Optim Lett
DOI 10.1007/s11590-017-1132-y

ORIGINAL PAPER

A semidefinite programming method for integer convex quadratic minimization

Jaehyun Park[1] · **Stephen Boyd**[2]

Received: 21 September 2016 / Accepted: 11 March 2017

Abstract We consider the NP-hard problem of minimizing a convex quadratic function over the integer lattice $\mathbf{Z}^n$. We present a simple semidefinite programming (SDP) relaxation for obtaining a nontrivial lower bound on the optimal value of the problem. By interpreting the solution to the SDP relaxation probabilistically, we obtain a randomized algorithm for finding good suboptimal solutions, and thus an upper bound on the optimal value. The effectiveness of the method is shown for numerical problem instances of various sizes.

Keywords Convex optimization · Integer quadratic programming · Mixed-integer programming · Semidefinite relaxation · Branch-and-bound

1 Introduction

We consider the NP-hard problem

$$\begin{array}{ll} \text{minimize} & f(x) = x^T P x + 2q^T x \\ \text{subject to} & x \in \mathbf{Z}^n, \end{array} \tag{1}$$

with variable x, where $P \in \mathbf{R}^{n\times n}$ is nonzero, symmetric, and positive semidefinite, and $q \in \mathbf{R}^n$.

✉ Jaehyun Park
jpark@cs.stanford.edu

Stephen Boyd
boyd@stanford.edu

[1] Stanford University, Packard Building, Room 243, 350 Serra Mall, Stanford, CA 94305, USA

[2] Stanford University, Packard Building, Room 254, 350 Serra Mall, Stanford, CA 94305, USA

Published online: 17 March 2017

A number of other problems can be reduced to the form of (1). The *integer least squares problem*,

$$\begin{array}{l}\text{minimize } \|Ax - b\|_2^2 \\ \text{subject to } x \in \mathbf{Z}^n,\end{array} \tag{2}$$

with variable x and data $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$, is easily reduced to the form of (1) by expanding out the objective function. The mixed-integer version of the problem, where some components of x are allowed to be real numbers, also reduces to an equivalent problem with integer variables only. This transformation uses the Schur complement to explicitly minimize over the noninteger variables [5, §A.5.5]. Another equivalent formulation of (1) is the *closest vector problem*,

$$\begin{array}{l}\text{minimize } \|v - z\|_2^2 \\ \text{subject to } z \in \{Bx \mid x \in \mathbf{Z}^n\},\end{array}$$

in the variable $z \in \mathbf{R}^m$. Typically, the columns of B are linearly independent. Although not equivalent to (1), the *shortest vector problem* is also a closely related problem, which in fact, is reducible to solving the closest vector problem:

$$\begin{array}{l}\text{minimize } \|z\|_2^2 \\ \text{subject to } z \in \{Bx \mid x \in \mathbf{Z}^n\} \\ \qquad\qquad z \neq 0.\end{array}$$

Problem (1) arises in several applications. For example, in position estimation using the Global Positioning System (GPS), resolving the integer ambiguities of the phase data is posed as a mixed-integer least squares problem [18]. In multiple-input multiple-output (MIMO) wireless communication systems, maximum likelihood detection of (vector) Boolean messages involves solving an integer least squares problem [19]. The mixed-integer version of the least squares problem appears in data fitting applications, where some parameters are integer-valued (See, e.g., [26]). The closest vector problem and shortest vector problem have numerous application areas in cryptanalysis of public key cryptosystem such as RSA [24]. The *spectral test*, which is used to check the quality of linear congruential random number generators, is an application of the shortest vector problem [20, §3.3.4].

1.1 Previous work

Several hardness results are known for the integer least squares problem (2). Given an instance of the integer least squares problem, define the approximation factor of a point x to be $\|Ax - b\|_2^2 / \|Ax^\star - b\|_2^2$, where $x^\star$ is the global (integer) solution of (2). Finding a constant factor approximation is an NP-hard problem [2]. In fact, finding an approximation still remains NP-hard even when the target approximation factor is relaxed to $n^{c/\log\log n}$, where $c > 0$ is some constant [11].

Standard methods for finding the global optimum of (1), in the case of positive definite P, work by enumerating all integer points within a suitably chosen box or ellipsoid

[6,13]. The worst case running time of these methods is exponential in n, making it impractical for problems of large size. Algorithms such as Lenstra–Lenstra–Lovász lattice reduction algorithm [21,25] can be used to find an approximate solution in polynomial time, but the approximation factor guarantee is exponential in n [17, §5.3].

A simple lower bound on $f^\star$, the optimal value of (1), can be obtained in $O(n^3)$ time, by removing the integer constraint. If $q \in \mathcal{R}(P)$, where $\mathcal{R}(P)$ denotes the range of P, then this continuous relaxation has a solution $x^{\text{cts}} = -P^\dagger q$, with objective value $f^{\text{cts}} = -q^T P^\dagger q$, where $P^\dagger$ denotes the Moore–Penrose pseudoinverse of P (When P is positive definite, the continuous solution reduces to $x^{\text{cts}} = -P^{-1}q$). If $q \notin \mathcal{R}(P)$, then the objective function is unbounded below and $f^\star = -\infty$.

There exist different approaches for obtaining tighter lower bounds than f^{cts}. The strongest bounds to date are based on semidefinite programming (SDP) relaxation [6–8]. The primary drawback of the SDP-based methods is their running time. In particular, if these methods are applied to branch-and-bound type enumeration methods to prune the search tree, the benefit of having a stronger lower bound is overshadowed by the additional computational cost it incurs, for all small- to medium-sized problems. Enumeration methods still take exponential time in the number of variables, whereas solving SDPs can be done (practically) in $O(n^3)$ time. Thus, for very large problems, SDP-based lower bounds are expected to reduce the total running time of the enumeration methods. However, such problems would be too big to have any practical implication. On the other hand, there exist weaker bounds that are quicker to compute; in [7], for example, these bounds are obtained by finding a quadratic function $\tilde{f}$ that is a global underestimator of f, that has the additional property that the integer point minimizing $\tilde{f}$ can be found simply by rounding x^{cts} to the nearest integer point. Another approach is given by [3], which is to minimize f outside an ellipsoid that can be shown to contain no integer point. Standard results on the $\mathcal{S}$-procedure state that optimizing a quadratic function outside an ellipsoid, despite being a nonconvex problem, can be done exactly and efficiently [4].

