MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Constructing Tight Quadratic Relaxations for Global Optimization: II. Underestimating Difference-of-Convex (D.C.) Functions

Strahl, William; Raghunathan, Arvind; Sahinidis, Nikolaos V.; Gounaris, Chrysanthose

TR2026-085 June 27, 2026

Abstract

Recent advances in the efficiency and robustness of algorithms solving convex quadratically constrained quadratic programming (QCQP) problems motivate developing techniques for cre- ating convex quadratic relaxations that, although more expensive to compute, provide tighter bounds than their classical linear counterparts. In the first part of this two-paper series [Strahl et al., 2024], we developed a cutting-plane algorithm to construct convex quadratic underestimators for twice-differentiable convex functions, which we extend here to address the case of non-convex difference-of-convex (d.c.) functions as well. Furthermore, we generalize our approach to consider a hierarchy of quadratic forms, thereby allowing the construction of even tighter underestimators. Utilizing a benchmark library of d.c. functions, we demonstrate note- worthy reduction in the hypervolume between our quadratic underestimators and linear ones constructed at the same points. Additionally, we construct convex QCQP relaxations at the root node of a spatial branch-and-bound tree for a set of systematically created d.c. optimization problems in up to four dimensions, and we show that our relaxations reduce the gap between the lower bound computed by the state-of-the-art global optimization solver BARON and the optimal solution by an excess of 90%, on average.

Journal of Global Optimization 2025

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Constructing Tight Quadratic Relaxations for Global Optimization: II. Underestimating Difference-of-Convex (D.C.) Functions

William R. Strahl[1,2], Arvind U. Raghunathan[3], Nikolaos V. Sahinidis[2,4], and Chrysanthos E. Gounaris[1,2*]

[1]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

[2]Center for Advanced Process Decision-making, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

[3]Mitsubishi Electric Research Labs, Cambridge, MA, 02139, USA

[4]H. Milton Stewart School of Industrial & Systems Engineering and School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA

Abstract

Recent advances in the efficiency and robustness of algorithms solving convex quadratically constrained quadratic programming (QCQP) problems motivate developing techniques for creating convex quadratic relaxations that, although more expensive to compute, provide tighter bounds than their classical linear counterparts. In the first part of this two-paper series [Strahl et al., 2024], we developed a cutting-plane algorithm to construct convex quadratic underestimators for twice-differentiable convex functions, which we extend here to address the case of non-convex difference-of-convex (d.c.) functions as well. Furthermore, we generalize our approach to consider a hierarchy of quadratic forms, thereby allowing the construction of even tighter underestimators. Utilizing a benchmark library of d.c. functions, we demonstrate noteworthy reduction in the hypervolume between our quadratic underestimators and linear ones constructed at the same points. Additionally, we construct convex QCQP relaxations at the root node of a spatial branch-and-bound tree for a set of systematically created d.c. optimization problems in up to four dimensions, and we show that our relaxations reduce the gap between the lower bound computed by the state-of-the-art global optimization solver BARON and the optimal solution by an excess of 90%, on average.

*Corresponding author: gounaris@cmu.edu

Keywords: deterministic global optimization, convex relaxation, quadratic underestimation, cutting-plane algorithm, difference-of-convex (d.c.) functions

1 Introduction

Global optimization is utilized in a variety of research fields, including recent applications in route planning for unmanned air vehicles [Öztürk and Köksalan, 2023], energy storage system selection, design, and operation [Zantye et al., 2023], heat exchanger network synthesis [Zhou et al., 2024], resource recovery from wastewater [Durkin et al., 2024], pathway optimization using kinetic metabolic models for mammalian cells [Lu et al., 2023], profit maximization of scheduling hydrogen production with solar power and grid energy supply [Yang et al., 2022], and minimizing nitrogen oxide emissions produced by incinerating explosive waste materials [Kim et al., 2022], to name but a few. The interested reader can find a plethora of additional historical applications of global optimization in eleven different research areas nicely organized in Table 2 of Boukouvala et al. [2016]. Indeed, the ubiquitous utilization of global optimization in the literature manifests its substantial impact as a tool for scientific discovery and also highlights the importance of any advancements in this area.

The key feature of global optimization algorithms is their guarantee for identification of a global–rather than local–optimal solution. In contrast to optimization problems that are convex (i.e, the objective function and the feasible set are convex), for which a local minimum is also global [Bertsekas et al., 2003], non-convex problems exhibit additional complexity in the sense that non-convex objective surfaces and constraints and/or disjoint feasible regions lead to the existence of multiple extrema. For such problems, many global optimization algorithms, such as spatial branch-and-bound, compute rigorous lower and upper bounds on the objective value and successively refine those on partitioned space until the bounds converge (within some ε tolerance) to a global optimal solution. Without question, the quality of the bounds impacts the convergence of these algorithms.

For minimization problems, the objective function evaluated at any feasible solution provides an upper bound, while the globally optimal objective value of a relaxation of the problem provides a lower bound. Convex relaxations find extensive applications in these algorithms for determining lower bounds because their local (and hence global) minimum are efficient to compute; also, the quality of the convex relaxation used (i.e., how closely the convex relaxation approximates the

original problem) directly impacts the quality of the computed bound. To this end, an enormous amount of research has focused on developing tight convex relaxations for various specific problem structures, which has entailed significant work on deriving convex envelopes for many types of functions. Additionally, after the required expense to construct the relaxations, algorithms available to solve the convex relaxation, such as linear programming (LP) or non-linear programming (NLP) solvers, contribute to the effort required to determine the bound, and thereby contribute greatly to the overall efficiency of the algorithm. Noting the improvement in the robustness and efficiency of algorithms that solve convex quadratically constrained quadratic programs (QCQPs) [Mittelmann, 2023], we focus in this work on creating tight convex quadratic relaxations of non-convex optimization problems. In particular, we extend our previous methodology for constructing quadratic outer approximations of twice-differentiable convex functions [Strahl et al., 2024] to the case of non-convex difference-of-convex (d.c.) functions, which is a very general class of functions arising ubiquitously in global optimization applications.

In the literature, quadratics are frequently utilized to create convex relaxations of non-convex functions, but in most cases they do not result in an actual *quadratic relaxation*. The αBB methodology, for example, uses quadratics to create convex relaxations for general non-convex functions by adding a sufficiently large convex quadratic term to overcome the non-convexities of the function over the entire domain [Maranas and Floudas, 1995, Androulakis et al., 1995, Adjiman et al., 1998b,b]. The approach has been generalized in Akrotirianakis et al. [2004], Skjäl et al. [2012] to include modifications to the diagonal and nondiagonal terms of the quadratic, extended in Meyer and Floudas [2005] to use subintervals of the domain to create a spline from piecewise quadratic functions, and utilized in Gounaris and Floudas [2008a,b] to create tight piecewise linear convex relaxations for non-convex functions. While the αBB methodology utilizes quadratics in the construction of underestimators, it only creates a quadratic underestimator if the function that is underestimated is itself quadratic. Excluding this special case, none of the αBB variants discussed above produce convex quadratic relaxations.

Other studies in the literature have proposed quadratic underestimators for convex functions, but these works restrict construction of their underestimators to specific classes of functions or problems. Su et al. [2018] proposed scaling the second-order term of a Taylor series approximation at a point of construction and provided a procedure for determining the tightest scaling parameter

for underestimating a restrictive class of functions, which has been the foundational study for our own work. Buchheim and Trieu [2013] use quadratic underestimators in integer programming, introducing quadratic cuts that are generated for a specific class of functions where a matrix, Q, is known a priori to satisfy $\nabla^2 f(x) \succcurlyeq Q$ for all $x \in \mathbb{R}^n$. Olama et al. [2023] includes quadratic cuts in a mixed integer nonlinear programming algorithm derived specifically for strongly convex functions where the strong convexity parameter is easily determined. Last, there exist studies in the literature that construct quadratic underestimators for general non-convex functions, but these procedures either lack guarantees of convexity for the quadratic underestimators, or they only apply in restricted contexts. In particular, Streeter and Dillon [2022, 2023] introduce quadratic over- and underestimators for general functions based on so-called Taylor polynomial enclosures, which are efficiently constructed even for functions of high dimension, but are not guaranteed to be convex. In their work, Ouanes et al. [2015] present a procedure to create a convex quadratic underestimator by subtracting a quadratic perturbation from a linear interpolant of a general non-convex function, but their method only applies in the restricted context of minimizing a single multivariate function subject to box constraints.

As discussed, in this paper we will focus on underestimating d.c. functions. The importance of such functions in optimization is evidenced by their extensive appearance in the literature, which is due to their general applicability, succinctly captured by Tuy's statement that *"every continuous global optimization problem on a compact set can be reformulated as a d.c. optimization problem"* in Horst and Pardalos [2013] (pp. 149-150). In fact, early pioneers established that any twice-differentiable continuous function defined over a convex set in $\mathbb{R}^n$ is representable as a d.c. function [Hartman, 1959], and a later study showed that any piecewise linear continuous function can be expressed as a d.c. function too [Melzer, 1986]. Difference-of-convex functions naturally arise in a diverse number of specific applications, including but not limited to problems in signal processing, communications, and networking [Gasso et al., 2009, Alvarado et al., 2014, Nguyen and Le Thi, 2023], transportation [Holmberg and Tuy, 1999], facility location [Chen et al., 1998], image processing [Lou et al., 2015], clustering [Bagirov et al., 2016, Bagirov and Ugon, 2018], and machine learning [Le Thi and Nguyen, 2017, Awasthi et al., 2024, Askarizadeh et al., 2023].

Relaxations for d.c. functions exist in the literature, but they are not convex quadratic, except in limited cases. Additionally, to the best of our knowledge, there are no available techniques tailored

specifically for non-convex d.c. functions, but existing approaches apply to general non-convex functions, such as αBB [Adjiman et al., 1998b,a], McCormick relaxations [McCormick, 1976, Mitsos et al., 2009, Tsoukalas and Mitsos, 2014], the reformulation-linearization technique (RLT) [Sherali and Adams, 2013], and the auxiliary variable method (AVM) [Ryoo and Sahinidis, 1995, 1996], to name but a few. Recently, the reformulation-perspectification technique published in Bertsimas et al. [2023] also creates convex relaxations for d.c. functions. Each of these techniques comes with the advantages of being very broadly applicable but also suffers from specific shortcomings. For example, the αBB method, which requires the computation of an approximate bound on the largest eigenvalue over the domain of interest, often produces very loose relaxations due to poor eigenvalue bounding. On the other hand, McCormick relaxations or those stemming from RLT and AVM require additional variables to be introduced and the problem is often elevated to a much higher dimensional space. While their relaxations can be formed and solved in the higher dimensional space, they sometimes lack tightness that could otherwise be achieved by a relaxation defined in the original space.

In summary, we propose in this manuscript a methodology to construct convex quadratic underestimators for twice-differentiable d.c. functions. Our contributions to the literature include:

- An extension of the cutting-plane algorithm in Strahl et al. [2024] that yields convex quadratic relaxations for d.c. functions.
- A hierarchy of quadratic forms established to generate relaxations of increasing tightness (at a computational cost).
- A computational experiment demonstrating, both qualitatively and quantitatively, the constructed underestimators on a set of functions extracted from optimization benchmark libraries.
- A comparison of relaxations for d.c. optimization problems at the root node of a spatial branch-and-bound tree, constructed using our methodology versus the state-of-the-art, which showcases the quality of our relaxations.

The remainder of the paper is organized as follows. In Section 2, we present our proposed methodology, including (i) the extension of the algorithm in Strahl et al. [2024] to construct convex quadratic underestimators for non-convex d.c. functions, and (ii) the hierarchy established

by generalizing the quadratic form and introducing a shift term, as necessary. In Section 3, we present our computational experiments, including (i) qualitative evidence and quantitative results for underestimators generated by our methodology, and (ii) a comparison of convex quadratic relaxations for systematically created d.c. optimization problems against relaxations created by the `BARON` solver [Sahinidis, 1996]. Finally, in Section 4, we offer some conclusions and state possible directions for future work.

2 Methodology

Henceforth, we follow a notation convention where we denote vectors in boldface and matrices in capital letters. Given a d.c. function $f : \mathbb{R}^n \mapsto \mathbb{R}$, defined over a box domain $\mathcal{B} := \{\boldsymbol{x} \in \mathbb{R}^n : x_i^L \leq x_i \leq x_i^U \, \forall \, i = 1, \ldots, n\}$, we define a quadratic underestimator of f as $q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0) : \mathbb{R}^n \mapsto \mathbb{R}$, where $\boldsymbol{x}_0 \in \mathcal{B}$ is a chosen *point of construction* and α is a scaling parameter required to realize underestimation over the full domain $\boldsymbol{x} \in \mathcal{B}$.

2.1 Synopsis of Prior Work

In the first part of this two-paper series [Strahl et al., 2024], we constructed a cutting-plane algorithm that determined the tightest value of the scaling parameter α (a *scalar*) of the quadratic underestimator (1), introduced in [Su et al., 2018], for general twice-differentiable convex functions. The cutting-plane algorithm, as presented in Strahl et al. [2024], solves the partial epigraph reformulation of (2), where the objective value provides a bound on the magnitude that the quadratic overestimates the function at any point in the domain, given a value for α.

$$q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0) := f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top \alpha \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \tag{1}$$

$$\min_{\boldsymbol{x} \in \mathcal{B}} f(\boldsymbol{x}) - q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0) \tag{2}$$

Additionally, we proposed a natural extension of the cutting-plane algorithm for underestimating nonlinear functions in the context of optimization problems that feature linear constraints, which we exploited to tighten the generated underestimators by allowing the latter to overestimate in infeasible regions. In this work, we introduce modifications at two specific parts of the methodology: (i) in the treatment of formulation (2), and (ii) in Step 5 of the cutting-plane algorithm,

where the quadratic underestimator is checked for possible overestimation at enumerated vertices and corrective action is taken, if necessary.

2.2 Extension to Non-convex D.C. Functions

We extend the cutting-plane algorithm, originally devised for twice-differentiable convex functions, to accommodate non-convex d.c. functions. In particular, we observe that (2) is a d.c. optimization problem for convex f, and by now redefining f as a d.c. function, i.e., $f(\boldsymbol{x}) := h(\boldsymbol{x}) - g(\boldsymbol{x})$, where h and g are both convex, (2) becomes (3), where $g(\boldsymbol{x}) + q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0)$ is convex.

$$\min_{\boldsymbol{x} \in \mathcal{B}} h(\boldsymbol{x}) - \big[g(\boldsymbol{x}) + q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0)\big] \tag{3}$$

Consequently, we can employ a partial epigraph reformulation to produce (4), for which the objective $t - \big(g(\boldsymbol{x}) + q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0)\big)$ is a concave function defined over a convex feasible set, and the same cutting-plane algorithm in Strahl et al. [2024] executes using only a modified objective.

