An Effective Branch-and-Bound Algorithm for Convex Quadratic Integer Programming *

Christoph Buchheim[†], Alberto Caprara[‡], and Andrea Lodi[§]

Abstract

We present a branch-and-bound algorithm for minimizing a convex quadratic objective function over integer variables subject to convex constraints. In a given node of the enumeration tree, corresponding to the fixing of a subset of the variables, a lower bound is given by the continuous minimum of the restricted objective function. We improve this bound by exploiting the integrality of the variables using suitably-defined lattice-free ellipsoids. Experiments show that our approach is very fast on both unconstrained problems and problems with box constraints. The main reason is that all expensive calculations can be done in a preprocessing phase, while a single node in the enumeration tree can be processed in linear time in the problem dimension.

1 Introduction

Nonlinear integer optimization has attracted a lot of attention recently. Besides its practical importance, this class of problems is challenging from a theoretical and methodological point of view. While intensive research has led to tremendous progress in the practical solution of integer *linear* programs in the last decades [12], practical methods for the nonlinear case are still rare [10]. This is true even in special cases, such as *(Strictly) Convex Quadratic Integer Programming* (CQIP):

$$\begin{array}{ll} \min & f(x) := x^\top Q x + L^\top x + c \\ \text{s.t.} & x \in \mathbb{Z}^n \cap X \ , \end{array} \tag{1}$$

where Q is a positive definite $n \times n$ matrix, $L \in \mathbb{R}^n$, $c \in \mathbb{R}$, and $X \subseteq \mathbb{R}^n$ is a convex set for which membership can be tested in polynomial time. Positive definiteness of Q guarantees strict convexity of f.

1.1 The two applications considered

Our original motivating application is in Electronics and arises in the development of pulse coders for actuation, signal synthesis and audio amplification. The aim is to synthesize periodic waveforms

[*] This work was partially supported by the University of Bologna within the OpIMA project. The first author is supported by the DFG under contract BU 2313/1-2. An extended abstract of this paper will appear in the Proceedings of IPCO 2010.

[†] Fakultät für Mathematik, Technische Universität Dortmund and DEIS, Università di Bologna email: christoph.buchheim@tu-dortmund.de

[‡] DEIS, Università di Bologna email: alberto.caprara@unibo.it

[§] DEIS, Università di Bologna email: andrea.lodi@unibo.it

by either bipolar or tripolar pulse codes. In Electronics, this problem is called *Filtered Approximation* (FA) and can be very shortly (and very roughly) summarized as follows for the tripolar case, see [4] for a complete description.

One is given a target signal $f\colon [0, nT] \to \mathbb{R}$ to be approximated by generating a ternary sequence $x_i \in \{-1, 0, +1\}$ for $i = 1, \ldots, n$ and filtering it through a linear causal filter, represented by a function $\tilde{g}\colon \mathbb{R} \to \mathbb{R}$, which is the output of the filter when it receives on input a rectangular function g; see Figure 1. The resulting signal is given by $\sum_{i=1}^{n} x_i \tilde{g}(t - (i-1)T)$ and one aims at minimizing the (square of the) standard deviation from f,

$$\int_{t=0}^{nT} \left(\sum_{i=1}^{n} x_i \tilde{g}(t - (i-1)T) - f(t) \right)^2 dt = x^\top Q x + L^\top x + c\ ,$$

where

$$\begin{aligned} q_{ij} &:= \int_{t=0}^{nT} \tilde{g}(t-(i-1)T) \cdot \tilde{g}(t-(j-1)T) dt\ , \\ L_i &:= -2 \int_{t=0}^{nT} \tilde{g}(t-(i-1)T) \cdot f(t) dt\ , \text{ and} \\ c &:= \int_{t=0}^{nT} f^2(t) dt\ . \end{aligned}$$

See Figures 2 and 3. In other words, we have a CQIP for $X = [-1, 1]^n$, where each variable can take three different values, leading to a *ternary* CQIP. The physical meaning of $x^\top Q x$, representing an energy, ensures that Q is positive definite. By increasing n and decreasing T accordingly, so as to leave nT constant, the approximation gets better and better; see Figure 3.

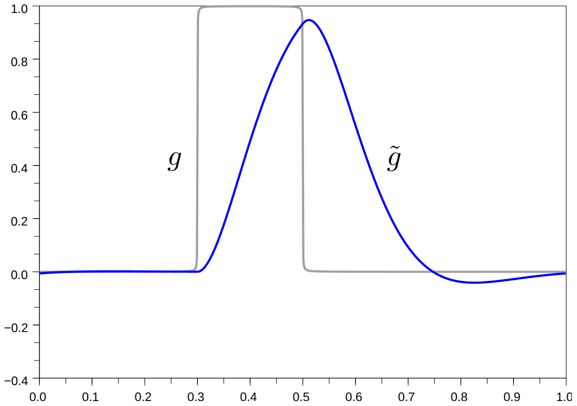


Figure 1: The rectangular function g is transformed into function $\tilde{g}$ by filtering.

Besides the original motivation, if Q is positive definite and $X = \mathbb{R}^n$, CQIP is equivalent to the *Closest Vector Problem* (CVP), which, given a basis $b^1, \ldots, b^n$ of $\mathbb{R}^n$ and an additional vector $v \in \mathbb{R}^n$, calls for an integer linear combination of the vectors of the basis which is as close as possible (with respect to the Euclidean distance) to v. Equivalently, the problem calls for scalars $\lambda_1, \ldots, \lambda_n \in \mathbb{Z}$ such that $\| \sum_{i=1}^{n} \lambda_i b^i - v\|^2$ is minimized. It is easy to check that this amounts to solving (1) for a positive definite matrix $Q = B^\top B$, where B is the $n \times n$ matrix whose columns

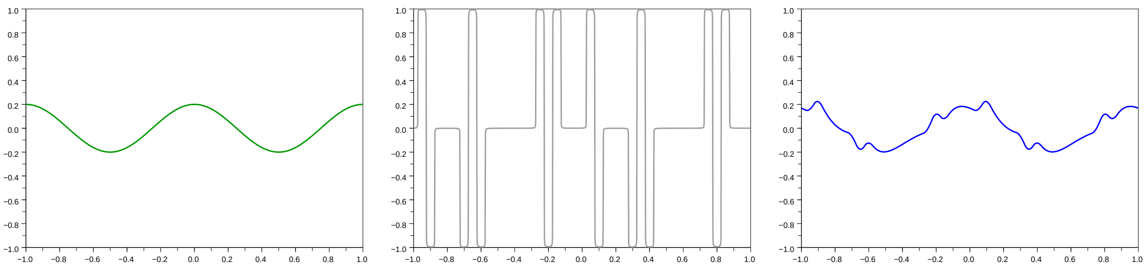


Figure 2: Target signal f (left), optimal sequence x_i for $n = 40$ (middle), and corresponding approximation $\sum_{i=1}^{n} x_i \tilde{g}(t - (i-1)T)$ of f (right).

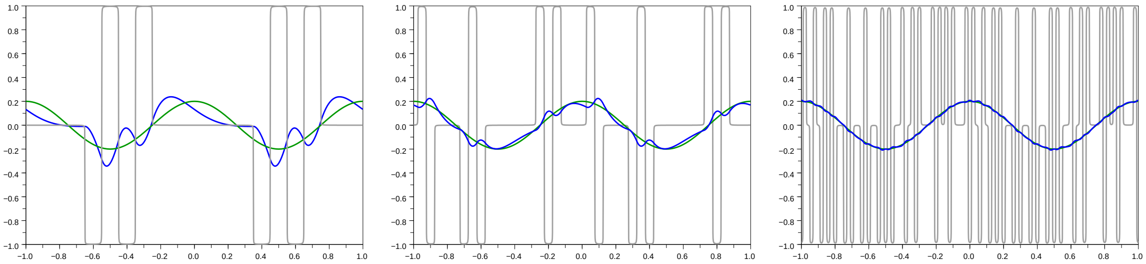


Figure 3: FA for $n = 20$ (left), $n = 40$ (middle), and $n = 100$ (right).

are $b^1, \dots, b^n$. Vice versa, given an instance of (1) in which Q is positive definite and symmetric, a corresponding CVP instance is obtained by computing a Cholesky decomposition $Q = B^\top B$, which yields the basis $b^1, \dots, b^n$, and by defining v accordingly. This problem has a wide relevance from both the theoretical and practical viewpoints. Specifically, it is very hard to approximate and it is used in cryptosystems (see, e.g., [14]).

1.2 Literature review and state-of-the-art

It is clear that the ternary CQIP is closely related to at least two famous combinatorial optimization problems, namely *Unconstrained Binary Quadratic Programming* (UBQP) and *Maximum Cut* (MC). More precisely, UBQP is the special case of CQIP in which $X = [0,1]^n$, while, given a graph $G = (V, E)$ and edge weights $c \in \mathbb{R}^E$, MC is the problem of finding a cut $\delta(W)$ of G having maximum total weight $c(\delta(W))$. Although UBQP and MC have been treated in an almost separate way in the literature, a well-known result in [6] shows that they have in fact the same polyhedral description. Thus, one might be tempted to use the algorithmic technology (as well as the software) available for MC and UBQP in order to solve the ternary case of CQIP as well. We tried three different approaches.

- *Rooted-semimetric MC relaxation.* An early attempt [5] was based on the simple observation that the classical rooted-semimetric MC relaxation is half integral, i.e., has vertex entries in $\{0, \frac{1}{2}, 1\}$, and can be solved efficiently by max-flow techniques [9]. It is not difficult to shift the ternary $\{-1, 0, 1\}$ problem into an equivalent ternary $\{0, \frac{1}{2}, 1\}$ problem and use the rooted relaxation embedded within a branch-and-bound algorithm. Of course, there are $\{0, \frac{1}{2}, 1\}$ solutions which are not feasible but they can be eliminated by branching, still maintaining the combinatorial structure that allows one to solve max-flow type problems. Unfortunately, the results were disappointing because the bound is weak and (based on what we will discuss later) still expensive to compute.