A simple upper bound on $f^\star$ can be obtained by observing some properties of the problem. First of all, $x = 0$ gives a trivial upper bound of $f(0) = 0$, which immediately gives $f^\star \leq 0$. Another simple approximate solution can be obtained by rounding each entry of x^{cts} to the nearest integer point, x^{rnd}. Let $f^{\text{rnd}} = f(x^{\text{rnd}})$. Assuming that $q \in \mathcal{R}(P)$, we can get a bound on f^{rnd} as follows. Start by rewriting the objective function as

$$f(x) = (x - x^{\text{cts}})^T P(x - x^{\text{cts}}) + f^{\text{cts}}.$$

Since rounding changes each coordinate by at most $1/2$, we have

$$\|x^{\text{rnd}} - x^{\text{cts}}\|_2^2 = \sum_{i=1}^{n} (x_i^{\text{rnd}} - x_i^{\text{cts}})^2 \leq n/4.$$

It follows that

$$f^{\text{rnd}} - f^{\text{cts}} = (x^{\text{rnd}} - x^{\text{cts}})^T P(x^{\text{rnd}} - x^{\text{cts}}) \leq \sup_{\|v\|_2 \leq \sqrt{n}/2} v^T P v = (n/4)\omega_{\max}, \quad (3)$$

where $\omega_{\max}$ is the largest eigenvalue of P. Since f^{cts} is a lower bound on $f^\star$, this inequality bounds the suboptimality of x^{rnd}. We note that in the special case of diagonal P, the objective function is separable, and thus the rounded solution is optimal. However, in general, x^{rnd} is not optimal, and in fact, f^{rnd} can be positive, which is even worse than the trivial upper bound $f(0) = 0$.

We are not aware of any efficient method of finding a strong upper bound on $f^\star$, other than performing a local search or similar heuristics on x^{rnd}. However, the well-known result by [14] gives provable lower and upper bounds on the optimal value of the NP-hard *maximum cut problem*, which, after a simple reformulation, can be cast as a Boolean nonconvex quadratic problem in the following form:

$$\begin{array}{ll} \text{maximize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n. \end{array} \tag{4}$$

These bounds were obtained by solving an SDP relaxation of (4), and subsequently running a randomized algorithm using the solution of the relaxation. The expected approximation factor of the randomized algorithm is approximately 0.878. There exist many extensions of the Goemans–Williamson SDP relaxation [22]. In particular, [8] generalizes this idea to a more general domain $D_1 \times \cdots \times D_n$, where each D_i is any closed subset of $\mathbf{R}$.

1.2 Our contribution

Our aim is to present a simple but powerful method of producing *both* lower and upper bounds on the optimal value $f^\star$ of (1). Our SDP relaxation is an adaptation of [14], but can also be recovered by appropriately using the method in [8]. By considering the binary expansion of the integer variables as a Boolean variable, we can reformulate (1) as a Boolean problem and directly apply the method of [14]. This reformulation, however, increases the size of the problem and incurs additional computational cost. To avoid this, we work with the formulation (1), at the expense of slightly looser SDP-based bound. We show that our lower bound still consistently outperforms other lower bounds shown in [6,7]. In particular, the new bound is better than the best axis-parallel ellipsoidal bound, which also requires solving an SDP.

Using the solution of the SDP relaxation, we construct a randomized algorithm that finds good feasible points. In addition, we present a simple local search heuristic that can be applied to every point generated by the randomized algorithm. Evaluating the objective at these points gives an upper bound on the optimal value. This upper bound provides a good starting point for enumeration methods, and can save a significant amount of time during the search process. We show this by comparing the running time of an enumeration method, when different initial upper bounds on the optimal value were given. Also, we empirically verify that this upper bound is much stronger than simply rounding a fractional solution to the nearest integer point, and in fact, is near-optimal for randomly generated problem instances.

2 Lagrange duality

In this section, we discuss a Lagrangian relaxation for obtaining a nontrivial lower bound on $f^\star$. We make three assumptions without loss of generality. Firstly, we assume that $q \in \mathcal{R}(P)$, so that the optimal value $f^\star$ is not unbounded below. Secondly, we assume that $x^{\mathrm{cts}} \notin \mathbf{Z}^n$, otherwise x^{cts} is already the global solution. Lastly, we assume that x^{cts} is in the box $[0, 1]^n$. For any arbitrary problem instance, we can translate the coordinates in the following way to satisfy this assumption. Note that for any $v \in \mathbf{Z}^n$, the problem below is equivalent to (1):

$$\begin{array}{l}\text{minimize } (x - v)^T P(x - v) + 2(Pv + q)^T (x - v) + f(v)\\ \text{subject to } x \in \mathbf{Z}^n.\end{array}$$

By renaming $x - v$ to x and ignoring the constant term $f(v)$, the problem can be rewritten in the form of (1). Clearly, this has different solutions and optimal value from the original problem, but the two problems are related by a simple change of coordinates: point x in the new problem corresponds to $x + v$ in the original problem. To translate the coordinates, find $x^{\mathrm{cts}} = -P^\dagger q$, and take elementwise floor to x^{cts} to get x^{flr}. Then, substitute x^{flr} in place of v above.

We note a simple fact that every integer point x satisfies either $x_i \leq 0$ or $x_i \geq 1$ for all i. Equivalently, this condition can be written as $x_i(x_i - 1) \geq 0$ for all i. Using this, we relax the integer constraint $x \in \mathbf{Z}^n$ into a set of nonconvex quadratic constraints: $x_i(x_i - 1) \geq 0$ for all i. The following nonconvex problem is then a relaxation of (1):

$$\begin{array}{l}\text{minimize } x^T Px + 2q^T x\\ \text{subject to } x_i(x_i - 1) \geq 0, \quad i = 1, \ldots, n.\end{array} \tag{5}$$

It is easy to see that the optimal value of (5) is greater than or equal to f^{cts}, because x^{cts} is not a feasible point, due to the two assumptions that $x^{\mathrm{cts}} \notin \mathbf{Z}^n$ and $x^{\mathrm{cts}} \in [0, 1]^n$. Note that the second assumption was necessary, for otherwise x^{cts} is the global optimum of (5), and the Lagrangian relaxation described below would not produce a lower bound that is better than f^{cts}.

The Lagrangian of (5) is given by

$$L(x, \lambda) = x^T Px + 2q^T x - \sum_{i=1}^{n} \lambda_i x_i(x_i - 1) = x^T(P - \mathbf{diag}(\lambda))x + 2(q + (1/2)\lambda)^T x,$$

where $\lambda \in \mathbf{R}^n$ is the vector of dual variables. Define $\tilde{q}(\lambda) = q + (1/2)\lambda$. By minimizing the Lagrangian over x, we get the Lagrangian dual function

$$g(\lambda) = \begin{cases} -\tilde{q}(\lambda)^T (P - \mathbf{diag}(\lambda))^\dagger \tilde{q}(\lambda) & \text{if } P - \mathbf{diag}(\lambda) \succeq 0 \text{ and } \tilde{q}(\lambda) \in \mathcal{R}(P - \mathbf{diag}(\lambda)) \\ -\infty & \text{otherwise,} \end{cases} \tag{6}$$

where the inequality $\succeq$ is with respect to the positive semidefinite cone. The Lagrangian dual problem is then

$$\begin{array}{l}\text{maximize } g(\lambda)\\ \text{subject to } \lambda \geq 0,\end{array} \tag{7}$$

in the variable $\lambda \in \mathbf{R}^n$, or equivalently,

$$\begin{array}{ll} \text{maximize} & -\tilde{q}(\lambda)^T \left(P - \mathbf{diag}(\lambda)\right)^\dagger \tilde{q}(\lambda) \\ \text{subject to} & P - \mathbf{diag}(\lambda) \succeq 0 \\ & \tilde{q}(\lambda) \in \mathcal{R}(P - \mathbf{diag}(\lambda)) \\ & \lambda \geq 0. \end{array}$$