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathcal{B}, t \in \mathbb{R}} \quad & t - \big[g(\boldsymbol{x}) + q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0)\big] \\ \text{s.t.} \quad & h(\boldsymbol{x}) - t \leq 0 \end{aligned} \tag{4}$$

We make the following remarks: (i) the globally optimal objective value of (4) for a valid underestimator is 0, and is attained at $\boldsymbol{x} = \boldsymbol{x}_0$, assuming that an underestimating quadratic of form (1) exists at $\boldsymbol{x}_0$; and (ii) the set of vertices maintained by the cutting-plane algorithm is an outer approximation of only $h(\boldsymbol{x})$, and thus $h(\boldsymbol{x})$ should be the only function utilized to initialize and enumerate vertices. We note that the reformulation preserves the monotonically increasing property of lower bounds generated by the cutting-plane algorithm, and thus the convergence proof in Strahl et al. [2024], which relies on the proof from Hoffman [1981], holds for (4) without any alterations. Furthermore, we highlight that we can directly utilize this reformulation in conjunction with the extension presented in Strahl et al. [2024] to exploit the possible presence of linear constraints in an optimization problem so as to construct even tighter relaxations.

Finally, note that the validity of the bound on overestimation computed by (4) is predicated on the premise that $g(\boldsymbol{x}) + q(\boldsymbol{x}; \alpha, \boldsymbol{x}_0)$ is convex, and if the selection of $\boldsymbol{x}_0$ nullifies this premise, then the bound yielded by (4) is no longer valid. In the next section, we discuss properties that $\boldsymbol{x}_0$ ought to possess to ensure the construction of a valid underestimator.

2.3 Point of Construction Selection

In the case of twice-differentiable convex functions, $\nabla^2 f(\boldsymbol{x}) \succcurlyeq 0$ holds for all $\boldsymbol{x} \in \text{dom}(f)$. However, in the extension introduced in Section 2.2, we now consider underestimating non-convex functions, where in many cases there exist $\boldsymbol{x} \in \text{dom}(f)$ such that $\nabla^2 f(\boldsymbol{x}) \not\succcurlyeq 0$. The quadratic (1) directly scales $\nabla^2 f(\boldsymbol{x}_0)$ as well as, consequently, the eigenvalues of $\nabla^2 f(\boldsymbol{x}_0)$. Thus, that formula will not produce a convex quadratic underestimator for a non-convex function, unless $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ as a necessary (yet not sufficient) condition.

We highlight that the condition $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ is not sufficient in the non-convex case, as a first-order Taylor series approximation constructed at a point $\boldsymbol{x}_0$ (even where $\nabla^2 f(\boldsymbol{x}_0) \succ 0$) is not guaranteed to underestimate a non-convex function over the entire domain. Figure 1 illustrates this concept with a non-convex d.c. function and three first-order Taylor series approximations, created at three candidate points of construction that each demonstrates different possibilities: (i) a point where $\nabla^2 f(\boldsymbol{x}_0) \not\succcurlyeq 0$, for which the tangent line does not even underestimate locally, (ii) a point where $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$, for which the tangent line underestimates locally but not throughout the whole domain, and (iii) a point where $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ for which the tangent line constitutes a valid underestimator. Obviously, it is only in the last case that one can construct a valid quadratic underestimator of the form of (1).

Finally, we remark that, while $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ does not suffice by itself, it can become sufficient as long as additional conditions apply. In particular, a sufficient condition for admitting the construction of a valid quadratic underestimator is that $f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) < f(\boldsymbol{x})\ \forall \boldsymbol{x} \neq \boldsymbol{x}_0$ and $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ with one of its eigenvalues being positive; however, the first part of this condition cannot be asserted without first executing the cutting-plane algorithm itself. This implies that, in practice, we ought to initially assert the positive semi-definiteness locally at $\boldsymbol{x}_0$, but also be prepared to abort the cutting-plance algorithm whenever a certificate of non-underestimation by even the tangent line is obtained. Later in this paper (see Section 2.4.4), we introduce a modification involving the notion of a "downward shift" that enables $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ to become a necessary and sufficient condition, ensuring that the cutting-plane algorithm can yield a valid underestimator for any such $\boldsymbol{x}_0$.

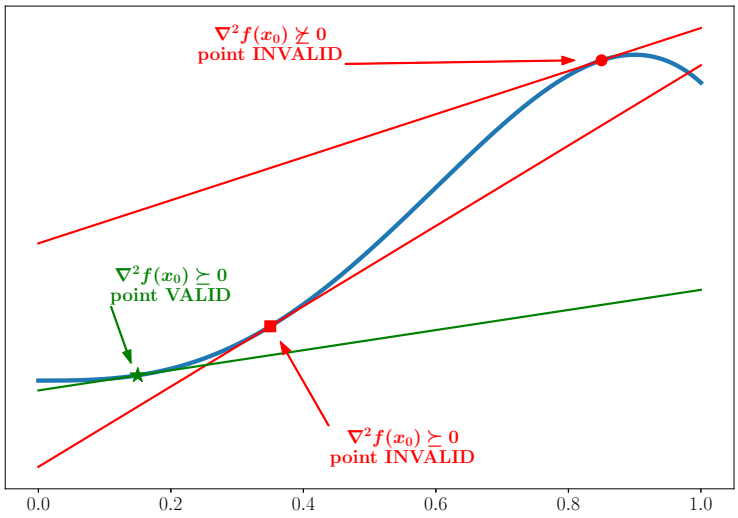

Figure 1. The univariate non-convex d.c. function $f(x) := 3x^3 - 2.5x^4$ over the domain $x \in [0, 1]$ with first-order Taylor series approximations constructed at (i) $x = 0.15$, (ii) $x = 0.35$, and (iii) $x = 0.85$, illustrating respectively three different cases: (i) underestimation over the entire domain, (ii) overestimation in part of the domain despite local convexity at the point of construction, and (iii) overestimation where the point of construction is not locally convex.

2.4 Generalization to a Hierarchy of Quadratic Underestimators

Considering the fact that the scalar α uniformly scales the elements of $\nabla^2 f(\boldsymbol{x}_0)$ in (1), we generalize the quadratic form and present a hierarchy of quadratic underestimators that allows for the construction of tighter underestimators. Before presenting the methodology for each underestimator, we restate key ideas and relevant algorithmic steps for understanding and implementing our hierarchy. In the remainder of the manuscript, we use $\boldsymbol{x}^*$ to indicate a point at which $f(\boldsymbol{x}^*) - q(\boldsymbol{x}^*; \alpha^{(k)}, \boldsymbol{x}_0) < -\varepsilon$, where ε is a user-specified tolerance and superscript "(k)" denotes the value of the scaling variable at iteration k of the cutting-plane algorithm.

Corrective action is taken in Step 5 of the cutting-plane algorithm to suitably recompute the parameter of the quadratic. Here, we restate Step 5 as well as Observation 1 from Strahl et al. [2024], which elucidates a key property for our quadratic underestimators, i.e., that the quadratic monotonically decreases with respect to a decrease in α, which can be exploited to improve algorithmic efficiency.

Step 5. *Evaluate Underestimation:*

$$\alpha^{(k)} \leftarrow \min_{(\boldsymbol{x}_v, t_v) \in \mathcal{H}^+} \left\{ \frac{f(\boldsymbol{x}_v) - \big(f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)\big)}{\frac{1}{2}(\boldsymbol{x}_v - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}_v - \boldsymbol{x}_0)} \;:\; f(\boldsymbol{x}_v) - q(\boldsymbol{x}_v; \alpha^{(k)}, \boldsymbol{x}_0) < -\varepsilon \right\},$$

where $\mathcal{H}^+$ is the set of new vertices enumerated with the introduction of a cutting plane.

Observation 1 (for proof, see Strahl et al. [2024])**.** *The quadratic underestimator in (1) monotonically decreases as α decreases; that is, $q(\boldsymbol{x}; \alpha_1, \boldsymbol{x}_0) \leq q(\boldsymbol{x}; \alpha_2, \boldsymbol{x}_0)$ for all $\alpha_1 < \alpha_2$.*

The goal of the following presentation is to generalize the quadratic underestimator form to allow the construction of tighter underestimators, while preserving the efficiency of the algorithm by enforcing the quadratic to monotonically nonincrease each time the parameters are updated. We note that changes at $\boldsymbol{x}^*$ are the same, regardless of whether the function is convex or d.c., and will therefore hereafter reference the function to be underestimated as $f(\boldsymbol{x})$, which in the d.c. case can be substituted directly with $h(\boldsymbol{x}) - g(\boldsymbol{x})$. We begin our presentation with the original method developed in Strahl et al. [2024], which for purposes of the hierarchy, we shall refer to as the "scalar" method.

2.4.1 Scalar Method

In the scalar method (also labeled method "S"), we decrement the scalar α of the quadratic form (1) using the update rule S. Here, we note the monotonic relationship of the quadratic with α, as per Observation 1, which implies that $(\boldsymbol{x}^* - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}^* - \boldsymbol{x}_0) \neq 0$ for all $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}^*) - q(\boldsymbol{x}^*; \alpha^{(k)}, \boldsymbol{x}_0) < -\varepsilon$.

$$\alpha^{(k)} \leftarrow \frac{f(\boldsymbol{x}^*) - \big(f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x}^* - \boldsymbol{x}_0)\big)}{\frac{1}{2}(\boldsymbol{x}^* - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x}^* - \boldsymbol{x}_0)} \tag{S}$$

2.4.2 Diagonal Method

We generalize the quadratic form in method "S" by introducing a diagonal matrix of scaling parameters, $A \in \mathbb{R}^{n \times n}$, into the eigenvalue decomposition of $\nabla^2 f(\boldsymbol{x}_0)$, $Q\Lambda Q^{-1}$, as shown in (5), which permits the scaling parameters, i.e., the diagonal elements of A, to modify each eigenvalue of $\nabla^2 f(\boldsymbol{x}_0)$ differently, while the off-diagonal elements are fixed to 0. Hence, we refer to this method as the "diagonal" method, or method "D".

$$q(\boldsymbol{x}; A, \boldsymbol{x}_0) := f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top Q A \Lambda Q^{-1}(\boldsymbol{x} - \boldsymbol{x}_0) \tag{5}$$

The new quadratic form (5) requires a different procedure for determining the scaling parameters, such that each new selection: (i) ensures the quadratic underestimates the function at $\boldsymbol{x}^*$; (ii) monotonically decreases the quadratic over the entire domain with each consecutive update to the parameters; and (iii) produces the tightest possible quadratic underestimator over the entire domain, when the algorithm converges. Observing that the (diagonal) elements of A participate linearly in the quadratic form, we can compute them via the linear program (D), where we explicitly enforce (i) and (ii) in the constraints.

$$\begin{aligned} \max_{A} \quad & \sum_{\boldsymbol{v}\in\mathcal{S}} q(A;\boldsymbol{v},\boldsymbol{x}_0) \\ \text{s.t.} \quad & q(A;\boldsymbol{x}^*,\boldsymbol{x}_0) \le f(\boldsymbol{x}^*) \\ & A_{ii} \le A_{ii}^{(k)} && \forall\, i \\ & A_{ii} \ge 0 && \forall\, i \\ & A_{ij} = 0 && \forall\, i,j : \{j \ne i\}, \end{aligned} \tag{D}$$

where $A_{ii}^{(k)}$ refers to the incumbent values for the diagonal elements of the scaling matrix (to be updated with the optimal solution of this LP). In the first iteration, $k = 0$, we initialize with $A_{ii}^{(0)} = 1$ for all rows $i = 1, 2, \ldots, n$.

To accomplish (iii), the above LP aims to explicitly optimize for "tightness" or quality of the produced underestimator. For this, we utilize the metric (6), defined in Strahl et al. [2024], that quantifies the tightness of the quadratic underestimator, $q(\boldsymbol{x})$, as the fractional reduction in hypervolume from a function to it, when compared to a linear underestimator, $\ell(\boldsymbol{x})$, generated at the same point of construction.

$$M_{\ell(\boldsymbol{x})}^{q(\boldsymbol{x})} = \frac{\int_{\boldsymbol{x}\in\mathrm{dom}(f)} \big(q(\boldsymbol{x}) - \ell(\boldsymbol{x})\big) d\boldsymbol{x}}{\int_{\boldsymbol{x}\in\mathrm{dom}(f)} \big(f(\boldsymbol{x}) - \ell(\boldsymbol{x})\big) d\boldsymbol{x}} \tag{6}$$

As we consider utilizing this metric as the objective in the LPs for our methods, we make the following observation, where $\mathcal{S}$ is a set of points $\boldsymbol{x} \in \mathrm{dom}(f)$.

Observation 2. *In the case that $|\mathcal{S}| \to \infty$, the constraint $q(A;\boldsymbol{v},\boldsymbol{x}_0) \le f(v)$ for all $\boldsymbol{v} \in \mathcal{S}$, in conjunction with the objective $\max_A \sum_{\boldsymbol{v}\in\mathcal{S}} q(A;\boldsymbol{v},\boldsymbol{x}_0)$, produces the tightest scaling parameters by construction, thereby achieving objective (iii).*

Since sampling an infinite amount of points at which to evaluate underestimating distances is not practical, we discretize the space by choosing $|\mathcal{S}| = 100n$ points, where n is the dimension of

the problem and the points are selected via a Latin hypercube approach. To embed the metric directly in the LP, we remove constant terms from (6) to produce the objective function in (D), which maximizes the value of the quadratic at each sampled point. However, the constraints in (D) only enforce underestimation at a single point, namely $\boldsymbol{x}^*$. This fact, considered together with the monotonicity requirement imposed on the parameters, results in the algorithm making greedy initial commitments to parameter values which, although optimal in light of $\boldsymbol{x}^*$, may ultimately yield sub-optimal parameter values at termination.