- *Reduction to quadratic 0–1 programming.* Replacing every variable $x_i \in \{-1, 0, 1\}$ by two variables $x_i^+, x_i^- \in \{0, 1\}$ such that $x_i = x_i^+ - x_i^-$, the function $\hat{f}(x^+, x^-) := f(x_1^+ - x_1^-, \dots, x_n^+ - x_n^-)$ is again a quadratic function, defined on $2n$ binary variables. In fact, it is easy to see that $\hat{f}$ is again convex (though not strictly convex). The result is a binary CQIP on $2n$ variables. All IP-based methods we used to address this binary problem suffered from the fact that the matrix Q is dense in our application. SDP-based approaches suffered from the large number of variables and from the fact that the entries of Q are not integer in general, so that subproblems cannot be pruned as quickly as in the integer case. Surprisingly, the best results for all approaches exploiting this reduction are obtained by running CPLEX MIQP on the transformed instance, i.e., without exploiting convexity explicitly. In any case, this is far from being competitive.

- *Convex MINLP approaches.* We also experimented with methods that can handle the ternary problem directly, which in turn were not able to solve instances on 50 variables in time. For Bonmin [3], the branch-and-bound variant `B-BB` is much faster than the outer approximation `B-OA` or the hybrid method `B-Hyb`. Nevertheless, even `B-BB` is way slower than our approach.

Out of all the methods listed above, all of which we tried, the best one turns out to be the direct application of the CPLEX MIQP solver [11] to the original problem, which allows us to solve instances with n up to 50 for our real-world application in Electronics. This is too slow to be of practical use to engineers.

Regarding the second application, besides CPLEX MIQP, we tried the software SHVEC [18], which was designed for solving shortest and closest vector problems, implementing the Fincke-Pohst algorithm [8]. Both methods are able to solve instances with n up to about 55.

1.3 Our contribution

In this paper, we present a branch-and-bound algorithm for CQIP that is very fast at processing nodes but still computes reasonable lower bounds. The main observation behind the method is that, in the unconstrained case $X = \mathbb{R}^n$, strict convexity leads to a unique continuous minimum $\bar{x}$ of f over $\mathbb{R}^n$ which is easy to compute, yielding a lower bound that is of course valid also for any X. This bound can be improved by taking the integrality constraints into account, using lattice-free ellipsoids. Roughly speaking, the general idea is to center a given ellipsoid E in $\bar{x}$ and to compute the value λ such that the scaled ellipsoid λE contains at least one integer point on its border and no integer point in its interior. This can be done quickly if E is chosen appropriately. Then we find the minimum of the function f over the border of λE, which yields an improved lower bound on f.

Our enumeration strategy is depth-first, branching by fixing the value of one of the n variables. A crucial property of our algorithm is that we restrict the order in which variables are fixed. In other words, the set of variables fixed only depends on the depth of the node in the tree. We thus lose the flexibility of choosing the best branching variable, but this strategy allows us to process a single node in the tree much faster. Our main observation is that all expensive calculations to be performed in a node actually only depend on the depth d, i.e., on the set of variables fixed, but not on the particular values to which variables are fixed. This allows us to move these calculations to a preprocessing phase. We will show that, after this preprocessing, the running time per node is only $O(n-d)$, i.e., sublinear in the problem input size which is $\Theta(n^2)$.

Experimentally, we show that our approach leads to the solution of large CQIP instances of Problem (1) for our real-world Filtered Approximation application in Electronics, with n up to about 120, allowing engineers to validate their practical approaches [4]. For the CVP we solve instances with n up to about 70. Even for the largest instances we can solve, our algorithm is able to process around $400,000$ nodes per second.

1.4 Organization of the paper

In Section 2 we discuss our methods for computing lower bounds, explaining how to incorporate constraints into our framework, whereas in Section 3 we present an outline of the overall branch-and-bound algorithm, illustrating how to compute the lower bounds in linear time. In Section 4 we present computational results for our algorithm. Some final remarks are listed in Section 5.

1.5 Basic definitions and notation

We denote scalars by lower case letters, matrices by upper case letters, and vectors by both lower and upper case letters. Given a (column) vector $x \in \mathbb{R}^n$, we will let $x^\top$ denote its transposed (row) vector and x_i its ith component. When x is given by an expression (e.g., a matrix product), we denote its ith component by $(x)_i$. In some cases it will be convenient to use superscripts to indicate vectors, such as x^i, and in this case we will denote the transposed vector by $(x^i)^\top$. As customary, given a matrix Q we will let q_{ij} denote its entry in the ith row and jth column. Given a scalar $a \in \mathbb{R}$, we will let $\lfloor a \rceil$ denote the integer value closest to a, which is uniquely defined if we break ties (if a is half integer) in favor of the value closest to 0. Analogously, given a vector x we will let $\lfloor x \rceil$ denote the result of componentwise rounding of x to the nearest integer.

Let $X \subseteq \mathbb{R}^n$. Given $\alpha \in \mathbb{R}_+$, we let $\alpha X := \{\alpha x : x \in X\}$. Moreover, given $t \in \mathbb{R}^n$, we let $t^\top X := \{t^\top x : x \in X\} \subseteq \mathbb{R}$. A *box* $X \subseteq \mathbb{R}^n$ is a set of the form $X = \{x \in \mathbb{R}^n : l \leq x \leq u\}$, where $l, u \in \mathbb{R}^n$, $l \leq u$. Let Q' be a positive semidefinite matrix. For $x' \in \mathbb{R}^n$ we consider the corresponding *ellipsoid*

$$E(Q', x') := \{x \in \mathbb{R}^n : (x - x')^\top Q'(x - x') \leq 1\} ,$$

which is the translation of $E(Q') := E(Q', 0)$ by the vector x'. Note that we allow Q' to be singular, so that $E(Q', x')$ may be a degenerate ellipsoid. Moreover, for $\alpha \in \mathbb{R}_+$, we let $\alpha E(Q', x')$ denote $E(Q', x')$ scaled by α with respect to the center x', i.e.,

$$\alpha E(Q', x') := x' + \alpha E(Q') = \{x' + \alpha x : x \in E(Q')\} .$$

Given a set X, we let $\operatorname{conv}(X)$ denote its convex hull. Given a closed convex set X, we let $\operatorname{int}(X)$ denote its interior and $\operatorname{bd}(X)$ denote its border.

2 Lower Bounds

As anticipated, the main inspiring observation for our method is that the computation of the minimum of the objective function (neglecting all constraints including integrality) simply requires solving a system of linear equations.

Remark 1 *The unique minimum of $f(x) = x^\top Qx + L^\top x + c$ in case Q is positive definite is attained at $\bar{x} = -\frac{1}{2}Q^{-1}L$ and has value $c - \frac{1}{4}L^\top Q^{-1}L$. Moreover, for every $x \in \mathbb{R}^n$, $f(x) = f(\bar{x}) + (x - \bar{x})^T Q(x - \bar{x})$.*

Our aim in this section is to get stronger bounds by exploiting the integrality of the variables and possibly the structure of X.

2.1 The unconstrained case

For a given $x' \in \mathbb{R}^n$, we let

$$\mu(Q', x') := \sup\{\alpha : \alpha E(Q', x') \cap \mathbb{Z}^n = \emptyset\} = \min\{\alpha : \alpha E(Q', x') \cap \mathbb{Z}^n \neq \emptyset\}$$

be the scaling factor α such that the ellipsoid $\alpha E(Q', x')$ contains some integer point on its border but no integer point in its interior (see for instance the light grey ellipsoid in Figure 4).

Observation 1 $\mu(Q', x') = \min\{\sqrt{(x - x')^\top Q'(x - x')} : x \in \mathbb{Z}^n\} = \max\{\alpha : (x-x')^\top Q'(x-x') \geq \alpha^2$ for all $x \in \mathbb{Z}^n\}$.

Note that, given our objective function $f(x) = x^\top Qx + L^\top x + c$ and the associated continuous minimum $\bar{x}$, the level sets of $f(x)$ are precisely the borders of ellipsoids of the form $\alpha E(Q, \bar{x})$. Given this, it is easy to visualize the fact that finding the integer point that minimizes f is equivalent to scaling $E(Q, \bar{x})$ by α starting from $\alpha = 0$ and stopping as soon as the border of $\alpha E(Q, \bar{x})$ contains an integer point. This is the same as computing $\mu(Q, \bar{x})$. Since this is as hard as solving (1) when $X = \mathbb{R}^n$, we rather do the same scaling for some other ellipsoid $E(Q', \bar{x})$, and then scale $E(Q, \bar{x})$ in turn until it touches the border of the first ellipsoid, see Figure 4. This requires one to be able to compute $\mu(Q', \bar{x})$ as well as the maximum $\alpha \in \mathbb{R}_+$ such that $\alpha E(Q)$ is contained in $E(Q')$:

$$\lambda(Q, Q') := \max\{\alpha : \alpha E(Q) \subseteq E(Q')\} ,$$

noting that this latter value does not depend on $\bar{x}$. By the following observation, $\lambda(Q, Q')$ can be computed by finding the largest α such that all eigenvalues of $Q - \alpha^2 Q'$ are nonnegative.

Observation 2 $\lambda(Q, Q') = \max\{\alpha : x^\top Qx \geq \alpha^2 x^\top Q'x$ for all $x \in \mathbb{R}^n\} = \max\{\alpha : Q - \alpha^2 Q' \succeq 0\} = \min\{1/\sqrt{x^\top Q'x} : x \in E(Q)\}$.

