# A Method for Convex Black-Box Integer Global Optimization

Jeffrey Larson* Sven Leyffer* Prashant Palkar† Stefan M. Wild*

March 26, 2019

**Abstract**

We study the problem of minimizing a convex function on the integer lattice when the function cannot be evaluated at noninteger points. We propose a new underestimator that does not require access to (sub)gradients of the objective but, rather, uses secant linear functions that interpolate the objective function at previously evaluated points. These linear mappings are shown to underestimate the objective in disconnected portions of the domain. Therefore, the union of these conditional cuts provides a nonconvex underestimator of the objective. We propose an algorithm that alternates between updating the underestimator and evaluating the objective function. We prove that the algorithm converges to a global minimum of the objective function on the integer lattice. We present two approaches for representing the underestimator and compare their computational effectiveness. We also compare implementations of our algorithm with existing methods for minimizing functions on the integer lattice. We discuss the noticeable difficulty of this problem class and provide insights into why a computational proof of optimality is challenging even for moderate problem sizes.

## 1 Introduction

We study the problem of minimizing a convex function on the integer lattice. In particular, we consider problems of the form

$$\underset{x}{\text{minimize}} \;\; f(x) \quad \text{subject to:} \;\; x \in \Omega \subset \mathbb{Z}^n, \tag{1}$$

under the following assumption.

**Assumption 1.** *$\Omega$ is bounded, $f$ is convex on $\Omega$, and $f$ cannot be evaluated at $x \notin \Omega \subset \mathbb{Z}^n$.*

Because we assume that $f$ cannot be evaluated at noninteger points, problem (1) can be referred to as a convex optimization problem with *unrelaxable integer constraints* [29]. We are especially interested in problems where the cost of evaluating $f$ is large.

We note that $f$ need be convex only on the finite set $\Omega$ and that $\Omega$ need not contain all integer points in its convex hull (i.e., our approach allows for situations where $\operatorname{conv}(\Omega) \cap \mathbb{Z}^n \neq \Omega$). Our assumption is a weaker assumption than *integer convexity* [24, Definition 15.2] and is equivalent to assuming that there exists a convex function with the same value as $f$ at every point in $\Omega$. Admittedly, it is rare to know that $f$ is convex when $f$ is not given in closed form (although one may be able to detect convexity [26]). Nevertheless, we believe that studying the convex case is important because we are unaware of any method (besides complete enumeration) for obtaining exact solutions to (1) when $f$ (convex or otherwise) cannot be evaluated at noninteger points.

One example where an objective is not given in closed form but is known to be convex arises in the combinatorial optimal control of PDEs. For example, Buchheim et al. [9, Lemma 2] show that the solution operator of certain semilinear elliptic PDEs is a convex function of the controls provided that the nonlinearities in the PDE and boundary conditions are concave and nondecreasing. Thus, any linear function of

*Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439 (jmlarson@anl.gov, leyffer@anl.gov, wild@anl.gov).

†Industrial Engineering and Operations Research, Indian Institute of Technology Bombay, Powai, Mumbai, MH 400076 (prashant.palkar@iitb.ac.in).

the states of the PDE (e.g., the $\max$-function) is a convex function of the controls when the states are eliminated. The authors of [9] propose using adjoint information to compute subgradients of the objective, but an alternative would be to consider a derivative-free approach.

We consider only pure-integer problems of the form (1); however, our developments are equally applicable to the mixed-integer case

$$\underset{x,y}{\text{minimize}} \; F(x,y) \quad \text{subject to:} \; (x,y) \in \Omega \times \Psi \subset \mathbb{Z}^n \times \mathbb{R}^m \tag{2}$$

provided $F$ is convex on $\Omega \times \Psi$. If we define the function

$$f(x) = \min_{y \in \Psi} F(x,y)$$

and if $f$ is well defined over $\Omega$, then (2) can be solved by minimizing $f$ on $\Omega \subset \mathbb{Z}^n$, where each evaluation $f(x)$ requires an optimization of the continuous variables $y$ for a fixed $x$. Because many of the results below rely only on the convexity of $f$ and not the discrete nature of $\Omega$, much of the analysis below readily applies to the mixed-integer case.

Problems of the form (1) or (2) where the objective is expensive to evaluate and some integer constraints are unrelaxable arise in a range of simulation-based optimization problems. For example, the optimal design of concentrating solar power plants gives rise to computationally expensive simulations for each set of design parameters [40]. Furthermore, some of the design parameters (e.g., the number of panels on the power plant receiver) cannot be relaxed to noninteger values. Similar problems arise when tuning codes to run on high-performance computers [6]. In this case, $f(x)$ may be the memory footprint of a code that is compiled with settings $x$, which can correspond to decisions such as loop unrolling or tiling factors that do not have meaningful noninteger values. Optimal material design problems may also constrain the choice of atoms to a finite set, resulting in unrelaxable integer constraints; see [21] for a derivative-free optimization algorithm designed explicitly for such a problem.

Motivated by such applications, we develop a method that will certifiably converge to the solution of (1) under Assumption 1 without access to $\partial f$. Using only evaluations of $f$, we construct *secants*, which are linear functions that interpolate $f$ at a set of $n+1$ points. These secant functions underestimate $f$ in certain parts of $\Omega$. We use these secants to define *conditional cuts* that are valid in disconnected portions of the domain. The complete set of secants and the conditions that describe when they are valid are used to construct an underestimator of a convex $f$. While access to $\partial f$ could strengthen such an underestimator, we do not address such considerations in this paper.

Solving (1) under Assumption 1 without access to $\partial f$ poses a number of theoretical and computational challenges. Because the integer constraints are unrelaxable, one cannot apply traditional branch-and-bound approaches. In particular, model-based continuous derivative-free methods would require evaluating the objective at noninteger points to ensure convergence for the continuous relaxation of (1). In addition, other traditional techniques for mixed-integer optimization—such as Benders decomposition [20] or outer approximation [17, 18]—cannot be used to solve (1) when $\partial f$ is unavailable. Since we know of no method (other than complete enumeration) for obtaining global minimizers of (1) under Assumption 1, we know of no potential algorithm to address this problem when a (sub)gradient is unavailable.

We make three contributions in this paper: (1) we develop a new underestimator for convex functions on the integer lattice that is based solely on function evaluations; (2) we present an algorithm that alternates between updating this underestimator and evaluating the objective in order to identify a global solution of (1) under Assumption 1; and (3) we show empirically that certifying global optimality when optimizing on the integer lattice is a challenging problem. In our experiments, we are unable to prove optimality for many problems when $n \geq 5$, and we provide insights into why a proof of optimality remains computationally challenging.

**Outline** Section 2 surveys recent methods for addressing (1). Section 3 introduces valid conditional cuts using only the function values of a convex objective and discusses the theoretical properties of these cuts. Section 4 presents an algorithm for solving (1) and shows that this algorithm identifies a global minimizer of (1) under Assumption 1. Section 5 considers two approaches for formulating the underestimator and presents the method SUCIL—secant underestimator of convex functions on the integer lattice. Section 6

provides detailed numerical studies for implementations of SUCIL on a set of convex problems. Section 7 discusses many of the challenges in obtaining global solutions to (1).

## 2 Background

Developing methods to solve (1) without access to derivatives of $f$ is an active area of research. Most methods address general (i.e., nonconvex) functions $f$, and heuristic approaches are commonly adopted to handle integer decision variables for such derivative-free optimization problems. For example, the method in [36] rounds noninteger components of candidate points to the nearest feasible integer values. The method's asymptotic convergence results are based on the inclusion of points drawn uniformly from the domain (and rounding noninteger values as necessary).

Integer-constrained pattern-search methods [2, 4] generalize their continuous counterparts and target local minimizers. These modified pattern-search methods can be shown to converge to *mesh-isolated minimizers*: points with function values that are better than all neighboring points on the integer lattice. Unfortunately, such mesh-isolated minimizers can be arbitrarily far from a global minimizer, even when $f$ is convex; see [1, Fig. 2] for an example function. Other methods that converge to mesh-isolated minimizers include direct-search methods that update the integer variables via a local search [30, 31] and mesh adaptive direct-search methods adapted to address discrete and granular variables (those that have a controlled number of decimals) [3, 5]. The direct-search method in [19] accounts for integer constraints by constructing a set of directions that have a nonnegative span of $\mathbb{R}^n$ and that ensure that all intermediate iterates will be integer valued. This method is shown to converge to a stationary point that, even in the convex case, may not be a global minimizer. See [37] for various definitions of local minimizers of (1) and a discussion of associated properties. The BFO method [38] has a recursive step that explores points near the current iterate by fixing each of the discrete variables to its value plus or minus a step-size parameter.

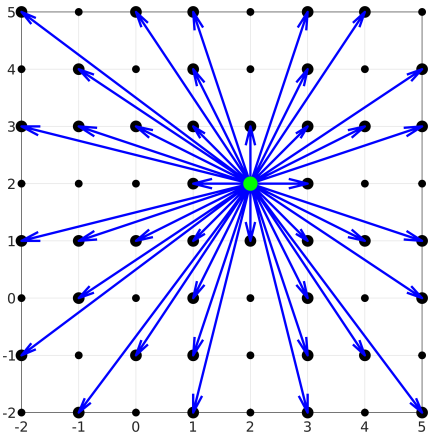



Figure 1: Primitive directions emanating from $(2, 2)$ in the domain $\Omega = [-2, 5]^2 \cap \mathbb{Z}^2$.

| | $n=2$ | | $n=3$ | | $n=4$ | | $n=5$ | |
|---|---|---|---|---|---|---|---|---|
| $k$ | $\lvert\Omega\rvert$ | # | $\lvert\Omega\rvert$ | # | $\lvert\Omega\rvert$ | # | $\lvert\Omega\rvert$ | # |
| 1 | 9 | 8 | 27 | 26 | 81 | 80 | 243 | 242 |
| 2 | 25 | 16 | 125 | 98 | 625 | 544 | 3,125 | 2,882 |
| 3 | 49 | 32 | 343 | 290 | 2,403 | 2,240 | 16,807 | 16,322 |
| 4 | 81 | 48 | 729 | 578 | 6,561 | 5,856 | 59,049 | 55,682 |

Table 1: Number of primitive directions, $\# = |\mathcal{N}(x_c, 1)|$, that emanate from the origin $x_c$ of the domain $\Omega = [-k, k]^n \cap \mathbb{Z}^n$ and that correspond to points in $\Omega$.

The method of Liuzzi et al. [32] uses line searches over a set of *primitive directions*; that is, a set of scaled directions $D$ where no vector $d_i \in D$ is a positive multiple of a different $d_j \in D$. This method explores a discrete set of directions around the current iterate until finding a local minimum $x_c$ in a $\beta-$neighborhood, defined as $\mathcal{N}(x_c, \beta) = \{x_c + \alpha d \in \Omega : d \in D, \alpha \in \mathbb{N}, \alpha \leq \beta\}$ for $\beta \in \mathbb{N}$. Although they target nonconvex objectives, their approach will converge to a global minimum $x_c$ of a convex objective $f$ if all points in $\mathcal{N}(x_c, 1)$ are evaluated. Figure 1 illustrates such a discrete 1-neighborhood. Unfortunately, $|\mathcal{N}(x_c, 1)|$ can be large; see Table 1.

Model-based methods approximate objective functions on the integer lattice by using surrogate models; see, for example, [25], [39], and [13]. The surrogate model is used to determine points where the objective should be evaluated; the model is typically refined after each objective evaluation. The methodology in [13]

specifically uses *radial basis function* models and does automatic model selection at each iteration. Mixed-integer nonlinear optimization solvers are used to minimize the surrogate to obtain the next integral point for evaluation. The model-based methods of [33, 34, 35] modify the sampling strategies and local searches used to solve continuous objective versions. The approaches in [33, 34] restart when a suitably defined local minimizer is encountered, continuing to evaluate the objective until the available budget of function evaluations is exhausted. These model-based methods differ in the initial sampling method, the type of surrogate model, and the sampling strategy used to select the next points to be evaluated. See [7], for a survey and taxonomy of continuous and discrete model-based optimization approaches.

In a different line of research, Davis and Ierapetritou [14] propose a branch-and-bound framework to address binary variables; a solution to the relaxed nonlinear subproblems is obtained via a combination of global kriging models and local response surface models. Similarly, Hemker et al. [23] replace the black-box portions of the objective function (and constraints) by a stochastic surrogate; the resulting mixed-integer nonlinear programs are solved by using branch-and-bound. Both approaches, however, assume that the integer constraints are relaxable.

# 3 Underestimator of Convex Functions on the Integer Lattice

To construct an underestimator of a convex objective function $f$, we now discuss secant functions, which are linear mappings that interpolate $f$ at $n+1$ points. We provide conditions for where these cuts will underestimate $f$. We then discuss necessary conditions on the set of evaluated points so that if all possible secants are constructed, these conditional cuts underestimate $f$ on all its domain $\Omega$. This underestimator is essential for obtaining a global minimizer of (1) under Assumption 1, as we will see in Section 4.

## 3.1 Secant Functions and Conditional Cuts

Constructing secant functions requires a set of interpolation points $X$ (satisfying $|X| \geq n+1$) where $f$ has been evaluated. To define a secant function for $f$, we introduce a multi-index $\boldsymbol{i}$ of $n+1$ distinct indices, $1 \leq i_1 < \ldots < i_{n+1} \leq |X|$, as $\boldsymbol{i} = (i_1, \ldots, i_{n+1})$. With a slight abuse of notation, we will refer to elements $i_j \in \boldsymbol{i}$.

Given the set of points $X^{\boldsymbol{i}} = \{x^{i_j} : i_j \in \boldsymbol{i}\}$, we construct the secant

$$m^{\boldsymbol{i}}(x) = (c^{\boldsymbol{i}})^{\mathrm{T}} x + b^{\boldsymbol{i}},$$

where the coefficients $c^{\boldsymbol{i}}$ and $b^{\boldsymbol{i}}$ are the solution to the linear system

$$\left[\bar{X}^{\boldsymbol{i}}\ e\right] \begin{bmatrix} c^{\boldsymbol{i}} \\ b^{\boldsymbol{i}} \end{bmatrix} = f^{\boldsymbol{i}}, \text{ where } \bar{X}^{\boldsymbol{i}} = \begin{bmatrix} (x^{i_1})^{\mathrm{T}} \\ \vdots \\ (x^{i_{n+1}})^{\mathrm{T}} \end{bmatrix}, \ e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \text{ and } f^{\boldsymbol{i}} = \begin{bmatrix} f(x^{i_1}) \\ \vdots \\ f(x^{i_{n+1}}) \end{bmatrix}. \tag{3}$$

The secant $(c^{\boldsymbol{i}})^{\mathrm{T}} x + b^{\boldsymbol{i}}$ is unique provided that the set $X^{\boldsymbol{i}}$ is *poised*, which we now define.

**Definition 1.** *The set of points $X^{\boldsymbol{i}}$ is* poised *if the matrix $\left[\bar{X}^{\boldsymbol{i}}\ e\right]$ is nonsingular.*

Note that Definition 1 is equivalent to $X^{\boldsymbol{i}}$ being affinely independent. We show that the secant $m^{\boldsymbol{i}}$ underestimates $f$ in certain polyhedral cones: namely, the cones

$$\mathcal{U}^{\boldsymbol{i}} = \bigcup_{i_j \in \boldsymbol{i}} \operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right), \text{ where} \tag{4}$$

$$\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right) = \{x^{i_j} + \sum_{l=1, l\neq j}^{n+1} \lambda_l (x^{i_j} - x^{i_l}) : \ i_j \in \boldsymbol{i}, i_l \in \boldsymbol{i}, \lambda_l \geq 0\}. \tag{5}$$

**Lemma 1** (Conditional Cuts). *If $f$ is convex and $X^{\boldsymbol{i}}$ is poised, then the unique linear mapping $m^{\boldsymbol{i}}$ satisfying $m^{\boldsymbol{i}}(x^{i_j}) = f(x^{i_j})$ for each $i_j \in \boldsymbol{i}$ satisfies $m^{\boldsymbol{i}}(x) \leq f(x)$ for all $x \in \mathcal{U}^{\boldsymbol{i}}$.*

*Proof.* The uniqueness of the linear mapping follows directly from the affine independence guaranteed by Definition 1 for poised $X^{\boldsymbol{i}}$.