By using the Schur complements, the problem can be reformulated into an SDP:

$$\begin{array}{ll} \text{maximize} & -\gamma \\ \text{subject to} & \begin{bmatrix} P - \mathbf{diag}(\lambda) & q + (1/2)\lambda \\ (q + (1/2)\lambda)^T & \gamma \end{bmatrix} \succeq 0 \\ & \lambda \geq 0, \end{array} \tag{8}$$

in the variables $\lambda \in \mathbf{R}^n$ and $\gamma \in \mathbf{R}$. We note that while (8) is derived from a nonconvex problem (5), it is convex and thus can be solved in polynomial time.

2.1 Comparison to simple lower bound

Due to weak duality, we have $g(\lambda) \leq f^\star$ for any $\lambda \geq 0$, where $g(\lambda)$ is defined by (6). Using this property, we show a provable bound on the Lagrangian lower bound. Let $f^{\text{cts}} = -q^T P^\dagger q$ be the simple lower bound on $f^\star$, and $f^{\text{sdp}} = \sup_{\lambda \geq 0} g(\lambda)$ be the lower bound obtained by solving the Lagrangian dual. Also, let $\omega_1 \geq \cdots \geq \omega_n$ be the eigenvalues of P. For clarity of notation, we use $\omega_{\max}$ and $\omega_{\min}$ to denote the largest and smallest eigenvalues of P, namely ω_1 and ω_n. Let $\mathbf{1}$ represent a vector of an appropriate length with all components equal to one. Then, we have the following result.

Theorem 1 *The lower bounds satisfy*

$$f^{\text{sdp}} - f^{\text{cts}} \geq \frac{n\omega_{\min}^2}{4\omega_{\max}} \left(1 - \frac{\left\|x^{\text{cts}} - (1/2)\mathbf{1}\right\|_2^2}{n/4}\right)^2. \tag{9}$$

Proof When $\omega_{\min} = 0$, the right-hand side of (9) is zero, and there is nothing else to show. Thus, without loss of generality, we assume that $\omega_{\min} > 0$, i.e., $P \succ 0$.

Let $P = Q\,\mathbf{diag}(\omega)Q^T$ be the eigenvalue decomposition of P, where $\omega = (\omega_1, \ldots, \omega_n)$. We consider λ of the form $\lambda = \alpha\mathbf{1}$, and rewrite the dual function in terms of α, where α is restricted to the range $\alpha \in [0, \omega_{\min})$:

$$g(\alpha) = -(q + (1/2)\alpha\mathbf{1})^T (P - \alpha I)^{-1} (q + (1/2)\alpha\mathbf{1}).$$

We note that $g(0) = -q^T P^{-1} q = f^{\text{cts}}$, so it is enough to show the same lower bound on $g(\alpha) - g(0)$ for any particular value of α.

Let $s = Q^T\mathbf{1}$, and $\tilde{q} = Q^T q$. By expanding out $g(\alpha)$ in terms of $s, \tilde{q}$, and ω, we get

$$\begin{aligned} g(\alpha) - g(0) &= -\sum_{i=1}^{n} \frac{\tilde{q}_i^2 + \alpha s_i \tilde{q}_i + (1/4)\alpha^2 s_i^2}{\omega_i - \alpha} + \sum_{i=1}^{n} \frac{\tilde{q}_i^2}{\omega_i} \\ &= -\sum_{i=1}^{n} \frac{(\alpha/\omega_i)(\tilde{q}_i + (1/2)\omega_i s_i)^2 - (1/4)\alpha s_i^2(\omega_i - \alpha)}{\omega_i - \alpha} \\ &= \frac{\alpha}{4}\sum_{i=1}^{n} s_i^2 - \sum_{i=1}^{n} \frac{\alpha(\tilde{q}_i + (1/2)\omega_i s_i)^2}{\omega_i(\omega_i - \alpha)} \\ &= \frac{\alpha n}{4} - \alpha \sum_{i=1}^{n} \left(1 - \frac{\alpha}{\omega_i}\right) \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha}\right)^2. \end{aligned}$$

By differentiating the expression above with respect to α, we get

$$g'(\alpha) = \frac{n}{4} - \sum_{i=1}^{n} \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha}\right)^2.$$

We note that g' is a decreasing function in α in the interval $[0, \omega_{\min})$. Also, at $\alpha = 0$, we have

$$\begin{aligned} g'(0) &= \frac{n}{4} - \sum_{i=1}^{n} (\tilde{q}_i/\omega_i + (1/2)s_i)^2 \\ &= \frac{n}{4} - \left\| \mathbf{diag}(\omega)^{-1} Q^T q + (1/2) Q^T \mathbf{1} \right\|_2^2 \\ &= \frac{n}{4} - \left\| -Q\,\mathbf{diag}(\omega)^{-1} Q^T q - (1/2) Q Q^T \mathbf{1} \right\|_2^2 \\ &= \frac{n}{4} - \left\| -P^{-1} q - (1/2)\mathbf{1} \right\|_2^2 \\ &= \frac{n}{4} - \left\| x^{\text{cts}} - (1/2)\mathbf{1} \right\|_2^2 \geq 0. \end{aligned}$$

The last line used the fact that x^{cts} is in the box $[0, 1]^n$.

Now, we distinguish two cases depending on whether the equation $g'(\alpha) = 0$ has a solution in the interval $[0, \omega_{\min})$.