Proof. The first equality is simply a restatement of the definition. As to the second equality, the relation $\alpha E(Q) \subseteq E(Q')$ is equivalent to $\alpha^2 x^\top Q'x \leq 1$ for every $x \in \mathbb{R}^n$ such that $x^\top Qx \leq 1$. Given that both Q and Q' are positive semidefinite, this is equivalent to $\alpha^2 x^\top Q'x \leq x^\top Qx$ for every $x \in \mathbb{R}^n$, i.e., to $x^\top(Q - \alpha^2 Q')x \geq 0$ for every $x \in \mathbb{R}^n$. For the third equality, note that we have $x^\top Qx \geq \alpha^2 x^\top Q'x$ for all $x \in \mathbb{R}^n$ as soon as this holds for all $x \in \mathrm{bd}(E(Q))$. Thus $\lambda(Q, Q') = \max\{\alpha : \alpha^2 x^\top Q'x \leq 1$ for all $x \in \mathrm{bd}(E(Q))\}$, which in turn equals $\min\{1/\sqrt{x^\top Q'x} : x \in \mathrm{bd}(E(Q))\} = \min\{1/\sqrt{x^\top Q'x} : x \in E(Q)\}$. □

Proposition 1 *Given $f(x) = x^\top Qx + L^\top x + c$ with Q positive definite and continuous minimum $\bar{x}$ and a positive semidefinite matrix Q' of the same size as Q,*

$$\min\{f(x) : x \in \mathbb{Z}^n\} \geq f(\bar{x}) + \lambda^2(Q, Q')\mu^2(Q', \bar{x}) .$$

Proof. For every $x \in \mathbb{Z}^n$, using Remark 1 and Observations 1 and 2, we have

$$\begin{aligned} f(x) &= f(\bar{x}) + (x-\bar{x})^\top Q(x-\bar{x}) \\ &\geq f(\bar{x}) + \lambda^2(Q,Q')(x-\bar{x})^\top Q'(x-\bar{x}) \\ &\geq f(\bar{x}) + \lambda^2(Q,Q')\mu^2(Q',\bar{x}) \ . \end{aligned}$$

□

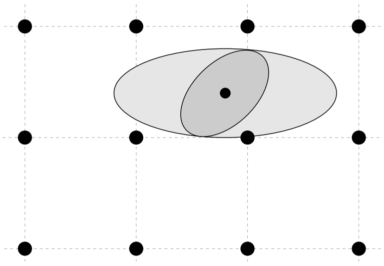


Figure 4: Improving the lower bounds: the light gray ellipsoid is $E(Q',\bar{x})$ scaled by $\mu(Q',\bar{x})$; the dark gray ellipsoid is $E(Q,\bar{x})$ scaled by $\lambda(Q,Q')\mu(Q',\bar{x})$.

Note that, in order to find hopefully strong lower bounds, one would like to have matrices Q' such that on the one hand $E(Q')$ is as similar as possible to $E(Q)$ and on the other hand $\mu(Q',\bar{x})$ is fast to compute. It is particularly fast to compute $\mu(Q',\bar{x})$ if Q' is a *split*, i.e., if $Q' = tt^\top$ for some vector $t \in \mathbb{Z}^n \setminus \{0\}$ with $t_1,\ldots,t_n$ coprime, as the first integer point $x \in \mathbb{Z}^n$ touched when scaling $E(Q',\bar{x})$ is such that $t^\top x = \lfloor t^\top \bar{x} \rceil$. Formally, we have

Observation 3 $\mu(tt^\top, x') = |\lfloor t^\top x' \rceil - t^\top x'|$.

Proof. Recalling Observation 1, for $Q' = tt^\top$ we have that $(x-x')^\top Q'(x-x') \geq \alpha^2$ is equivalent to $|t^\top(x-x')| \geq \alpha$. The claim follows by noting that $|t^\top(x-x')| = |t^\top x - t^\top x'| \geq |\lfloor t^\top x' \rceil - t^\top x'|$ for each $x \in \mathbb{Z}^n$, and that there exists $x \in \mathbb{Z}^n$ such that $t^\top x = \lfloor t^\top x' \rceil$ in case the entries of t are coprime. □

In order to derive strong lower bounds, we aim at splits Q' that yield large factors $\lambda(Q,Q')$. To this end, we consider *flat directions* of the ellipsoid $E(Q)$, i.e., vectors $t \in \mathbb{Z}^n \setminus \{0\}$ minimizing the *width* of $E(Q)$ along t, defined as

$$\max\{t^\top x : x \in E(Q)\} - \min\{t^\top x : x \in E(Q)\} = 2\max\{t^\top x : x \in E(Q)\} \ .$$

Observation 4 *$t \in \mathbb{Z}^n \setminus \{0\}$ maximizes $\lambda(Q, tt^\top)$ if and only if it is a flat direction of $E(Q)$.*

Proof. By Observation 2, $\lambda(Q,tt^\top) = \min\{1/(t^\top x) : x \in E(Q)\}$. Thus t maximizes $\lambda(Q,tt^\top)$ if and only if it maximizes $-\min\{t^\top x : x \in E(Q)\}$, i.e., if it minimizes $\max\{t^\top x : x \in E(Q)\}$. □

The following remark is stated explicitly in, e.g., [7]. For the sake of completeness, we report here an explicit proof.

Remark 2 *If $Q = B^\top B$ for a nonsingular $n \times n$ matrix B, then the width of $E(Q)$ along t is given by $2\|t^\top B^{-1}\|$.*

Proof. The width is given by $2\max\{t^\top x : \|Bx\|^2 \leq 1\} = 2\max\{t^\top B^{-1}y : \|y\|^2 \leq 1\}$, where $y := Bx$. Since y ranges over the unit ball, the maximum is attained when y is parallel to $t^\top B^{-1}$, and has value $2\|t^\top B^{-1}\|$. □

In other words, finding a flat direction of $E(Q)$ is equivalent to finding the coefficients $t_1, \ldots, t_n$ yielding a shortest non-zero vector in the lattice generated by the columns of $(B^{-1})^\top$, which is well known to be NP-hard. A natural heuristic to compute short vectors is obtained by taking as candidates the vectors in a reduced basis of the lattice. Let $t^1, \ldots, t^n \in \mathbb{Z}^n \setminus \{0\}$ be the columns of the corresponding transformation matrix T, from the original basis to the reduced basis, such that $(r^i)^\top = (t^i)^\top B^{-1}$, where r^i is the ith vector in the reduced basis. Recall that T is a unimodular integer matrix, i.e., $\det(T) \in \{-1, 1\}$, and therefore T^{-1} is integer as well. We use the splits $Q'_i := t^i(t^i)^\top$ and compute $\mu(Q'_i, \bar{x}) = |(t^i)^\top \bar{x} - \lfloor (t^i)^\top \bar{x} \rceil|$ as in Observation 3.

Moreover, we consider the matrix $Q'_0 := \sum_{i=1}^n \lambda^2(Q, Q'_i) Q'_i$.

Observation 5 $\mu(Q'_0, x') = \sqrt{\sum_{i=1}^n \lambda^2(Q, Q'_i)\mu^2(Q'_i, x')}$.

Proof. We apply a basis change with respect to $T^\top$. Since $T^{-1}Q'_0(T^{-1})^\top$ is a diagonal matrix, the transformed ellipsoid $T^\top E(Q'_0, x') := E(T^{-1}Q'_0(T^{-1})^\top, T^\top x')$ is axis-parallel. Symmetry implies that the first integer point touched when scaling this ellipsoid is $\lfloor T^\top x' \rceil$. Hence, the first integer point touched when scaling $E(Q'_0, x')$ is $z = (T^\top)^{-1}\lfloor T^\top x' \rceil$. This point is the unique (integer) solution to the equations $(t^i)^\top z = \lfloor (t^i)^\top x' \rceil$ for $i = 1, \ldots, n$. Thus, recalling Observation 1, we have

$$\mu^2(Q'_0, x') = (z - x')^\top Q'_0 (z - x') = \sum_{i=1}^n (z - x')^\top \lambda^2(Q, Q'_i) Q'_i (z - x') = \sum_{i=1}^n \lambda^2(Q, Q'_i)\mu^2(Q'_i, x') ,$$

noting that, by Observation 3, we have $\mu(Q'_i, x') = |(t^i)^\top z - (t^i)^\top x'|$ for $i = 1, \ldots, n$, since $(t^i)^\top z = \lfloor (t^i)^\top x' \rceil$. □

Note that $\lambda(Q, Q'_0)$ can be strictly smaller than one, so that the lower bound derived from Q'_0,

$$f(\bar{x}) + \lambda^2(Q, Q'_0)\mu^2(Q'_0, \bar{x}) = f(\bar{x}) + \lambda^2(Q, Q'_0) \sum_{i=1}^n \lambda^2(Q, Q'_i)\mu^2(Q'_i, \bar{x}) ,$$

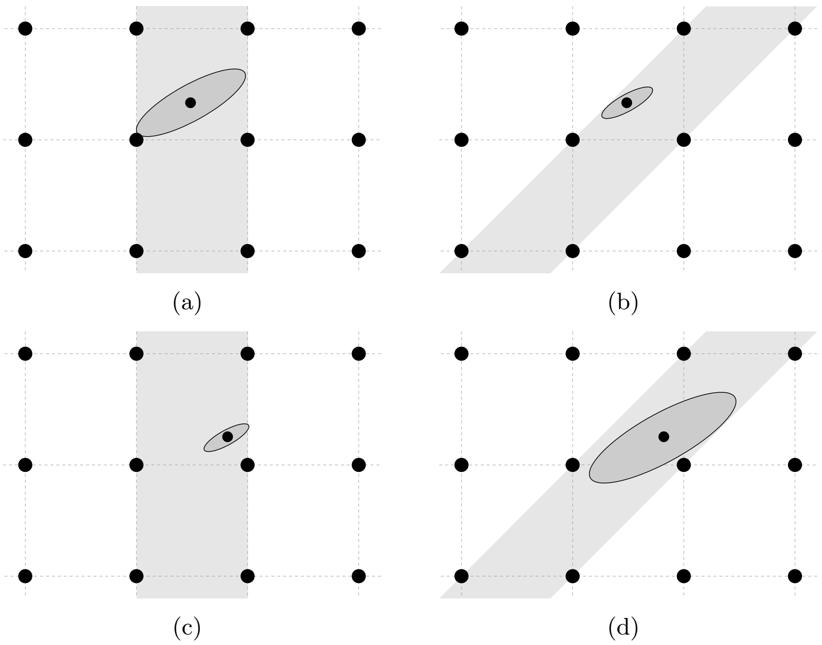
can be weaker than the bound $f(\bar{x}) + \lambda^2(Q, Q'_i)\mu^2(Q'_i, \bar{x})$ derived from Q'_i for some $i \geq 1$. In general, which Q'_i gives the strongest lower bound depends on the position of $\bar{x}$; see Figure 5.