To address this behavior, we recommend augmenting the formulation of the first LP we ever solve with the constraints $q(A; \boldsymbol{v}, \boldsymbol{x}_0) \leq f(\boldsymbol{v})$ for all $\boldsymbol{v} \in \mathcal{S}$, resulting in formulation $(D^{(1)})$, which provides an update to the parameters using a holistic treatment of the space. These additional constraints are only required for the first LP executed in the cutting-plane algorithm; subsequent LPs can be relaxed to contain only the single constraint $q(A; \boldsymbol{x}^*, \boldsymbol{x}_0) \leq f(\boldsymbol{x}^*)$ (formulation D), since monotonicity guarantees underestimation for these points through the remaining execution of the algorithm.

$$
\begin{aligned}
\max_{A} \quad & \sum_{\boldsymbol{v} \in \mathcal{S}} q(A; \boldsymbol{v}, \boldsymbol{x}_0) & & \\
\text{s.t.} \quad & q(A; \boldsymbol{x}^*, \boldsymbol{x}_0) \leq f(\boldsymbol{x}^*) & & \\
& q(A; \boldsymbol{v}, \boldsymbol{x}_0) \leq f(\boldsymbol{v}) & \forall\, \boldsymbol{v} \in \mathcal{S} & \\
& A_{ii} \leq A_{ii}^{(0)} & \forall\, i & \\
& A_{ii} \geq 0 & \forall\, i & \\
& A_{ij} = 0 & \forall\, i, j : \{j \neq i\} &
\end{aligned}
\tag{$D^{(1)}$}
$$

We highlight that an interesting tradeoff arises between the number of points in set $\mathcal{S}$ and the number of LPs that have to be solved throughout the algorithm. Despite yielding a larger LP $(D^{(1)})$, which however need only be solved once at the first iteration, larger $|\mathcal{S}|$ typically leads to detecting fewer points of overestimation, creating tighter underestimators while requiring fewer LPs (D) to be solved in the long run. For problems addressed in the computational study of this manuscript, we selected $|\mathcal{S}|$ to be $100n$ inasmuch as increasing the sample size beyond $100n$ yielded very small gains (<1.5% after a 10-fold increase) in the quality of the underestimators with additional computational expense.

We remark the following regarding the diagonal scaling method: (i) $(D^{(1)})$ has $|\mathcal{S}| + 2n + 1$ constraints in the first iteration, after which (D) has $2n + 1$ constraints; thus, these LPs are small

in size; (ii) we know that the value of 1 is a valid upper bound on each element of the diagonal of A, A_{ii}, because the quadratic would overestimate locally at the point of construction if any $A_{ii} > 1$; and (iii) "S" can be viewed as a restriction of method "D", where additional constraints are added in (D) to achieve equivalency for all A_{ii}; indeed, the final parameter α determined by method "S" used for all the diagonal elements of A will be feasible to (D), but due to the inexactness introduced from sampling the space (finite set $\mathcal{S}$), formulation (D) can possibly produce an inferior underestimator, the mitigation of which is encouraged by ($D^{(1)}$).

In the remainder of the manuscript, we present a number of additional methods associated with their own formulations. While we do so in the context of a general iteration k, we recommend augmenting them in a manner similar to $D^{(1)}$ to avoid suboptimal parameter values.

2.4.3 Matrix Method

We further generalize the diagonal scaling method to allow modifications to the off-diagonal elements of A as well, where the quadratic underestimator form is the same as (5), but we remove the restriction that the off-diagonal elements of A must be 0. We shall refer to this method as the "matrix" method, or method "M".

While the quadratic form does not change from the diagonal scaling method, the introduction of off-diagonal variables in A necessitates a more involved LP to preserve the required properties of the quadratic. First and foremost, we are interested in producing *convex* quadratic underestimators, for which we had trivial guarantees in previous methods, as we directly scaled eigenvalues with $\nabla f(\boldsymbol{x}_0) \succcurlyeq 0$. However, modifying the off-diagonal elements gives rise to the possibility of creating non-convex quadratics even with non-negative diagonal elements. To that end, we explicitly enforce convexity during each parameter update by requiring A to be diagonally dominant, which is a sufficient condition for the positive semi-definiteness of A. To invoke this property, we first require symmetry on $A\Lambda$ and non-negativity of all diagonal elements A_{ii}, which ensures that $[A\Lambda]_{ii} \geq 0$ due to $\Lambda_{ii} \geq 0$ at the selected point of construction. Then, we impose the diagonal dominance property by requiring that $[A\Lambda]_{ii} \geq \sum_{j \neq i} |\, [A\Lambda]_{ij} \,|$ for all rows $i = 1, 2, \ldots, n$.

Lastly, to preserve the algorithmic efficiency of the cutting-plane algorithm achieved by exploiting the monotonicity property of the quadratic (Observation 1), we utilize the fact that $A^{(k)}\Lambda \succcurlyeq A\Lambda$ implies that $\boldsymbol{x}A^{(k)}\Lambda\boldsymbol{x} \geq \boldsymbol{x}A\Lambda\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^n$, where $A^{(k)}$ are the incumbent values of the param-

eters at the time the LP is executed. This realizes the desired monotonicity property, since the orthogonality of Q in the eigenvalue decomposition preserves the eigenvalues of $A\Lambda$. For the constraint requiring $A^{(k)}\Lambda \succcurlyeq A\Lambda$, we similarly impose the diagonal dominance property on matrices $(A^{(k)} - A)\Lambda$. Based on the above, the computation of A can be achieved via formulation (M), which can be trivially reformulated to the LP in (M′) after the introduction of auxiliary variables $S \in \mathbb{R}^{n\times n}$ and $T \in \mathbb{R}^{n\times n}$.

$$
\begin{aligned}
\max_{A} \quad & \sum_{\boldsymbol{v}\in\mathcal{S}} q(A;\boldsymbol{v},\boldsymbol{x}_0) \\
\text{s.t.} \quad & q(A;\boldsymbol{x}^*,\boldsymbol{x}_0) \le f(\boldsymbol{x}^*) \\
& A_{ii} \le A_{ii}^{(k)} && \forall\, i \\
& A_{ij}\Lambda_{jj} = A_{ji}\Lambda_{ii} && \forall\, i,j:\{j\neq i\} \\
& A_{ii} \ge 0 && \forall\, i \\
& A_{ii}\Lambda_{ii} \ge \sum_{j\neq i} |A_{ij}\Lambda_{jj}| && \forall\, i \\
& (A_{ii}^{(k)} - A_{ii})\Lambda_{ii} \ge \sum_{j\neq i} \left|(A_{ij}^{(k)} - A_{ij})\Lambda_{jj}\right| && \forall\, i,
\end{aligned}
\tag{M}
$$

$$
\begin{aligned}
\max_{A,S,T} \quad & \sum_{\boldsymbol{v}\in\mathcal{S}} q(A;\boldsymbol{v},x_0) \\
\text{s.t.} \quad & q(A;\boldsymbol{x}^*,\boldsymbol{x}_0) \le f(\boldsymbol{x}^*) \\
& A_{ii} \le A_{ii}^{(k)} && \forall\, i \\
& A_{ij}\Lambda_{jj} = A_{ji}\Lambda_{ii} && \forall\, i,j:\{j\neq i\} \\
& A_{ii} \ge 0 && \forall\, i \\
& A_{ii}\Lambda_{ii} \ge \sum_{j\neq i} S_{ij} && \forall\, i \\
& -S_{ij} \le A_{ij}\Lambda_{jj} \le +S_{ij} && \forall\, i,j:\{j\neq i\} \\
& (A_{ii}^{(k)} - A_{ii})\Lambda_{ii} \ge \sum_{j\neq i} T_{ij} && \forall\, i \\
& -T_{ij} \le (A_{ij}^{(k)} - A_{ij})\Lambda_{jj} \le +T_{ij} && \forall\, i,j:\{j\neq i\},
\end{aligned}
\tag{M′}
$$

where $A_{ij}^{(k)}$ refers to the incumbent values for the elements of the scaling matrix (to be updated with the optimal solution of this LP). Initially, we define $A^{(0)}$ to be the identity matrix.

We highlight that the LP (M′) has $n^2 + 2n(n-1)$ variables and $4n + 5n(n-1) + 1$ constraints. Despite being a more involved LP than the one encountered in the diagonal method, it is still expected to be very tractable in practice.

2.4.4 Shift

In Section 2.3, we established a necessary–but not sufficient–condition for the selection of a point of construction to yield a valid underestimator generated by our cutting-plane algorithm. Here, we introduce a term into the quadratic forms (1) and (5) to allow for their vertical (downward) shift. We show how to properly update (S), (D) (or $D^{(1)}$) and (M′) to accommodate the augmented form, and how the introduction of the shift causes the necessary condition $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ to also become sufficient for our methodology to produce valid underestimators. For all methods, we denote the scalar variable for the shift as $\gamma \in [0, \infty)$, which is initialized to $\gamma^{(0)} = 0$, and which is to be negated from the applicable quadratic form (1) or (5) in each case. To differentiate from the previous methods without shift, we augment the label of our methods that include the shift with the letter "S" (e.g., "DS" refers to the diagonal method *with shift*).

For the scalar method, we utilize a shift only if method "S" computes $\alpha < 0$, whereupon we reset $\alpha \leftarrow 0$ and update γ throughout each iteration using (SS). While computationally inexpensive, this approach (dubbed method "SS") will effectively produce a linear underestimator when the shift is indeed required (i.e., a vertical shift of the first-order Taylor series approximation).

$$\gamma^{(k)} \leftarrow q(\boldsymbol{x}^*, 0, \boldsymbol{x}_0) - f(\boldsymbol{x}^*) \tag{SS}$$

Before we present the incorporation of shift in the methods utilizing a scaling matrix A in lieu of a scalar α, we consider a hybrid method that requires identical elements in a diagonal matrix. Consistent with the rest of the hierarchy, we will refer to this as method as "UDS" (i.e., "uniform diagonal with shift"). In this method, both scalar parameters α (the common element in the diagonal) and γ can be optimized simultaneously via the LP (UDS). We also highlight the inclusion of the constraint $\gamma \geq \gamma^{(k)}$, which imposes the monotonicity property of the quadratic from iteration to iteration. Importantly, this enforcement of monotonicity preserves the requisite properties for the cutting-plane algorithm's convergence proof [Strahl et al., 2024] after the addition of the shift. We also remark that method "UDS" is a generalization of both methods "SS" and "S" inasmuch as any results determined by the latter two are also feasible to "UDS", noting however

that the relative tightness of these underestimators might be affected by the selection of the set $\mathcal{S}$.

$$\begin{aligned}
\max_{A,\gamma} \quad & \sum_{\boldsymbol{v}\in\mathcal{S}} \left[q(A;\boldsymbol{v},\boldsymbol{x}_0) - \gamma\right] \\
\text{s.t.} \quad & q(A;\boldsymbol{x}^*,\boldsymbol{x}_0) - \gamma \leq f(\boldsymbol{x}^*) \\
& A_{ii} \leq A_{ii}^{(k)} && \forall\, i \\
& A_{ii} \geq 0 && \forall\, i \\
& A_{11} = A_{jj} && \forall\, j : \{j \neq 1\} \\
& \gamma \geq \gamma^{(k)},
\end{aligned} \tag{UDS}$$

where $\gamma^{(k)}$ is the incumbent value for the shift variable (to be updated with the optimal solution of this LP).

In summary, in the hierarchy of using a mere scalar in the quadratic form, we have: (i) method "S", which may fail to produce a valid underestimator for certain points of construction in the case of non-convex d.c. functions; (ii) method "SS", which is efficient to compute but will only produce a linear underestimator, and should hence be reserved only for the points of construction where "S" fails; and (iii) method "UDS", which requires solving LPs but will produce a valid underestimator for all points of construction where $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ (at worst a linear underestimator, but likely a quadratic one). Figure 2 demonstrates an example where utilizing method "UDS" constructs a successful quadratic underestimator when method "S" would otherwise fail due to poor choice of the point of construction. Finally, we highlight that, in certain instances, a combination of $\gamma > 0$ and $\alpha > 1$ in method "UDS" could achieve even tighter underestimators, as measured by relative hypervolumes; however, for our computational studies, we keep the upper bound of parameter α at 1 and initialize $\alpha^{(0)} \leftarrow 1$.

We now turn our attention to the incorporation of shift in the methods utilizing scaling matrices, introducing methods "DS" and "MS". For this, the linear programs defined by (D) and (M$'$) are updated analogously to (UDS), where we incorporate the variable γ and impose the constraint to preserve monotonicity. This results in linear programs (DS) and (MS), respectively for the two methods.