1. Suppose that $g'(\alpha^\star) = 0$ for some $\alpha^\star \in [0, \omega_{\min})$. Then, we have

$$\begin{aligned}
g(\alpha^\star) - g(0) &= \frac{\alpha^\star n}{4} - \alpha^\star \sum_{i=1}^n \left(1 - \frac{\alpha^\star}{\omega_i}\right) \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha^\star}\right)^2 \\
&= \alpha^\star \left(\frac{n}{4} - \sum_{i=1}^n \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha^\star}\right)^2\right) \\
&\quad + \sum_{i=1}^n \frac{\alpha^{\star 2}}{\omega_i} \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha^\star}\right)^2 \\
&= \sum_{i=1}^n \frac{\alpha^{\star 2}}{\omega_i} \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha^\star}\right)^2 \\
&\geq \frac{\alpha^{\star 2}}{\omega_{\max}} \sum_{i=1}^n \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha^\star}\right)^2 \\
&= \frac{n\alpha^{\star 2}}{4\omega_{\max}}.
\end{aligned}$$

Using this, we go back to the equation $g'(\alpha^\star) = 0$ and establish a lower bound on $\alpha^\star$:

$$\begin{aligned}
\frac{n}{4} &= \sum_{i=1}^n \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha^\star}\right)^2 \\
&= \sum_{i=1}^n \frac{\omega_i}{\omega_i - \alpha^\star} (\tilde{q}_i/\omega_i + (1/2)s_i)^2 \\
&\leq \frac{\omega_{\min}}{\omega_{\min} - \alpha^\star} \sum_{i=1}^n (\tilde{q}_i/\omega_i + (1/2)s_i)^2 \\
&= \frac{\omega_{\min}}{\omega_{\min} - \alpha^\star} \left\| x^{\text{cts}} - (1/2)\mathbf{1} \right\|_2^2.
\end{aligned}$$

From this inequality, we have

$$\alpha^\star \geq \omega_{\min} \left(1 - \frac{\|x^{\text{cts}} - (1/2)\mathbf{1}\|_2^2}{n/4}\right).$$

Plugging in this lower bound on $\alpha^\star$ gives

$$f^{\text{sdp}} - g(0) \geq g(\alpha^\star) - g(0) \geq \frac{n\omega_{\min}^2}{4\omega_{\max}} \left(1 - \frac{\|x^{\text{cts}} - (1/2)\mathbf{1}\|_2^2}{n/4}\right)^2.$$

2. If $g'(\alpha) \neq 0$ for all $\alpha \in [0, \omega_{\min})$, then by continuity of g', it must be the case that $g'(\alpha) \geq 0$ on the range $[0, \omega_{\min})$. Then, for all $\alpha \in [0, \omega_{\min})$,

$$\begin{aligned} f^{\text{sdp}} - g(0) &\geq g(\alpha) - g(0) \\ &= \frac{\alpha n}{4} - \alpha \sum_{i=1}^{n} \left(1 - \frac{\alpha}{\omega_i}\right) \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha}\right)^2 \\ &\geq \frac{\alpha n}{4} - \alpha \left(1 - \frac{\alpha}{\omega_{\max}}\right) \sum_{i=1}^{n} \left(\frac{\tilde{q}_i + (1/2)\omega_i s_i}{\omega_i - \alpha}\right)^2 \\ &\geq \frac{\alpha n}{4} - \alpha \left(1 - \frac{\alpha}{\omega_{\max}}\right) \frac{n}{4} \\ &= \frac{n\alpha^2}{4\omega_{\max}}, \end{aligned}$$

and thus,

$$f^{\text{sdp}} - g(0) \geq \lim_{\alpha \to \omega_{\min}} \frac{n\alpha^2}{4\omega_{\max}} = \frac{n\omega_{\min}^2}{4\omega_{\max}}.$$

Therefore, in both cases, the increase in the lower bound is guaranteed to be at least

$$\frac{n\omega_{\min}^2}{4\omega_{\max}} \left(1 - \frac{\|x^{\text{cts}} - (1/2)\mathbf{1}\|_2^2}{n/4}\right)^2,$$

as claimed. □

Now we discuss several implications of Theorem 1. First, we note that the right-hand side of (9) is always nonnegative, and is monotonically decreasing in $\left\|x^{\text{cts}} - (1/2)\mathbf{1}\right\|_2$. In particular, when x^{cts} is an integer point, then we must have $f^{\text{cts}} = f^{\text{sdp}} = f^\star$. Indeed, for $x^{\text{cts}} \in \{0, 1\}^n$, we have $\left\|x^{\text{cts}} - (1/2)\mathbf{1}\right\|_2^2 = n/4$, and the right-hand side of (9) is zero. Also, when P is positive definite, i.e., $\omega_{\min} > 0$, then (9) implies that $f^{\text{sdp}} > f^{\text{cts}}$.

In order to obtain the bound on f^{sdp}, we only considered vectors λ of the form $\alpha\mathbf{1}$. Interestingly, solving (7) with this additional restriction is equivalent to solving the following problem:

$$\begin{aligned} &\text{minimize } x^T P x + 2q^T x \\ &\text{subject to } \|x - (1/2)\mathbf{1}\|_2^2 \geq n/4. \end{aligned} \tag{10}$$

The nonconvex constraint enforces that x lies outside the n-dimensional sphere centered at $(1/2)\mathbf{1}$ that has every lattice point $\{0, 1\}^n$ on its boundary. Even if (10) is not a convex problem, it can be solved exactly; the $\mathcal{S}$-lemma implies that the SDP relaxation of (10) is tight (see, e.g., [4]). For completeness, we give the dual of the SDP relaxation [which has the same optimal value as (10)] below, which is exactly what we used to prove Theorem 1:

$$\begin{array}{ll}\text{maximize} & -\gamma \\ \text{subject to} & \begin{bmatrix} P-\alpha I & q+(\alpha/2)\mathbf{1} \\ (q+(\alpha/2)\mathbf{1})^T & \gamma \end{bmatrix} \succeq 0 \\ & \alpha \geq 0. \end{array}$$

3 Semidefinite relaxation

In this section, we show another convex relaxation of (5) that is equivalent to (8), but with a different form. By introducing a new variable $X = xx^T$, we can reformulate (5) as:

$$\begin{array}{ll}\text{minimize} & \mathbf{Tr}(PX) + 2q^T x \\ \text{subject to} & \mathbf{diag}(X) \geq x \\ & X = xx^T, \end{array}$$

in the variables $X \in \mathbf{R}^{n\times n}$ and $x \in \mathbf{R}^n$. Observe that the constraint $\mathbf{diag}(X) \geq x$, along with $X = xx^T$, is a rewriting of the constraint $x_i(x_i - 1) \geq 0$ in (5).

Then, we relax the nonconvex constraint $X = xx^T$ into $X \succeq xx^T$, and rewrite it using the Schur complement to obtain a convex relaxation:

$$\begin{array}{ll}\text{minimize} & \mathbf{Tr}(PX) + 2q^T x \\ \text{subject to} & \mathbf{diag}(X) \geq x \\ & \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0. \end{array} \tag{11}$$

The optimal value of problem (11) is a lower bound on $f^\star$, just as the Lagrangian relaxation (8) gives a lower bound f^{sdp} on $f^\star$. In fact, problems (8) and (11) are duals of each other, and they yield the same lower bound f^{sdp} [27].