Let $x$ be a point in $\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ for arbitrary $x^{i_j} \in X^{\boldsymbol{i}}$. By definition (5),

$$x = x^{i_j} + \sum_{l=1, l\neq j}^{n+1} \lambda_l \left(x^{i_j} - x^{i_l}\right), \tag{6}$$

with $\lambda_l \geq 0$ (for $l = 1, \ldots, n+1; l \neq j$). Rearranging (6) yields

$$x^{i_j} = \frac{1}{1 + \sum_{k=1,k\neq j}^{n+1} \lambda_k} x + \frac{1}{1 + \sum_{k=1,k\neq j}^{n+1} \lambda_k} \sum_{l=1,l\neq j}^{n+1} \lambda_l x^{i_l},$$

showing that $x^{i_j}$ can be expressed as a convex combination of $\{x\} \cup \{x^{i_l} : l = 1, \ldots, n+1; l \neq j\}$. Therefore, by convexity of $f$ and Jensen's inequality,

$$f(x^{i_j}) \leq \frac{1}{1 + \sum_{k=1,k\neq j}^{n+1} \lambda_k} f(x) + \frac{1}{1 + \sum_{k=1,k\neq j}^{n+1} \lambda_k} \sum_{l=1,l\neq j}^{n+1} \lambda_l f\left(x^{i_l}\right).$$

Solving for $f(x)$ and using the fact that $m^{\boldsymbol{i}}$ interpolates $f$ at points in $X^{\boldsymbol{i}}$, we obtain

$$\begin{aligned}
f(x) &\geq \left(1 + \sum_{k=1,k\neq j}^{n+1} \lambda_k\right) f(x^{i_j}) - \sum_{l=1,l\neq j}^{n+1} \lambda_l f(x^{i_l}) \\
&= \left(1 + \sum_{k=1,k\neq j}^{n+1} \lambda_k\right) m^{\boldsymbol{i}}(x^{i_j}) - \sum_{l=1,l\neq j}^{n+1} \lambda_l m^{\boldsymbol{i}}(x^{i_l}) \\
&= m^{\boldsymbol{i}}(x^{i_j}) + \sum_{l=1,l\neq j}^{n+1} \lambda_l \left(m^{\boldsymbol{i}}(x^{i_j}) - m^{\boldsymbol{i}}(x^{i_l})\right) \\
&= m^{\boldsymbol{i}}(x),
\end{aligned}$$

where the last equality holds by (6) and the linearity of $m^{\boldsymbol{i}}$. Because $x$ is an arbitrary point in $\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ for arbitrary $x^{i_j}$, the result is shown. □

We now prove that the $n+1$ cones in $\mathcal{U}^{\boldsymbol{i}}$ do not intersect.

**Lemma 2** (A Point Is in One Cone). *If $X^{\boldsymbol{i}}$ is a poised set, no point $x \in \mathbb{R}^n$ satisfies $x \in \operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ and $x \in \operatorname{cone}\left(x^{i_k} - X^{\boldsymbol{i}}\right)$ for $x^{i_j}, x^{i_k} \in X^{\boldsymbol{i}}$ and $x^{i_j} \neq x^{i_k}$.*

*Proof.* Let $x^{i_1}$ and $x^{i_2}$ be different, but otherwise arbitrary, points in $X^{\boldsymbol{i}}$. In order to arrive at a contradiction, suppose that there exists $x$ satisfying $x \in \operatorname{cone}\left(x^{i_1} - X^{\boldsymbol{i}}\right)$ and $x \in \operatorname{cone}\left(x^{i_2} - X^{\boldsymbol{i}}\right)$. That is, $x = x^{i_1} + \sum_{l=2}^{n+1} \lambda_l \left(x^{i_1} - x^{i_l}\right)$ and $x = x^{i_2} + \sum_{l=1,l\neq 2}^{n+1} \sigma_l \left(x^{i_2} - x^{i_l}\right)$ for $\lambda_l \geq 0$ ($l \in \{2, \ldots, n+1\}$) and $\sigma_l \geq 0$ ($l \in \{1, 3, \ldots, n+1\}$). Subtracting these two expressions for $x$ yields

$$\begin{aligned}
0 &= x^{i_1} - x^{i_2} + \sum_{l=2}^{n+1} \lambda_l (x^{i_1} - x^{i_l}) - \sum_{l=1,l\neq 2}^{n+1} \sigma_l (x^{i_2} - x^{i_l}) \\
&= x^{i_1} - x^{i_2} + \sum_{l=2}^{n+1} \lambda_l x^{i_1} - \sum_{l=2}^{n+1} \lambda_l x^{i_2} + \sum_{l=2}^{n+1} \lambda_l x^{i_2} - \sum_{l=2}^{n+1} \lambda_l x^{i_l} - \sum_{l=1,l\neq 2}^{n+1} \sigma_l (x^{i_2} - x^{i_l})
\end{aligned}$$

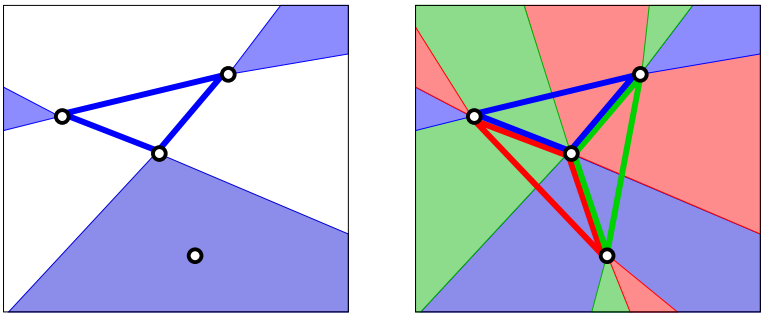


Figure 2: Illustration of areas in $\mathbb{R}^2$ where conditional cuts are valid. Left shows the regions of the domain where the secant through three points (the vertices of the blue triangle) will underestimate $f$. Right shows that one point in the interior of $n+1$ points is sufficient to underestimate $f$. The conditional cuts correspond to the $n+1$ points in the triangle of the same color.

$$= \left(1 + \sum_{l=2}^{n+1} \lambda_l\right)\left(x^{i_1} - x^{i_2}\right) - \sum_{l=2}^{n+1} \lambda_l \left(x^{i_l} - x^{i_2}\right) + \sum_{l=1, l\neq 2}^{n+1} \sigma_l \left(x^{i_l} - x^{i_2}\right)$$
$$= \left(1 + \sigma_1 + \sum_{l=2}^{n+1} \lambda_l\right)\left(x^{i_1} - x^{i_2}\right) + \sum_{l=3}^{n+1} (\sigma_l - \lambda_l)\left(x^{i_l} - x^{i_2}\right). \tag{7}$$

Since $X^{\boldsymbol{i}}$ is a poised set, Definition 1 ensures that the vectors $\left\{x^{i_l} - x^{i_2} : i_l \in \boldsymbol{i}, i_l \neq i_2\right\}$ are linearly independent. Hence the dependence relation in (7) can only be satisfied if the coefficient on $(x^{i_1} - x^{i_2})$ vanishes. That is,

$$1 + \sigma_1 + \sum_{l=2}^{n+1} \lambda_l = 0,$$

which contradicts $\lambda_l \geq 0$ (for $l = 2, \ldots, n+1$) and $\sigma_1 \geq 0$. Since $x^{i_1}$ and $x^{i_2}$ were arbitrary points in $X^{\boldsymbol{i}}$, the result is shown. □

For each poised set $X^{\boldsymbol{i}}$, Lemma 1 ensures that the secant $\eta \geq (c^{\boldsymbol{i}})^{\mathrm{T}} x + b^{\boldsymbol{i}}$ underestimates $f$ in $n+1$ cones within $\Omega$. We can therefore underestimate $f$ via a model that consists of the pointwise maximum of the underestimators for which the point is in $\mathcal{U}^{\boldsymbol{i}}$ for some poised set $X^{\boldsymbol{i}}$ of previously evaluated points. Minimizing this nonconvex model on the integer lattice can then provide a lower bound on the global minimum of $f$ on $\Omega$.

Figure 2 shows a two-dimensional example of points that produce such secant functions and the regions in which they will underestimate any convex function $f$. For the $n+2$ points (black dots), we consider three poised sets indicated by triangles linking $n+1$ points. The left image shows a poised set (blue line triangle), and the three cones (shaded blue area) in which the secant through these points is a valid underestimator. The right image shows that conditional cuts using $n+2$ points can cover all of $\mathbb{R}^n$.

### 3.2 Lower Bound on $f$

We now describe an optimization problem whose solution provides a lower bound on $f$ on $\Omega$. Let $W(X)$ denote the set of all multi-indices corresponding to poised subsets of $X$:

$$W(X) = \left\{\boldsymbol{i} : X^{\boldsymbol{i}} \subseteq X, X^{\boldsymbol{i}} \text{ poised}\right\}. \tag{8}$$

If $f$ has been evaluated at every point in $X$, we can construct a secant $(c^{\boldsymbol{i}})^{\mathrm{T}}x + b^{\boldsymbol{i}}$ interpolating $f$ on $X^{\boldsymbol{i}}$ for every multi-index $\boldsymbol{i} \in W(X)$. We then collect all such conditional cuts in the piecewise linear program

$$\begin{aligned} &\underset{x,\eta}{\text{minimize}}\ \eta \\ &\text{subject to: } \eta \geq (c^{\boldsymbol{i}})^{\mathrm{T}}x + b^{\boldsymbol{i}}, \text{ if } x \in \mathcal{U}^{\boldsymbol{i}}, \text{ for all } \boldsymbol{i} \in W(X) \\ &\qquad\qquad x \in \Omega, \end{aligned} \tag{PLP}$$

where $\mathcal{U}^{\boldsymbol{i}}$ is defined in (4). For the set of points $X$ and corresponding $W(X)$, let $\eta(\bar{x})$ denote the value of (PLP) when the constraint $x = \bar{x}$ is added to (PLP) for a particular $\bar{x} \in \Omega$. As we will see below, $\eta$ represents the largest lower bound on $f$ induced by the set $X$, and the solution to (PLP) provides a lower bound on $f$ on $\Omega$.

**Lemma 3** (Underestimator of $f$)**.** *If $f$ is convex, then the optimal value $\eta_*$ of* (PLP) *satisfies $\eta_* \leq f(x)$ for all $x \in \Omega$.*

*Proof.* If $W(X)$ is empty, the result holds trivially since $\eta$ is unconstrained. Otherwise, since (PLP) minimizes $\eta$, it suffices to show that $\eta(x) \leq f(x)$ for arbitrary $x \in \Omega$. Two cases can occur. First, if $x \notin \mathcal{U}^{\boldsymbol{i}}$ for every $\boldsymbol{i} \in W(X)$, then no conditional cut exists at $x$. Thus $\eta(x) = -\infty$ and $\eta(x) < f(x)$. Second, if $x \in \mathcal{U}^{\boldsymbol{i}}$ for some $\boldsymbol{i} \in W(X)$,

$$(c^{\boldsymbol{i}})^{\mathrm{T}}x + b^{\boldsymbol{i}} \leq f(x),$$

by Lemma 1, where the poisedness of $X^{\boldsymbol{i}}$ follows from the definition of $W(X)$. Therefore, $\eta(x) \leq f(x)$ for all $x \in \Omega$. Since $\eta_* = \min_{x \in \Omega} \eta(x)$, the result is shown. □

If $W(X)$ in (PLP) is replaced by a proper subset $W'(X) \subset W(X)$ of multi-indices, then Lemma 3 still holds. (This relaxation of (PLP) associated with removing constraints cannot increase $\eta_*$.) Such a replacement may be necessary if $W(X)$ becomes too large to allow considering every poised subset of $X$ when forming (PLP).

### 3.3 Covering $\mathbb{R}^n$ with Conditional Cuts

Because the cuts in (PLP) are valid only within $\mathcal{U}^{\boldsymbol{i}}$, the resulting model takes an optimal value of $\eta_* = -\infty$ if there is a point $x \in \Omega$ that is not in the union of $\mathcal{U}^{\boldsymbol{i}}$ over all $\boldsymbol{i} \in W(X)$. Thus, we find it beneficial to ensure $X$ contains points that result in a finite objective value for the underestimator described by (PLP). In this section, we provide one sufficient condition that ensures that the union of conditional cuts induced by $X$ covers $\mathbb{R}^n$, and therefore $\Omega$.

We say that a point $x^0$ belongs to the interior of the convex hull of a set of points $X = \{x^1, \ldots, x^{n+1}\}$ if scalars $\alpha_j$ exist such that

$$x^0 = \sum_{j=1}^{n+1} \alpha_j x^j, \text{ where } \sum_{j=1}^{n+1} \alpha_j = 1 \text{ and } \alpha_j > 0 \text{ for } j = 1, \ldots, n+1. \tag{9}$$

This is denoted by $x^0 \in \operatorname{int}(\operatorname{conv}(X))$.

**Lemma 4** (Poisedness of Initial Points)**.** *If $X = \{x^1, \ldots, x^{n+1}\} \subset \mathbb{R}^n$ is a poised set and if $x^0$ satisfies $x^0 \in \operatorname{int}(\operatorname{conv}(X))$, then all subsets of $n+1$ points in $\{x^0\} \cup X$ are poised.*

*Proof.* For contradiction, suppose that the set $\{x^0\} \cup X \setminus \{x^{n+1}\}$ is not poised and therefore is affinely dependent. Therefore, there must exist scalars $\beta_j$ not all zero, and (without loss of generality) $x^n \in X$ such that

$$\sum_{j=0}^{n-1} \beta_j (x^j - x^n) = 0. \tag{10}$$

Replacing $x^0$ with (9) in the left-hand side above yields

$$\beta_0 \left( \sum_{j=1, j\neq n}^{n+1} \alpha_j x^j + (\alpha_n - 1)x^n \right) + \sum_{j=1}^{n-1} \beta_j (x^j - x^n) = 0$$

$$\sum_{j=1}^{n-1} (\beta_0 \alpha_j + \beta_j) x^j + \left( \beta_0(\alpha_n - 1) - \sum_{j=1}^{n-1} \beta_j \right) x^n + \beta_0 \alpha_{n+1} x^{n+1} = 0. \tag{11}$$

Since $X$ is poised, the vectors $\{x^1, \ldots, x^{n+1}\}$ are affinely independent; by definition of affine independence, the only solution to $\sum_{j=1}^{n+1} \gamma_j x^j = 0$ and $\sum_{j=1}^{n+1} \gamma_j = 0$ is $\gamma_j = 0$ for $j = 1, \ldots, n+1$. Because the sum of the coefficients from (11) satisfies

$$\sum_{j=1}^{n-1} (\beta_0 \alpha_j + \beta_j) + \beta_0(\alpha_n - 1) - \sum_{j=1}^{n-1} \beta_j + \beta_0 \alpha_{n+1} = \beta_0 \sum_{j=1}^{n+1} \alpha_j - \beta_0 = 0,$$

because $\sum_{j=1}^{n+1} \alpha_j = 1$. Since $\alpha_{n+1} > 0$, the last term from (11) implies that $\beta_0 = 0$. Considering the remaining coefficients in (11), we conclude that $\beta_0 \alpha_j + \beta_j = 0$, which implies that $\beta_j = 0$ for $j = 1, \ldots, n-1$. This contradicts the assumption that not all $\beta_j = 0$. Hence, the result is proved. □

We now show a simple set of points that produces conditional cuts that cover $\mathbb{R}^n$ and, therefore, the domain $\Omega$.