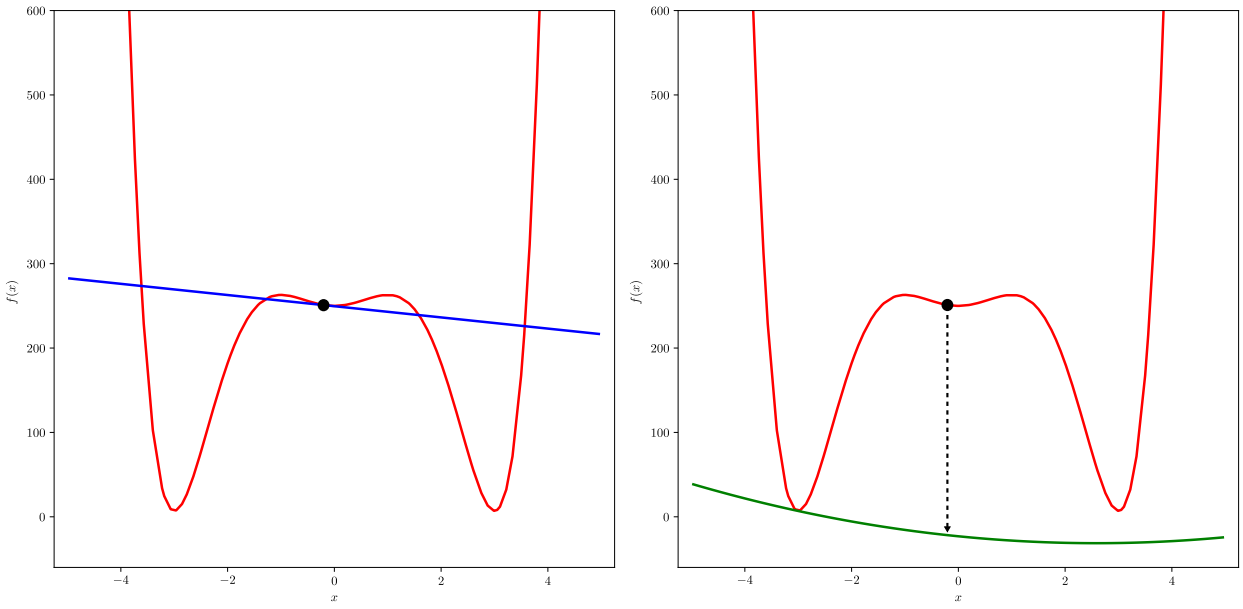

Figure 2. Introducing the shift parameter allows the construction of a valid quadratic underestimator (right, in green) for the d.c. function $f(x) = \left(27x^2 + x^6 + 250\right) - 15x^4$ at $x_0 = -0.125$ even when, at the same point of construction, a linear underestimator is not valid (left, in blue).

$$
\begin{aligned}
\max_{A,\gamma} \quad & \sum_{\boldsymbol{v}\in\mathcal{S}} \left[q(A;\boldsymbol{v},\boldsymbol{x}_0) - \gamma\right] \\
\text{s.t.} \quad & q(A;\boldsymbol{x}^*,\boldsymbol{x}_0) - \gamma \leq f(\boldsymbol{x}^*) \\
& A_{ii} \leq A_{ii}^{(k)} \qquad\qquad \forall\, i \\
& A_{ii} \geq 0 \qquad\qquad \forall\, i \\
& \gamma \geq \gamma^{(k)}
\end{aligned}
\tag{DS}
$$

$$
\begin{aligned}
\max_{A,\gamma,S,T} \quad & \sum_{\boldsymbol{v}\in\mathcal{S}} \left[q(A;\boldsymbol{v},\boldsymbol{x}_0) - \gamma\right] & \\
\text{s.t.} \quad & q(A;\boldsymbol{x}^*,\boldsymbol{x}_0) - \gamma \leq f(\boldsymbol{x}^*) & \\
& A_{ii} \leq A_{ii}^{(k)} & \forall\, i \\
& A_{ij}\Lambda_{jj} = A_{ji}\Lambda_{ii} & \forall\, i,j : \{j \neq i\} \\
& A_{ii} \geq 0 & \forall\, i \\
& A_{ii}\Lambda_{ii} \geq \sum_{j\neq i} S_{ij} & \forall\, i \\
& -S_{ij} \leq A_{ij}\Lambda_{jj} \leq +S_{ij} & \forall\, i,j : \{j \neq i\} \\
& (A_{ii}^{(k)} - A_{ii})\Lambda_{ii} \geq \sum_{j\neq i} T_{ij} & \forall\, i \\
& -T_{ij} \leq (A_{ij}^{(k)} - A_{ij})\Lambda_{jj} \leq +T_{ij} & \forall\, i,j : \{j \neq i\} \\
& \gamma \geq \gamma^{(k)} &
\end{aligned}
\tag{MS}
$$

Restating the motivation for considering the shift parameter, we highlight that its introduction allows the various methods to yield at worst a linear underestimator, for any $\boldsymbol{x}_0$ where $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$, whereas the methods without the shift may fail. Here, we offer one theoretical condition and one empirical explanation for the case where the methods incorporating shift (i.e., SS, UDS, DS, MS) will yield a linear underestimator. Consider a function and a point of construction where $\nabla^2 f(\boldsymbol{x}_0) = 0$; in this case, all these methods (which scale $\nabla^2 f(\boldsymbol{x}_0)$) generate a zero second-order term, thereby producing a linear underestimator. Empirically, it is also the case that, depending on the specific instance under consideration, the algorithm may drive a handful of diagonal elements A_{ii} (alongside their respective rows A_{ij}) to zero to maximize the metric at beginning iterations, and these parameters will remain at the value of zero over future iterations because of the monotonicity requirement. Hence, any points of overestimation that the algorithm may identify subsequently will put further pressure towards decreasing any remaining non-zero elements (to possibly also equal zero) so as to validly underestimate. In such cases, a linear underestimator could again be constructed despite there possbily existing a valid quadratic underestimator. We re-emphasize that the constraints over the point sample $\mathcal{S}$ introduced in the LP for the first iteration is intended to remediate this suboptimal result.

Overall, we have presented a hierarchy of methods that can produce quadratic underestimators of non-convex d.c. functions at user-specified points of construction. This hierarchy is presented in a diagram in Figure 3, where tighter–but more computationally expensive–underestimators lie at

the bottom and to the right. We remind readers that all the methods not in blue require solving an LP to update their parameters. Selecting a particular method to use in practice depends on the particular problem to be solved. First, disregarding the shift, the more complex methods can achieve the greatest gains compared to simpler methods in higher dimensions, where there is great variability amongst the eigenvalues of the function at each point. Otherwise, if the eigenvalues are all approximately equivalent, a uniform scaling can achieve a close result. Second, the shift parameter adds a very slight increase in the computation time and is generally recommended for the diagonal and matrix methods, which already require solving an LP. However, constructing quadratic underestimators using the (UDS) method–rather than linear underestimators using the SS method–and including the shift elevates the complexity from an analytic evaluation to an LP, and would thus have to be judged empirically based on the resulting tightness-tractability tradeoff in the context of functions of interest.

Finally, we emphasize the fact that all quadratic forms and update procedures are immediately amenable to, without further modification, the methodological extension presented in Strahl et al. [2024], where information of external linear constraints in optimization problems can be exploited to produce tighter underestimators by allowing them to overestimate in infeasible regions.

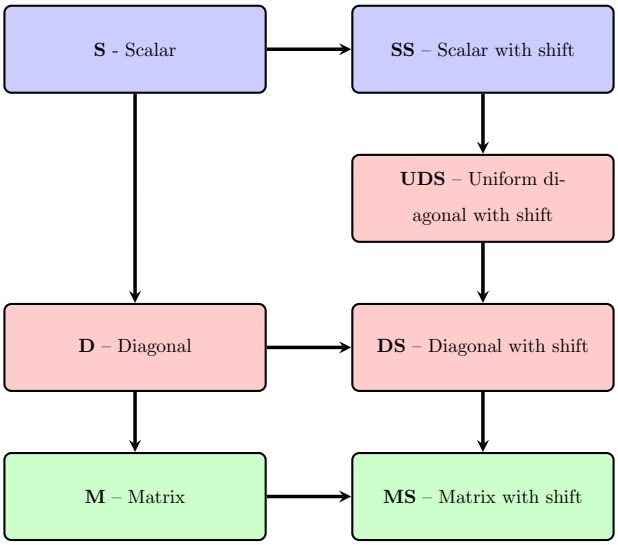

Figure 3. The hierarchy of our methods producing quadratic underestimators, based on scalar (blue), diagonal (red), and matrix (green) scalings, with the addition of the shift in those methods appearing on the right column. The arrows indicate the generation of tighter underestimators.

3 Computational Study

We demonstrate the performance of our hierarchy of quadratic underestimators generated by our cutting-plane algorithm for non-convex d.c. functions in two distinct computational experiments. In the first experiment in Section 3.1, we extract d.c. functions from optimization problems found in benchmark libraries, construct underestimators for them, and showcase their efficiency and tightness. In the second experiment in Section 3.2, we compare the root node relaxation of the state-of-the-art global optimization solver `BARON` with a convex quadratic relaxation constructed via our methodology on optimization problems with d.c objectives and constraints.

3.1 Hierarchy Comparison Study

For our study on the tightness and computational efficiency of our hierarchy of underestimators, we first looked into collecting d.c. functions from the COCONUT library [Shcherbina et al., 2003]. The location of the problems in this library and the explicit expressions adopted for our computational study are included in Table A1 in the Appendix. We note that we employed the relationships $x_1x_2 = \frac{1}{2}\left(x_1+x_2\right)^2 - \frac{1}{2}\left(x_1^2+x_2^2\right)$ and $x_1^2x_2^2 = \frac{1}{2}\left(x_1^2+x_2^2\right)^2 - \frac{1}{2}\left(x_1^4+x_2^4\right)$ to create d.c. representations for many of the functions found in the libraries. Where necessary, we assigned bounds to otherwise unbounded variables as noted in Table A1. Remarkably, the COCONUT library does not contain many d.c. functions, and through the above process, we were only able to extract 3 one-dimensional and 7 two-dimensional d.c. functions. To add to those, as well as to extend the database with functions of three and four dimensions, we followed a complementary approach. More specifically, we begin with a core set of six univariate non-convex d.c. functions (see Table A2 in the Appendix), which we devised by subtracting chosen convex functions, and which we add up to construct higher dimensional d.c. functions while also adding a "linking" term, $\mathcal{L}_i(\boldsymbol{x}) := \frac{1}{2p}(\sum_{j=1}^{p} x_j)^4$, where p is the number variables included in a particular multi-dimensional d.c. function. The linking term induces inseparable variable dependencies on the function output, which prevents trivially separating terms that could otherwise have been underestimated separately. The overall process takes special care to ensure the functions are non-convex and multi-modal over the prescribed domain. The exact functions derived via this approach and augmenting the initial dataset are displayed in Table A3 in the Appendix. Among the two approaches, we compiled a set of 9 d.c. functions per each dimension, one through four. For all of the functions in our study, we scale their

range to lie within the interval $[-1, 1]$ by using the scaling factor $1/\max\{|\min_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x})|, |\max_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x})|\}$.

For each function, we generated sample points from the domain using Latin hypercube sampling and assessed the local convexity of the function at these points via eigenvalue decomposition of the Hessian. We continued this process until we accumulated 10 points of construction where the function is locally convex, discarding along the way any points for which this was not true. The underlying motivation for utilizing Latin hypercube sampling here is that it provides a practical technique to source points from diverse regions of the domain where the condition $\nabla^2 f(\boldsymbol{x}) \succcurlyeq 0$ might happen to be true. Finally, we clarify that we used the value $\varepsilon = 0.001$ for the convergence tolerance of the cutting-plane algorithm. Our computational experiments were executed on a machine equipped with a 1.80GHz Intel(R) Core(TM) i7-8565U CPU running on a Ubuntu 22.04 virtual machine with 8GB RAM and 4 logical processors, where we have implemented the algorithms in Python. To solve LPs, we used the open source HiGHS [Huangfu and Hall, 2018] linear program solver, which is included in the Python `SciPy` package [Virtanen et al., 2020], and which was found to be adequate to address the small sized LPs encountered in this study.

3.1.1 Results

Qualitatively, Figure 4 depicts the underestimators created by the methods "S", "D" and "M" for a function from the `dipigri` optimization problem, highlighting the possible improvement in underestimator tightness by using ever more sophisticated techniques. Additionally, Figures 5 and 6 visualize some examples of our quadratic underestimators constructed for non-convex d.c. functions, as generated by the "S" method.

Performing a quantitative comparison among the methods in the hierarchy requires categorizing the functions along with points of construction into two groups: (i) those for which methods that do not utilize shift can construct a successful quadratic underestimator, where the entire hierarchy can be compared; and (ii) those for which construction procedures require the shift for success (i.e., first-order Taylor series approximations do not underestimate across the full domain of interest), and hence, where only the methods "SS", "UDS", "DS" and "MS" can be compared. Accordingly, Tables 1 and 2 display the average tightness metric for $1000n$ sample points, computed via (6), and the CPU time (in milliseconds) required to compute the underestimators for groups (i) and (ii), respectively.

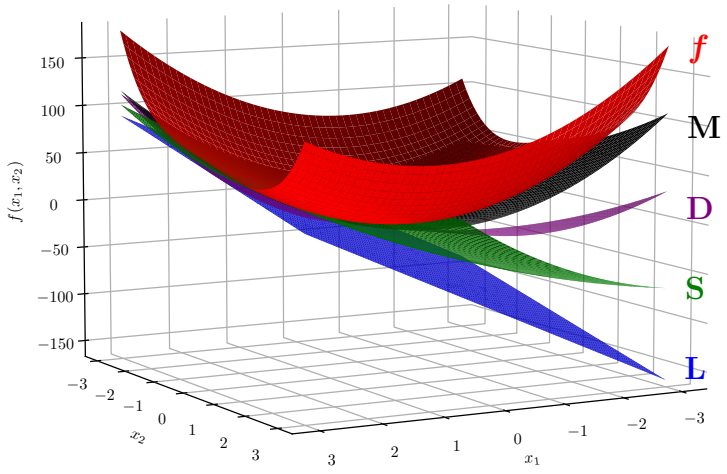

Figure 4. Qualitative evidence highlighting the improvement in the quality of underestimators by using more general methods from our hierarchy. The d.c. function $f(x_1, x_2) = \left(x_2^4 + 9x_1^2 + 2x_2^2\right) - 2\left(x_1 + x_2\right)^2$ is underestimated in the box domain $[-3, 3]^2$ using $(1.84, -1.04)$ as the point of construction. Here, "f" is the function, "L" is the linear underestimator, while "S", "D", and "M" are the quadratic underestimators constructed using the respective methods.

Table 1. Computational results for different methods in the hierarchy, for points of construction that admit a valid underestimator without the shift.