3.1 Randomized algorithm

The semidefinite relaxation (11) has a natural probabilistic interpretation, which can be used to construct a simple randomized algorithm for obtaining good suboptimal solutions, i.e., feasible points with low objective value. Let $(X^\star, x^\star)$ be any solution to (11). Suppose $z \in \mathbf{R}^n$ is a Gaussian random variable with mean μ and covariance matrix Σ. Then, $\mu = x^\star$ and $\Sigma = X^\star - x^\star x^{\star T}$ solve the following problem of minimizing the expected value of a quadratic form, subject to quadratic inequalities:

$$\begin{array}{ll}\text{minimize} & \mathbf{E}(z^T Pz + 2q^T z) \\ \text{subject to} & \mathbf{E}(z_i(z_i - 1)) \geq 0, \quad i = 1, \ldots, n, \end{array}$$

in variables $\mu \in \mathbf{R}^n$ and $\Sigma \in \mathbf{R}^{n\times n}$. Intuitively, this distribution $\mathcal{N}(\mu, \Sigma)$ has mean close to x^{cts} so that the expected objective value is low, but each diagonal entry of Σ is large enough so that when z is sampled from the distribution, $z_i(z_i - 1) \geq 0$ holds in expectation. While sampling z from $\mathcal{N}(\mu, \Sigma)$ does not give a feasible point to (1)

immediately, we can simply round it to the nearest integer point to get a feasible point. Using these observations, we present the following randomized algorithm.

Algorithm 3.1 *Randomized algorithm for suboptimal solution to (1).*

given number of iterations K.

1. *Solve SDP.* Solve (11) to get $X^\star$ and $x^\star$.
2. *Form covariance matrix.* $\Sigma := X^\star - x^\star x^{\star T}$, and find Cholesky factorization $LL^T = \Sigma$.
3. *Initialize best point.* $x^{\text{best}} := 0$ and $f^{\text{best}} := 0$.

for $k = 1, 2, \ldots, K$

4. *Random sampling.* $z^{(k)} := x^\star + Lw$, where $w \sim \mathcal{N}(0, I)$. (Same as $z^{(k)} \sim \mathcal{N}(x^\star, \Sigma)$.)
5. *Round to nearest integer.* $x^{(k)} := \mathbf{round}(z^{(k)})$.
6. *Update best point.* If $f^{\text{best}} > f(x^{(k)})$, then set $x^{\text{best}} := x^{(k)}$ and $f^{\text{best}} := f(x^{(k)})$.

The SDP in step 1 takes $O(n^3)$ time to solve, assuming that the number of iterations required by an interior point method is constant. Step 2 is dominated by the computation of Cholesky factorization, which uses roughly $n^3/3$ flops. Steps 4 through 6 can be done in $O(n^2)$ time. The overall time complexity of the method is then $O(n^2(K+n))$. By choosing $K = O(n)$, the time complexity can be made $O(n^3)$.

4 Greedy algorithm for obtaining a 1-opt solution

Here we discuss a simple greedy descent algorithm that starts from an integer point, and iteratively moves to another integer point that has a lower objective value. This method can be applied to the simple suboptimal point x^{rnd}, or every $x^{(k)}$ found in Algorithm 3.1, to yield better suboptimal points.

We say that $x \in \mathbf{Z}^n$ is *1-opt* if the objective value at x does not improve by changing a single coordinate, i.e., $f(x + ce_i) \geq f(x)$ for all indices i and integers c. The difference in the function values at x and $x + ce_i$ can be written as

$$f(x + ce_i) - f(x) = c^2 P_{ii} + cg_i = P_{ii}(c + g_i/(2P_{ii}))^2 - g_i^2/(4P_{ii}),$$

where $g = 2(Px + q)$ is the gradient of f at x. It is easily seen that given i, the expression above is minimized when $c = \mathbf{round}(-g_i/(2P_{ii}))$. For x to be optimal with respect to x_i, then c must be 0, which is the case if and only if $P_{ii} \geq |g_i|$. Thus, x is 1-opt if and only if $\mathbf{diag}(P) \geq |g|$, where the absolute value on the right-hand side is taken elementwise.

Also, observe that

$$P(x + ce_i) + q = (Px + q) + cP_i,$$

where P_i is the ith column of P. Thus, when x changes by a single coordinate, the value of g can be updated just by referencing a single column of P. These observations suggest a simple and quick greedy algorithm for finding a 1-opt point from any given integer point x.

Algorithm 4.1 *Greedy descent algorithm for obtaining 1-opt point.*

given an initial point $x \in \mathbf{Z}^n$.

1. *Compute initial gradient.* $g = 2(Px + q)$.

repeat

2. *Stopping criterion.* **quit** if $\mathbf{diag}(P) \geq |g|$.
3. *Find descent direction.* Find index i and integer c minimizing $c^2 P_{ii} + cg_i$.
4. *Update* x. $x_i := x_i + c$.
5. *Update gradient.* $g := g + 2cP_i$.

Initializing g takes $O(n^2)$ flops, but each subsequent iteration only takes $O(n)$ flops. This is because steps 2 and 3 only refer to the diagonal elements of P, and step 5 only uses a single column of P. Though we do not give an upper bound on the total number of iterations, we show, using numerical examples, that the average number of iterations until convergence is roughly $0.14n$, when the initial points are sampled according to the probability distribution given in Sect. 3.1. The overall time complexity of Algorithm 4.1 is then $O(n^2)$ on average. Thus, we can run the greedy 1-opt descent on every $x^{(k)}$ in Algorithm 3.1, without changing its overall time complexity $O(n^2(K + n))$.

5 Examples

In this section, we consider numerical examples to show the performance of the SDP-based lower bound and randomized algorithm, developed in previous sections.

5.1 Method

We combine the techniques developed in previous sections to find lower and upper bounds on $f^\star$, as well as suboptimal solutions to the problem. By solving the simple relaxation and rounding the solution, we immediately get a lower bound f^{cts} and an upper bound $f^{\text{rnd}} = f(x^{\text{rnd}})$. We also run Algorithm 4.1 on x^{rnd} to get a 1-opt point, namely $\hat{x}^{\text{rnd}}$. This gives another upper bound $\hat{f}^{\text{rnd}} = f(\hat{x}^{\text{rnd}})$.

Then, we solve the semidefinite relaxation (11) to get a lower bound f^{sdp}. Using the solution to the SDP, we run Algorithm 3.1 to obtain suboptimal solutions, and keep the best suboptimal solution x^{best}. In addition, we run Algorithm 4.1 on every feasible point considered in step 4 of Algorithm 3.1, and find the best 1-opt suboptimal solution $\hat{x}^{\text{best}}$. The randomized algorithm thus yields two additional upper bounds on $f^\star$, namely $f^{\text{best}} = f(x^{\text{best}})$ and $\hat{f}^{\text{best}} = f(\hat{x}^{\text{best}})$.

The total number of iterations K in Algorithm 3.1 is set to $K = 3n$, so that the overall time complexity of the algorithm, not counting the running time of the 1-opt greedy descent algorithm, is $O(n^3)$. We note that the process of sampling points and running Algorithm 4.1 trivially parallelizes.