**Lemma 5** (Initial Points and Coverage of $\Omega$)**.** *Let $X$ be a poised set of $n+1$ points, let $x^0 \in \operatorname{int}(\operatorname{conv}(X))$, and let $W(X \cup \{x^0\})$ be defined as in* (8)*. Then,*

$$\bigcup_{\boldsymbol{i} \in W(X \cup \{x^0\})} \mathcal{U}^{\boldsymbol{i}} = \mathbb{R}^n.$$

*Proof.* Since $x^0 \in \operatorname{int}(\operatorname{conv}(X))$, there exist $\alpha_j > 0$ such that

$$0 = (\sum_{j=1}^{n+1} \alpha_j)(x^0 - x^0) = (\sum_{j=1}^{n+1} \alpha_j) x^0 - \sum_{j=1}^{n+1} \alpha_j x^j = \sum_{j=1}^{n+1} \alpha_j (x^0 - x^j), \tag{12}$$

where the second equality follows from (9). The existence of $\alpha_j > 0$ such that $\sum_{j=1}^{n+1} \alpha_j (x^0 - x^j) = 0$ implies that the vectors $\{x^0 - x^j : j \in \{1, \ldots, n+1\}\}$ are a positive spanning set by [12, Theorem 2.3 (iii)]. Therefore arbitrary $x \in \mathbb{R}^n$ can be expressed as

$$x = \sum_{j=1}^{n} \alpha_j (x^0 - x^j),$$

with $\alpha_j \geq 0$ for all $j$.

We will show that any $x \in \mathbb{R}^n$ belongs to $\mathcal{U}^{\boldsymbol{i}}$ for some multi-index $\boldsymbol{i}$ containing $x^0$. By Lemma 4, every set of $n$ distinct vectors of the form $(x^0 - x^j)$ for $x^j \in X$ is a linearly independent set. Thus we can express

$$x - x^0 = \sum_{j=1, j\neq l}^{n+1} \lambda_j (x^0 - x^j), \tag{13}$$

for some $l \in \{1, \ldots n+1\}$. If $\lambda_j \geq 0$ for each $j$, then we are done, and $x \in \operatorname{cone}\big(x^0 - X \setminus \{x^l\}\big)$.

Otherwise, choose an index $j'$ such that $\lambda_{j'}$ is the most negative coefficient on the right of (13) (breaking ties arbitrarily). Using (12), we can exchange the indices $l$ and $j'$ in (13) by observing that

$$\lambda_{j'}(x^0 - x^{j'}) = \frac{-\lambda_{j'}}{\alpha_{j'}} \left( \sum_{j=1, j\neq j'}^{n+1} \alpha_j (x^0 - x^j) \right).$$

Note that $\frac{-\lambda_{j'}}{\alpha_{j'}}\alpha_j > 0$ by (9), and we can rewrite (13) as

$$x - x^0 = \sum_{j=1, j\neq j'}^{n+1} \mu_j (x^0 - x^j), \tag{14}$$

with new coefficients $\mu_j$ that are strictly larger than $\lambda_j$:

$$\mu_j = \begin{cases} \lambda_j - \dfrac{\lambda_{j'}}{\alpha_{j'}}\alpha_j > \lambda_j, & j \neq l, j \neq j' \\ -\frac{\lambda_{j'}}{\alpha_{j'}}\alpha_j & j = l. \end{cases}$$

Observe that (14) has the same form as (13), but with coefficients $\mu_j$ that are strictly greater than $\lambda_j$. We can now define $\lambda = \mu$ and repeat the process. If there is some $\lambda_{j'} < 0$, the process will strictly increase all $\lambda_j$. Because there are only a finite number of subsets of size $n$, we must eventually have all $\lambda_j \geq 0$. Once $\lambda_{j'}$ has been pivoted out, it can reenter only with a positive value (like $\mu_l$ above), so eventually all $\lambda_j$ will be nonnegative. □

Lemma 5 ensures that any poised set of $n + 1$ points with an additional point in their interior will produce conditional cuts that cover $\mathbb{R}^n$. Figure 2 illustrates this for $n = 2$. An alternative set of $n + 2$ points is

$$X = \{0, e_1, e_2, \ldots, e_n, -e\},$$

where $e_i$ is the $i$th unit vector and $e$ is the vector of ones. Larger sets, such as those of the form

$$X = \{0, e_1, -e_1, \ldots, e_n, -e_n\},$$

will similarly guarantee coverage of $\mathbb{R}^n$.

We note that the results in this section do not rely on $X$ or $\Omega$ being a subset of $\mathbb{Z}^n$. Therefore, the results are readily applicable to the case when $f$ has continuous and integer variables.

# 4 Convergence Analysis

We now present Algorithm 1 to identify global solutions to (1) under Assumption 1. This algorithm constructs a sequence of underestimators of the form (PLP). Section 5.1.1 and Section 5.1.2 show two approaches for modeling the underestimator; Section 5.2 highlights other details that are important for an efficient implementation of Algorithm 1. For example, the next iterate can be a solution of (PLP) but not necessarily so.

Note that (PLP) provides a valid lower bound on $f$ on $\Omega$. If $X \subseteq \Omega$ are points where $f$ has been evaluated, then $\min\{f(x) : x \in X\}$ is an upper bound on the minimum of $f$ on $\Omega$. Algorithm 1 terminates when the upper bound is equal to the lower bound provided by (PLP). We observe that Algorithm 1 produces a nondecreasing sequence of lower bounds provided that conditional cuts are not removed from (PLP); we show in Theorem 1 that this sequence of lower bounds will converge to the global minimum of $f$ on $\Omega$. (Algorithm 1 resembles a traditional outer-approximation approach [8, 17, 18] in that it obtains a sequence of lower bounds of (1) using an underestimator that is updated after each function evaluation. These function evaluations provide a nonincreasing sequence of upper bounds on the objective; when the upper bound equals the lower bound provided by the underestimator, the method can terminate with a certificate of optimality.)

Algorithm 1 leaves open a number of important decisions concerning how (PLP) is formulated and solved and how the next iterate is selected. While we will discuss more involved options for addressing these concerns, a simple choice would be to add all new possible cuts and let the next iterate be a minimizer of (PLP). Although such choices can result in computational difficulties, these choices are useful for showing the behavior of Algorithm 1, which we do now. In Figure 3 we see three iterations of Algorithm 1 solving the one-dimensional problem

$$\text{minimize } f(x) = x^2 \quad \text{subject to: } x \in [-3, 3], x \in \mathbb{Z}.$$

**Input:** A set of evaluated points $X^0 \subseteq \Omega$ satisfying $\left|W(X^0)\right| > 0$

1 Set $\hat{x} \in \underset{x \in X^0}{\arg\min}\, f(x)$, upper bound $u_0 \leftarrow f(\hat{x})$, and lower bound $l_0 \leftarrow -\infty$; $k \leftarrow 0$

2 **while** $l_k < u_k$ **do**

3 Update: Update the piecewise linear program (PLP) using $W(X^k)$

4 Lower Bound: Solve (PLP) and let its optimal value be $l_{k+1}$

5 Next Iterate: Select a new trial point $x^{k+1} \in \Omega \setminus X^k$

6 Evaluate $f(x^{k+1})$ and set $X^{k+1} \leftarrow X^k \cup \{x^{k+1}\}$

7 **if** $f(x^{k+1}) < u_k$ **then**

8 Upper Bound: New incumbent $\hat{x} \leftarrow x^{k+1}$ and upper bound $u_{k+1} \leftarrow f(x^{k+1})$

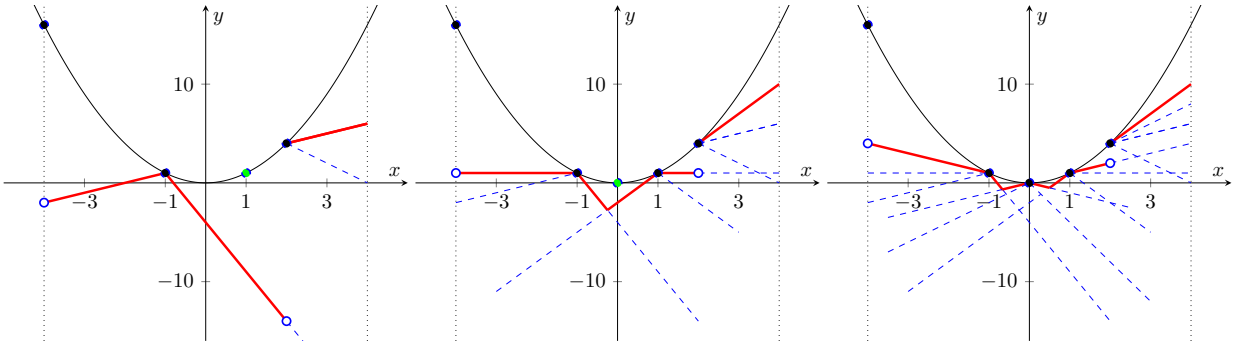
9 **else**

10 $u_{k+1} \leftarrow u_k$

11 $k \leftarrow k + 1$

**Output:** $\hat{x}$, a global minimizer of $f$ on $\Omega$

**Algorithm 1:** Identifying a global minimizer of a convex objective on the integer lattice.




Iter. 1: $l_k = -9, u_k = 1, \hat{x} = 1$ Iter. 2: $l_k = -0.27, u_k = 0, \hat{x} = 0$ Iter. 3: $l_k = 0, u_k = 0, \hat{x} = 0$

Figure 3: Illustration of Algorithm 1 minimizing $f(x) = x^2$ on $[-4, 4] \cap \mathbb{Z}$.

Black dots indicate interpolation points where $f$ has been previously evaluated, and green dots indicate the solution to (PLP) in each iteration. The solid red lines show the piecewise linear underestimator of the function. We observe that Lemma 1 can be strengthened for one-dimensional problems where conditional cuts underestimate convex $f$ at all points outside the convex hull of the points used to determine the corresponding secant function. (This is not true for $n > 1$.)

We now prove that Algorithm 1 identifies a global minimizer of convex $f$.

**Theorem 1** (Convergence of Algorithm 1)**.** *If* Assumption 1 *holds,* Algorithm 1 *terminates at an optimal solution* $x^*$ *of* (1) *in finitely many iterations.*

*Proof.* Algorithm 1 will terminate in a finite number of iterations because Assumption 1 ensures that $\Omega$ is bounded and Line 5 ensures that $x^k$ is not a previously evaluated element of $\Omega$.

For contradiction, assume that Algorithm 1 terminates at iteration $k'$ with $f(\hat{x}) > f(x^*)$ for some $x^* \in \underset{x \in \Omega}{\arg\min}\, f(x)$. It follows from Line 8 that $x^* \notin X^{k'}$, because $f(x^*) < f(\hat{x})$. Lemma 3 ensures that the value of each conditional cut at $x^*$ is not larger than $f(x^*)$, which implies that $\eta(x^*) \leq f(x^*)$. Thus, the lower bound satisfies

$$l_{k'} \leq f(x^*) < f(\hat{x}) = u_{k'}.$$

Since $l_{k'} < u_{k'}$, Algorithm 1 did not terminate at iteration $k'$, giving a contradiction. Therefore, the result is shown. □

A special case of Theorem 1 ensures that Algorithm 1 terminates with a global solution of (1) when $x^k$ is an optimal solution of (PLP).

# 5 Implementation Details

Algorithm 1 relies critically on the underestimator described by (PLP). Section 5.1 develops two approaches for formulating (PLP), and Section 5.2 discusses important details for efficiently implementing Algorithm 1. Section 5.3 combines these details in a description of our preferred method for solving (1), SUCIL

## 5.1 Formulating (PLP)

We present two methods for encoding (PLP) and thereby obtain lower bounds on (1). The first approach formulates (PLP) as a mixed-integer linear program (MILP) using binary variables to indicate when a point $x$ is in $\mathcal{U}^{\boldsymbol{i}}$ for some multi-index $\boldsymbol{i}$. Unfortunately, the resulting MILP is difficult to solve for even small problem instances. This motivates the development of the second approach, which directly builds an enumerative model of (PLP) in the space of the original variables only.

### 5.1.1 Mixed-Integer Linear Programming Approach

Formulating (PLP) as an MILP requires forming the secant function $m^{\boldsymbol{i}}(x) = (c^{\boldsymbol{i}})^{\mathrm{T}}x + b^{\boldsymbol{i}}$ corresponding to each multi-index $\boldsymbol{i} \in W(X)$. Since $m^{\boldsymbol{i}}$ is valid only in $\mathcal{U}^{\boldsymbol{i}}$ (see Lemma 3), we use binary variables to encode when $x \in \mathcal{U}^{\boldsymbol{i}}$. Explicitly, for each $\boldsymbol{i} \in W(X)$ and each $i_j \in \boldsymbol{i}$, our MILP model sets the binary variable $z^{i_j}$ to be 1 if and only if $x \in \operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$. While the forward implication can be easily modeled by using continuous variables $\lambda^{i_j}$, we must introduce additional binary variables $w^{i_j}$ for the reverse implication.