Example We conclude this section by illustrating the above ideas by an example in the plane. Let

$$Q = \begin{pmatrix} 1 & -2 \\ -2 & 8 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -2 & -2 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ 0 & -2 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -2 \\ 0 & -2 \end{pmatrix}, \quad B^{-1} = \begin{pmatrix} 1 & -1 \\ 0 & -1/2 \end{pmatrix} .$$

The ellipse $E(Q)$ is shown in Figure 6. Short vectors of the lattice generated by the rows of B^{-1}, the vectors $(1, -1)^\top$ and $(0, -1/2)^\top$, are $r^1 = (0, -1/2)^\top$ and $r^2 = (1, 0)^\top$. These correspond to the following transformation matrix T:

$$T = \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}, \quad T^{-1} = \begin{pmatrix} -2 & 1 \\ 1 & 0 \end{pmatrix} ,$$



(a)

(b)

(c)

(d)

Figure 5: Depending on the position of $\bar{x}$, different ellipsoids $E(Q'_i)$ give rise to the strongest lower bound. In (a) and (c), the ellipsoid is the split defined by $(1,0)^\top$; in (b) and (d), it is the split defined by $(1,-1)^\top$.

and hence to the (hopefully) flat directions $t^1 = (0,1)^\top$ and $t^2 = (1,-2)^\top$. Thus

$$Q'_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad Q'_2 = \begin{pmatrix} 1 & -2 \\ -2 & 4 \end{pmatrix}$$

and $\lambda(Q,Q'_1) = 2$, $\lambda(Q,Q'_2) = 1$. The ellipses $E(Q'_1)$ and $E(Q'_2)$ are illustrated in Figure 7. Finally, in this case we are lucky to obtain $Q'_0 = 4Q'_1 + Q'_2 = Q$, so that the improved lower bound given by Q'_0 agrees with the optimal solution of the problem, independently of L and c.

In case, for instance, $\bar{x} = (1/2,1/2)^\top$, we have that $\mu(Q'_1,\bar{x}) = \mu(Q'_2,\bar{x}) = 1/2$ and $\mu(Q'_0,\bar{x}) = \sqrt{5}/2$ and the first integer point touched when scaling $E(Q'_0,\bar{x})$ is, e.g., $z = (0,0)^\top$.

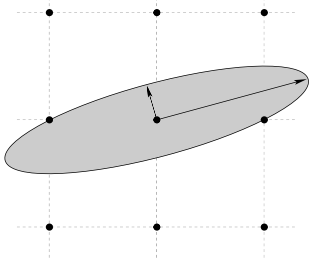


Figure 6: The ellipse $E(Q)$.

□

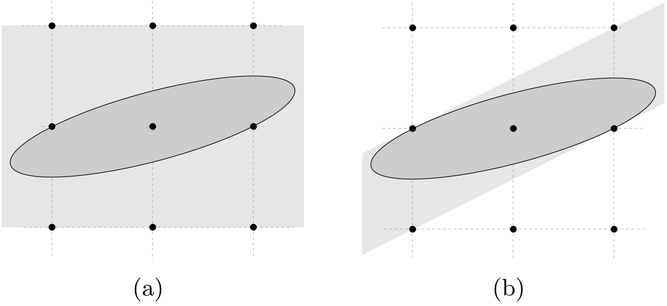


Figure 7: The split $E(Q'_1)$ given by $(0,1)^\top$ in (a); the split $E(Q'_2)$ given by $(1,-2)^\top$ in (b).

We conclude this section by observing that, given $\bar{x}$, the *axis-parallel* ellipsoid yielding the largest lower bound improvement according to Proposition 1 can be found by solving an SDP. Although this does not fit nicely within our branch-and-bound framework, in which we aim at computing bounds very quickly, we report the result due to its simplicity. Recall that an axis-parallel ellipsoid $E(Q')$ arises when $Q = \text{Diag}(t)$ for some $t \in \mathbb{R}^n_+$.

Observation 6 *Given $x' \in \mathbb{R}^n$, the vector $t \in \mathbb{R}^n_+$ such that $\lambda^2(Q, \text{Diag}(t))\mu^2(\text{Diag}(t), x')$ is maximum is given by the solution of the following SDP:*

$$\max \left\{ \sum_{i=1}^n t_i (\lfloor x'_i \rceil - x'_i)^2 : Q \succeq \text{Diag}(t),\ t \geq 0 \right\} .$$

Proof. The SDP can be restated as

$$\max \left\{ \alpha^2 : \sum_{i=1}^n t_i (\lfloor x'_i \rceil - x'_i)^2 = \alpha^2,\ Q \succeq \text{Diag}(t),\ t \geq 0 \right\} .$$

By applying the transformation $s := t/\alpha^2$, this is equivalent to

$$\max \left\{ \alpha^2 : \sum_{i=1}^n s_i (\lfloor x'_i \rceil - x'_i)^2 = 1,\ Q - \alpha^2 \text{Diag}(s) \succeq 0,\ s \geq 0 \right\} .$$

Now, by Observation 2, $Q - \alpha^2 \text{Diag}(s) \succeq 0$ is equivalent to $\alpha \leq \lambda(Q, \text{Diag}(s))$. Moreover, given that the first integer point touched when scaling any axis-parallel ellipsoid centered in x' is $\lfloor x' \rceil$, the constraint $\sum_{i=1}^n s_i (\lfloor x'_i \rceil - x'_i)^2 = 1$ is equivalent to $\mu(\text{Diag}(s), x') = 1$. This proves that the optimal value α^2 of the SDP is such that $\lambda^2(Q, \text{Diag}(s))\mu^2(\text{Diag}(s), x') = \lambda^2(Q, \text{Diag}(t))\mu^2(\text{Diag}(t), x')$ is largest. □

2.2 Dealing with constraints

Even if X is a box, determination of the continuous minimum of f over X is much more complex than in the unconstrained case [15] and would not fit well into our method, which is aimed at exploring branch-and-bound nodes quickly. On the other hand, even if we stick to the computation of the unconstrained continuous minimum, we can take into account the structure of X in the lower bound improvement of Section 2.1.

In particular, we can replace the definition of $\mu(Q',x')$ by

$$\mu_X(Q',x') := \sup\{\alpha : \alpha E(Q',x') \cap X \cap \mathbb{Z}^n = \emptyset\} \ .$$

In other words, we scale $E(Q',x')$ until it touches a feasible point, instead of simply any integer point. The obvious counterpart of Observation 1 is:

Observation 7 $\mu_X(Q',x') = \min\{\sqrt{(x-x')^\top Q'(x-x')} : x \in X \cap \mathbb{Z}^n\} = \max\{\alpha : (x-x')^\top Q'(x-x') \geq \alpha^2$ for all $x \in X \cap \mathbb{Z}^n\}$.

In addition, the counterpart of Proposition 1 reads:

Proposition 2 *Given $f(x) = x^\top Qx + L^\top x + c$ with Q positive definite and continuous minimum $\bar{x}$ and a positive semidefinite matrix Q' of the same size as Q,*

$$\min\{f(x) : x \in X \cap \mathbb{Z}^n\} \geq f(\bar{x}) + \lambda^2(Q,Q')\mu_X^2(Q',\bar{x}) \ .$$

For a split, computation of $\mu_X(tt^\top, x')$ amounts to solving $\min\{|t^\top x - t^\top x'| : x \in X \cap \mathbb{Z}^n\}$, whose objective function can be linearized yielding $\min\{z : z \geq t^\top x - t^\top x',\ z \geq t^\top x' - t^\top x,\ x \in X \cap \mathbb{Z}^n\}$. It is easy to see that calculation of $\mu_X(tt^\top, x')$ is NP-hard even if X is a box (e.g., if $X = \mathbb{R}^n_+$). On the other hand, optimization of $t^\top x$ over $X \cap \mathbb{R}^n$ is trivial in this case. This inspires the following lower bound on $\mu_X(tt^\top, x')$, that can be applied to an arbitrary X whenever we are given a linear optimization oracle for $X \cap \mathbb{Z}^n$.