5.2 Numerical examples

We use random instances of the integer least squares problem (2) generated in the following way. First, the entries of $A \in \mathbf{R}^{m \times n}$ are sampled independently from $\mathcal{N}(0, 1)$.

The dimensions are set as $m = 2n$. We set $q = -Px^{\text{cts}}$, where $P = A^T A$, and x^{cts} is randomly drawn from the box $[0, 1]^n$. The problem is then scaled so that the simple lower bound is -1, i.e., $f^{\text{cts}} = -q^T P^\dagger q = -1$.

There are other ways to generate random problem instances. For example, the eigenspectrum of P is controlled by the magnitude of m relative to n. We note that P becomes a near-diagonal matrix as m diverges to infinity, because the columns of A are uncorrelated. This makes the integer least squares problem easier to solve. On the contrary, smaller m makes the problem harder to solve. Another way of generating random problem instances is to construct P from a predetermined eigenspectrum $\omega_1, \ldots, \omega_n$, as $P = Q\,\mathbf{diag}(\omega)Q^T$, where Q is a random rotation matrix. This makes it easy to generate a matrix with a desired condition number. Our method showed the same qualitative behavior on data generated in these different ways, for larger or smaller m, and also for different eigenspectra.

The SDP (11) was solved using CVX [15,16] with the MOSEK 7.1 solver [23], on a 3.40 GHz Intel Xeon machine. For problems of relatively small size $n \leq 70$, we found the optimal point using MILES [10], a branch-and-bound algorithm for mixed-integer least squares problems, implemented in MATLAB. MILES solves (1) by enumerating lattice points in a suitably chosen ellipsoid. The enumeration method is based on various algorithms developed in [1,9,13,25].

5.3 Results

Lower bounds We compare various lower bounds on $f^\star$. In [7], three lower bounds on $f^\star$ are shown, which we denote by f^{axp}, f^{qax}, and f^{qrd}, respectively. These bounds are constructed from underestimators of f that have a *strong rounding property*, i.e., the integer point minimizing the function is obtained by rounding the continuous solution. We denote the lower bound obtained by solving the following trust region problem in [3] by f^{tr}:

$$\begin{array}{l}\text{minimize } x^T Px + 2q^T x \\ \text{subject to } \|x - x^{\text{cts}}\|_2^2 \geq \|x^{\text{cts}} - x^{\text{rnd}}\|_2^2.\end{array}$$

To the best of our knowledge, there is no standard benchmark test set for the integer least squares problem. Thus, we compared the lower bounds on randomly generated problem instances; for each problem size, we generated 100 random problem instances. In Table 1, we compare the lower bounds averaged over the random problem instances. Note that in all instances, the simple lower bound was $f^{\text{cts}} = -1$. We found not only that our method found a tighter lower bound on average, but also that our method performed consistently better, i.e., in all problem instances, the SDP based lower bound was higher than any other lower bound. We found that the pairs of lower bounds (f^{axp}, f^{qax}) and (f^{qrd}, f^{tr}) were practically equal, although the results of [7] show that they can be all different. We conjecture that this disparity comes from different problem sizes and eigenspectra of the random problem instances. The solution $f^\star$ was not computed for $n > 70$ as MILES was unable to find it within a reasonable amount of time.

Table 1 Average lower bound by number of variables

n	$f^\star$	f^{sdp}	f^{axp}	f^{qax}	f^{qrd}	f^{tr}
50	−0.8357	−0.9162	−0.9434	−0.9434	−0.9736	−0.9736
60	−0.8421	−0.9202	−0.9459	−0.9459	−0.9740	−0.9740
70	−0.8415	−0.9212	−0.9471	−0.9471	−0.9747	−0.9747
100	N/A	−0.9268	−0.9509	−0.9509	−0.9755	−0.9755
500	N/A	−0.9401	−0.9606	−0.9606	−0.9777	−0.9777
1000	N/A	−0.9435	−0.9630	−0.9630	−0.9781	−0.9781

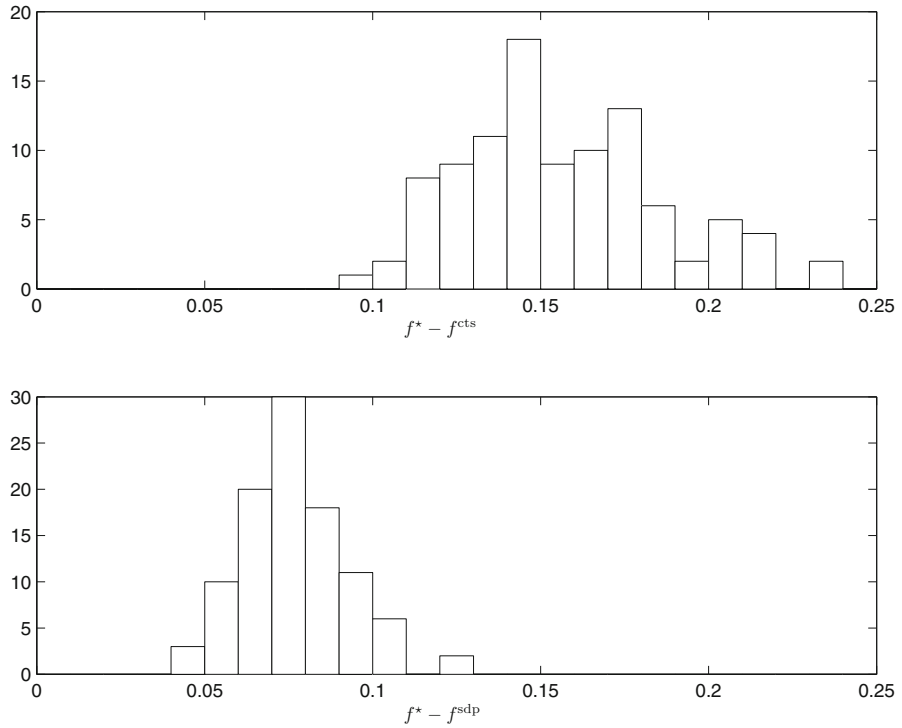

Fig. 1 Histograms of the gap between the optimal value $f^\star$ and the two lower bounds f^{cts} and f^{sdp}, for 100 random problem instances of size $n = 60$

To see how tight f^{sdp} compared to the simple lower bound is, we focus on the set of 100 random problem instances of size $n = 60$, and show, in Fig. 1, the distribution of the gap between $f^\star$ and the lower bounds.