Dimension	# Functions	# Points of Construction	Avg. Metric							Avg. CPU (ms)						
			S	D	M	SS	UDS	DS	MS	S	D	M	SS	UDS	DS	MS
1	9	37	0.663	0.663	0.663	0.663	0.663	0.663	0.663	18	19	22	13	21	21	25
2	9	42	0.435	0.502	0.510	0.435	0.480	0.523	0.524	120	157	190	89	133	170	191
3	9	18	0.503	0.725	0.726	0.503	0.673	0.725	0.727	710	1350	1600	527	736	1439	1660
4	9	9	0.383	0.681	0.682	0.383	0.605	0.683	0.695	4180	10598	11173	3397	3201	9277	9743

For functions of one dimension, Table 1 clearly shows that all methods perform similarly for points of construction that admit valid underestimators without requiring a shift, which is simply explained by the equivalency of the methods in the univariate case. In the case of higher dimensional functions, however, Table 1 shows that the shift provides some flexibility to create tighter underestimators. This is evidenced by the sharp increase in the average metric by approximately 10%, 34%, and 58% between methods "S" and "UDS" for dimensions 2, 3, and 4, respectively. Notably, for dimensions 3 and 4, elevating the complexity of the method from "S" to "D" yielded

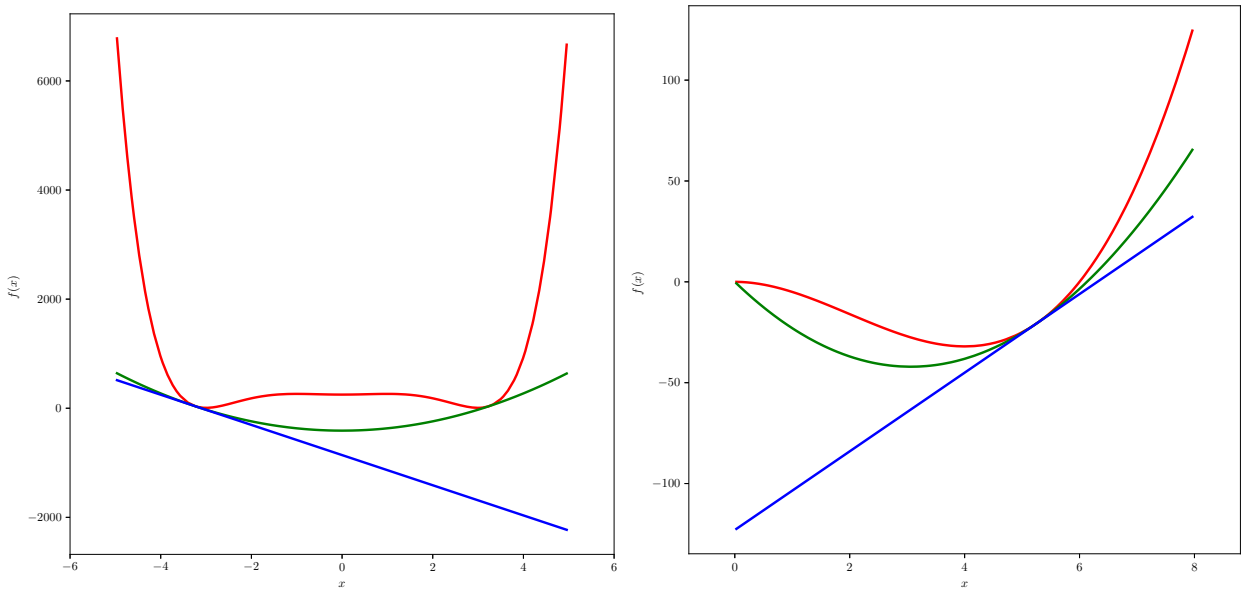

Figure 5. Quadratic underestimators (green) for the univariate d.c. functions $f(x) = \left(27x^2 + x^6 + 250\right) - 15x^4$ at $x_0 = -3.24$ (left) and $f(x) = x^3 - 6x^2$ at $x_0 = 5.24$ (right), compared to linear underestimators (blue) using the "S" method.

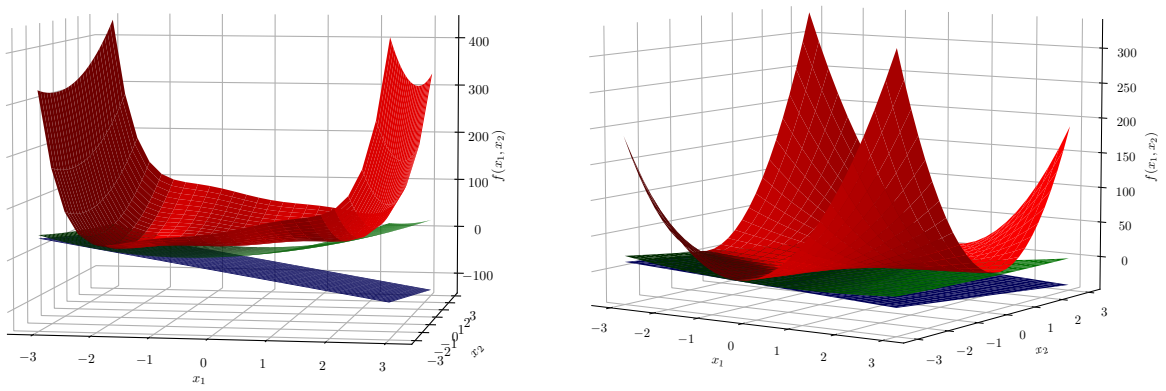

Figure 6. Quadratic underestimators (green) for the bivariate d.c. functions $f(\boldsymbol{x}) = \left(15.0x_1^2 + 9x_2^2 + x_1^6\right) - \left[3(x_1 + x_2)^2 + 6.3x_1^4\right]$ at $\boldsymbol{x}_0 = (-1.95, -1.81)$ (left) and $f(\boldsymbol{x}) = \left[5x_1^2 + 5x_2^2 + \frac{3}{2}(x_1^2 + x_2^2)^2\right] - \left[4(x_1 + x_2)^2 + \frac{3}{2}x_1^4 + \frac{3}{2}x_2^4\right]$ at $\boldsymbol{x}_0 = (-0.68, -2.21)$ (right), compared to linear underestimators (blue) using the "S" method.

Table 2. Computational results for different methods in the hierarchy, for points of construction that require the shift to admit a valid underestimator.

Dimension	# Functions	# Points of Construction	Avg. Metric				Avg. CPU (ms)			
			SS	UDS	DS	MS	SS	UDS	DS	MS
1	9	53	0.000	0.022	0.022	0.022	8	15	15	19
2	9	48	0.000	0.202	0.213	0.223	49	110	109	130
3	9	72	0.000	0.509	0.535	0.542	461	964	1202	1450
4	9	81	0.000	0.487	0.526	0.538	3162	8868	14741	16986

an average improvement in the metric of approximately 44% and 78%, respectively. However, using the matrix methods in lieu of the diagonal ones (i.e., "M" vs. "D", or "MS" vs. "DS") yields only marginal improvements in underestimator tightness. As expected, method "MS" constructs the tightest underestimators, but it also exacts the greatest computational expense. In all methods, the computational time to construct these underestimators increases sharply with function dimension.

Table 2 demonstrates the effectiveness of the shift and the capability for the methodologies to construct *quadratic* underestimators at points where, without the shift, the methods would fail. First, we clearly state that all methods with the shift successfully create underestimators for all points of construction, without exception. However, given that the first-order Taylor series approximation constructed at these points is not a valid underestimator, we ought to use an altered definition of the metric to convey tightness. More specifically, in Table 2, the reported metric compares the linear underestimator generated by method "SS" with underestimators constructed by the other methods; hence, method "SS" is reported in Table 2 as having a metric of 0, by definition. Notably, the other methods show a metric greater than 0, indicating that, on average, the final underestimators generated by these methods retain some curvature. Indeed, for the higher dimensional cases and the "MS" method, we observe that simultaneously optimizing the scaling parameters with the shift produces underestimators that reduce the volume between the function and the linear underestimators generated by "SS" by approximately 22%, 54%, and 54% for dimensions 2, 3, and 4, respectively. A slight decrease in tightness would be suffered if one utilized methods "DS" or "UDS" instead of "MS", albeit at the benefit of a more expedient computation. These results demonstrate the importance of using the shift parameter to successfully achieve

quadratic underestimation at any and all points of construction for which $\nabla^2 f(\boldsymbol{x}_0) \succcurlyeq 0$ and to improve the tightness of these underestimators.

Tables A4–A9 in the Appendix provide detailed data (per dimension) regarding the average and standard deviation of cutting-plane algorithm iterations, number of vertices enumerated, metric, CPU time, and number of LP solves exhibited by each method in this computational study. Finally, we remark that, in our extraction of d.c. functions, certain pathological cases were identified but not removed from our benchmark set. In these cases, *pathological* refers to functions for which the cutting-plane algorithm will fail to generate a successful quadratic underestimator at all points $\{\boldsymbol{x} \in \text{dom}(f) : \nabla^2 f(\boldsymbol{x}) \succcurlyeq 0\}$ without the shift. In other words, regardless of local convexity, first-order Taylor series expansions constructed at all points of these functions will overestimate them at some part of the domain. As an example, consider the function depicted in Figure 1; had this function been defined only over the smaller domain $x \in [0.35, 1.0]$, it would have been a pathological function. Another clear example of a pathological case is a function from the `sisser` optimization problem showed in Figure A1 in the Appendix.

3.2 Root Node Relaxation Comparison with State-of-the-art

In this study, we demonstrate the quality of relaxations constructed using our quadratic underestimation methodology on a set of d.c. optimization problems. In particular, we employ method "DS", which affords a good tradeoff between quality and tractability, to construct quadratic underestimators for each non-linear function in these problems, setting the tolerance of the cutting-plane algorithm to $\varepsilon = 1\text{e}{-3}$. We then solve the resulting convex QCQP relaxation using `IPOPT` v3.14.13 [Wächter and Biegler, 2006] with the linear solver `MUMPS` v5.6.2 [Amestoy et al., 2001, 2006] to determine a lower bound at the root node of each problem, which we compare with the lower bound at the root node computed by `BARON` v24.5.8 [Sahinidis, 1996]. We use all default settings of `BARON`, including the option `NoutPerVar = 4` that dictates the number of outer approximations per variable for convex multivariate functions. For a fair comparison, we mirror this by choosing to construct four quadratic underestimators per dimension at points of construction determined using Latin hypercube sampling, for each non-linear function present in the benchmark problems. Aside from this, we use the same settings for the cutting-plane algorithm, as presented in Section 3.1. We also highlight that we include all auxiliary techniques available by default in `BARON` (e.g., bounds

tightening), even though equivalent techniques are not utilized in our methodology. Despite such an unfavorable setup, however, we will demonstrate the superior quality of the lower bounds computed using our convex quadratic relaxation for problems defined in dimensions greater than one.

3.2.1 Generation of D.C. Optimization Problem Library

Noting the scarcity of benchmark d.c. optimization problems available in the literature, we systematically created optimization problems that are defined in one to four dimensions and that include linear, convex, and (non-convex) d.c. functions. Whereas the explicit problem instances used in our computational study are provided in the Appendix, in this section we outline the general procedure we employed in creating these instances.

We generate optimization problems parameterized by a tuple, (n, m_ℓ, m_c, m_{dc}), where n is the dimension of the problem, m_ℓ is the number of linear constraints, m_c is the number of convex constraints, and m_{dc} is the number of non-convex d.c. constraints. For our study, we produce instances defined using all combinations of $n \in \{1, 2, 3, 4\}$, $m_\ell = 1$ if $n > 1$ (no linear constraints added in univariate optimization problems), $m_c \in \{1, 2\}$ and $m_{dc} \in \{1, 2, 3\}$. In total, we produce 24 problems (6 for each dimension). In all cases, we restrict the domain of the variables to $[-1, 1]$.

To create a new instance, we first assign n to the desired dimension, and then select a d.c. function to serve as the objective, followed by m_ℓ, m_c, and m_{dc} linear, convex, and d.c. functions, respectively, to serve as constraints. As we add a new constraint to the problem, we ensure it is not redundant (i.e., it does reduce the feasible space) by utilizing a set of $100n$ points selected via Latin hypercube sampling to check for feasibility. More specifically, given the box domain of the variables, we determine a priori the upper and lower bounds of the range of each function, where the upper bound defines a right-hand side for the constraint to encompass the full feasible space (i.e., makes the constraint redundant), and the lower bound defines a right-hand side that eliminates the entire feasible space (i.e., causes the problem to be infeasible). Via binary search, we pick the right-hand side to use for this constraint such that 20% of the remaining sampled feasible points are eliminated. For example, after we sample 400 points for a four-dimensional problem, the first constraint added will eliminate 80 of those from the feasible space, while the next constraint will eliminate an additional 64 (20% of the remaining 320) points, resulting in 256 feasible points in the original sample. Consequently, the order in which the constraints are added to the optimization

problem is important: we add the linear constraint first, followed by the convex constraints, and add the d.c. constraints last. We highlight that adding the constraints in this order maximizes the contribution of the non-convex d.c. functions in defining the feasible space, which aligns with the goal of this work.

We generate linear constraints for use in our optimization problems by considering the linear form $\sum_{i=1}^{n} \beta_i x_i \leq \phi$, where $\beta_i = 1$ or $\beta_i = -1$ (chosen randomly) for each dimension i. As per the process explained above, we identify bounds on range of each linear expression and execute a binary search on ϕ until the feasible space (as represented by the sample points) has been reduced by 20%. In regards to generating convex functions, we do so by randomly sampling from a set of convex functional forms extracted from benchmark libraries and expanded using established rules that preserve convexity. For d.c. functions, we follow an approach similar to the one used in the hierarchy comparison study of Section 3.1.1, utilizing the core set of functions from Table A2 to construct higher-dimensional representatives. After randomly shuffling all d.c. functions available to us from this process, we use the first d.c. function in the objective, and subsequent functions in the d.c. constraints until we have added a total of m_{dc} constraints.

While we omit many specific details from our problem construction description, we explicitly include the resulting formulations in Tables A10–A13 in the Appendix for reference. We note that these tables only present floating point numbers to three decimals of precision; consequently, we supply the `.nl` files defining the problems with full decimal precision as executed in this study as Supplementary Material to this paper. We additionally provide a comprehensive list of the points of construction for each function of the study, included as comments in each file.

3.2.2 Results

Table 3 provides aggregate results for the root node relaxation comparison, where we report the number of instances for which each alternative approach–QCQP relaxation versus `BARON`–computes a superior root node lower bound as well as the average reduction of the gap between the optimal solution and the `BARON` lower bound that the QCQP relaxation affords us. These results demonstrate that the convex QCQP relaxation produces substantially superior lower bounds compared to `BARON`, reducing the root node relaxation gap by an excess of 92%, on average, for problems defined in two or more dimensions. For univariate problems, the QCQP relaxation produces tighter lower

bounds than BARON at the root node for five out of the six problem instances, while for the sixth problem, BARON reduced the gap between the QCQP bound and the optimal solution by only 11%. These results provide strong evidence of the tightness of the convex QCQP relaxation constructed by our methodology. We acknowledge that, while this comparison provides evidence for the quality of the quadratic underestimators generated by our methodology, the latter requires more computation time than BARON. However, given that we have implemented our algorithms in Python, it remains to be seen whether efficiency gains that would be achieved by utilizing a compiled language can provide for a more straightforward comparison.

Table 3. Comparison of QCQP relaxation with BARON root node.

Dimension	# Problems	BARON Root Node Better	QCQP Bound Better	Avg. Gap Reduction by QCQP
1D	6	1	5	78.8%*
2D	6	0	6	92.1%
3D	6	0	6	94.4%
4D	6	0	6	94.5%

*Reduction for the 5 problems where the QCQP provided a better bound.