Observation 8 *Consider $t \in \mathbb{Z}^n$ and let $t_{\min}$ and $t_{\max}$ denote the minimum and maximum of $t^\top x$ over $X \cap \mathbb{Z}^n$, respectively. Then,*

$$\mu_X(tt^\top, x') \geq \max \begin{cases} |t^\top x' - \lfloor t^\top x' \rceil| \\ t^\top x' - t_{\max} \\ t_{\min} - t^\top x' \end{cases} .$$

Proof. By definition, we have $\mu_X(tt^\top, x') \geq \mu(tt^\top, x') = |t^\top x' - \lfloor t^\top x' \rceil|$. In order to show that $\mu_X(tt^\top, x') \geq t^\top x' - t_{\max}$, we may assume $t^\top x' \geq t_{\max}$. Now $t^\top E(tt^\top, x')$ is the interval $[t^\top x' - 1, t^\top x' + 1]$, which may be scaled by exactly $t^\top x' - t_{\max}$ until it touches $t^\top(X \cap \mathbb{Z}^n)$, thus $E(tt^\top, x')$ may be scaled by the same amount until it touches $X \cap \mathbb{Z}^n$. Analogously, one can show $\mu_X(tt^\top, x') \geq t_{\min} - t^\top x'$. □

Note that we have equality in Observation 8 if the hyperplane $t^\top x = t^\top x'$ does not intersect $\text{int}(\text{conv}(X \cap \mathbb{Z}^n))$, in which case the bound is given by $t^\top x' - t_{\max}$ or $t_{\min} - t^\top x'$. Letting Q'_i be defined as in the previous section for $i = 0, \ldots, n$, Observation 8 applies of course to Q'_i with $i = 1, \ldots, n$. As to Q'_0, we have

Observation 9 $\mu_X(Q'_0, x') \geq \sqrt{\sum_{i=1}^n \lambda^2(Q,Q'_i)\mu_X^2(Q'_i,x')}$.

Proof. Let z be the first point in $X \cap \mathbb{Z}^n$ touched when scaling $E(Q'_0, x')$. As in the proof of Observation 5, we then have

$$\mu_X^2(Q'_0, x') = \sum_{i=1}^n \lambda^2(Q,Q'_i)(z-x')^\top Q'_i(z-x') \ ,$$

and the claim follows since $(z-x')^\top Q'_i(z-x') \geq \mu_X^2(Q'_i, x')$ by Observation 7. □

Observe that strict inequality may hold in Observation 9 as there may be a point $z' \in X \cap \mathbb{Z}^n$ that is touched before z when scaling $E(Q'_i, x')$.

Example (continued) In the example of Section 2.1, if $\bar{x} = (1/2, 1/2)^\top$ and $X = \{(0,1)\}$ (a single point), we have $\mu_X(Q'_1, \bar{x}) = 1/2$, $\mu_X(Q'_2, \bar{x}) = 1$ and $\mu_X(Q'_0, \bar{x}) = \sqrt{13}/2 > \sqrt{4 \cdot 1/4 + 1 \cdot 1} = \sqrt{2}$. □

Finally, note that the approach is still correct if the oracle optimizes over X instead of $X \cap \mathbb{Z}^n$, or any set containing $X \cap \mathbb{Z}^n$, but this may lead to weaker bounds.

3 The Branch-and-Bound Algorithm

Our branch-and-bound algorithm adopts a depth-first enumeration strategy. Branching consists of fixing a single variable to an integer value, and is illustrated in detail in the following.

Assume that the next variable to be fixed is x_i. We consider the value $\bar{x}_i$ of x_i in the continuous minimum computed in the current node. We fix x_i to integer values by increasing distance from $\bar{x}_i$. More precisely, if $\lfloor \bar{x}_i \rceil = \lfloor \bar{x}_i \rfloor$, the variable x_i is fixed to

$$\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil, \lfloor \bar{x}_i \rfloor - 1, \lceil \bar{x}_i \rceil + 1, \ldots \tag{2}$$

while otherwise it is fixed to

$$\lceil \bar{x}_i \rceil, \lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil + 1, \lfloor \bar{x}_i \rfloor - 1, \ldots \tag{3}$$

By the convexity of f and its symmetry with respect to $\bar{x}$, the continuous minima with respect to these fixings are non-decreasing, so that we can stop as soon as one of these minima exceeds the current upper bound. In particular, we get a finite algorithm even without bounds on the variables, since we assume that f is strictly convex.

In order to enumerate subproblems as quickly as possible, our aim is to perform the most time-consuming computations in a preprocessing phase. Specifically, having fixed d variables, we get a reduced objective function $\bar{f} : \mathbb{R}^{n-d} \to \mathbb{R}$ of the form

$$\bar{f}(x) = x^\top \bar{Q}_d x + \bar{L}^\top x + \bar{c} \,.$$

If x_i is fixed to r_i for $i = 1, \ldots, d$, we have $\bar{c} = c + \sum_{i=1}^d L_i r_i + \sum_{i=1}^d \sum_{j=1}^d q_{ij} r_i r_j$ and $\bar{L}_{j-d} = L_j + 2\sum_{i=1}^d q_{ij} r_i$ for $j = d+1, \ldots, n$. On the other hand, the matrix $\bar{Q}_d$ is obtained from Q by deleting the first d rows and columns, and therefore is positive definite and does not depend on the values at which the first d variables are fixed.

3.1 Achieving quadratic time per node

For $\bar{Q}_d$, we need the inverse matrix and all scaling factors $\lambda(\bar{Q}_d, Q')$ for the computation of lower bounds. For this reason, we do not change the order of fixing variables, i.e., we always fix the first unfixed variable according to an order that is determined before starting the enumeration. This implies that, in total, we only have to consider n different matrices $\bar{Q}_d$, which we know in advance as soon as the fixing order is determined. If the variables to be fixed were chosen freely, the number of such matrices would be exponential.

See Algorithm 1 for an outline of our method, for the case in which $X = \mathbb{R}^n$ and we simply use the continuous lower bound. Clearly, the running time of this algorithm is exponential in general.

Algorithm 1: Outline of the basic algorithm

```
Input: a strictly-convex function $f\colon \mathbb{R}^n \to \mathbb{R},\ x \mapsto x^\top Q x + L^\top x + c$
Output: a vector $x \in \mathbb{Z}^n$ minimizing $f(x)$
determine a variable order $x_1, \dots, x_n$;
let $\bar{Q}_d$ be the submatrix of $Q$ given by rows and columns $d+1, \dots, n$;
compute the inverse matrices $\bar{Q}_d^{-1}$ for $d = 1, \dots, n$;
set $d := 0$, ub $:= \infty$;
while $d \geq 0$ do
    define $\bar{f} : \mathbb{R}^{n-d} \to \mathbb{R}$ by $\bar{f}(x) := f((r_1, \dots, r_d, x_1, \dots, x_{n-d}))$;
    compute $\bar{L}$ and $\bar{c}$ such that $\bar{f}(x) = x^\top \bar{Q}_d x + \bar{L}^\top x + \bar{c}$;
    // compute lower bound
    compute the continuous minimum $\bar{x} := -\frac{1}{2}(\bar{Q}_d^{-1}\bar{L}) \in \mathbb{R}^{n-d}$ of $\bar{f}$;
    set lb $:= \bar{f}(\bar{x})$;
    // compute upper bound
    set $r_j := \lfloor \bar{x}_{j-d} \rceil$ for $j = d+1, \dots, n$ to form $r \in \mathbb{Z}^n$;
    apply primal heuristics to improve $r$;
    // update solution
    if $\bar{f}((r_{d+1}, \dots, r_n)) <$ ub then
        set $r^* := r$;
        set ub $:= \bar{f}((r_{d+1}, \dots, r_n))$;
    end
    // prepare next node
    if lb $<$ ub then
        // branch on variable $x_{d+1}$
        set $d := d + 1$;
        set $r_d := \lfloor \bar{x}_1 \rceil$;
    else
        // always holds if $d = n$
        // prune current node
        set $d := d - 1$;
        if $d > 0$ then
            // go to next node
            increment $r_d$ according to (2) or (3);
        end
    end
end
```

However, every node in the enumeration tree can be processed in $O(n^2)$ time (if the primal heuristics obey the same runtime bound), the bottleneck being the computation of the continuous minimum given the pre-computed inverse matrix $\bar{Q}_d^{-1}$. Note that Algorithm 1 can easily be adapted to the constrained case where $X \neq \mathbb{R}^n$. In this case, we just prune all nodes with invalid variable fixings.

For the computation of stronger lower bounds as explained in Section 2, at each node we consider the matrices $\bar{Q}'_0, \dots, \bar{Q}'_n$ derived from $\bar{Q}_d$. It is crucial that the values $\lambda(\bar{Q}_d, \bar{Q}'_i)$ can be computed in the preprocessing phase for each depth d and for $i = 0, \ldots, n$. In the unconstrained case, the running time per node is then affected only by the time needed to compute $\mu(\bar{Q}'_i, \bar{x})$. This requires $O(n)$ time for $i = 1, \ldots, n$ by Observation 3 and an additional $O(n)$ time for $i = 0$ by Observation 5, i.e., $O(n^2)$ time in total.

The same applies to the constrained case, where in each node we compute the stronger bounds given by Observations 8 and 9. For this, we determine all t^i_{min} and t^i_{max} in the preprocessing phase by calling the linear optimization oracle $2n^2$ times in total, namely $2n$ times for the t^i vectors associated with each depth of the tree. After that, we only need two additional comparisons for each t^i in order to compute the improved bounds.

The above discussion is summarized in the following proposition.

Proposition 3 *The running time per node in Algorithm 1, with lower bounds improved as illustrated in Section 2, is $O(n^2)$.*

3.2 Improvement to linear time per node

With some adjustments, the running time to process a node of depth d in our algorithm can be decreased from $O(n^2)$ to $O(n-d)$. For this, we have to improve the running time of two different components: the computation of the continuous minima of $\bar{f}$ and the lower bound improvement.