Upper bounds Algorithm 3.1 gives a better suboptimal solution as the number of samples K grows. To test the relationship between the number of samples and the quality of suboptimal solutions, we considered a specific problem instance of size $n = 500$ and sampled $K = 50n$ points. The result suggested that $K = 3n$ is a large enough

Table 2 Average upper bound by number of variables

n	$f^\star$	$\hat{f}^{\text{best}}$	f^{best}	$\hat{f}^{\text{rnd}}$	f^{rnd}	Optimal
50	−0.8357	−0.8353	−0.8240	−0.8186	−0.7365	90%
60	−0.8421	−0.8420	−0.8268	−0.8221	−0.7397	94%
70	−0.8415	−0.8412	−0.8240	−0.8235	−0.7408	89%
100	N/A	−0.8465	−0.8235	−0.8296	−0.7488	N/A
500	N/A	−0.8456	−0.7991	−0.8341	−0.7466	N/A
1000	N/A	−0.8445	−0.7924	−0.8379	−0.7510	N/A

number of samples for most problems; in order to decrease $\hat{f}^{\text{best}}$ further, many more samples were necessary. All subsequent experiments discussed below used $K = 3n$ as the number of sample points.

In Table 2, we compare different upper bounds on $f^\star$ using the same set of test data considered above. We found that Algorithm 3.1 combined with the 1-opt heuristic gives a feasible point whose objective value is, on average, within 5×10^{-3} from the optimal value. The last column of Table 2 indicates the percentage of the problem instances for which $\hat{f}^{\text{best}} = f^\star$ held; we not only found near-optimal solutions, but for most problems, the randomized algorithm actually terminated with the global solution. We expect the same for larger problems, but have no evidence since there is no efficient way to verify optimality.

In Fig. 2, we take the same test data used to produce Fig. 1, and show histograms of the suboptimality of x^{rnd}, $\hat{x}^{\text{rnd}}$, x^{best}, and $\hat{x}^{\text{best}}$. The mean suboptimality of x^{rnd} was 0.1025, and simply finding a 1-opt point from x^{rnd} improved the mean suboptimality to 0.0200. Algorithm 3.1 itself, without 1-opt refinement, produced suboptimal points of mean suboptimality 0.0153, and running Algorithm 4.1 on top of it reduced the suboptimality to 0.0002.

Finally, we take problems of size $n = 1000$, where all existing global methods run too slowly. As the optimal value is unobtainable, we consider the gap given by the difference between the upper and lower bounds. Figure 3 shows histograms of the four optimality gaps obtained from our method, namely $f^{\text{rnd}} - f^{\text{cts}}$, $\hat{f}^{\text{rnd}} - f^{\text{cts}}$, $f^{\text{best}} - f^{\text{sdp}}$, and $\hat{f}^{\text{best}} - f^{\text{sdp}}$. The mean value of these quantities were: 0.2490, 0.1621, 0.1511, and 0.0989. As seen in Table 2, we believe that the best upper bound $\hat{f}^{\text{best}}$ is very close to the optimal value, whereas the lower bound f^{sdp} is farther away from the optimal value.

Running time In Table 3, we compare the running time of our method and that of MILES for problems of various sizes. The running time of MILES varied depending on particular problem instances. For example, for $n = 70$, the running time varied from 6.6 s to 25 min. Our method showed more consistent running time, and terminated within 3 min for every problem instance of the biggest size $n = 1000$. It should be noted that MILES always terminates with a global optimum, whereas our method does not have such a guarantee, even though the experimental results suggest that the best suboptimal point found is close to optimal. From the breakdown of the running time of our method, we see that none of the three parts of our method clearly dominates

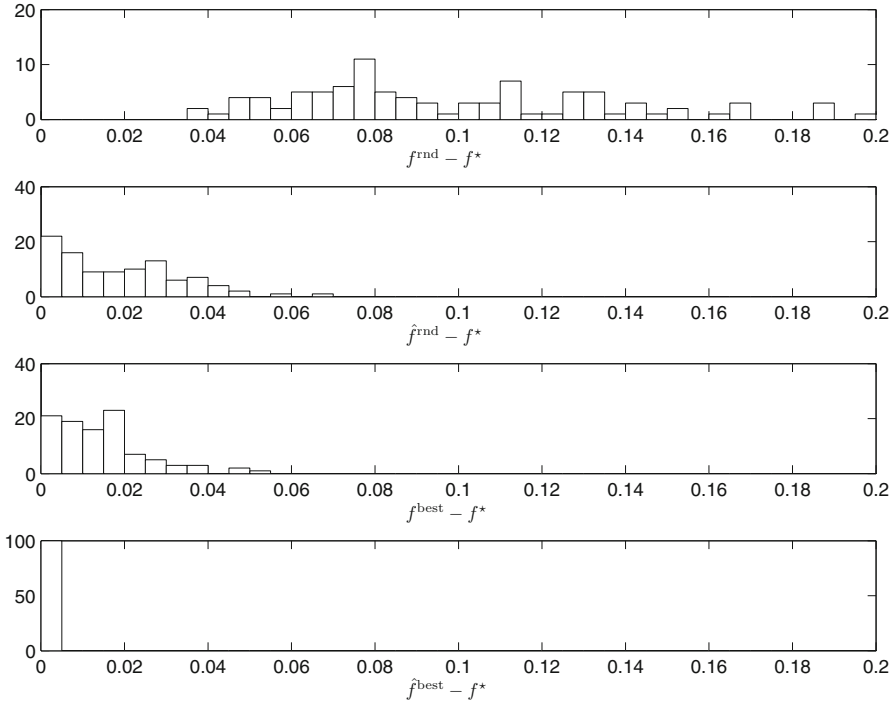

Fig. 2 Histograms of the suboptimality of f^{rnd}, $\hat{f}^{\mathrm{rnd}}$, f^{best}, and $\hat{f}^{\mathrm{best}}$, for 100 random problem instances of size $n = 60$

Table 3 Average running time of MILES and our method in seconds (left), and breakdown of the running time of our method (right)

n	Total running time		Breakdown of running time		
	MILES	Our method	SDP	Random sampling	Greedy 1-opt
50	3.069	0.397	0.296	0.065	0.036
60	28.71	0.336	0.201	0.084	0.051
70	378.2	0.402	0.249	0.094	0.058
100	N/A	0.690	0.380	0.193	0.117
500	N/A	20.99	12.24	4.709	4.045
1000	N/A	135.1	82.38	28.64	24.07

the total running time. We also note that the total running time grows subcubically, despite the theoretical running time of $O(n^3)$.