4 Conclusions

In this work, we presented a hierarchy of methodologies to construct convex quadratic underestimators for non-convex d.c. functions. Focusing on d.c. functions extracted from optimization benchmark libraries, we generated quadratic underestimators that reduce, on average, the hypervolume between the function and a linear underestimator constructed at the same point of construction by 66% for one-dimensional functions and, depending on the variant of our methodology used, by a range of 38%–73% for functions of higher dimensions. Furthermore, we demonstrated in our computational study that, with variants that include a shift in the quadratic form, we could generate valid underestimators for all points of construction where $\nabla^2 f(\boldsymbol{x}) \succcurlyeq 0$, including points at which the first-order Taylor series approximation is not a valid underestimator. We showcase the tightness of our quadratic underestimators by providing qualitative results on several example func-

tions, highlighting also the improvements that can be achieved by utilizing more involved methods in our hierarchy. Finally, we show the quality of convex QCQP relaxations constructed using our quadratic underestimation methodology in a comparison with `BARON` for lower bounds computed at the root node of a set of systematically created d.c. optimization problems. Notably, our convex QCQP relaxation is able to produce superior lower bounds than those computed by `BARON` at the root node in the vast majority of cases, closing the root node relaxation gap by 90%, on average.

Future work could investigate generalizing the hierarchy even further to include variants that determine a matrix for the second-order term of the Taylor series approximation from the entire positive semi-definite cone, rather than from a restricted subset limited to matrices satisfying diagonal dominance properties. In our experience, maintaining a quadratic that is monotonic in its parameters is much more involved with a parameter modifying the first-order term of the quadratic. Thus, future work could explore ways to tailor the cutting-plane algorithm to achieve the monotonicity property in the context of modifying the first-order term, or altogether removing the monotonicity requirement from the algorithm while preserving efficiency and convergence. Finally, the opportunity now arises to incorporate our quadratic underestimators within the relaxations of established global optimization software, and future work shall investigate the bound benefits and computational tradeoffs of doing so.

Acknowledgments

We acknowledge financial support from Mitsubishi Electric Research Labs (MERL) through the Center for Advanced Process Decision-making (CAPD) at Carnegie Mellon University. William Strahl also gratefully acknowledges support from the R.R. Rothfus Graduate Fellowship in Chemical Engineering and the Chevron Graduate Fellowship in Chemical Engineering. We also acknowledge the contribution of two anonymous referees who helped us improve this manuscript through their insightful comments.

Appendix

The material supplied in this appendix provides the explicit function and problem formulations used in our computational study as well as supplemental details for some of the results. More specifically,

Tables A1–Table A3 display the precise functions used in Section 3.1.1, while Tables A4–A9 provide additional details for the computational results of the same section. Figure A1 provides a visual example of a pathological function discovered in the study for quadratic underestimator forms without the shift parameter. Finally, for the study in Section 3.2, where we compare root node relaxations of optimization problems, Tables A10–A13 provide the explicit formulations of the problems used in the study. All models are supplied in the Supplementary Material accompanying this paper in the form of `.nl` files, which also include information about the points of construction used in that study.

Table A1. Details of functions from the COCONUT library used in our computational studies, where the equation names are as in the GAMS files available at `https://arnold-neumaier.at/glopt/coconut/Benchmark/Benchmark.html`.

Dimension	Library	Problem	Name	Expression	Variable Bounds	Assigned Bounds
1	GlobalLib	ex4_1_6	objcons	$f(x_1) = \left(27x_1^2 + x_1^6 + 250\right) - 15x_1^4$	$[-5, 5]$	–
1	CUTE	zy2	objcons	$f(x_1) = x_1^3 - 6x_1^2$	$[0, 8]$	upper
1	GlobalLib	ex4_1_9	con1	$f(x_1) = 8x_1^3 - \left(8x_1^2 + 2x_1^4\right)$	$[0, 3]$	–
2	CSTP	conform1	con2,	$f(x_1, x_2) = \left[5x_1^2 + 5x_2^2 + \frac{3}{2}(x_1^2 + x_2^2)^2\right] - \left[4(x_1 + x_2)^2 + \frac{3}{2}x_1^4 + \frac{3}{2}x_2^4\right]$	$[-3, 3] \times [-3, 3]$	both
2	GlobalLib	ex8_1_4	objcons	$f(x_1, x_2) = \left[15x_1^2 + 9x_2^2 + x_1^6\right] - \left[3(x_1 + x_2)^2 + 6.3x_1^4\right]$	$[-3, 3] \times [-3, 3]$	both
2	CUTE	camel6	objcons	$f(x_1, x_2) = \left[\frac{7}{2}x_1^2 + 0.5(x_1 + x_2)^2 + 4x_2^4 + \frac{1}{3}x_1^6\right] - \left(\frac{9}{2}x_2^2 + 2.1x_1^4\right)$	$[-3, 3] \times [-1.5, 1.5]$	–
2	CUTE	sisser	objcons	$f(x_1, x_2) = \left(4x_1^2 + 4x_2^2\right) - \left(x_1^2 + x_2^2\right)^2$	$[-3, 3] \times [-3, 3]$	both
2	CSTP	cyclo	con1	$f(x_1, x_2) = \left[264.5x_1^2 + 79.5x_2^2 + 694.5(x_1 + x_2)^2 + 656.5\left(x_1^2 + x_2^2\right)^2\right] - \left(656.5x_1^4 + 656.5x_2^4\right)$	$[-10, 10] \times [-10, 10]$	–
2	GlobalLib	ex4_1_5	objcons	$f(x_1, x_2) = \left(\frac{3}{2}x_2^2 + \frac{5}{2}x_1^2 + \frac{1}{6}x_1^6\right) - \left[0.5(x_1 + x_2)^2 + 1.05x_1^4\right]$	$[-5, 5] \times [-5, 5]$	upper for x_1, lower for x_2
2	CUTE	dipigri	objcons	$f(x_1, x_2) = \left(x_2^4 + 9x_1^2 + 2x_2^2\right) - 2(x_1 + x_2)^2$	$[-3, 3] \times [-3, 3]$	both

Table A2. Core functions utilized to construct d.c. functions for our computational studies.

Functions
$f_1(x) := \left(20x^{10} + 4x^2\right) - 12x^4$
$f_2(x) := \left(4x^6 + x^4\right) - 3x^2$
$f_3(x) := \left(32x^6 + 8x^2\right) - 31x^4$
$f_4(x) := \left(4x^2 + 8x^8\right) - \left(e^{2.35x} + e^{-2.35x} - 2.35\right)$
$f_5(x) := \left(e^{4x} + e^{-4x}\right) - \left(6e^{2x} + 6e^{-2x} - 8.6\right)$
$f_6(x) := \left(20x^{10} + 36x^6 + 8x^2\right) - 38x^4$
$\mathcal{L}_i(\boldsymbol{x}) := \frac{1}{2p}(\sum_{j=1}^{p} x_j)^4$

Table A3. Additional d.c. functions utilized in our computational studies. Functions $f_{(\cdot)}$ and $\mathcal{L}_{(\cdot)}$ are as defined in Table A2. In all cases, the variables are bounded in $[-1, 1]$.

Dimension	Function
1	$f_1(x)$
1	$f_2(x)$
1	$f_3(x)$
1	$f_4(x)$
1	$f_5(x)$
1	$f_6(x)$
2	$f_1(x_1) + f_2(x_2) + \mathcal{L}_2(x_1, x_2)$
2	$f_2(x_1) + f_4(x_2) + \mathcal{L}_2(x_1, x_2)$
3	$f_1(x_1) + f_1(x_2) + f_3(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_1(x_1) + f_1(x_2) + f_5(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_1(x_1) + f_2(x_2) + f_5(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_1(x_1) + f_2(x_2) + f_6(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_1(x_1) + f_6(x_2) + f_6(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_2(x_1) + f_4(x_2) + f_5(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_3(x_1) + f_4(x_2) + f_4(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_3(x_1) + f_6(x_2) + f_6(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
3	$f_5(x_1) + f_5(x_2) + f_5(x_3) + \mathcal{L}_3(x_1, x_2, x_3)$
4	$f_1(x_1) + f_1(x_2) + f_1(x_3) + f_3(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_1(x_1) + f_1(x_2) + f_1(x_3) + f_4(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_1(x_1) + f_1(x_2) + f_2(x_3) + f_2(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_1(x_1) + f_3(x_2) + f_5(x_3) + f_5(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_1(x_1) + f_4(x_2) + f_5(x_3) + f_5(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_2(x_1) + f_2(x_2) + f_2(x_3) + f_2(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_2(x_1) + f_2(x_2) + f_2(x_3) + f_6(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_2(x_1) + f_2(x_2) + f_4(x_3) + f_5(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$
4	$f_3(x_1) + f_4(x_2) + f_4(x_3) + f_4(x_4) + \mathcal{L}_4(x_1, x_2, x_3, x_4)$

Table A4. Computational details (averages ± standard deviations) for producing underestimators of 2D functions using the 42 points of construction (in total, across all functions) that did not require shift for successful construction.

Method	Iterations	Vertices	Metric	CPU (ms)	LP Solves
S	45.8 ± 34.1	214.0 ± 161.9	0.435 ± 0.298	120 ± 127	0.0 ± 0.0
D	59.8 ± 38.6	280.3 ± 183.3	0.502 ± 0.292	157 ± 119	5.0 ± 2.8
M	64.0 ± 41.5	300.4 ± 197.6	0.510 ± 0.291	190 ± 121	5.3 ± 3.2
SS	45.8 ± 34.1	214.0 ± 161.9	0.435 ± 0.298	89 ± 91	0.0 ± 0.0
UDS	49.4 ± 36.5	230.9 ± 173.4	0.480 ± 0.294	133 ± 100	4.5 ± 2.2
DS	62.1 ± 41.6	290.8 ± 197.6	0.523 ± 0.288	170 ± 130	5.6 ± 2.8
MS	65.7 ± 42.8	308.1 ± 204.0	0.524 ± 0.290	191 ± 129	5.7 ± 3.2

Table A5. Computational details (averages ± standard deviations) for producing underestimators of 2D functions using the 48 points of construction (in total, across all functions) that required the shift for successful construction.

Method	Iterations	Vertices	Metric	CPU (ms)	LP Solves
SS	28.2 ± 20.7	131.4 ± 97.7	0.000 ± 0.000	49 ± 46	0.0 ± 0.0
UDS	39.9 ± 30.5	186.5 ± 148.7	0.202 ± 0.257	110 ± 84	4.2 ± 2.5
DS	40.1 ± 30.7	187.2 ± 149.1	0.213 ± 0.256	109 ± 83	4.5 ± 2.7
MS	41.5 ± 31.2	193.3 ± 151.8	0.223 ± 0.257	130 ± 81	4.7 ± 3.0

Table A6. Computational details (averages ± standard deviations) for producing underestimators of 3D functions using the 18 points of construction (in total, across all functions) that did not require shift for successful construction.

Method	Iterations	Vertices	Metric	CPU (ms)	LP Solves
S	93.6 ± 33.6	1173.4 ± 551.2	0.503 ± 0.209	710 ± 467	0.0 ± 0.0
D	155.8 ± 68.0	2033.0 ± 1184.1	0.725 ± 0.028	1350 ± 1091	15.3 ± 5.3
M	162.6 ± 60.3	2135.8 ± 1045.8	0.726 ± 0.032	1600 ± 904	18.6 ± 6.0
SS	93.6 ± 33.6	1173.4 ± 551.2	0.503 ± 0.209	527 ± 379	0.0 ± 0.0
UDS	100.7 ± 43.2	1237.3 ± 683.8	0.673 ± 0.073	736 ± 515	12.3 ± 3.4
DS	155.0 ± 68.5	2020.9 ± 1191.4	0.725 ± 0.028	1439 ± 1208	15.9 ± 5.0
MS	162.3 ± 61.2	2126.7 ± 1057.3	0.727 ± 0.028	1660 ± 918	19.6 ± 5.4

Table A7. Computational details (averages ± standard deviations) for producing underestimators of 3D functions using the 72 points of construction (in total, across all functions) that required the shift for successful construction.

Method	Iterations	Vertices	Metric	CPU (ms)	LP Solves
SS	83.2 ± 42.9	1161.9 ± 750.1	0.000 ± 0.000	461 ± 519	0.0 ± 0.0
UDS	124.1 ± 52.1	1768.7 ± 913.9	0.509 ± 0.216	964 ± 652	11.3 ± 4.4
DS	144.3 ± 55.9	2081.4 ± 987.0	0.535 ± 0.215	1202 ± 801	13.2 ± 4.4
MS	152.4 ± 63.3	2217.9 ± 1119.2	0.542 ± 0.209	1450 ± 991	14.7 ± 5.4

Table A8. Computational details (averages ± standard deviations) for producing underestimators of 4D functions using the 9 points of construction (in total, across all functions) that did not require shift for successful construction.

Method	Iterations	Vertices	Metric	CPU (ms)	LP Solves
S	161.9 ± 63.8	7127.4 ± 3899.3	0.383 ± 0.188	4180 ± 3792	0.0 ± 0.0
D	303.8 ± 91.0	14299.9 ± 5620.7	0.681 ± 0.062	10598 ± 6448	23.4 ± 9.6
M	293.0 ± 98.3	14011.1 ± 6206.6	0.682 ± 0.064	11173 ± 6981	25.6 ± 4.1
SS	161.9 ± 63.8	7127.4 ± 3899.3	0.383 ± 0.188	3397 ± 2795	0.0 ± 0.0
UDS	151.9 ± 44.2	6286.6 ± 2177.7	0.605 ± 0.068	3201 ± 1292	16.4 ± 4.1
DS	283.2 ± 71.0	12959.4 ± 3804.9	0.683 ± 0.059	9277 ± 3784	22.8 ± 7.7
MS	273.2 ± 82.0	12491.9 ± 4866.7	0.695 ± 0.045	9743 ± 5808	25.0 ± 4.5

Table A9. Computational details (averages ± standard deviations) for producing underestimators of 4D functions using the 81 points of construction (in total, across all functions) that required the shift for successful construction.