For the computation of the continuous minima, we can replace the $O((n-d)^2)$ method by an incremental technique needing $O(n-d)$ time only, which determines the new continuous minimum from the old one in linear time whenever a new variable is fixed. For this, we exploit the basic observation that in a given node, the continuous minima according to all possible fixings of the next variable lie on a line. Moreover, the direction of this line only depends on which variables have been fixed so far, but not on the values to which they were fixed. This implies that, in our algorithm, the direction of this line is fully determined by the depth of the current node. Additional care has to be taken for computing the objective function values of the continuous minima, since a direct evaluation of the objective function would take quadratic time.

Formally, recall that by $\bar{f}(x) = x^\top \bar{Q}_d x + \bar{L}^\top x + \bar{c}$ we denote the function obtained from f by fixing variable x_i to r_i for $i = 1, \ldots, d$, and let $\bar{x} \in \mathbb{R}^{n-d}$ be the continuous minimum of $\bar{f}$, noting that $\bar{x}_1$ corresponds to the value of the original variable x_{d+1}. Finally, let

$$z^d := (1, -(q_{d+1,d+2}, q_{d+1,d+3}, \ldots, q_{d+1,n})\, \bar{Q}^{-1}_{d+1})^\top \in \mathbb{R}^{n-d} .$$

Observation 10 *If we fix variable x_{d+1} to r_{d+1} and re-optimize $\bar{f}$, the resulting continuous minimum is given by $\bar{x} + (r_{d+1} - \bar{x}_1)z^d$.*

Proof. Let $\bar{L}(u) \in \mathbb{R}^{n-d-1}$ denote the linear term in the objective function after fixing variable x_{d+1} to u and removing the associated component. Then $\bar{L}(u)_{j-d-1} = L_j + 2\sum_{i=1}^{d} q_{i,j} r_i + 2q_{d+1,j}u$ for $j = d+2, \ldots, n$. Moreover, let $\bar{\bar{x}} \in \mathbb{R}^{n-d}$ denote the continuous minimum after fixing variable x_{d+1} to r_{d+1} and keeping the associated component, i.e., $\bar{\bar{x}}_1 = r_{d+1}$. By Remark 1, we have that $\bar{\bar{x}} = (r_{d+1}, -\frac{1}{2}\bar{Q}^{-1}_{d+1}\bar{L}(r_{d+1}))$.

Now observe that, in case $r_{d+1} = \bar{x}_1$, i.e., in case variable x_{d+1} was fixed to its optimal continuous value (putting aside the fact that this may not be integer), then $\bar{\bar{x}} = \bar{x}$, i.e., the continuous minimum would not change. With the current notation, this reads $\bar{x} = (\bar{x}_1, -\frac{1}{2}\bar{Q}^{-1}_{d+1}\bar{L}(\bar{x}_1))$. This implies $\bar{\bar{x}} - \bar{x} = (r_{d+1} - \bar{x}_1, -\frac{1}{2}\bar{Q}^{-1}_{d+1}(\bar{L}(r_{d+1}) - \bar{L}(\bar{x}_1)))$, from which the claim follows by elementary calculations. □

The above discussion implies that, in order to find the continuous minimizer for the nodes generated by branching from a given node, as well as the associated objective function value, we simply have to compute $\bar{x} + \alpha z^d$ and $\bar{f}(\bar{x} + \alpha z^d)$ for a given $\alpha \in \mathbb{R}$. As to the latter, if we define

$$v^d := 2\bar{Q}_d z^d \in \mathbb{R}^{n-d} , \qquad s_d := (z^d)^\top \bar{Q}_d z^d \in \mathbb{R} ,$$

then we get

$$\bar{f}(\bar{x} + \alpha z^d) = \bar{f}(\bar{x}) + \alpha(\bar{x}^\top v^d + \bar{L}^\top z^d) + \alpha^2 s_d .$$

Since $\bar{L}$ can be computed incrementally in $O(n-d)$ time, we get:

Proposition 4 *If, in the preprocessing phase of Algorithm 1, we compute z^d, v^d, s_d as defined above for $d = 0, \dots, n-1$, then the computation of the continuous minimizer and the associated lower bound can be carried out in $O(n-d)$ time per node.*

When improving lower bounds by ellipsoids as illustrated in Section 2, the following natural restriction leads to linear time per node: if the splits in the root node are defined by the columns of the transformation matrix T, then the splits on level d are defined by the columns $\bar{t}_d^1, \dots, \bar{t}_d^{n-d}$ of the matrix $\bar{T}_d$ arising from T by deleting the first d rows and columns. Indeed, in this case, for the continuous minimum $\bar{\bar{x}} = \bar{x} + (r_{d+1} - \bar{x}_1) z^d$ obtained after having fixed variable x_{d+1} to r_{d+1}, we have to compute $\bar{T}_{d+1}^\top(\bar{\bar{x}}_2, \dots, \bar{\bar{x}}_{n-d})^\top$ in order to determine the scaling factors $\mu(\bar{Q}_i', (\bar{\bar{x}}_2, \dots, \bar{\bar{x}}_{n-d})^\top)$ for $i = 1, \dots, n-d-1$, where $\bar{Q}_i' := \bar{t}_{d+1}^i (\bar{t}_{d+1}^i)^\top$. If we define

$$w^{d+1} := \bar{T}_{d+1}^\top (z_2^d, \dots, z_{n-d}^d)^\top \in \mathbb{R}^{n-d-1} ,$$

we have

$$\begin{aligned} \bar{T}_{d+1}^\top(\bar{\bar{x}}_2, \dots, \bar{\bar{x}}_{n-d})^\top &= \bar{T}_{d+1}^\top(\bar{x}_2, \dots, \bar{x}_{n-d})^\top + \bar{T}_{d+1}^\top(r_{d+1} - \bar{x}_1)(z_2^d, \dots, z_{n-d}^d)^\top \\ &= ((\bar{T}_d^\top \bar{x})_2, \dots, (\bar{T}_d^\top \bar{x})_{n-d}) - \bar{x}_1(t_{d+1,d+2}, \dots, t_{d+1,n})^\top + (r_{d+1} - \bar{x}_1) w^{d+1} . \end{aligned}$$

Now $\bar{T}_d^\top \bar{x}$ has already been determined, hence all of the $n-d-1$ factors $\mu(\bar{Q}_i', (\bar{\bar{x}}_2, \dots, \bar{\bar{x}}_{n-d})^\top)$ can be computed in $O(n-d)$ time. After that, the last factor $\mu(\bar{Q}_0', (\bar{\bar{x}}_2, \dots, \bar{\bar{x}}_{n-d})^\top)$, where $\bar{Q}_0' := \sum_{i=1}^{n-d-1} \lambda^2(\bar{Q}_d, \bar{Q}_i') Q_i'$ can be computed in $O(n-d)$ time by Observation 5. In the constrained case, we can determine improved bounds as explained in Section 3.1.

In summary, we thus have

Proposition 5 *If, in the preprocessing phase of Algorithm 1, we compute w^{d+1} and all $t_{\min}^i$ and $t_{\max}^i$ for $d = 0, \dots, n-1$, then the lower bound improvement as illustrated in Section 2 can be carried out in $O(n-d)$ time per node.*

4 Computational Results

In this section, we present experimental results for the two special cases of CQIP mentioned in the introduction, namely the cases $X = [-1, 1]^n$ (Section 4.1) and $X = \mathbb{R}^n$ (Section 4.2). In all cases, we compare our algorithm with the CPLEX MIQP solver [11], which, as mentioned in the introduction, turned out to be by far the best method among those available when we approached the problem. For our algorithm, we also state results obtained without the lower bound improvements discussed in Section 2. We implemented the version running in linear time per node, see Section 3.2. For

instance		trivial	standard basis		reduced basis		best heur
γ	n		improved	best axis-parallel	improved	best axis-parallel	
0.2	50	6.930943e-11	1.108101e-06	1.138748e-06	6.483600e-06	1.507646e-05	1.088225e-04 $\star$
0.2	100	5.657194e-13	6.909366e-08	7.038127e-08	1.124626e-07	6.040816e-07	9.677417e-06 $\star$
0.2	150	<1e-15	1.364557e-08	1.395173e-08	1.544313e-08	8.028670e-08	2.419792e-06
0.2	200	<1e-15	2.292244e-09	2.755499e-09	3.481464e-09	1.998845e-08	9.263187e-07
0.3	50	1.559461e-10	2.493226e-06	2.561368e-06	7.104197e-06	2.043416e-05	1.453055e-04 $\star$
0.3	100	1.272753e-12	1.554607e-07	1.570978e-07	1.279994e-07	6.829632e-07	9.448114e-06 $\star$
0.3	150	<1e-15	3.070253e-08	3.120370e-08	1.294113e-08	7.248821e-08	2.492522e-06
0.3	200	<1e-15	5.157549e-09	5.676386e-09	3.533325e-09	1.987283e-08	1.025988e-06
0.4	50	2.772375e-10	4.432402e-06	4.553818e-06	7.435790e-06	1.641745e-05	1.518196e-04 $\star$
0.4	100	2.262501e-12	2.763746e-07	2.787844e-07	1.113016e-07	4.776112e-07	1.167412e-05 $\star$
0.4	150	<1e-15	5.458228e-08	5.525709e-08	1.452634e-08	8.543101e-08	2.487505e-06
0.4	200	<1e-15	9.168976e-09	9.600037e-09	3.749173e-09	2.093379e-08	9.845103e-07

Table 1: Lower bound values for second order Butterworth filters.

comparison, in the ternary case we also state results for the version using quadratic time per node, see Section 3.1.

In our algorithm, we use two primal heuristics based on the genetic algorithm for quadratic 0–1 programming presented in [13]. For general X, we can apply this binary heuristic locally: if $\bar{x}$ is the current continuous minimum, we consider the box

$$B = \{\lfloor \bar{x}_1 \rfloor, \lfloor \bar{x}_1 \rfloor + 1\} \times \cdots \times \{\lfloor \bar{x}_n \rfloor, \lfloor \bar{x}_n \rfloor + 1\}$$

and try to minimize f over $B \cap X$. Moreover, following the discussion in Section 1.2, in the special case $X = [-1, 1]^n$, every binary heuristic also gives rise to a global heuristic for minimizing f by replacing every variable x_i by $x_i^+ - x_i^-$, where $x^+, x^- \in \{0, 1\}^n$. Both heuristics are only applied once at the beginning.