We now describe the constraints in the MILP model. The first set of constraints ensures that $\eta$ is no smaller than any of the conditional cuts that underestimate $f$:

$$\eta \geq (c^{\boldsymbol{i}})^{\mathrm{T}}x + b^{\boldsymbol{i}} - M_\eta \left(1 - \sum_{j=1}^{n+1} z^{i_j}\right), \quad \forall \boldsymbol{i} \in W(X), \tag{15}$$

where $M_\eta$ is a sufficiently large constant. By Lemma 2, we can add constraints to ensure that $x \in \Omega$ belongs to no more than one of the cones in $\mathcal{U}^{\boldsymbol{i}}$ for a given $\boldsymbol{i}$:

$$\sum_{j=1}^{n+1} z^{i_j} \leq 1, \ \forall \boldsymbol{i} \in W(X). \tag{16}$$

The following constraints define each point $x \in \Omega$ as a linear combination of the extreme rays of each $\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$:

$$x = x^{i_j} + \sum_{l=1, l\neq j}^{n+1} \lambda_l^{i_j} \left(x^{i_j} - x^{i_l}\right), \quad \forall \boldsymbol{i} \in W(X), \ \forall i_j \in \boldsymbol{i}. \tag{17}$$

To indicate that $x \in \operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$, the following constraints enforce a lower bound of 0 on $\lambda$ when the corresponding $z^{i_j} = 1$:

$$\lambda_l^{i_j} \geq -M_\lambda \left(1 - z^{i_j}\right), \quad \forall \boldsymbol{i} \in W(X), \ \forall i_j, i_l \in \boldsymbol{i}, j \neq l, \tag{18}$$

where $M_\lambda$ is a sufficiently large constant. Next, we introduce the binary variables $w_l^{i_j}$ that are 1 when the corresponding variable $\lambda_l^{i_j}$ is nonnegative. The following constraints model the condition: $w_l^{i_j} = 0$ implies that the corresponding $\lambda_l^{i_j}$ takes a negative value:

$$\lambda_l^{i_j} \leq -\epsilon_\lambda + M_\lambda w_l^{i_j}, \quad \forall \boldsymbol{i} \in W(X), \ \forall i_j, i_l \in \boldsymbol{i}, j \neq l, \tag{19}$$

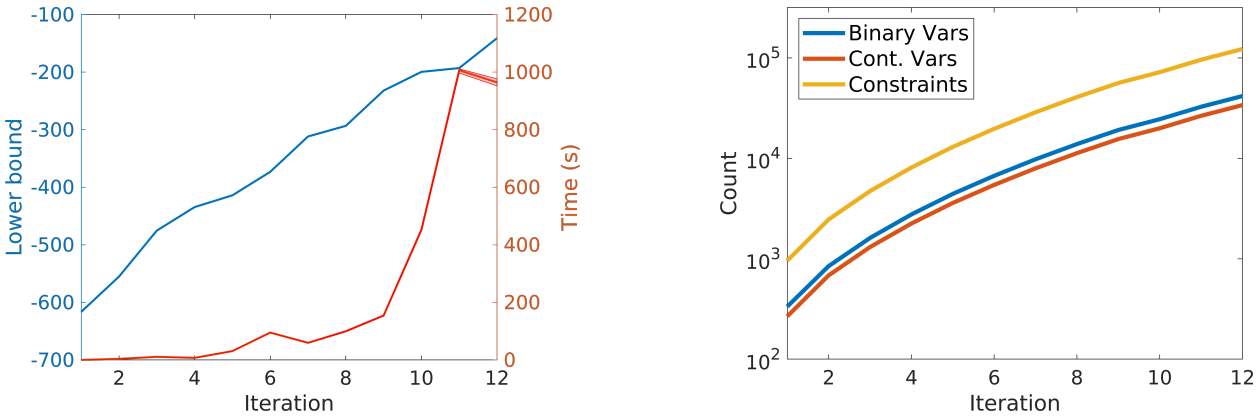



Figure 4: Characteristics of the first 12 instances of (CPF) generated by Algorithm 1 minimizing the convex quadratic `abhi` on $\Omega = [-2, 2]^3 \cap \mathbb{Z}^3$. Left shows the lower bound and solution time (mean of five replications, maximum and minimum times are also shown); right shows the number of binary and continuous variables and constraints. For further details of these 12 MILP models, see Table 4 in Supplement 9.

where $\epsilon_\lambda$ is a sufficiently small positive constant. The last set of constraints force at least one of the $w$ variables to be 0 if the corresponding $z$ is 0:

$$nz^{i_j} \leq \sum_{l=1, l\neq j}^{n+1} w_l^{i_j} \leq n - 1 + z^{i_j} \quad \forall \boldsymbol{i} \in W(X),\ \forall i_j \in \boldsymbol{i}. \tag{20}$$

The full MILP model encoding of (PLP) is

$$\begin{aligned} &\underset{x,\lambda,z,w}{\text{minimize}}\ \eta \\ &\text{subject to: } (15) - (20) \\ &\qquad w_l^{i_j}, z^{i_j} \in \{0,1\}, \quad \forall l, j \in \{1,\dots,n+1\}, l \neq j; \quad \forall \boldsymbol{i} \in W(X) \\ &\qquad x \in \Omega. \end{aligned} \tag{CPF}$$

The constants $M_\eta$, $M_\lambda$, and $\epsilon_\lambda$ must be chosen carefully in order to avoid numerical issues when solving (CPF). In early numerical results, we observed that taking large values for $M_\eta, M_\lambda$ and small values for $\epsilon_\lambda$ resulted in numerical issues for the MILP solvers. In an attempt to remedy this situation, we derived cuts in which $c^{\boldsymbol{i}}, b^{\boldsymbol{i}}$ are integer valued. One can then show, for example, that $1/\|c^{\boldsymbol{i}}\|_2$ is a valid lower bound on $\epsilon_\lambda$, and similar tight bounds can be derived for $M_\lambda$. With these tighter constants, some numerical issues were resolved. Yet, the growth of the number of constraints in (CPF) prevented its application to problems with $n \geq 3$.

Initial versions of the MILP model (CPF) resulted in large times to solution. Figure 4 shows the behavior of Algorithm 1—when adding all possible cuts when updating (PLP) and choosing the next iterate be a minimizer of (PLP)—when minimizing the convex quadratic function `abhi` (defined in Table 3 of Supplement 8) on $\Omega = [-2, 2]^3 \cap \mathbb{Z}^3$. We note, that the variations in CPU time are consistent over five repeated runs and vary by less than 2.4% for the last two iterations. We find that this growth in CPU time is due to the increasing number of conditional cuts and the associated explosion in the number of binary and continuous variables. This trend appears to limit the applicability of the MILP approach. Note that the global minimum of `abhi` on $[-2, 2]^3 \cap \mathbb{Z}^3$ has not yet been encountered when the MILPs become too large to solve. (The iteration 13 MILP was not solved in 30 minutes.)

### 5.1.2 Enumerative Approach

Whereas the MILP from Section 5.1.1 encodes information about every conditional cut in a single model, this section considers an alternative approach of updating the value of $\eta(x)$ for each $x \in \Omega$ as new condi-

tional cuts are encountered. After the information from a new secant function is used to update $\eta(x)$, the secant is discarded.

Ordering the set of feasible integer lattice points as $\{x^1, x^2, \ldots, x^{|\Omega|}\}$, then our approach maintains and updates a vector of bounds

$$\left[\eta(x^1), \eta(x^2), \ldots, \eta(x^{|\Omega|})\right]^{\mathrm{T}} \in \mathbb{R}^{|\Omega|}, \tag{21}$$

where $\eta(x^j)$ is the value of (PLP) when $x = x^j$. The value of $\eta(x^j)$ is initialized to $-\infty$, and as each secant is constructed, $\eta(x^j)$ is set to the maximum of its current value and the value of the conditional cut at $x^j$. This procedure is described in Algorithm 2. Since the important information about each conditional cut will be stored in $\eta(x)$, the secants defining each cut do not need to be stored. Furthermore, if $\eta_k(x)$ is the value of the underestimator (21) at iteration $k$, then solving each instance of (PLP) corresponds to looking up $\underset{j\in\{1,\ldots,|\Omega|\}}{\arg\min}\ \eta(x^j)$ (breaking ties arbitrarily). Similarly, termination of Algorithm 1 requires testing only that $\underset{j\in\{1,\ldots,|\Omega|\}}{\min}\ \eta_k(x^j) \geq u_k$.

Note that when solving (1), updating $\eta(x)$ for all $x \in \Omega$ is unnecessary. Rather, one needs to update $\eta(x)$ only at points that could possibly be a global minimum of $f$ on $\Omega$. When $f$ is evaluated at $x^{k+1}$ and a multi-index $\boldsymbol{i} \in W(X^k \cup x^{k+1})$ is encountered that is not in $W(X^k)$, we update the lower bound only at points in $\mathcal{U}^{\boldsymbol{i}}$ that are also in

$$\Omega_k = \{x \in \Omega \setminus X^k : \eta_k(x) < u_k\}. \tag{22}$$

That is, we update $\eta_k(x)$ for points in $\mathcal{U}_k^{\boldsymbol{i}} = \Omega_k \cap \mathcal{U}^{\boldsymbol{i}}$ for each newly encountered $\boldsymbol{i}$.

**Function** `UpdateEta(`$X^{\boldsymbol{i}}$,$b^{\boldsymbol{i}}$,$c^{\boldsymbol{i}}$,$\mathcal{U}_k^{\boldsymbol{i}}$,$\eta(x)$`)`**:**
  **for** $i_k \in \boldsymbol{i}$ **do**
    **for** $j = 1, \ldots, |\Omega|$ **do**
      **if** $x^j \in \mathrm{cone}\left(x^{i_k} - X^{\boldsymbol{i}}\right) \cap \mathcal{U}_k^{\boldsymbol{i}}$ **then**
        $\eta(x^j) \leftarrow \max\left(\eta(x^j), (c^{\boldsymbol{i}})^{\mathrm{T}} x^j + b^{\boldsymbol{i}}\right)$

**Algorithm 2:** Routine for updating lower bound on $f$ at each point in $\Omega$.

## 5.2 Other Implementation Details

The enumerative approach of maintaining the value of the underestimator $\eta(x)$ described in Section 5.1.2 avoids many of the computational pitfalls of the MILP model discussed in Section 5.1.1. Below, we discuss additional computational enhancements that lead to an efficient implementation of Algorithm 1 in conjunction with Algorithm 2.

### 5.2.1 Checking Whether $X^{\boldsymbol{i}}$ Is Poised and Whether $x \in \mathcal{U}^{\boldsymbol{i}}$

We now describe a numerically efficient representation of $\mathrm{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ for $i_j \in \boldsymbol{i}$. Given a poised set of $n+1$ points, $X^{\boldsymbol{i}}$, for each $i_j \in \boldsymbol{i}$ we define a secant function satisfying

$$(c^{i_j})^{\mathrm{T}} x^{i_l} + b^{i_j} = 0, \text{ for all } i_l \in \boldsymbol{i}, i_l \neq i_j, \text{ and} \tag{23}$$

$$(c^{i_j})^{\mathrm{T}} x^{i_j} + b^{i_j} > 0. \tag{24}$$

Only one such secant exists for each $i_j \in \boldsymbol{i}$; however, the representation of this secant is not unique since $(c^{i_j}, b^{i_j})$ are obtained by solving an underdetermined system of equations. Given $(c^{i_j}, b^{i_j})$ satisfying (23) and (24), we define the corresponding halfspace,

$$H^{i_j} = \{x : (c^{i_j})^{\mathrm{T}} x + b^{i_j} \leq 0\}. \tag{25}$$

We now show that $\mathrm{cone}\left(x^j - X^{\boldsymbol{i}}\right)$ (defined in (5)) can be represented as the intersection of $n$ such halfspaces.

**Lemma 6** (Set Equality). *For a poised set $X^{\boldsymbol{i}}$, $\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right) = F^{i_j} = \bigcap_{i_l \neq i_j} H^{i_l}$ for each $i_j \in \boldsymbol{i}$.*

*Proof.* Let $\boldsymbol{i}$ be given and $i_j \in \boldsymbol{i}$ fixed. We first show that $\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right) \subseteq F^{i_j}$ by showing that an arbitrary $x \in \operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ satisfies (25) for each $i_l \in \boldsymbol{i}, i_l \neq i_j$. Given $(c^{i_l}, b^{i_l})$ satisfying (23) and (24), then using the definition (5) yields

$$\begin{aligned}
(c^{i_l})^{\mathrm{T}} x + b^{i_l} &= (c^{i_l})^{\mathrm{T}} \left( x^{i_j} + \sum_{k=1, k\neq j}^{n+1} \lambda_k (x^{i_j} - x^{i_k}) \right) + b^{i_l} \\
&= (c^{i_l})^{\mathrm{T}} x^{i_j} + b^{i_l} + \sum_{k=1, k\neq j}^{n+1} \lambda_k (c^{i_l})^{\mathrm{T}} x^{i_j} - \sum_{k=1, k\neq j}^{n+1} \lambda_k (c^{i_l})^{\mathrm{T}} x^{i_k} \\
&= 0 + \sum_{k=1, k\neq j}^{n+1} \lambda_k \left( (c^{i_l})^{\mathrm{T}} x^{i_j} + b^{i_l} \right) - \sum_{k=1, k\neq j}^{n+1} \lambda_k \left( (c^{i_l})^{\mathrm{T}} x^{i_k} + b^{i_l} \right) \\
&= 0 - \sum_{k=1, k\neq j}^{l-1} \lambda_k \left( (c^{i_l})^{\mathrm{T}} x^{i_k} + b^{i_l} \right) - \lambda_l \left( (c^{i_l})^{\mathrm{T}} x^{i_l} + b^{i_l} \right) - \sum_{k=l+1, k\neq j}^{n+1} \lambda_k \left( (c^{i_l})^{\mathrm{T}} x^{i_k} + b^{i_l} \right) \\
&= -\lambda_l \left( (c^{i_l})^{\mathrm{T}} x^{i_l} + b^{i_l} \right) \leq 0,
\end{aligned}$$

where we have used (23) in the last three equations. The final inequality holds because $\lambda_l \geq 0$ by (5) and $(c^{i_l})^{\mathrm{T}} x^{i_l} + b^{i_l} > 0$ by (24). Because $i_l$ is arbitrary, it follows that any $x$ in $\operatorname{cone}\left(x^{i_j} - X\right)$ is also in $F^{i_j}$.

We now show that $F^{i_j} \subseteq \operatorname{cone}\left(x^j - X^{\boldsymbol{i}}\right)$ by contradiction. If $x \notin \operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ for a set of $n+1$ poised points $X^{\boldsymbol{i}}$, then $x$ can be represented as $x^{i_j} + \sum_{l=1, l\neq j}^{n+1} \lambda_l \left( x^{i_j} - x^{i_l} \right)$ only with some $\lambda_l < 0$. Thus, (25) is violated for some $l$, and hence $x \notin F^{i_j}$. □

Lemma 6 gives a representation of each $\operatorname{cone}\left(x^{i_j} - X^{\boldsymbol{i}}\right)$ involving $n$ halfspaces that differs from $\operatorname{cone}\left(x^{i_l} - X^{\boldsymbol{i}}\right)$ for $i_l \in \boldsymbol{i}$, $i_l \neq i_j$ in only one component. Therefore, we can represent $\mathcal{U}^{\boldsymbol{i}}$ via only $n+1$ halfspaces. We efficiently calculate these halfspaces by utilizing the QR factorization $\left[ Q^{\boldsymbol{i}} \; R^{\boldsymbol{i}} \right] = \left[ \bar{X}^{\boldsymbol{i}} \; e \right]^{\mathrm{T}}$. If $R^{\boldsymbol{i}}$ has positive diagonal entries, then the multi-index $\boldsymbol{i}$ corresponds to a poised set $X^{\boldsymbol{i}}$. The coefficients in each $(c^{i_j}, b^{i_j})$ can be obtained by updating $Q^{\boldsymbol{i}}$, $R^{\boldsymbol{i}}$ by deleting the corresponding column from $\left[ \bar{X}^{\boldsymbol{i}} \; e \right]^{\mathrm{T}}$. The sign of $(c^{i_j}, b^{i_j})$ can be changed in order to ensure that (24) holds.

### 5.2.2 Approximating $W(X^k \cup \{x^{k+1}\})$

The use of $\eta_k(x)$ to store the lower bound at each $x \in \Omega_k$ allowed us to avoid encoding all secants in $W(X^k)$. After $f$ has been evaluated at a new point $x^{k+1}$, constructing the tightest possible underestimator in $\eta_k$ requires considering multi-indices $\boldsymbol{i}$ in $W(X^k \cup \{x^{k+1}\})$ that contain $x^{k+1}$. (Combinations not containing $x^{k+1}$ have already been considered in previous iterations.) While not storing secants is significantly more computationally efficient than encoding and storing all secants in $W(X^k)$, it still results in checking the poisedness of prohibitively many sets of $n+1$ points. For example, if $\left|X^k\right| = 100$ and $n = 5$, over 75 million QR factorizations must be performed, as discussed in Section 5.2.1.

Therefore, as an alternative, we seek a small, representative subset of multi-indices of $W(X^k)$ by identifying a subset of points that will yield the best conditional cuts.