Method	Iterations	Vertices	Metric	CPU (ms)	LP Solves
SS	140.4 ± 77.1	7282.8 ± 5581.0	0.000 ± 0.000	3162 ± 3531	0.0 ± 0.0
UDS	227.9 ± 149.2	12404.1 ± 11227.1	0.487 ± 0.193	8868 ± 11509	16.6 ± 5.4
DS	301.2 ± 183.0	17118.2 ± 14460.3	0.526 ± 0.188	14741 ± 20844	20.4 ± 6.2
MS	317.6 ± 188.9	18212.8 ± 14949.4	0.538 ± 0.181	16986 ± 24619	23.3 ± 7.2

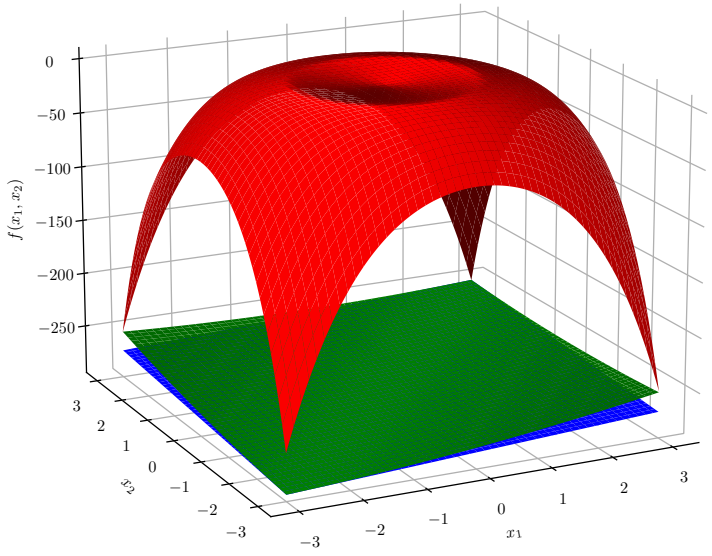

Figure A1. The pathological d.c. function from problem *sisser*, where no locally convex point of construction admits a valid quadratic underestimator unless a method with shift is employed. The green surface depicts the quadratic underestimator generated using the shift-utilizing "DS" method at a central point, $\boldsymbol{x}_0 = [0.592, 0.555]$, whereas its linear relaxation counterpart (i.e., after dropping the second-order term) is shown in blue.

Table A10. 1D d.c. optimization problems utilized in root node study.

Problem #	Definition	
01	$\min_{\mathbf{x}\in[-1,1]}$	$f_1(x_1)$
	s.t.	$0.053\left(0.909x_1^{10}+1\right)^3 e^{1.0x_1^{10}} \leq 0.075$
		$f_2(x_1) \leq -0.072$
02	$\min_{\mathbf{x}\in[-1,1]}$	$f_6(x_1)$
	s.t.	$\frac{0.004}{\left(1-\frac{0.091}{x_1+1.1}\right)^{2.333}} \leq 0.006$
		$f_2(x_1) \leq -0.031$
		$f_5(x_1) \leq 0.466$
03	$\min_{\mathbf{x}\in[-1,1]}$	$f_4(x_1)$
	s.t.	$-1.103\left(1-0.334e^{-x_1}\right)^{0.75} \leq -0.540$
		$f_1(x_1) \leq 0.320$
		$f_6(x_1) \leq 0.381$
		$f_5(x_1) \leq 0.579$
04	$\min_{\mathbf{x}\in[-1,1]}$	$f_4(x_1)$
	s.t.	$\frac{0.053\left(-\frac{0.909\log(1.1-x_1)}{\log(10)}+1\right)^3}{(1.1-x_1)^{\frac{1.0}{\log(10)}}} \leq 0.152$
		$-1.0\left(1-0.909x_1^{10}\right)^{0.1} \leq -0.998$
		$f_6(x_1) \leq 0.422$
05	$\min_{\mathbf{x}\in[-1,1]}$	$f_6(x_1)$
	s.t.	$\frac{0.202}{\left(-0.048(0.909x_1+1)^3e^{x_1}+1\right)^{0.667}} \leq 0.263$
		$-0.616\left(0.852\left(0.909x_1+1\right)^{0.1}+1\right)^{0.75} \leq -0.946$
		$f_2(x_1) \leq -0.059$
		$f_5(x_1) \leq 0.271$
06	$\min_{\mathbf{x}\in[-1,1]}$	$f_5(x_1)$
	s.t.	$\frac{0.1}{-\frac{0.074e^{-x_1}}{(0.909x_1+1)^{0.667}}+1.1} \leq 0.115$
		$\frac{0.202}{(0.909x_1+1)^{0.667}} \leq 0.247$
		$f_2(x_1) \leq -0.018$
		$f_4(x_1) \leq 0.305$
		$f_3(x_1) \leq 0.616$

Table A11. 2D d.c. optimization problems utilized in root node study.

Problem #	Definition	
07	$\min_{\mathbf{x}\in[-1,1]^2}$	$f_2(x_1)+f_5(x_2)+\mathcal{L}_2(x_1,x_2)$
	s.t.	$-x_1+x_2\leq 0.797$
		$\frac{1.887e^{-0.469(0.909x_1+1)^{0.1}-0.308(0.909x_2+1)^{0.75}}}{\left(0.426(0.909x_1+1)^{0.1}+0.28(0.909x_2+1)^{0.75}+1\right)^{0.667}}\leq 0.760$
		$f_4(x_1)+f_6(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.730$
08	$\min_{\mathbf{x}\in[-1,1]^2}$	$f_1(x_1)+f_5(x_2)+\mathcal{L}_2(x_1,x_2)$
	s.t.	$-x_1-x_2\leq 0.688$
		$\frac{0.053\left(\frac{0.034e^{-x_2}}{(0.909x_2+1)^{0.667}}-\frac{0.455\log(1.1-x_1)}{\log(10)}+1\right)^3 e^{\frac{0.037e^{-x_2}}{(0.909x_2+1)^{0.667}}}}{(1.1-x_1)^{\frac{0.5}{\log(10)}}}\leq 0.132$
		$f_3(x_1)+f_4(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.963$
		$f_2(x_1)+f_6(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.219$
09	$\min_{\mathbf{x}\in[-1,1]^2}$	$f_1(x_1)+f_4(x_2)+\mathcal{L}_2(x_1,x_2)$
	s.t.	$x_1-x_2\leq 0.727$
		$0.144\left(0.024\left(0.909x_2+1\right)^3e^{x_2}+1+\frac{0.045}{x_1+1.1}\right)^3\leq 0.307$
		$f_2(x_1)+f_3(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.580$
		$f_1(x_1)+f_3(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.888$
		$f_2(x_1)+f_4(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.052$
10	$\min_{\mathbf{x}\in[-1,1]^2}$	$f_4(x_1)+f_5(x_2)+\mathcal{L}_2(x_1,x_2)$
	s.t.	$-x_1+x_2\leq 0.741$
		$\frac{0.45e^{-0.308(0.909x_1+1)^{0.75}+0.184e^{-x_2}}}{\left(0.28(0.909x_1+1)^{0.75}+1-0.167e^{-x_2}\right)^{0.667}}\leq 0.515$
		$\frac{0.074e^{\frac{0.101}{(0.909x_1+1)^{0.667}}+0.072(0.909x_2+1)^3}}{\left(-\frac{0.092}{(0.909x_1+1)^{0.667}}-0.065(0.909x_2+1)^3+1\right)^{0.667}}\leq 0.133$
		$f_2(x_1)+f_3(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.559$
11	$\min_{\mathbf{x}\in[-1,1]^2}$	$f_2(x_1)+f_4(x_2)+\mathcal{L}_2(x_1,x_2)$
	s.t.	$x_1-x_2\leq 0.719$
		$0.053\left(0.065\left(0.909x_1+1\right)^3+0.167e^{x_2}+1\right)^3e^{0.072(0.909x_1+1)^3+0.184e^{x_2}}\leq 0.275$
		$-0.967\left(\frac{0.455(-1.478x_1-1.626)\log(x_1+1.1)}{\log(10)}-0.167e^{x_2}+1\right)^{0.75}\leq -0.718$
		$f_1(x_1)+f_2(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.364$
		$f_5(x_1)+f_5(x_2)+\mathcal{L}_2(x_1,x_2)\leq 2.221$
12	$\min_{\mathbf{x}\in[-1,1]^2}$	$f_4(x_1)+f_5(x_2)+\mathcal{L}_2(x_1,x_2)$
	s.t.	$x_1-x_2\leq 0.810$
		$0.144\left(0.455x_1+0.455x_2+1\right)^3\leq 0.363$
		$-1.017\left(-\frac{0.002}{(0.909x_1+1)^{2.333}}+1-\frac{0.045}{x_2+1.1}\right)^{0.75}\leq -0.942$
		$f_1(x_1)+f_2(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.234$
		$f_2(x_1)+f_3(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.474$
		$f_1(x_1)+f_3(x_2)+\mathcal{L}_2(x_1,x_2)\leq 0.768$

Table A12. 3D d.c. optimization problems utilized in root node study.

Problem #	Definition	
13	$\min_{\mathbf{x}\in[-1,1]^3}$	$f_2(x_1)+f_4(x_2)+f_5(x_3)+\mathcal{L}_3(x_1,x_2,x_3)$
	s.t.	$x_1-x_2+x_3\leq 0.938$
		$\frac{0.679}{-0.333x_3^4+0.312(0.909x_1+1)^{0.1}-0.048(0.909x_2+1)^3+1.1}\leq 0.605$
		$f_4(x_1)+f_4(x_2)+f_4(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 0.750$
14	$\min_{\mathbf{x}\in[-1,1]^3}$	$f_1(x_1)+f_2(x_2)+f_6(x_3)+\mathcal{L}_3(x_1,x_2,x_3)$
	s.t.	$x_1+x_2+x_3\leq 0.891$
		$\frac{0.074e^{\frac{0.067}{(0.909x_1+1)^{0.667}}+0.123e^{x_3}}}{(1.1-x_2)^{\frac{0.333}{\log(10)}}\left(-\frac{0.061}{(0.909x_1+1)^{0.667}}-0.111e^{x_3}+\frac{0.303\log(1.1-x_2)}{\log(10)}+1\right)^{0.667}}\leq 0.130$
		$f_1(x_1)+f_4(x_2)+f_4(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 1.277$
		$f_4(x_1)+f_5(x_2)+f_6(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 2.414$
15	$\min_{\mathbf{x}\in[-1,1]^3}$	$f_1(x_1)+f_1(x_2)+f_6(x_3)+\mathcal{L}_3(x_1,x_2,x_3)$
	s.t.	$x_1-x_2+x_3\leq 0.873$
		$\frac{0.004}{\left(-0.044(0.909x_1+1)^3-\frac{0.023e^{-x_3}}{(0.909x_3+1)^{0.667}}+\frac{0.303\log(1.1-x_2)}{\log(10)}+1\right)^{2.333}}\leq 0.008$
		$f_2(x_1)+f_3(x_2)+f_4(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 1.158$
		$f_4(x_1)+f_4(x_2)+f_4(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 0.604$
		$f_3(x_1)+f_4(x_2)+f_4(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 0.887$
16	$\min_{\mathbf{x}\in[-1,1]^3}$	$f_1(x_1)+f_3(x_2)+f_4(x_3)+\mathcal{L}_3(x_1,x_2,x_3)$
	s.t.	$-x_1-x_2+x_3\leq 0.879$
		$-0.975\left(0.187\left(0.909x_1+1\right)^{0.75}-0.044\left(0.909x_2+1\right)^3+1-\frac{0.03}{x_3+1.1}\right)^{0.1}\leq -0.969$
		$0.053\left(0.044\left(0.909x_1+1\right)^3+\frac{0.023e^{-x_3}}{(0.909x_3+1)^{0.667}}+1+\frac{0.03}{x_2+1.1}\right)^3 e^{0.048(0.909x_1+1)^3+\frac{0.025e^{-x_3}}{(0.909x_3+1)^{0.667}}+\frac{0.033}{x_2+1.1}}\leq 0.158$
		$f_5(x_1)+f_5(x_2)+f_5(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 4.602$
17	$\min_{\mathbf{x}\in[-1,1]^3}$	$f_1(x_1)+f_1(x_2)+f_3(x_3)+\mathcal{L}_3(x_1,x_2,x_3)$
	s.t.	$-x_1-x_2-x_3\leq 0.867$
		$0.053\left(\frac{0.061}{(0.909x_1+1)^{0.667}}+1+0.111e^{-x_2}+\frac{0.03}{x_3+1.1}\right)^3 e^{\frac{0.067}{(0.909x_1+1)^{0.667}}+0.123e^{-x_2}+\frac{0.033}{x_3+1.1}}\leq 0.160$
		$0.867\left(-0.284\left(0.909x_1+1\right)^{0.1}-0.187\left(0.909x_2+1\right)^{0.75}+0.111e^{x_3}+1\right)^3\leq 0.360$
		$f_1(x_1)+f_6(x_2)+f_6(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 3.062$
		$f_1(x_1)+f_2(x_2)+f_5(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 0.837$
18	$\min_{\mathbf{x}\in[-1,1]^3}$	$f_2(x_1)+f_2(x_2)+f_3(x_3)+\mathcal{L}_3(x_1,x_2,x_3)$
	s.t.	$-x_1-x_2+x_3\leq 0.855$
		$0.531e^{-0.205(0.909x_1+1)^{0.75}+\frac{0.001}{(0.909x_2+1)^{2.333}}+\frac{0.033}{x_3+1.1}}\leq 0.501$
		$\frac{0.917e^{-0.312(0.909x_1+1)^{0.1}-0.205(0.909x_2+1)^{0.75}-\frac{0.333(-1.478x_3-1.626)\log(x_3+1.1)}{\log(10)}}}{\left(0.284(0.909x_1+1)^{0.1}+0.187(0.909x_2+1)^{0.75}+\frac{0.303(-1.478x_3-1.626)\log(x_3+1.1)}{\log(10)}+1\right)^{0.667}}\leq 0.511$
		$f_3(x_1)+f_5(x_2)+f_6(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 5.649$
		$f_5(x_1)+f_5(x_2)+f_5(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 3.941$
		$f_3(x_1)+f_6(x_2)+f_6(x_3)+\mathcal{L}_3(x_1,x_2,x_3)\leq 1.192$

Table A13. 4D d.c. optimization problems utilized in root node study.