All experiments were run on Intel Xeon processors running at 2.33 GHz. Our implementation uses BLAS [1] and LAPACK [2] for the main matrix and vector operations. For computing short vectors, we use the Block Korkin-Zolotarev basis reduction algorithm [16] implemented in NTL [17]. The runtime limit for all instances and solution methods was 8 hours. Besides total running times, we investigated the time needed for preprocessing and the time per node in the enumeration tree. Moreover, we state the total number of nodes processed.

4.1 Convex Quadratic Ternary Optimization

In this section, we present the experimental results for the first application illustrated in Section 1.1 for instances corresponding to second order Butterworth filters and a sinusoidal target signal f, whose amplitude is given by $\gamma := \max\{f(t) : t \in [0, nT]\} = -\min\{f(t) : t \in [0, nT]\}$.

We start by reporting in Table 1 the values of the lower bounds found by computing simply the continuous minimum ("trivial"), and improving the bound by, respectively, taking as t^i vectors the n unit vectors in $\mathbb{R}^n$ ("standard basis, improved"), finding the best axis-parallel ellipsoid according to Observation 6 ("standard basis, best axis-parallel"), taking as t^i vectors the n columns of a transformation matrix T yielding a reduced basis ("reduced basis, improved"), and finding the best axis-parallel ellipsoid according to Observation 6 after having applied a basis change with respect to $T^\top$ ("reduced basis, best axis-parallel"). Finally, in the last column we report the value "best heur" of the best solution we could find, marked with a "$\star$" in case it is provably optimal. Note that all values are numbers much smaller than 1.

The table shows that there is a notable improvement (a few orders of magnitude) from the continuous lower bound by applying the tools in Section 2, which would be more significant (sometimes one additional order of magnitude) by finding the best axis-parallel ellipsoid, which is however too expensive to be done (outside the root node) within our branch-and-bound algorithm. On the other hand, the bounds obtained by either taking as T the identity or the transformation matrix of a reduced basis are comparable, with the first one often dominating the second. This indicates that also improving with respect to the standard basis is fine for these instances, which is what we did within branch-and-bound. This leads to slightly better runtime results than using a reduced basis, due to the fact that computing bounds from axis-parallel ellipsoids is faster in practice although its asymptotic running time is the same, dominating the negative effect of having (on average) slightly weaker lower bounds. Finally, and unfortunately, all the lower bounds are a couple of orders of magnitude smaller than the optimum, which makes them essentially useless to certify the quality of a heuristic solution.

In Table 2, we present results for Algorithm 1 using only continuous minima ("trivial bounds"), Algorithm 1 with the lower bound improvement when T is the identity ("improved bounds"), Algorithm 1 with the lower bound improvement when T is the identity ("improved bounds") but with a running time of $O(n^2)$ per node, without the improvement in Section 3.2, ("$O(n^2)$, impr. bounds"), and for CPLEX MIQP. We report the total time in seconds needed to solve the instance to optimality ("tt/s"), the time in seconds needed for the preprocessing ("pt/s"), and the total number of nodes processed in the enumeration tree ("nodes").

Table 2 shows that our algorithm outperforms CPLEX by several orders of magnitude. CPLEX could not solve instances with more than 50 variables within the time limit of 8 hours, while our algorithm solves all instances up to 120 variables, both with and without improved lower bounds. The method using only trivial bounds is the fastest, but the method using improved lower bounds starts to catch up for larger instances. This can also be observed in Figure 8, where we plot the running times of both methods on a logarithmic scale. Unfortunately, $n = 120$ appears to be the largest size our method can solve, even if setting a much larger time limit. Finally, as expected, the $O(n^2)$ running time per node leads to much slower total times. However, even this slower version clearly outperforms CPLEX.

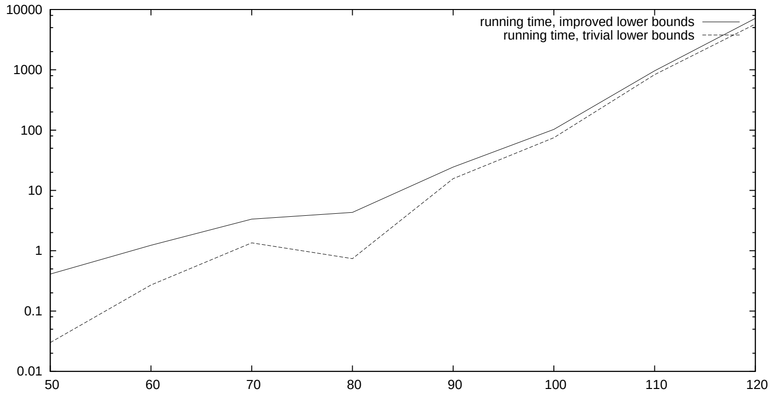


Figure 8: Running times for the ternary case ($\gamma = 0.2$).

instance		Algo. 1 (trivial bounds)			Algo. 1 (improved bounds)			Algo. 1 ($O(n^2)$, impr. bounds)			CPLEX MIQP		optimum
γ	n	tt/s	pt/s	nodes	tt/s	pt/s	nodes	tt/s	pt/s	nodes	tt/s	nodes	
0.2	30	0.00	0.00	2.72e+03	0.04	0.04	1.98e+03	0.05	0.04	1.84e+03	1.11	1.20e+04	6.657892e-04
0.2	40	0.01	0.00	2.53e+04	0.16	0.15	1.84e+04	0.31	0.15	1.68e+04	61.55	5.14e+05	2.703067e-04
0.2	50	0.03	0.00	4.58e+04	0.42	0.38	3.41e+04	0.88	0.41	3.00e+04	20915.93	1.20e+08	1.088225e-04
0.2	60	0.28	0.01	3.94e+05	1.20	0.88	2.98e+05	6.24	0.90	2.62e+05	—	—	5.987001e-05
0.2	70	1.38	0.03	1.80e+06	3.50	1.83	1.39e+06	32.96	1.91	1.09e+06	—	—	3.811803e-05
0.2	80	0.74	0.04	7.90e+05	4.38	3.50	6.09e+05	25.91	3.64	5.05e+05	—	—	1.863941e-05
0.2	90	15.91	0.07	1.80e+07	26.18	6.18	1.39e+07	523.11	6.41	1.09e+07	—	—	1.398550e-05
0.2	100	76.29	0.11	8.09e+07	106.46	10.26	6.28e+07	2958.38	10.68	5.12e+07	—	—	9.677417e-06
0.2	110	836.34	0.17	8.62e+08	1080.66	16.24	6.76e+08	—	—	—	—	—	7.226546e-06
0.2	120	5831.52	0.22	5.72e+09	7441.20	24.57	4.47e+09	—	—	—	—	—	5.332916e-06
0.3	30	0.00	0.00	3.68e+03	0.05	0.05	2.74e+03	0.06	0.04	2.49e+03	1.84	1.86e+04	7.691583e-04
0.3	40	0.04	0.01	5.97e+04	0.18	0.14	4.53e+04	0.52	0.15	3.89e+04	212.71	1.79e+06	3.624281e-04
0.3	50	0.10	0.00	1.47e+05	0.49	0.38	1.12e+05	1.78	0.40	9.36e+04	—	—	1.453055e-04
0.3	60	0.89	0.02	1.36e+06	1.92	0.86	1.05e+06	16.94	0.90	8.75e+05	—	—	7.434848e-05
0.3	70	1.65	0.03	2.18e+06	3.82	1.83	1.70e+06	39.34	1.91	1.34e+06	—	—	4.060845e-05
0.3	80	7.42	0.05	9.28e+06	12.68	3.49	7.24e+06	213.88	3.66	5.87e+06	—	—	2.452317e-05
0.3	90	19.76	0.08	2.26e+07	30.79	6.13	1.76e+07	688.08	6.41	1.41e+07	—	—	1.513070e-05
0.3	100	55.95	0.11	6.10e+07	80.84	10.26	4.71e+07	2135.50	10.69	3.84e+07	—	—	9.448114e-06
0.3	110	256.83	0.16	2.55e+08	338.45	16.18	1.97e+08	10425.21	16.85	1.46e+08	—	—	6.957344e-06
0.3	120	8383.02	0.23	8.39e+09	10500.86	24.58	6.54e+09	—	—	—	—	—	5.685910e-06
0.4	30	0.00	0.00	3.19e+03	0.04	0.04	2.38e+03	0.06	0.04	2.04e+03	1.23	1.29e+04	8.177199e-04
0.4	40	0.01	0.00	1.68e+04	0.15	0.15	1.28e+04	0.26	0.15	1.06e+04	80.71	6.80e+05	3.136527e-04
0.4	50	0.09	0.01	1.24e+05	0.47	0.38	9.50e+04	1.53	0.40	7.70e+04	—	—	1.518196e-04
0.4	60	0.26	0.02	3.57e+05	1.16	0.87	2.73e+05	5.49	0.90	2.15e+05	—	—	7.385543e-05
0.4	70	2.31	0.03	3.14e+06	4.57	1.83	2.45e+06	51.64	1.92	1.88e+06	—	—	4.473205e-05
0.4	80	0.21	0.05	1.81e+05	3.69	3.49	1.34e+05	8.60	3.65	1.04e+05	—	—	1.776351e-05
0.4	90	15.51	0.08	1.80e+07	25.17	6.13	1.37e+07	520.05	6.85	9.98e+06	—	—	1.587669e-05
0.4	100	198.42	0.11	2.24e+08	258.12	10.25	1.73e+08	6615.51	11.62	1.24e+08	—	—	1.167412e-05
0.4	110	659.69	0.16	6.91e+08	834.28	16.19	5.32e+08	24945.14	16.85	3.86e+08	—	—	7.950806e-06
0.4	120	6852.12	0.23	7.05e+09	8720.60	24.58	5.45e+09	—	—	—	—	—	5.842935e-06

Table 2: Branch-and-bound results for second order Butterworth filters.