**Definition 2.** *Let $\bar{W}_k$ be the set of multi-indices in $W(X^k)$ that define the largest lower bound at some point in $\Omega_k$ (defined in* (22)*). That is, $\bar{W}_k = \{\boldsymbol{i} : \exists x \in \Omega_k$ such that $\eta_k(x) = m^{\boldsymbol{i}}(x)\}$. We denote to the* generator *set of points as $G^k = \{x^j : \exists \boldsymbol{i} \in \bar{W}_k$ such that $j \in \boldsymbol{i}\}$.*

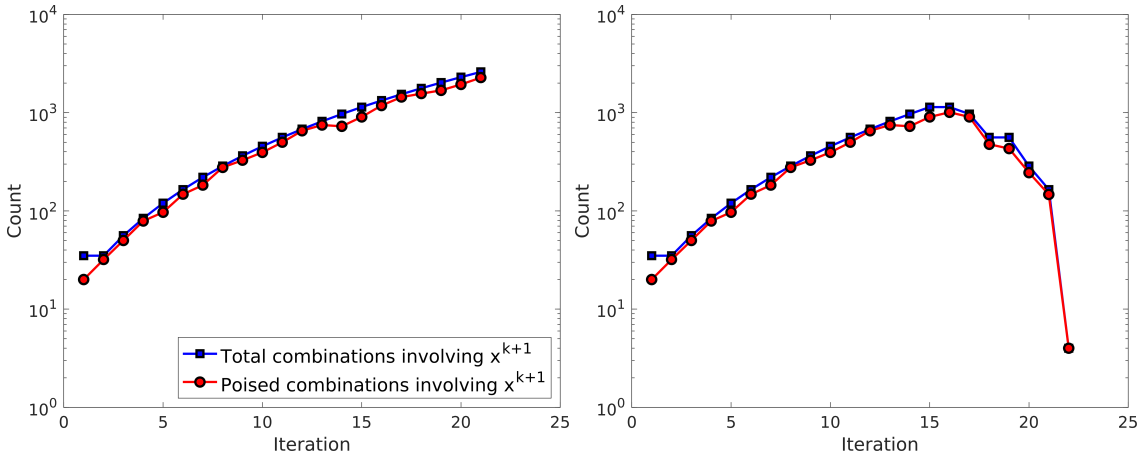



Figure 5: Number of total combinations and poised combinations that include $x^{k+1}$ in $W(X^k)$ (left) and $W(G^k)$ (right) when minimizing `quad` on $\Omega = [-4, 4]^3 \cap \mathbb{Z}^3$.

Hence, $G^k$ contains points that define $\eta_k(x)$ for at least one $x \in \Omega_k$. Using $W(G^k)$ in place of $W(X^k)$ does relax (PLP), yet the lower bounding property of (PLP) still remains. We show below that this change does not affect the finite termination property of Algorithm 1 provided at least one cut is added for every new $x^{k+1}$.

Figure 5 compares the growth of the number of subsets of indices that must be considered when determining whether a multi-index $\boldsymbol{i}$ is poised or not when using Algorithm 1 to minimize `quad` (Table 3 in Supplement 8) on $\Omega = [-4, 4]^3 \cap \mathbb{Z}^3$. Preliminary numerical experiments showed that although a high percentage of all combinations in $W(X^k \cup x^{k+1})$, which involve the new iterate $x^{k+1}$ at an iteration $k$, are poised, only a small fraction of these actually update the lower bound at any point in $\Omega_k$ (we elaborate more on this in Section 7).

#### 5.2.3 Selecting $x^{k+1}$

Early experiments with our algorithm showed that it wasted many early iterations evaluating points at the boundary of $\Omega$. Although Section 3.3 provides a method for ensuring that all $x \in \Omega$ are bounded by at least one conditional cut, the solution to (PLP) is often at the boundary of $\Omega$. Rather than moving so far from a candidate solution, we consider a trust-region approach to keep iterates close to the current incumbent. As long as we maintain a lower bound on $f$ on $\Omega$, the convergence proof in Theorem 1 does not depend on $x^{k+1}$ being the global minimizer of our lower bound.

In practice, we use an infinity-norm trust region and set the minimum trust-region radius, $\Delta_{\min}$ to 1. At iteration $k$, the maximum radius that must be considered is $\max_{x,y \in \Omega_k, x \neq y} \|x - y\|_\infty$.

### 5.3 The SUCIL Method

We now present the SUCIL method for obtaining global solutions to (1) under Assumption 1. The algorithm using the trust-region step is shown in Algorithm 3. We observe that Algorithm 3 maintains a valid lower bound $\eta_k(x)$ at every point, $x \in \Omega_k$, and that the trust-region mechanism ensures that the algorithm terminates only when the lower bound equals the best observed function value.

We note that $G^k$ may not be a subset of $G^{k+1}$, because $\Omega_k$ can contain fewer points as the upper and lower bounds on $f$ are improved. However, the following generalization of Theorem 1 ensures that Algorithm 3 still returns a global minimizer of (1).

**Theorem 2** (Convergence of Algorithm 3)**.** *If* Assumption 1 *holds and if $W(G^k)$ includes at least one cut for every $x \in \Omega_k$, then* Algorithm 3 *terminates at an optimal solution $x^*$ of* (1) *in finitely many iterations.*

**Input:** A set of evaluated points $X^0 \subseteq \Omega : \bigcup_{\boldsymbol{i} \in W(X^0)} \mathcal{U}^{\boldsymbol{i}} = \mathbb{R}^n$ and trust-region radius lower bound $\Delta_{\min} \geq 1$

1 Set $\hat{x} \in \arg\min_{x \in X^0} f(x)$, upper bound $u_0 \leftarrow f(\hat{x})$, $\Omega_0 \leftarrow \Omega$, and $k \leftarrow 0$

2 Initialize lower bounding function $\eta_{-1}(x) \leftarrow -\infty$ for all $x \in \Omega$; set lower bound $l_0 \leftarrow -\infty$

3 **while** $l_k < u_k$ **do**

4 Update:

5 Generate $G^k$ (according to Definition 2) using $X^k$

6 **for** $\boldsymbol{i} \in W(G^k)$ **do**

7 Compute $QR$ factors: $[Q, R] \leftarrow \mathrm{qr}([e\ X^{\boldsymbol{i}}])$

8 **if** $X^{\boldsymbol{i}}$ *is poised* **then**

9 Find coefficients $c^{\boldsymbol{i}}, b^{\boldsymbol{i}}$ and form set $\mathcal{U}_k^{\boldsymbol{i}} \leftarrow \Omega_k \cap \mathcal{U}^{\boldsymbol{i}}$ using $QR$ factors

10 Update look-up: $\eta_k \leftarrow$ `UpdateEta(`$X^{\boldsymbol{i}}$,$b^{\boldsymbol{i}}$,$c^{\boldsymbol{i}}$,$\mathcal{U}_k^{\boldsymbol{i}}$,$\eta_{k-1}$`)`; see Algorithm 2

11 Lower Bound:

12 $l_{k+1} \leftarrow \min_{x \in \Omega_k} \eta_k(x)$ from look-up table

13 **if** $l_{k+1} = u_k$ **then**

14 **break**

15 Next Iterate:

16 Update $\Omega_k \leftarrow \{x \in \Omega \setminus X^k : \eta_k(x) < u_k\}$

17 **if** $\{x \in \Omega_k : \|x - \hat{x}\| \leq \Delta_k\} = \emptyset$ **then**

18 Increase trust-region radius: $\Delta_k \leftarrow \Delta_k + 1$ until $\{x \in \Omega_k : \|x - \hat{x}\| \leq \Delta_k\} \neq \emptyset$

19 **else**

20 Set $x^{k+1} \in \arg\min_{x \in \Omega_k : \|x - \hat{x}\| \leq \Delta_k} \eta_k(x)$

21 Evaluate $f(x^{k+1})$ and set $X^{k+1} \leftarrow X^k \cup \{x^{k+1}\}$

22 **if** $f(x^{k+1}) < u_k$ **then**

23 Upper Bound:

24 New incumbent $\hat{x} \leftarrow x^{k+1}$ and upper bound $u_{k+1} \leftarrow f(x^{k+1})$

25 Increase trust-region radius $\Delta_{k+1} \leftarrow \Delta_k + 1$

26 **else**

27 No progress: $u_{k+1} \leftarrow u_k$ and reduce trust-region radius $\Delta_{k+1} \leftarrow \max\left\{\Delta_{\min}, \frac{\Delta_k}{2}\right\}$

28 $k \leftarrow k + 1$

**Output:** $\hat{x}$, a global minimizer of $f$ on $\Omega$

**Algorithm 3:** SUCIL: secant underestimator of convex functions on the integer lattice.

*Proof.* Algorithm 3 will terminate in a finite number of iterations because $\Omega$ is bounded and Line 16 ensures that $x^k$ is not a previously evaluated element of $\Omega$. Because $W(G^k) \subset W(X^k)$, it follows that $\eta_k(x)$ is a valid lower bound on $f$ on $\Omega$, and the trust-region mechanism in Line 18 ensures that we terminate only if $l_{k+1} = u_k$. Therefore, the result is shown. □

# 6 Numerical Experiments

We now describe numerical experiments that we performed on multiple versions of SUCIL; see Table 2. These methods differ in how $x^{k+1}$ is selected and in the set of points used within (PLP). The last two methods are idealized because they assume access to the true function value at every point in $\Omega_k$. They are included in order to provide a best-case performance for a SUCIL implementation. In the numerical experiments to follow, we set $\Delta_{\min} \leftarrow 1$ in Algorithm 3 and use an infinity-norm trust region. All SUCIL

| Method | $X$ in (PLP)? | $x^{k+1} =?$ |
|---|---|---|
| SUCIL | $G^k$ | $\arg\min_{x\in\Omega_k:\|x-\hat{x}\|_\infty\leq\Delta_k} \eta_k(x)$ |
| SUCIL-noTR | $G^k$ | $\arg\min_{x\in\Omega_k} \eta_k(x)$ |
| SUCIL-ideal1 | $X^k$ | $\arg\min_{x\in\Omega\setminus X^k} f(x)$ |
| SUCIL-ideal2 | $G^k$ | $\arg\min_{x\in\Omega\setminus X^k} f(x)$ |

Table 2: Description of how SUCIL versions choose $X$ in (PLP) and the next iterate $x^{k+1}$ (breaking ties in $\arg\min$ arbitrarily). $G^k$ is defined in Definition 2, and $X^k$ is all points evaluated before iteration $k$.

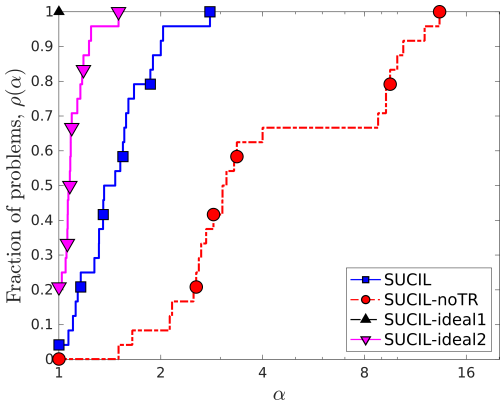



Figure 6: Performance profiles for SUCILs. Convergence measured by number of function evaluations before a method terminates with a certificate of global optimality.

instances begin by evaluating the starting point $\bar{x}$ and $\{\bar{x} \pm e_1, \ldots, \bar{x} \pm e_n\}$ ensuring a finite lower bound at every point in $\Omega$.

Below, we compare SUCIL implementations with a direct-search method, DFLINT [32], a model-based method, MATSuMoTo [33], and a hybrid method, NOMAD [3]. We tested the default nonmonotone DFLINT in MATLAB, as well as the monotone version, denoted DFLINT-M. We tested the default C++ version of NOMAD (v.3.9.0) as well as the same version with *DISABLE MODELS* set to true, denoted NOMAD-NM; the rest of the settings are default. MATSuMoTo is a surrogate-model toolbox explicitly designed for computationally expensive, black-box, global optimization problems. Since MATSuMoTo has a restarting mechanism that ensures that any budget of function evaluations will be exhausted, we input the optimal objective function value to MATSuMoTo and allowed it to run (and make as many restarts as required) until the global optimal value was identified. The default settings were used: surrogate models using cubic radial basis function, sampling at the minimum of the surrogate, and using an initial symmetric Latin hypercube design. We performed 20 replications of MATSuMoTo for each problem instance; the details of each run are shown in Tables 8–10 in Supplement 10. We report the floor of the average number of function evaluations incurred in the last row of these tables and use this statistic for our comparisons. A common starting point is given to all methods; the starting point for the `maxq` and `mxhilb` problems is the global minimizer. A maximum function evaluation limit of $1,000$ is set for all the methods.

We perform numerical experiments minimizing the convex objectives in Table 3 in Supplement 8 on the domains $[-4, 4]^n$ for $n \in \{3, 4, 5\}$ to yield 24 problem instances. (The last row of Table 1 shows $|\Omega|$ for these test problems.) Of note is the `KLT` function that generalizes the example function from [28] that shows how coordinate search methods can fail to find descent. The function from [28] is itself a modification of the Dennis-Woods function [15], is strongly convex, and for points $x$ along the line $x_1 = \cdots = x_n$ satisfies $f(x) < f(x \pm \epsilon e_i)$ for all $i$ and for all $\epsilon > 0$. The problems `CB3II`, `CB3I`, `LQ`, `maxq`, and `mxhilb` were introduced in [22] and also used in [32]. These five problems are either summation or maximization of generalizations of simple convex functions, constructed by extending or chaining nonsmooth convex functions or making smooth functions nonsmooth. The function `LQ` takes a global minimum at any $x \in [0, 1]^n \cap \mathbb{Z}^n$ that does not have zeros in consecutive coordinates. For example, for $n = 3$, the vectors $[0, 1, 0]^{\mathrm{T}}, [0, 1, 1]^{\mathrm{T}}, [1, 0, 1]^{\mathrm{T}}, [1, 1, 0]^{\mathrm{T}}$ and $[1, 1, 1]^{\mathrm{T}}$ are optimal but $[0, 0, 0]^{\mathrm{T}}, [0, 0, 1]^{\mathrm{T}}$, and $[1, 0, 0]^{\mathrm{T}}$ are not.

The wall-clock time taken by different DFO methods is not a suitable performance measure because it can depend on the implementation of the algorithm, its ability to exploit hardware, and other concerns. Instead we compare methods using performance profiles [16] based on the number of function evaluations required to satisfy a convergence criterion. For each method $s$, $\rho_s(\alpha) = \frac{|\{p\in P: r_{p,s}\leq\alpha\}|}{|P|}$, for a scalar $\alpha \geq 1$, $P$ is the collection of benchmark problems, and $r_{p,s} = \frac{N_{p,s}}{\min_{s\in S}\{N_{p,s}\}}$ is the performance ratio. We consider two measures of $N_{p,s}$: (1) the number of function evaluations before a method $s$ terminates on a problem $p$ and

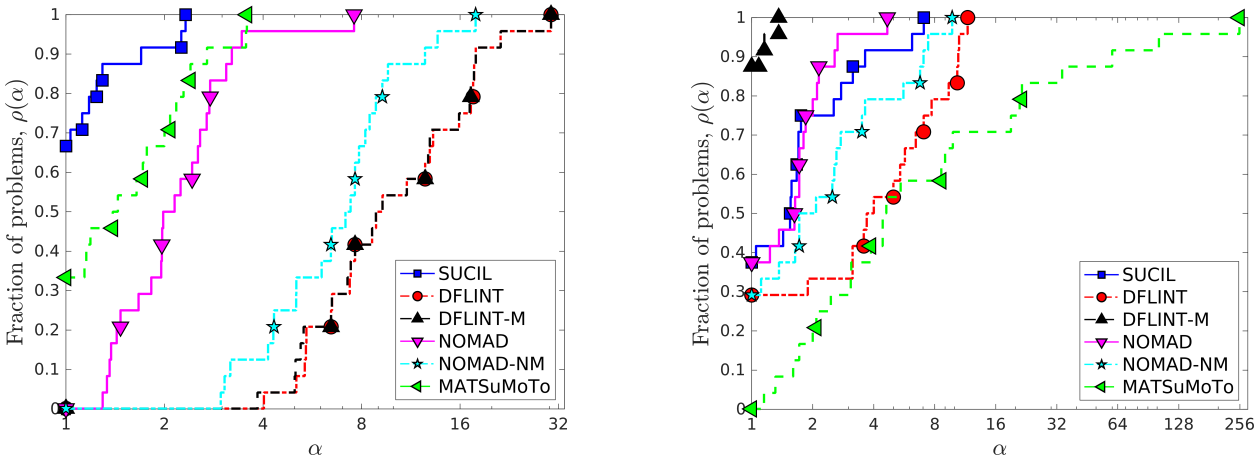



Figure 7: Performance profiles of different methods solving 24 problem instances. Left compares the number of evaluations until a method terminates; right compares the number of evaluations before a method first evaluates global minimizer. Performance for each solver on each problem instance can be found in Tables 5–7 in Supplement 10.