Problem #	Definition
19	$\min_{\mathbf{x}\in[-1,1]^4} \; f_1(x_1)+f_1(x_2)+f_1(x_3)+f_4(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)$ s.t. $x_1-x_2+x_3-x_4\leq 1.016$ $-\frac{1.0\log\left(-\frac{0.001}{(0.909x_1+1)^{2.333}}-0.013(0.909x_4+1)^3e^{x_4}+1.1-0.092e^{-x_2}-\frac{0.025}{x_3+1.1}\right)}{\log(10)}\leq 0.089$ $f_2(x_1)+f_2(x_2)+f_3(x_3)+f_5(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 3.270$
20	$\min_{\mathbf{x}\in[-1,1]^4} \; f_1(x_1)+f_5(x_2)+f_6(x_3)+f_6(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)$ s.t. $x_1-x_2+x_3-x_4\leq 0.906$ $-0.976\left(0.14\left(0.909x_1+1\right)^{0.75}+\frac{0.227(-1.478x_2-1.626)\log(x_2+1.1)}{\log(10)}-\frac{0.001}{(0.909x_3+1)^{2.333}}-\frac{0.017e^{-x_4}}{(0.909x_4+1)^{0.667}}+1\right)^{0.1}\leq -0.969$ $f_1(x_1)+f_1(x_2)+f_2(x_3)+f_2(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 1.404$ $f_2(x_1)+f_2(x_2)+f_2(x_3)+f_3(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 0.157$
21	$\min_{\mathbf{x}\in[-1,1]^4} \; f_1(x_1)+f_4(x_2)+f_5(x_3)+f_5(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)$ s.t. $x_1+x_2-x_3+x_4\leq 0.988$ $0.053\left(0.033\left(0.909x_1+1\right)^3+\frac{0.227\cdot(1.478x_2+1.626)\log(x_2+1.1)}{\log(10)}+1+0.084e^{-x_3}+\frac{0.023}{x_4+1.1}\right)^3 e^{0.036(0.909x_1+1)^3+\frac{0.25\cdot(1.478x_2+1.626)\log(x_2+1.1)}{\log(10)}+0.092e^{-x_3}+\frac{0.025}{x_4+1.1}}\leq 0.151$ $f_3(x_1)+f_4(x_2)+f_4(x_3)+f_4(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 2.414$ $f_2(x_1)+f_3(x_2)+f_3(x_3)+f_6(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 2.160$ $f_2(x_1)+f_2(x_2)+f_2(x_3)+f_5(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq -0.047$
22	$\min_{\mathbf{x}\in[-1,1]^4} \; f_1(x_1)+f_3(x_2)+f_3(x_3)+f_3(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)$ s.t. $-x_1+x_2-x_3-x_4\leq 0.938$ $-0.969\left(-0.227x_3^4-\frac{0.046}{(0.909x_1+1)^{0.667}}-0.033\left(0.909x_2+1\right)^3+\frac{0.227\log(1.1-x_4)}{\log(10)}+1\right)^{0.75}\leq -0.770$ $0.219\left(0.227x_3^{10}-0.14\left(0.909x_1+1\right)^{0.75}+0.012\left(0.909x_4+1\right)^3e^{x_4}+1+\frac{0.023}{x_2+1.1}\right)^3\leq 0.228$ $f_2(x_1)+f_2(x_2)+f_2(x_3)+f_6(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 0.035$
23	$\min_{\mathbf{x}\in[-1,1]^4} \; f_2(x_1)+f_2(x_2)+f_4(x_3)+f_5(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)$ s.t. $-x_1-x_2-x_3+x_4\leq 1.062$ $\frac{0.488}{\left(-0.227x_2^4-0.227x_4^{10}+0.14(0.909x_1+1)^{0.75}+1-0.084e^{-x_3}\right)^{0.667}}\leq 0.520$ $\frac{0.074e^{\frac{0.051}{(0.909x_1+1)^{0.667}}+0.013(0.909x_4+1)^3e^{x_4}+0.092e^{x_2}+0.092e^{-x_3}}}{\left(-\frac{0.046}{(0.909x_1+1)^{0.667}}-0.012(0.909x_4+1)^3e^{x_4}-0.084e^{x_2}+1-0.084e^{-x_3}\right)^{0.667}}\leq 0.140$ $f_1(x_1)+f_4(x_2)+f_5(x_3)+f_5(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 8.879$ $f_3(x_1)+f_3(x_2)+f_3(x_3)+f_4(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 1.999$
24	$\min_{\mathbf{x}\in[-1,1]^4} \; f_1(x_1)+f_3(x_2)+f_5(x_3)+f_5(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)$ s.t. $x_1+x_2+x_3-x_4\leq 1.016$ $-\frac{1.0\log\left(-0.25x_4^{10}-\frac{0.051}{(0.909x_1+1)^{0.667}}-0.036(0.909x_2+1)^3+1.1-\frac{0.025}{x_3+1.1}\right)}{\log(10)}\leq 0.081$ $-0.998\left(-0.227x_4^{10}+\frac{0.227(-1.478x_1-1.626)\log(x_1+1.1)}{\log(10)}-\frac{0.001}{(0.909x_2+1)^{2.333}}+1-0.084e^{-x_3}\right)^{0.1}\leq -0.970$ $f_1(x_1)+f_1(x_2)+f_1(x_3)+f_3(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 3.115$ $f_1(x_1)+f_3(x_2)+f_5(x_3)+f_6(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 2.389$ $f_3(x_1)+f_6(x_2)+f_6(x_3)+f_6(x_4)+\mathcal{L}_4(x_1,x_2,x_3,x_4)\leq 1.444$

References

Claire S Adjiman, Ioannis P Androulakis, and Christodoulos A Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs—II. Implementation and computational results. *Computers & chemical engineering*, 22(9):1159–1179, 1998a.

Claire S Adjiman, Stefan Dallwig, Christodoulos A Floudas, and Arnold Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998b.

IG Akrotirianakis, CA Meyer, and CA Floudas. The role of the off-diagonal elements of the hessian matrix in the construction of tight convex underestimators for nonconvex functions. *Discovery Through Product and Process Design*, pages 501–504, 2004.

Alberth Alvarado, Gesualdo Scutari, and Jong-Shi Pang. A new decomposition method for multiuser dc-programming and its applications. *IEEE Transactions on Signal Processing*, 62(11): 2984–2998, 2014.

P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23 (1):15–41, 2001.

P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995.

Mohammad Askarizadeh, Alireza Morsali, Mostafa Zangiabadi, and Kim Khoa Nguyen. Difference convex (DC) programming approach as an alternative optimizer for neural networks. In *ICC 2023-IEEE International Conference on Communications*, pages 5179–5184. IEEE, 2023.

Pranjal Awasthi, Anqi Mao, Mehryar Mohri, and Yutao Zhong. DC-programming for neural network optimizations. *Journal of Global Optimization*, pages 1–17, 2024.

Adil M Bagirov and Julien Ugon. Nonsmooth DC programming approach to clusterwise linear regression: optimality conditions and algorithms. *Optimization methods and software*, 33(1): 194–219, 2018.

Adil M Bagirov, Sona Taheri, and Julien Ugon. Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. *Pattern Recognition*, 53:12–24, 2016.

Dimitri Bertsekas, Angelia Nedic, and Asuman Ozdaglar. *Convex analysis and optimization*, volume 1. Athena Scientific, 2003.

Dimitris Bertsimas, Danique de Moor, Dick den Hertog, Thodoris Koukouvinos, and Jianzhe Zhen. A novel algorithm for a broad class of nonconvex optimization problems. *Optimization Online*, 2023.

Fani Boukouvala, Ruth Misener, and Christodoulos A Floudas. Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. *European Journal of Operational Research*, 252(3):701–727, 2016.

Christoph Buchheim and Long Trieu. Quadratic outer approximation for convex integer programming with box constraints. In *Experimental Algorithms: 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings 12*, pages 224–235. Springer, 2013.

Pey-Chun Chen, Pierre Hansen, Brigitte Jaumard, and Hoang Tuy. Solution of the multisource Weber and conditional weber problems by d.-c. programming. *Operations Research*, 46(4):548–562, 1998.

Alex Durkin, Lennart Otte, and Miao Guo. Surrogate-based optimisation of process systems to recover resources from wastewater. *Computers & Chemical Engineering*, page 108584, 2024.

Gilles Gasso, Alain Rakotomamonjy, and Stéphane Canu. Recovering sparse signals with a certain family of nonconvex penalties and DC programming. *IEEE Transactions on Signal Processing*, 57(12):4686–4698, 2009.

Chrysanthos E Gounaris and Christodoulos A Floudas. Tight convex underestimators for-continuous problems: I. Univariate functions. *Journal of Global Optimization*, 42(1):51–67, 2008a.

Chrysanthos E Gounaris and Christodoulos A Floudas. Tight convex underestimators for-continuous problems: II. Multivariate functions. *Journal of Global Optimization*, 42(1):69–89, 2008b.

Philip Hartman. On functions representable as a difference of convex functions. 1959.

Karla Leigh Hoffman. A method for globally minimizing concave functions over convex sets. *mathematical Programming*, 20:22–32, 1981.

Kaj Holmberg and Hoang Tuy. A production-transportation problem with stochastic demand and concave production costs. *Mathematical programming*, 85:157–179, 1999.

Reiner Horst and Panos M Pardalos. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013.

Qi Huangfu and JA Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

Minsu Kim, Sunghyun Cho, Kyojin Jang, Seokyoung Hong, Jonggeol Na, and Il Moon. Data-driven robust optimization for minimum nitrogen oxide emission under process uncertainty. *Chemical Engineering Journal*, 428:130971, 2022.

Hoai An Le Thi and Manh Cuong Nguyen. DCA based algorithms for feature selection in multi-class support vector machine. *Annals of Operations Research*, 249(1-2):273–300, 2017.

Yifei Lou, Tieyong Zeng, Stanley Osher, and Jack Xin. A weighted difference of anisotropic and isotropic total variation model for image processing. *SIAM Journal on Imaging Sciences*, 8(3): 1798–1823, 2015.

Yen-An Lu, Conor M O' Brien, Douglas G Mashek, Wei-Shou Hu, and Qi Zhang. Kinetic-model-based pathway optimization with application to reverse glycolysis in mammalian cells. *Biotechnology and bioengineering*, 120(1):216–229, 2023.

Costas D Maranas and Christodoulos A Floudas. Finding all solutions of nonlinearly constrained systems of equations. *Journal of Global Optimization*, 7:143–182, 1995.

Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part I—convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

D Melzer. On the expressibility of piecewise-linear continuous functions as the difference of two piecewise-linear convex functions. *Quasidifferential Calculus*, pages 118–134, 1986.

Clifford A Meyer and Christodoulos A Floudas. Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: Spline αBB underestimators. *Journal of Global Optimization*, 32:221–258, 2005.

Alexander Mitsos, Benoit Chachuat, and Paul I Barton. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009.

Hans Mittelmann. Benchmarks for optimization software, 2023. URL `https://plato.asu.edu/bench.html`. Accessed on December 23, 2023.

Phuong Anh Nguyen and Hoai An Le Thi. DCA approaches for simultaneous wireless information power transfer in MISO secrecy channel. *Optimization and Engineering*, 24(1):5–29, 2023.

Alireza Olama, Eduardo Camponogara, and Paulo RC Mendes. Distributed primal outer approximation algorithm for sparse convex programming with separable structures. *Journal of Global Optimization*, 86(3):637–670, 2023.

Mohand Ouanes, Hoai An Le Thi, Trong Phuc Nguyen, and Ahmed Zidna. New quadratic lower bound for multivariate functions in global optimization. *Mathematics and Computers in Simulation*, 109:197–211, 2015.

Diclehan Tezcaner Öztürk and Murat Köksalan. Biobjective route planning of an unmanned air vehicle in continuous space. *Transportation Research Part B: Methodological*, 168:151–169, 2023.

H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19:551–566, 1995.

H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. 8:107–139, 1996.

Nikolaos V Sahinidis. BARON: A general purpose global optimization software package. *Journal of global optimization*, 8:201–205, 1996.

Oleg Shcherbina, Arnold Neumaier, Djamila Sam-Haroud, Xuan-Ha Vu, and Tuan-Viet Nguyen. Benchmarking global optimization and constraint satisfaction codes. In *Global Optimization and Constraint Satisfaction: First International Workshop on Global Constraint Optimization and Constraint Satisfaction, COCOS 2002, Valbonne-Sophia Antipolis, France, October 2002. Revised Selected Papers 1*, pages 211–222. Springer, 2003.

Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.

Anders Skjäl, Tapio Westerlund, Ruth Misener, and Christodoulos A Floudas. A generalization of the classical αBB convex underestimation via diagonal and nondiagonal quadratic terms. *Journal of Optimization Theory and Applications*, 154:462–490, 2012.

William R. Strahl, Arvind Raghunathan, Nikolaos V. Sahinidis, and Chrysanthos E. Gounaris. Constructing tight quadratic relaxations for global optimization: I. Outer-approximating twice-differentiable convex functions. *Forthcoming*, 2024.

Matthew Streeter and Joshua V Dillon. Automatically bounding the Taylor remainder series: Tighter bounds and new applications. *arXiv preprint arXiv:2212.11429*, 2022.

Matthew Streeter and Joshua V Dillon. Sharp taylor polynomial enclosures in one dimension. *arXiv preprint arXiv:2308.00679*, 2023.

Lijie Su, Lixin Tang, David E Bernal, and Ignacio E Grossmann. Improved quadratic cuts for convex mixed-integer nonlinear programs. *Computers & Chemical Engineering*, 109:77–95, 2018.

Angelos Tsoukalas and Alexander Mitsos. Multivariate mcCormick relaxations. *Journal of Global Optimization*, 59(2-3):633–662, 2014.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3):261–272, 2020.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

Yu Yang, Brenda De La Torre, Katherine Stewart, Laurianne Lair, Nguyen L Phan, Rupak Das, Demar Gonzalez, and Roger C Lo. The scheduling of alkaline water electrolysis for hydrogen production using hybrid energy sources. *Energy Conversion and Management*, 257:115408, 2022.

Manali S Zantye, Akhilesh Gandhi, Mengdi Li, Akhil Arora, Pavitra Senthamilselvan Sengalani, Yifan Wang, Sai Pushpitha Vudata, Debangsu Bhattacharyya, and MM Faruque Hasan. THESEUS: A techno-economic design, integration and downselection framework for energy storage. *Energy Conversion and Management*, 284:116976, 2023.

Wenjin Zhou, Linlin Liu, Yafeng Xing, and Jian Du. A novel two-step approach for multi-plant indirect HENs design. *Chemical Engineering Science*, 285:119559, 2024.