4.2 Closest Vector Problem

We are not aware of any benchmark instances or any computational evaluation of algorithms for CVP. Even for the practical applications arising in cryptography, we could not find any public library of test problems.

Therefore, in this section, we present results for random instances of the problem generated as follows. For given problem dimension n, an $n \times n$ matrix B is produced, with all entries chosen uniformly at random in $\{-3, \ldots, 3\}$. Moreover, we choose $\lambda \in \mathbb{R}^n$ with entries uniformly at random in $[-1, 1]$. The problem is then to minimize $f(x) = ||Bx - B\lambda||^2$ over all $x \in \mathbb{Z}^n$.

The results for these instances are presented in Table 3, where 10 random instances have been considered for each n. The first column for every solution method shows the number of instances solved to optimality, the remaining figures are averages over all solved instances. Again it turns out that Algorithm 1 is much faster than CPLEX, even if the difference is smaller than in the ternary case. The same is true when comparing our algorithm to SHVEC (mentioned in Section 1.1). Notice that the effect of the improvement of lower bounds by ellipsoids (using as T the identity also in this case) is more apparent now.

In Table 4, we show results for the same instances when a basis reduction is applied before starting the algorithm. Namely, according to Section 2, we re-define the problem at the beginning by applying a basis change with respect to T. This guarantees that all ellipsoids used to improve bounds are axis-parallel. Moreover, we now branch along hyperplanes that are parallel to flat directions, which improves the running time considerably. Note that we could apply a basis change also in the ternary case, but with the effect that X would not be a box any more. Basis reduction decreases running times by about one order of magnitude for our algorithm, but also decreases running time for CPLEX (by a smaller factor). Surprisingly, the running time of SHVEC often increases when applying it to a reduced basis.

5 Final Remarks

We presented an algorithm for solving strictly convex quadratic minimization problems over integer variables. The main limitation in applying our method to the case in which the objective function f is convex but not strictly convex, i.e., Q has zero eigenvalues, is the fact that the continuous minimum is computed without taking X into account. While in the strictly convex case we always have a bounded relaxation, this may not hold any more in the convex case (though one could verify that testing unboundedness of the relaxation and, if this is not the case, deriving the minimum, simply requires additional linear algebra operations). Note that, if Q is a rational matrix, then the relaxed problem is unbounded if and only if the problem with integrality constraints is unbounded. However, our impression is that X should be considered in the relaxation solved at every node.

Clearly, our algorithm can be extended to the mixed-integer case in a natural way. Only two changes are necessary: the branching may be applied only to integer variables, and all ellipsoids used for improving lower bounds must be parallel to the subspace spanned by the continuous variables.

Acknowledgments

We are grateful to Federico Bizzarri, Sergio Callegari, Riccardo Rovatti and Gianluca Setti for having posed us the ternary CQIP as a real-world problem arising in Electronics and having co-operated with us within the OpIMA project. Moreover, we would like to thank Fritz Eisenbrand,

	Algorithm 1 (trivial bounds)				Algorithm 1 (improved bounds)				CPLEX MIQP			SHVEC	
n	#	tt/s	pt/s	nodes	#	tt/s	pt/s	nodes	#	tt/s	nodes	#	tt/s
20	10	0.00	0.00	7.02e+03	10	0.01	0.01	2.27e+03	10	0.23	2.84e+03	10	0.00
25	10	0.02	0.00	4.48e+04	10	0.03	0.02	1.66e+04	10	1.49	1.71e+04	10	0.02
30	10	0.36	0.00	6.77e+05	10	0.26	0.04	2.00e+05	10	13.20	1.34e+05	10	0.59
35	10	0.98	0.00	1.70e+06	10	0.64	0.08	4.41e+05	10	43.62	3.81e+05	10	3.97
40	10	17.15	0.00	2.77e+07	10	10.02	0.14	6.84e+06	10	529.71	3.67e+06	10	72.36
45	10	83.82	0.00	1.28e+08	10	44.34	0.24	2.79e+07	10	3304.63	1.86e+07	10	1808.25
50	10	485.67	0.01	7.03e+08	10	277.73	0.38	1.69e+08	7	12488.02	5.65e+07	8	12155.87
55	10	4384.28	0.01	6.05e+09	10	1952.78	0.58	1.03e+09	3	16054.74	6.26e+07	3	11565.88
60	8	14707.46	0.01	1.92e+10	10	9620.62	0.87	4.71e+09	0	—	—	0	—
65	1	26227.67	0.02	3.37e+10	2	18610.53	1.28	9.17e+09	0	—	—	0	—
70	0	—	—	—	0	—	—	—	0	—	—	0	—

Table 3: Branch-and-bound results for CVP.

	Algorithm 1 (trivial bounds)				Algorithm 1 (improved bounds)				CPLEX MIQP			SHVEC	
n	#	tt/s	pt/s	nodes	#	tt/s	pt/s	nodes	#	tt/s	nodes	#	tt/s
20	10	0.00	0.00	5.84e+02	10	0.00	0.00	2.64e+02	10	0.07	7.98e+02	10	0.00
25	10	0.00	0.00	2.84e+03	10	0.02	0.02	1.13e+03	10	0.37	4.25e+03	10	0.03
30	10	0.01	0.00	1.97e+04	10	0.05	0.04	7.09e+03	10	2.90	2.99e+04	10	0.32
35	10	0.06	0.01	1.02e+05	10	0.12	0.08	3.42e+04	10	20.88	1.88e+05	10	5.69
40	10	0.45	0.00	7.52e+05	10	0.42	0.15	2.29e+05	10	205.92	1.52e+06	10	133.58
45	10	3.30	0.00	5.30e+06	10	2.33	0.24	1.57e+06	10	1766.50	1.04e+07	10	2490.55
50	10	27.57	0.01	4.13e+07	10	16.67	0.39	1.10e+07	10	11577.58	5.27e+07	3	6229.05
55	10	174.25	0.01	2.51e+08	10	101.54	0.59	6.11e+07	4	18798.12	7.29e+07	4	17429.85
60	10	970.84	0.01	1.31e+09	10	561.70	0.88	3.12e+08	0	—	—	0	—
65	9	7263.22	0.02	9.34e+09	10	6441.91	1.28	3.40e+09	0	—	—	0	—
70	3	16337.70	0.03	1.98e+10	4	14264.91	1.84	6.88e+09	0	—	—	0	—

Table 4: Branch-and-bound results for CVP after basis reduction.

Achill Schürmann and Frank Vallentin for helpful discussions on CVP, and Angelika Wiegele and Frauke Liers for help on the state-of-the-art SDP-based methods for quadratic 0-1 programming.

References

[1] BLAS (Basic Linear Algebra Subprograms), 2009. `www.netlib.org/blas`.

[2] LAPACK – Linear Algebra PACKage, 2009. `www.netlib.org/lapack`.

[3] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–2004, 2008.

[4] S. Callegari, F. Bizzarri, R. Rovatti, and G. Setti. Discrete quadratic programming problems by $\Delta\Sigma$ modulation: the case of circulant quadratic forms. Technical report, Arces, University of Bologna, 2009.

[5] S. Dash, A. Lodi, and D. Rajan. $\{-1, 0, 1\}$ unconstrained quadratic programs using max-flow based relaxations. Technical Report OR/05/13, DEIS, University of Bologna, 2005.

[6] C. De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79:71–75, 1989.

[7] F. Eisenbrand. Integer programming and algorithmic geometry of numbers. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L.A. Wolsey, editors, *50 Years of Integer Programming 1958-2008. The Early Years and State-of-the-Art Surveys*. Springer-Verlag, 2009.

[8] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44:463–471, 1985.

[9] A. Frangioni, A. Lodi, and G. Rinaldi. Optimizing over semimetric polytopes. In D. Bienstock and G. Nemhauser, editors, *Integer Programming and Combinatorial Optimization – IPCO 2004*, volume 3064 of *Lecture Notes in Computer Science*, pages 431–443. Springer-Verlag, 2004.

[10] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel. Nonlinear integer programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008. The Early Years and State-of-the-Art Surveys*. Springer-Verlag, 2009.

[11] ILOG, Inc. ILOG CPLEX 12.1, 2009. `www.ilog.com/products/cplex`.

[12] A. Lodi. MIP computation and beyond. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L.A. Wolsey, editors, *50 Years of Integer Programming 1958-2008. The Early Years and State-of-the-Art Surveys*. Springer-Verlag, 2009.

[13] A. Lodi, K. Allemand, and T. M. Liebling. An evolutionary heuristic for quadratic 0–1 programming. *European Journal of Operational Research*, 119:662–670, 1999.

[14] D. Micciancio and S. Goldwasser. *Complexity Of Lattice Problems: A Cryptographic Perspective*. Springer-Verlag, 2002.

[15] J. J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1:93–113, 1991.

[16] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *8th International Symposium on Fundamentals of Computation Theory*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85, 1991.

[17] V. Shoup. NTL: A Library for doing Number Theory (version 5.5.2), 2009.
`www.shoup.net/ntl`.

[18] F. Vallentin. SHVEC: shortest and closest vectors in lattices, 2006.
`fma2.math.uni-magdeburg.de/~latgeo/SHVEC-1.0/shvec-1.0.tar.gz`.