(2) the number of function evaluations taken by method $s$ to find a global minimizer on problem $p$.

Figure 6 compares the number of evaluations required for four implementations of SUCIL to terminate (with a certificate of optimality) on the set of benchmark problems. While SUCIL-ideal1 is no slower than any other implementation on all the benchmark problems, it is not a realistic method in that it evaluates points based on their known function values. SUCIL requires no more than three times the evaluations as SUCIL-ideal1 for the set of benchmark problems. We do observe that using a trust region in SUCIL is a significant advantage. For many of the problems considered, SUCIL-noTR spent many function evaluations in the corners of $\Omega$.

As a point of comparison with the results in Figure 6, a different estimate of the number of function evaluations (or primitive directions explorations) required for the proof of optimality for our instances can be seen in Table 1, in columns corresponding to $n \in \{3, 4, 5\}$ and $k = 4$. As evident from the results, our method incurs a remarkably low number of function evaluations, which can be attributed to exploitation of convexity and subsequent formation of the underestimators, as explained in Section 2.

Now, we analyze the performance of SUCIL compared with the other methods. Figure 7 (left) shows the performance profiles for methods to terminate on the 24 benchmark problems; Figure 7 (right) compares the number of function evaluations required before each method first evaluates a global minimizer. This comparison is nontrivial because each solver has its own design considerations and notions of local optimality. However, none of the methods except ours claims to converge to a global optimal solution practically. Figure 7 shows that our algorithm requires the least number of function evaluations for more than $65\%$ of the instances and provides a global optimality certificate, in addition. In reaching the global optimal solution quickly, however, DFLINT-M wins for more than $85\%$ of the instances. Although SUCIL is not particularly designed to greedily descend to the global optimum, it is still competitive with the rest of the methods on this front.

# 7 Discussion

The order of results in this paper tells the story of how we arrived at the implementation of SUCIL. We first attempted to classify where linear interpolation models provide lower bounds for convex functions, yielding the results in Section 3; we then proved that such linear functions can underlie a convergent algorithm, as in Section 4. We initially modeled the secants and the conditions in which they are valid as an MILP, as in Section 5.1.1. After observing that the number of variables in the MILP model was larger than the number of points in the domain, we were motivated to develop the enumerative model in Section 5.1.2.

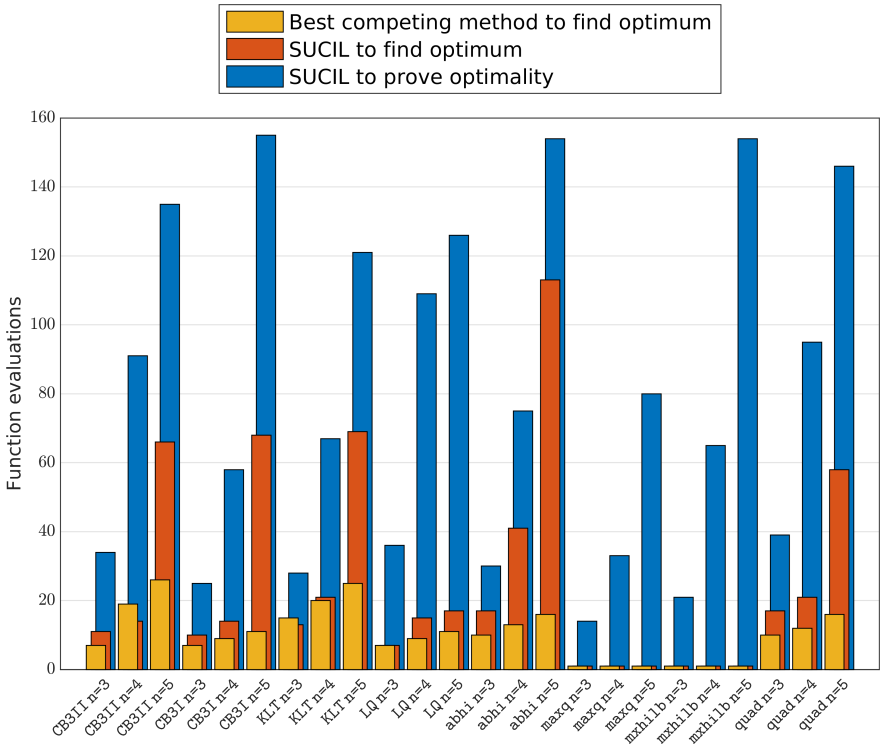



Figure 8: Number of function evaluations before SUCIL first identifies a global minimum and evaluations required to prove its global optimality. The fewest number of evaluations required by any of DFLINT, DFLINT-M, NOMAD, NOMAD-NM, and MATSuMoTo is shown for comparison.

Our computational developments expose a number of fundamental challenges for integer derivative-free optimization. The complexity of our piecewise linear model (PLP) is made worse by the fact that each secant function is valid only in the union of $n+1$ cones $\mathcal{U}^{\boldsymbol{i}}$, resulting in conditional cuts. We note that it may not be possible to derive *unconditional cuts*, that is, cuts that are valid in the whole domain, $\Omega$. For example, we might initially consider secants interpolating a convex $f$ at the $n+1$ points $x \in \mathbb{Z}^n$ and $x \pm e_i \in \mathbb{Z}^n$, where for every $i$ we can choose either $+$ or $-$. Such points form a unit simplex that has no integer lattice points in its interior. Consequently, one might suspect that the resulting cut is valid everywhere in $\Omega$. However, the following example shows that the resulting cut is *not* unconditionally valid. Consider $f(x) = x_1^2 - x_1x_2 + x_2^2$ and the set of points $\{[1,1]^{\mathrm{T}}, [0,1]^{\mathrm{T}}, [1,0]^{\mathrm{T}}\}$. It follows that $f(x) = 1$ at these points, and hence the unique interpolating secant function is the constant function, $m(x) = 1$. Now consider the point $x = [0,0]^{\mathrm{T}}$ for which $f(x) = 0$, which is not underestimated by $m(x) = 1$.

In Figure 8 we show the number of function evaluations needed to first evaluate a global minimizer and the additional number of evaluations used to prove it is a global minimizer. As is common, the effort required to certify optimality can be significantly larger than the cost of finding the optimum. In terms of number of function evaluations required, the proof of optimality is even more time consuming. Because the iterations where $X^k$ or $G^k$ is large require checking many potential secant functions, in SUCIL the computational cost of iterations can differ by orders of magnitude as the algorithm progresses.

Although our method provides a practical iterative way to check sufficiency of a set of points (optimality conditions) for a given convex instance, each iteration involves construction and evaluation of a large number of combinations of different $n+1$ points, which limits the scalability of Algorithm 3 in solving instances of higher dimensions. Yet, in our numerical experiments, we observe that only a small fraction of the total cuts evaluated are useful. We call $(c^{\boldsymbol{i}}, b^{\boldsymbol{i}})$ an *updating* cut at an iteration $k$ if there exists an $x \in \Omega_k$ such that $m^{\boldsymbol{i}}(x) > \eta_k(x)$, that is, a cut that improves the lower bound at at least one $x \in \Omega_k$. In addition, if $m^{\boldsymbol{i}}(x) \geq u_k$, we call it a *pruning* cut. A pruning cut helps eliminate points to be considered in the next

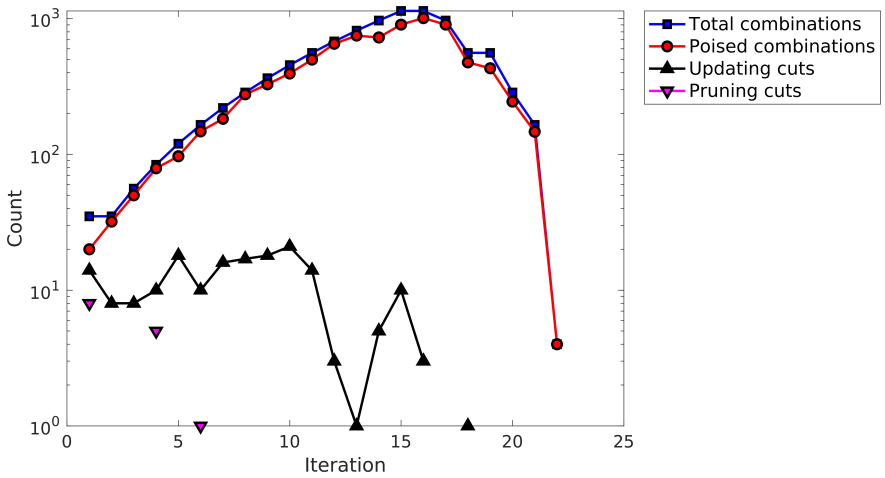



Figure 9: Number of total and poised combinations of $n+1$ points, the secant functions that update, and the secant functions that prune at least one point when minimizing `quad` on $\Omega = [-4,4]^3 \cap \mathbb{Z}^3$ using SUCIL. (Markers are removed when there is no updating or no pruning cut in an iteration.)

iteration ($\Omega_{k+1}$). Figure 9 shows the number of updating and pruning cuts generated per iteration of SUCIL when minimizing `quad` on $[-4,4]^3 \cap \mathbb{Z}^3$. The fact that few cuts prune a point or update the lower bound at any point where the minimum could be suggests that there may be some way to exclude a large set of multi-indices from consideration, possibly yielding dramatic computational savings.

Ideally, we would like to evaluate only the combinations that yield updating or pruning cuts. However, this approach requires the solution of a separate problem that we believe is especially hard to solve. Even the following simpler problem of finding a pruning cut at a given candidate point seems difficult.

**Problem 1.** *Given a point $\bar{x} \in \mathbb{Z}^n$, a set of (integer) points $X$ where $f$ has been evaluated, and scalar $u$, find a multi-index $\boldsymbol{i}$ corresponding to a pruning cut: a $\boldsymbol{i}$ such that $\bar{x} \in \mathcal{U}^{\boldsymbol{i}}$ and $(c^{\boldsymbol{i}})^{\mathrm{T}} \bar{x} + b^{\boldsymbol{i}} \geq u$, and $(c^{\boldsymbol{i}}, b^{\boldsymbol{i}})$ solves* (3)*, or show that no such multi-index $\boldsymbol{i}$ exists.*

If we choose a small subset $\bar{X^k}$ of $X^k$ to form $W(\bar{X^k})$, the SUCIL algorithm can end up using a large number of function evaluations to obtain a certificate of optimality. The reason is that points are evaluated that would be ruled worse than optimal if secants were built by using all combinations of points in $X^k$. This situation occurred when setting $\bar{X^k}$ to be a random subset of $X^k$, a subset of the points closest to $\hat{x}$, or a subset of points with best function values. Using $G^k$ avoids discarding too many points from $X^k$; but we observe a significant increase in $\left|W(G^k)\right|$, and thus we incur heavy computational costs during some iterations. The wall-clock time required per iteration for solving instances of dimension less than $5$ in our setup is not significant, but we present the same for 5-dimensional instances using SUCIL on a 96-core Intel Xeon computer with 1.5 TB of RAM. The complexity of our approach is better quantified by counting the number of combinations of points (or potential secants) considered at iteration $k$. Using $G^k$, we typically produce a strict subset of all possible combinations in such a way that the size of $W(G^k)$ decreases during the later iterations. This is shown in Figure 10: the number of secants added per iteration for all 5-dimensional test instances using SUCIL. Once $\Omega_k$, the number of points with $\eta(x)$ less than $f(\hat{x})$, starts decreasing, so do $G^k$ and $\left|W(G^k)\right|$. In general, it is difficult to predict when the number of combinations (or the wall-clock time curve) would be at the peak, but we suspect this peak will be worse as $n$ increases, by both the size and the iteration number where it occurs. This limits the applicability of the current implementation of SUCIL on higher-dimensional problems.

Again, since nearly all cuts in $W(G^k)$ do not update $\eta(x)$ at any point in $\Omega_k$ (see Figure 9), we believe there may be some approach for intelligently selecting points from $X^k$ using their geometry, their function values, and distance from $\hat{x}$ that will rule some multi-indices $\boldsymbol{i}$ as unnecessary to consider. We did attempt to identify minimal sets of points that were necessary for SUCIL to certify optimality for a variety of $n = 2$ test cases, but no general rule was apparent.

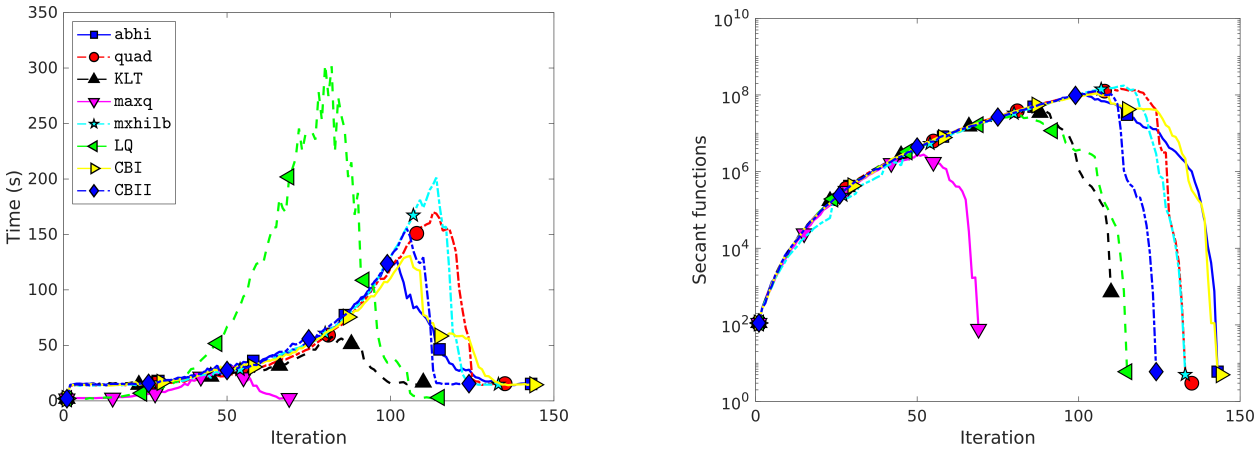



Figure 10: Wall-clock time recorded and number of secants constructed per iteration of SUCIL for 8 convex test problems on $\Omega = [-4, 4]^5 \cap \mathbb{Z}^5$.

We note that the storage requirements for the enumerative model may be prohibitive, even for moderate problem sizes. For example, an array storing the value of $\eta(x)$ as an 8-byte scalar for all $x \in \Omega = [-10, 10]^{10} \cap \mathbb{Z}^{10}$ would require over 200 GB of storage.

