3.5)**	0.626	**(3.3)**	0.304	**(2.06)**	0.108	**(2.81)**

Table 10: Comparison of L_1-error. Factor of improvement over the previous value for L is shown in bold.

To investigate the outlook of sparsely approximating x^3 with triangular functions directly, we solved the following MIP to obtain the L_1-optimal triangular approximation to x^3 on the interval $[0, 1]$ using re-scaled versions of the basis functions above, and explicitly including a linear shift. We discretely approximate the $L - 1$ error via the error at uniformly-spaced points $x_1, \ldots, x_{N_p} \in [0, 1]$, allowing the inclusion of only N_f triangular functions.

$$\begin{aligned} \min \quad & \tfrac{1}{N_p} \textstyle\sum_{j=1}^{N_p} t_j \\ s.t. \quad & t_j \geq \textstyle\sum_{i=I} (\lambda_i f_i(x_j) + \lambda_0 x_j + f_c) - x_j^3 && \forall j \\ & t_j \geq -(\textstyle\sum_{i \in I} (\lambda_i f_i(x_j) + \lambda_0 x_j + f_c) - x_j^3) && \forall j \\ & -M \cdot \alpha_i \leq \lambda_i \leq M \cdot \alpha_i && \forall i \geq 1 \\ & \textstyle\sum_{i=1}^{N} \alpha_i \leq N_f \\ & \lambda_i \in [-M, M] && \forall i \\ & \alpha_i \in \{0, 1\} && \forall i \end{aligned} \tag{61}$$

The result, shown in Figure 4, suggests that it is not possible to use this triangular basis to obtain a similar-quality sparse approximation for x^3 as for x^2: the best achievable error rate for x^3 is roughly $O(N_f^{-2})$, compared to $O(2^{-2N_f})$ for the quadratic. See also Table 10 where we compare the convergence of the two approximations.

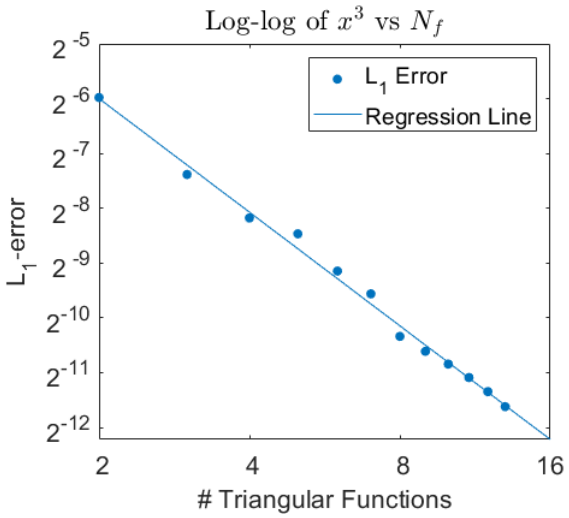


Fig. 4: The L_1-error of x^3 vs. the number of approximating triangular functions N_f. The equation of the regression line suggests an asymtotic error rate of roughly $O(N_f^{-2})$, compared to $O(2^{-2N_f})$ for the quadratic.