Ultimately, we believe further insights are yet to be discovered that will facilitate better algorithms for minimizing convex functions on the integer lattice.

# 8 Benchmark problems

Table 3 shows the set of benchmark problems considered in our paper.

Table 3: Set of convex benchmark problems.

| Name | Expression | $f(x^*)$ | $x^*$ |
|---|---|---|---|
| `CB3II` [11] | $\max\left\{\sum_{i=1}^{n-1} x_i^4 + x_{i+1}^2, \sum_{i=1}^{n-1} (2-x_i)^2 + (2-x_{i+1})^2, \sum_{i=1}^{n-1} 2e^{-x_i+x_{i+1}}\right\}$ | $2(n-1)$ | $e$ |
| `CB3I` [10] | $\sum_{i=1}^{n-1} \max\left\{x_i^4 + x_{i+1}^2, (2-x_i)^2 + (2-x_{i+1})^2, 2e^{-x_i+x_{i+1}}\right\}$ | $2(n-1)$ | $e$ |
| `KLT` [28] | $\max_{i\in\{1,\ldots,n\}}\left\{\|x - c_i - 2e\|^2\right\}, c_i = 2e_i - e$ | $n$ | $2e$ |
| `LQ` [22] | $\sum_{i=1}^{n-1} \max\left\{-x_i - x_{i+1}, -x_i - x_{i+1} + x_i^2 + x_{i+1}^2 - 1\right\}$ | $-(n-1)$ | many |
| `abhi` [1] | $\sum_{i=1}^{n}\left[64\left(c_1(x_i-2) - c_2(x_{i+1}-2)\right)^2 + \left(c_2(x_i-2) - c_1(x_{i+1}-2)\right)^2\right]$, $c_1 = \cos\left(\frac{\pi}{8}\right), c_2 = \sin\left(\frac{\pi}{8}\right)$ | 0 | $2e$ |
| `maxq` [22] | $\max_{i\in\{1,\ldots,n\}}\left\{x_i^2\right\}$ | 0 | 0 |
| `mxhilb` [27] | $\max_{i\in\{1,\ldots,n\}}\left\{\sum_{j=1}^{n}\left\|\frac{x_j}{i+j-1}\right\|\right\}$ | 0 | 0 |
| `quad` | $\sum_{i=1}^{n}(x_i-2)^2$ | 0 | $2e$ |

Table 4: Characteristics of the first 12 instances of (CPF) generated by Algorithm 1 minimizing the convex quadratic function `abhi` on $\Omega = [-2, 2]^3 \cap \mathbb{Z}^3$. (CPF) instances are generated by AMPL and solved by CPLEX; times are the mean of five replications.

| *k* | *sHyp* | *LB* | *UB* | *time* | *simIter* | *nodes* | *bVars* | *cVars* | *cons* | $\hat{x}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 20 | -616.3 | 79.9 | 0.1 | 374 | 0 | 335 | 268 | 960 | $[2; 2; -2]$ |
| 2 | 52 | -555.1 | 79.9 | 3.9 | 13,180 | 8,089 | 847 | 685 | 2,466 | $[2; 2; -1]$ |
| 3 | 100 | -475.2 | 44.7 | 10.9 | 31,423 | 8,555 | 1,615 | 1,310 | 4,724 | $[2; 1; -2]$ |
| 4 | 172 | -434.4 | 44.7 | 7.3 | 19,267 | 1,728 | 2,767 | 2,247 | 8,110 | $[1; 2; -2]$ |
| 5 | 276 | -413.9 | 19.1 | 30.8 | 68,264 | 5,874 | 4,431 | 3,600 | 13,000 | $[2; 1; -1]$ |
| 6 | 418 | -373.1 | 19.1 | 95.1 | 84,031 | 7,933 | 6,703 | 5,447 | 19,676 | $[1; 2; -1]$ |
| 7 | 611 | -311.7 | 19.1 | 59.6 | 83,102 | 5,440 | 9,791 | 7,957 | 28,749 | $[2; -2; -2]$ |
| 8 | 866 | -293.2 | 19.1 | 99.6 | 86,318 | 3,933 | 13,871 | 11,273 | 40,736 | $[1; 1; -2]$ |
| 9 | 1,196 | -232.0 | 19.1 | 154.2 | 84,440 | 4,568 | 19,151 | 15,564 | 56,248 | $[1; 1; -1]$ |
| 10 | 1,532 | -199.5 | 19.1 | 452.3 | 235,473 | 6,400 | 24,527 | 19,933 | 72,042 | $[2; -2; -1]$ |
| 11 | 2,038 | -192.9 | 19.1 | 1,006 | 387,491 | 9,686 | 32,623 | 26,512 | 95,826 | $[2; -1; -2]$ |
| 12 | 2,605 | -140.9 | 19.1 | 964.3 | 455,939 | 29,279 | 41,695 | 33,884 | 122,477 | $[1; -1; -2]$ |

# 9 Performance of MILP model

Table 4 shows the size of the MILP model at each iteration and the computational effort required for solving it. The column *k* refers to the iteration of Algorithm 1, *sHyp* denotes the number of secants (i.e., $|W(X)|$), and *LB* and *UB* give the lower and upper bound on $f$ on $\Omega$, respectively. We show the computational effort needed to solve each MILP via *time*, the mean solution time (in seconds) for 5 replications; *simIter*, the number of simplex iterations; and *nodes*, the number of branch-and-bound nodes explored by the MILP solver. The size of each MILP (after presolve) is shown in terms of *bVars*, the number of binary variables; *cVars*, the number of continuous variables; and *cons*, the number of constraints. Table 4 also shows the optimal solution $\hat{x}$ of each MILP. These experiments were performed by using CPLEX (v.12.6.1.0) on a 2.20 GHz, 12-core Intel Xeon computer with 64 GB of RAM. For this small problem we see that the size of the MILP grows exponentially as the iterations proceed, which results in an exponential growth in solution time as illustrated in Figure 4. The iteration 13 MILP was not solved after 30 minutes.

# 10 Detailed Numerical Results

Tables 5–10 contain detailed numerical results for the interested reader. Note that some solvers do not respect the given budget of function evaluations. We have used a different stopping criterion for MATSuMoTo: it is set to stop only when a point with the optimal value has been identified. Also, although the global minimum is the starting point for `maxq` and `mxhilb`, MATSuMoTo instead uses its initial symmetric Latin hypercube design. The last row of Table 1 in Section 2 shows $|\Omega|$ for these problems.

Table 5: Number of function evaluations before solvers terminate for $n = 3$ test problems. Parentheses show number of evaluations until $x^*$ is evaluated.

| | SUCIL-ideal1 | SUCIL | DFLINT | DFLINT-M | NOMAD | NOMAD-NM | MATSuMoTo |
|---|---|---|---|---|---|---|---|
| `abhi` | 19 (8) | 30 (17) | 161 (57) | 150 (10) | 59 (20) | 129 (56) | 60 (31) |
| `quad` | 25 (8) | 39 (17) | 157 (54) | 150 (10) | 53 (18) | 119 (35) | 45 (16) |
| `KLT` | 22 (8) | 28 (13) | 152 (48) | 146 (15) | 51 (24) | 116 (34) | 46 (17) |
| `maxq` | 14 (1) | 14 (1) | 181 (1) | 181 (1) | 34 (1) | 107 (1) | 50 (21) |
| `mxhilb` | 16 (1) | 21 (1) | 181 (1) | 181 (1) | 35 (1) | 106 (1) | 48 (19) |
| `LQ` | 18 (8) | 36 (7) | 182 (7) | 181 (7) | 48 (7) | 107 (7) | 41 (12) |
| `CB3I` | 22 (8) | 25 (10) | 184 (22) | 181 (7) | 56 (12) | 108 (12) | 60 (31) |
| `CB3II` | 22 (8) | 34 (11) | 184 (22) | 181 (7) | 44 (12) | 108 (12) | 60 (31) |

Table 6: Number of function evaluations before solvers terminate for $n = 4$ test problems. Parentheses show number of evaluations until $x^*$ is evaluated.

| | SUCIL-ideal1 | SUCIL | DFLINT | DFLINT-M | NOMAD | NOMAD-NM | MATSuMoTo |
|---|---|---|---|---|---|---|---|
| abhi | 45 (10) | 75 (41) | 993 (137) | 959 (13) | 110 (16) | 453 (88) | 89 (60) |
| quad | 51 (10) | 95 (21) | 982 (124) | 959 (13) | 111 (12) | 460 (89) | 56 (25) |
| KLT | 51 (10) | 67 (21) | 954 (141) | 953 (20) | 116 (42) | 457 (55) | 54 (23) |
| maxq | 30 (1) | 33 (1) | 1,001 (1) | 1,001 (1) | 91 (1) | 451 (1) | 89 (60) |
| mxhilb | 32 (1) | 65 (1) | 1,001 (1) | 1,001 (1) | 90 (1) | 450 (1) | 63 (34) |
| LQ | 39 (10) | 109 (15) | 1,000 (17) | 1,001 (9) | 145 (9) | 453 (10) | 47 (18) |
| CB3I | 50 (10) | 58 (14) | 1,001 (58) | 1,001 (9) | 149 (42) | 456 (23) | 126 (81) |
| CB3II | 48 (10) | 91 (14) | 1,001 (50) | 1,001 (19) | 125 (30) | 460 (35) | 131 (76) |

Table 7: Number of function evaluations before solvers terminate for $n = 5$ test problems. Parentheses show number of evaluations until $x^*$ is evaluated.

| | SUCIL-ideal1 | SUCIL | DFLINT | DFLINT-M | NOMAD | NOMAD-NM | MATSuMoTo |
|---|---|---|---|---|---|---|---|
| abhi | 105 (12) | 154 (113) | 1,001 (167) | 1,000 (16) | 301 (41) | 1,000 (156) | 214 (157) |
| quad | 108 (12) | 146 (58) | 1,001 (186) | 1,000 (16) | 286 (26) | 1,000 (58) | 113 (62) |
| KLT | 108 (12) | 121 (69) | 1,001 (193) | 1,001 (25) | 296 (43) | 1,000 (176) | 108 (77) |
| maxq | 75 (1) | 80 (1) | 1,001 (1) | 1,001 (1) | 257 (1) | 1,000 (1) | 284 (255) |
| mxhilb | 96 (1) | 154 (1) | 1,001 (1) | 1,001 (1) | 257 (1) | 1,000 (1) | 131 (102) |
| LQ | 83 (12) | 126 (17) | 1,001 (44) | 1,001 (11) | 425 (15) | 1,000 (15) | 56 (27) |
| CB3I | 114 (12) | 155 (68) | 1,001 (103) | 1,001 (11) | 417 (18) | 1,000 (18) | 266 (237) |
| CB3II | 100 (12) | 135 (66) | 1,001 (130) | 1,001 (26) | 465 (69) | 1,000 (54) | 281 (224) |

Table 8: Number of function evaluations taken by 20 replications of MATSuMoTo for each of the 8 convex test problems for $n = 3$.

| | abhi | quad | KLT | maxq | mxhilb | LQ | CB3I | CB3II |
|---|---|---|---|---|---|---|---|---|
| 1 | 48 (19) | 48 (19) | 48 (19) | 43 (14) | 48 (19) | 43 (14) | 59 (30) | 83 (54) |
| 2 | 63 (34) | 44 (15) | 44 (15) | 48 (19) | 44 (15) | 43 (14) | 43 (14) | 54 (25) |
| 3 | 44 (15) | 43 (14) | 45 (16) | 48 (19) | 49 (20) | 43 (14) | 48 (19) | 106(77) |
| 4 | 58 (29) | 43 (14) | 48 (19) | 53 (24) | 64 (35) | 39 (10) | 48 (19) | 63 (34) |
| 5 | 79 (50) | 48 (19) | 49 (20) | 54 (25) | 44 (15) | 38 (9) | 53 (24) | 53 (24) |
| 6 | 73 (44) | 37 (9) | 48 (19) | 39 (10) | 53 (24) | 39 (10) | 64 (35) | 53 (24) |
| 7 | 88 (59) | 43 (14) | 44 (15) | 58 (29) | 49 (20) | 44 (15) | 54 (25) | 59 (30) |
| 8 | 60 (31) | 48 (19) | 43 (14) | 38 (9) | 43 (14) | 44 (15) | 73 (44) | 73 (44) |
| 9 | 79 (50) | 43 (14) | 48 (19) | 53 (24) | 43 (14) | 39 (10) | 68 (39) | 58 (29) |
| 10 | 65 (36) | 53 (24) | 49 (20) | 53 (24) | 49 (20) | 44 (15) | 49 (20) | 43 (14) |
| 11 | 63 (34) | 48 (19) | 48 (19) | 48 (19) | 58 (29) | 43 (14) | 63 (34) | 48 (19) |
| 12 | 53 (24) | 44 (15) | 43 (14) | 48 (19) | 48 (19) | 38 (9) | 58 (29) | 53 (24) |
| 13 | 48 (19) | 49 (20) | 48 (19) | 54 (25) | 49 (20) | 53 (24) | 68 (39) | 37 (9) |
| 14 | 43 (14) | 48 (19) | 48 (19) | 43 (14) | 49 (20) | 43 (14) | 78 (49) | 49 (20) |
| 15 | 44 (15) | 48 (19) | 48 (19) | 48 (19) | 48 (19) | 43 (14) | 39 (10) | 73 (44) |
| 16 | 58 (29) | 48 (19) | 43 (14) | 58 (29) | 37 (9) | 38 (9) | 58 (29) | 73 (44) |
| 17 | 48 (19) | 44 (15) | 43 (14) | 53 (24) | 49 (20) | 37 (9) | 68 (39) | 48 (19) |
| 18 | 64 (35) | 43 (14) | 48 (19) | 63 (34) | 53 (24) | 38 (9) | 93 (64) | 69 (40) |
| 19 | 74 (45) | 43 (14) | 48 (19) | 58 (29) | 53 (24) | 37 (9) | 68 (39) | 63 (34) |
| 20 | 49 (20) | 43 (14) | 43 (14) | 58 (29) | 48 (19) | 44 (15) | 54 (25) | 58 (29) |
| $\lfloor$mean$\rfloor$ | **60 (31)** | **45 (16)** | **46 (17)** | **50 (21)** | **48 (19)** | **41 (12)** | **60 (31)** | **60 (31)** |

Table 9: Number of function evaluations taken by 20 replications of MATSuMoTo for each of the 8 convex test problems for $n = 4$.

| | abhi | quad | KLT | maxq | mxhilb | LQ | CB3I | CB3II |
|---|---|---|---|---|---|---|---|---|
| 1 | 110 (81) | 56 (27) | 51 (22) | 66 (37) | 60 (31) | 41 (12) | 230 (20) | 85 (56) |
| 2 | 70 (41) | 55 (26) | 60 (31) | 120 (91) | 60 (31) | 46 (17) | 76 (47) | 85 (56) |
| 3 | 50 (21) | 50 (21) | 50 (21) | 94 (65) | 50 (21) | 41 (12) | 150 (121) | 196 (11) |
| 4 | 104 (75) | 50 (21) | 50 (21) | 65 (36) | 80 (51) | 45 (16) | 110 (81) | 186 (16) |
| 5 | 91 (62) | 60 (11) | 55 (26) | 112 (83) | 80 (51) | 41 (12) | 100 (11) | 161 (132) |
| 6 | 66 (37) | 55 (26) | 56 (27) | 90 (61) | 56 (27) | 55 (26) | 101 (72) | 196 (167) |
| 7 | 86 (57) | 65 (36) | 45 (16) | 65 (36) | 50 (21) | 46 (17) | 90 (61) | 141 (112) |
| 8 | 90 (61) | 55 (26) | 65 (11) | 81 (52) | 65 (36) | 41 (12) | 105 (63) | 346 (161) |
| 9 | 166 (137) | 56 (27) | 56 (27) | 55 (26) | 50 (21) | 51 (22) | 96 (67) | 100 (71) |
| 10 | 85 (56) | 65 (21) | 55 (11) | 60 (31) | 65 (36) | 55 (26) | 55 (26) | 106 (77) |
| 11 | 147 (118) | 70 (41) | 46 (17) | 70 (41) | 65 (36) | 46 (17) | 95 (66) | 80 (51) |
| 12 | 96 (67) | 45 (16) | 46 (17) | 112 (83) | 71 (42) | 50 (21) | 112 (83) | 231 (202) |
| 13 | 39 (11) | 60 (31) | 71 (42) | 119 (90) | 71 (42) | 41 (12) | 50 (21) | 81 (52) |
| 14 | 85 (56) | 61 (32) | 50 (21) | 85 (56) | 55 (26) | 55 (26) | 205 (176) | 55 (26) |
| 15 | 101 (72) | 60 (31) | 60 (31) | 94 (65) | 55 (26) | 41 (12) | 160 (131) | 114 (85) |
| 16 | 66 (37) | 51 (22) | 45 (16) | 50 (21) | 75 (46) | 41 (12) | 65 (36) | 65 (36) |
| 17 | 76 (47) | 65 (36) | 56 (27) | 55 (26) | 60 (31) | 60 (31) | 85 (56) | 131 (102) |
| 18 | 70 (41) | 56 (27) | 51 (22) | 110 (81) | 75 (46) | 65 (36) | 314 (285) | 70 (41) |
| 19 | 81 (52) | 56 (11) | 60 (31) | 221 (192) | 71 (42) | 41 (12) | 117 (16) | 141 (52) |
| 20 | 105 (76) | 46 (17) | 60 (31) | 66 (37) | 50 (21) | 56 (27) | 210 (13) | 50 (21) |
| $\lfloor$mean$\rfloor$ | **89 (60)** | **56 (25)** | **54 (23)** | **89 (60)** | **63 (34)** | **47 (18)** | **126 (81)** | **131 (76)** |

Table 10: Number of function evaluations taken by 20 replications of MATSuMoTo for each of the $8$ convex test problems for $n = 5$.

| | abhi | quad | KLT | maxq | mxhilb | LQ | CB3I | CB3II |
|---|---|---|---|---|---|---|---|---|
| 1 | 472 (443) | 82 (53) | 83 (54) | 137 (108) | 149 (120) | 52 (23) | 199 (170) | 83 (54) |
| 2 | 358 (329) | 83 (54) | 317 (288) | 229 (200) | 208 (179) | 63 (34) | 267 (238) | 518 (489) |
| 3 | 193 (164) | 83 (54) | 172 (143) | 115 (86) | 175 (146) | 62 (33) | 377 (348) | 1,626 (133) |
| 4 | 194 (165) | 73 (44) | 62 (33) | 171 (142) | 72 (43) | 57 (28) | 223 (194) | 325 (296) |
| 5 | 160 (131) | 72 (43) | 73 (44) | 504 (475) | 87 (58) | 53 (24) | 835 (806) | 487 (458) |
| 6 | 396 (367) | 187 (127) | 78 (49) | 92 (63) | 152 (123) | 53 (24) | 127 (98) | 196 (18) |
| 7 | 82 (53) | 82 (53) | 120 (91) | 135 (106) | 247 (218) | 62 (33) | 184 (155) | 603 (574) |
| 8 | 62 (33) | 106 (15) | 58 (29) | 969 (940) | 186 (157) | 52 (23) | 164 (135) | 67 (38) |
| 9 | 107 (78) | 83 (54) | 68 (39) | 276 (247) | 103 (74) | 57 (28) | 399 (370) | 671 (642) |
| 10 | 186 (13) | 78 (18) | 73 (44) | 352 (323) | 82 (53) | 73 (44) | 364 (335) | 205 (18) |
| 11 | 181 (152) | 130 (65) | 62 (33) | 255 (226) | 53 (24) | 58 (29) | 414 (385) | 240 (211) |
| 12 | 239 (210) | 115 (86) | 125 (96) | 403 (374) | 181 (152) | 47 (18) | 83 (54) | 462 (433) |
| 13 | 295 (211) | 87 (18) | 72 (43) | 167 (138) | 151 (122) | 48 (19) | 128 (99) | 324 (33) |
| 14 | 354 (325) | 78 (49) | 72 (43) | 236 (207) | 82 (53) | 62 (33) | 200 (171) | 154 (125) |
| 15 | 113 (13) | 73 (44) | 52 (14) | 512 (483) | 88 (59) | 53 (24) | 271 (242) | 87 (58) |
| 16 | 227 (23) | 82 (53) | 120 (91) | 189 (160) | 93 (64) | 68 (39) | 326 (297) | 63 (34) |
| 17 | 102 (73) | 135 (106) | 270 (241) | 195 (166) | 57 (28) | 58 (29) | 180 (151) | 599 (570) |
| 18 | 139 (110) | 268 (80) | 150 (121) | 173 (144) | 112 (83) | 58 (29) | 228 (199) | 112 (83) |
| 19 | 169 (19) | 67 (38) | 72 (43) | 108 (79) | 117 (88) | 42 (13) | 204 (175) | 93 (64) |
| 20 | 268 (239) | 301 (195) | 62 (13) | 465 (436) | 237 (208) | 53 (24) | 162 (133) | 180 (151) |
| $\lfloor$mean$\rfloor$ | **214 (157)** | **113 (62)** | **108 (77)** | **284 (255)** | **131 (102)** | **56 (27)** | **266 (237)** | **281 (224)** |

## Acknowledgments

We are grateful to Eric Ni for his insights on derivative-free algorithms for unrelaxable integer variables. Sven Leyffer also wishes to acknowledge the insightful discussions on an early draft of this work during the Dagstuhl seminar 18081. This material is based upon work supported by the applied mathematics and SciDAC activities of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

## References

[1] K. ABHISHEK, S. LEYFFER, AND J. T. LINDEROTH, *Modeling without categorical variables: A mixed-integer nonlinear program for the optimization of thermal insulation systems*, Optimization and Engineering, 11 (2010), pp. 185–212, https://doi.org/10.1007/s11081-010-9109-z.

[2] M. ABRAMSON, C. AUDET, J. CHRISSIS, AND J. WALSTON, *Mesh adaptive direct search algorithms for mixed variable optimization*, Optimization Letters, 3 (2009), pp. 35–47, https://doi.org/10.1007/s11590-008-0089-2.

[3] M. A. ABRAMSON, C. AUDET, G. COUTURE, J. E. DENNIS JR, S. LE DIGABEL, AND C. TRIBES, *The NOMAD project*, 2014, https://www.gerad.ca/nomad.

[4] C. AUDET AND J. E. DENNIS, JR., *Pattern search algorithms for mixed variable programming*, SIAM Journal on Optimization, 11 (2000), pp. 573–594, https://doi.org/10.1137/S1052623499352024.

[5] C. AUDET, S. LE DIGABEL, AND C. TRIBES, *The mesh adaptive direct search algorithm for granular and discrete variables*, Tech. Report 6526, Optimization Online, 2018, http://www.optimization-online.org/DB_HTML/2018/03/6526.html.

[6] P. BALAPRAKASH, A. TIWARI, AND S. M. WILD, *Multi-objective optimization of HPC kernels for performance, power, and energy*, in High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation, S. A. Jarvis, S. A. Wright, and S. D. Hammond, eds., vol. 8551, Springer, 2014, pp. 239–260, https://doi.org/10.1007/978-3-319-10214-6_12.

[7] T. BARTZ-BEIELSTEIN AND M. ZAEFFERER, *Model-based methods for continuous and discrete global optimization*, Applied Soft Computing, 55 (2017), pp. 154–167, https://doi.org/10.1016/j.asoc.2017.01.039.

[8] P. BONAMI, L. BIEGLER, A. CONN, G. CORNUÉJOLS, I. GROSSMANN, C. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, 5 (2008), pp. 186–204, https://doi.org/10.1016/j.disopt.2006.10.011.

[9] C. BUCHHEIM, R. KUHLMANN, AND C. MEYER, *Combinatorial optimal control of semilinear elliptic PDEs*, Computational Optimization and Applications, 70 (2018), pp. 641–675, https://doi.org/10.1007/s10589-018-9993-2.

[10] C. CHARALAMBOUS AND J. W. BANDLER, *Non-linear minimax optimization as a sequence of least pth optimization with finite values of p*, International Journal of Systems Science, 7 (1976), pp. 377–391, https://doi.org/10.1080/00207727608941924.

[11] C. CHARALAMBOUS AND A. R. CONN, *An efficient method to solve the minimax problem directly*, SIAM Journal on Numerical Analysis, 15 (1978), pp. 162–187, https://doi.org/10.1137/0715011.

[12] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Introduction to Derivative-Free Optimization*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics, 2009.

[13] A. COSTA AND G. NANNICINI, *RBFOpt: An open-source library for black-box optimization with costly function evaluations*, Mathematical Programming Computation, 10 (2018), pp. 597–629.

[14] E. DAVIS AND M. IERAPETRITOU, *A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions*, Journal of Global Optimization, 43 (2009), pp. 191–205, https://doi.org/10.1007/s10898-007-9217-2.

[15] J. E. DENNIS, JR. AND D. J. WOODS, *Optimization on microcomputers: The Nelder-Mead simplex algorithm*, in New Computing Environments: Microcomputers in Large-Scale Computing, A. Wouk, ed., SIAM, 1987, pp. 116–122.

[16] E. D. DOLAN AND J. MORÉ, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213, https://doi.org/10.1007/s101070100263.

[17] M. A. DURAN AND I. E. GROSSMANN, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, 36 (1986), pp. 307–339, https://doi.org/10.1007/bf02592064.

[18] R. FLETCHER AND S. LEYFFER, *Solving mixed integer nonlinear programs by outer approximation*, Mathematical Programming, 66 (1994), pp. 327–349, https://doi.org/10.1007/bf01581153.

[19] U. M. GARCÍA-PALOMARES AND P. S. RODRÍGUEZ-HERNÁNDEZ, *Unified approach for solving box-constrained models with continuous or discrete variables by non monotone direct search methods*, Optimization Letters, 13 (2019), pp. 95–111, https://doi.org/10.1007/s11590-018-1253-y.

[20] A. M. GEOFFRION AND R. E. MARSTEN, *Integer programming algorithms: A framework and state-of-the-art survey*, Management Science, 18 (1972), pp. 465–491, https://doi.org/10.1287/mnsc.18.9.465.

[21] P. A. GRAF AND S. BILLUPS, *MDTri: Robust and efficient global mixed integer search of spaces of multiple ternary alloys*, Computational Optimization and Applications, 68 (2017), pp. 671–687, https://doi.org/10.1007/s10589-017-9922-9.

[22] M. HAARALA, K. MIETTINEN, AND M. M. MÄKELÄ, *New limited memory bundle method for large-scale nonsmooth optimization*, Optimization Methods and Software, 19 (2004), pp. 673–692, https://doi.org/10.1080/10556780410001689225.

[23] T. HEMKER, K. FOWLER, M. FARTHING, AND O. VON STRYK, *A mixed-integer simulation-based optimization approach with surrogate functions in water resources management*, Optimization and Engineering, 9 (2008), pp. 341–360, https://doi.org/10.1007/s11081-008-9048-0.

[24] R. HEMMECKE, M. KÖPPE, J. LEE, AND R. WEISMANTEL, *Nonlinear integer programming*, in 50 Years of Integer Programming 1958–2008, Springer, 2010, pp. 561–618, https://doi.org/10.1007/978-3-540-68279-0_15.

[25] K. HOLMSTRÖM, N.-H. QUTTINEH, AND M. EDVALL, *An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization*, Optimization and Engineering, 9 (2008), pp. 311–339, https://doi.org/10.1007/s11081-008-9037-3.

[26] N. JIAN, S. G. HENDERSON, AND S. R. HUNTER, *Sequential detection of convexity from noisy function evaluations*, in Proceedings of the Winter Simulation Conference, IEEE, 2014, https://doi.org/10.1109/wsc.2014.7020215.

[27] K. C. KIWIEL, *An ellipsoid trust region bundle method for nonsmooth convex minimization*, SIAM Journal on Control and Optimization, 27 (1989), pp. 737–757, https://doi.org/10.1137/0327039.

[28] T. G. KOLDA, R. M. LEWIS, AND V. J. TORCZON, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482, https://doi.org/10.1137/S003614450242889.

[29] S. LE DIGABEL AND S. M. WILD, *A taxonomy of constraints in black-box simulation-based optimization*, Preprint ANL/MCS-P5350-0515, Argonne, 2015-01, http://www.mcs.anl.gov/papers/P5350-0515.pdf.

[30] G. LIUZZI, S. LUCIDI, AND F. RINALDI, *Derivative-free methods for bound constrained mixed-integer optimization*, Computational Optimization and Applications, 53 (2011), pp. 505–526, https://doi.org/10.1007/s10589-011-9405-3.

[31] G. LIUZZI, S. LUCIDI, AND F. RINALDI, *Derivative-free methods for mixed-integer constrained optimization problems*, Journal of Optimization Theory and Applications, 164 (2015), pp. 933–965, https://doi.org/10.1007/s10957-014-0617-4.

[32] G. LIUZZI, S. LUCIDI, AND F. RINALDI, *An algorithmic framework based on primitive directions and nonmonotone line searches for black box problems with integer variables*, Tech. Report 6471, Optimization Online, 2018, http://www.optimization-online.org/DB_HTML/2018/02/6471.html.

[33] J. MÜLLER, *MATSuMoTo: The MATLAB surrogate model toolbox for computationally expensive black-box global optimization problems*, Tech. Report 1404.4261, arXiv, 2014, https://arxiv.org/abs/1404.4261.

[34] J. MÜLLER, *MISO: Mixed-integer surrogate optimization framework*, Optimization and Engineering, 17 (2016), pp. 177–203, https://doi.org/10.1007/s11081-015-9281-2.

[35] J. MÜLLER, C. A. SHOEMAKER, AND R. PICHÉ, *SO-I: A surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications*, Journal of Global Optimization, 59 (2013), pp. 865–889, https://doi.org/10.1007/s10898-013-0101-y.

[36] J. MÜLLER, C. A. SHOEMAKER, AND R. PICHÉ, *SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems*, Computers & Operations Research, 40 (2013), pp. 1383–1400, https://doi.org/10.1016/j.cor.2012.08.022.

[37] E. NEWBY AND M. M. ALI, *A trust-region-based derivative free algorithm for mixed integer programming*, Computational Optimization and Applications, 60 (2015), pp. 199–229, https://doi.org/10.1007/s10589-014-9660-1.

[38] M. PORCELLI AND P. L. TOINT, *BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables*, ACM Transactions on Mathematical Software, 44 (2017), pp. 1–25, https://doi.org/10.1145/3085592.

[39] K. RASHID, S. AMBANI, AND E. CETINKAYA, *An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization*, Engineering Optimization, 45 (2012), pp. 185–206, https://doi.org/10.1080/0305215X.2012.665450.

[40] P. RICHTER, E. ÁBRAHÁM, AND G. MORIN, *Optimisation of concentrating solar thermal power plants with neural networks*, in Adaptive and Natural Computing Algorithms, A. Dobnikar, U. Lotrič, and B. Šter, eds., vol. 6593, Springer, 2011, pp. 190–199, https://doi.org/10.1007/978-3-642-20282-7_20.