In a practical setting, if a near-optimal solution is good enough, then running Algorithm 3.1 is more effective than enumeration algorithms. Algorithm 3.1 is particularly useful if the problem size is 60 or more; this is the range where branch-and-bound type algorithms become unviable due to their exponential running time. In practice, one may run an enumeration algorithm (such as MILES) and terminate after a cer-

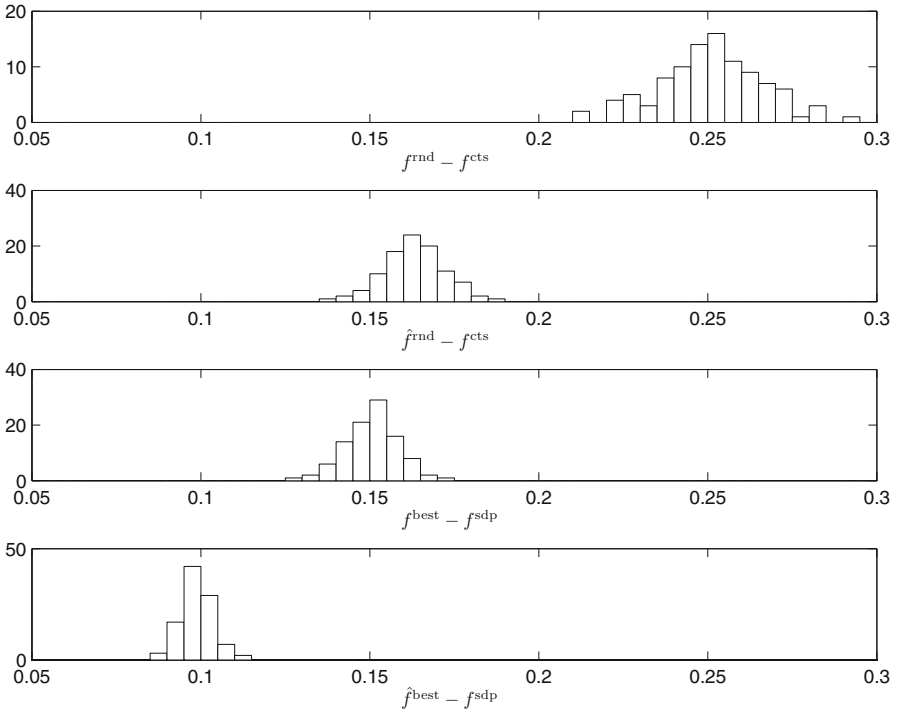

Fig. 3 Histograms of the four optimality gaps for 100 random problem instances of large size $n = 1000$

tain amount of time and use the best found point as an approximation. To show that Algorithm 3.1 is a better approach of obtaining a good suboptimal solution, we compare our method against MILES in the following way. First, we compute $\hat{f}^{\mathrm{best}}$ via Algorithm 3.1, and record the running time T. Then, we run MILES on the same problem instance, but with time limit T, i.e., we terminate MILES when its running time exceeds T, and record the best suboptimal point found. Let f^{miles} be the objective value of this suboptimal point. Table 4 shows the average value of $f^{\mathrm{miles}} - \hat{f}^{\mathrm{best}}$ and the percentage of problem instances for which $\hat{f}^{\mathrm{best}} \leq f^{\mathrm{miles}}$ held. The experiment was performed on 100 random problem instances of each size. We observe that on every problem instance of size $n \geq 100$, our method produced a better point than MILES, when alloted the same running time.

There is no simple bound for the number of iterations that Algorithm 4.1 takes, as it depends on both P and q, as well as the initial point. In Table 5, we give the average number of iterations for Algorithm 4.1 to terminate at a 1-opt point, when the initial points are sampled from the distribution found in Sect. 3.1. We see that the number of iterations when $n \leq 1000$ is roughly $0.14n$. Although the asymptotic growth in the number of iterations appears to be slightly superlinear, as far as practical applications are concerned, the number of iterations is effectively linear. This is because the SDP (11) cannot be solved when the problem becomes much larger (e.g., $n > 10^5$).

Table 4 Comparison of the best suboptimal point found by our method and by MILES, when alloted the same running time

n	$f^{\text{miles}} - \hat{f}^{\text{best}}$	$\hat{f}^{\text{best}} \leq f^{\text{miles}}$ (%)
50	0.0117	98
60	0.0179	99
70	0.0230	99
100	0.0251	100
500	0.0330	100
1000	0.0307	100

Table 5 Average number of iterations of Algorithm 4.1

n	Iterations
50	6.06
60	7.35
70	8.59
100	11.93
500	65.89
1000	141.1

Branch-and-bound method The results of Table 4 suggest that enumeration methods that solve (1) globally can utilize Algorithm 3.1 by taking the suboptimal solution $\hat{x}^{\text{best}}$ and using its objective value $\hat{f}^{\text{best}}$ as the initial bound on the optimal value. In Table 6, we show the average running time of MILES versus n, when different initial bounds were provided to the algorithm: 0, f^{rnd}, $\hat{f}^{\text{rnd}}$, $\hat{f}^{\text{best}}$, and $f^{\star}$. For a fair comparison, we included the running time of computing the respective upper bounds in the total execution time, except in the case of $f^{\star}$. We found that when $n = 70$, if we start with $\hat{f}^{\text{best}}$ as the initial upper bound of MILES, the total running time until the global solution is found is roughly 24% lower than running MILES with the trivial upper bound of 0. Even when $f^{\star}$ is provided as the initial bound, branch-and-bound methods will still traverse a search tree to look for a better point (though it will eventually fail to do so). This running time, thus, can be thought as the baseline performance. If we compare the running time of the methods with respect to this baseline running time, the effect of starting with a tight upper bound becomes more apparent. When 0 is given as the initial upper bound, MILES spends roughly 3 more minutes traversing the nodes that would have been pruned if it started with $f^{\star}$ as the initial upper bound. However, if the initial bound is $\hat{f}^{\text{best}}$, then the additional time spent is only 10 s, as opposed to 3 min. Finally, we note that the baseline running time accounts for more than 70% of the total running time even when the trivial upper bound of zero is given; this suggests that the core difficulty in the problem lies more in proving the optimality than in finding an optimal point itself. In our experiments, the simple lower bound f^{cts} was used to prune the search tree. If tighter lower bounds are used instead, this baseline running time changes, depending on how easy it is to evaluate the lower bound, and how tight the lower bound is.

Directly using the SDP-based lower bound to improve the performance of branch-and-bound methods is more challenging, due to the overhead of solving the SDP. The

Table 6 Average running time of MILES, given different initial upper bounds

n	Initial upper bound				
	0	f^{rnd}	$\hat{f}^{\text{rnd}}$	$\hat{f}^{\text{best}}$	$f^\star$
50	3.046	3.039	2.941	2.900	2.537
60	29.02	29.09	28.14	24.07	23.56
70	379.6	379.4	361.1	290.0	280.6

results by [7] suggest that in order for a branch-and-bound scheme to achieve faster running time, quickly evaluating a lower bound is more important than the quality of the lower bound itself. Even if our SDP-based lower bound is superior to any other lower bound shown in related works, solving an SDP at every node in the branch-and-bound search tree is computationally too costly. Indeed, the SDP-based axis-parallel ellipsoidal bound in [7] fails to improve the overall running time of a branch-and-bound algorithm when applied to every node in the search tree. An outstanding open problem is to find an alternative to f^{sdp} that is quicker to compute. One possible approach would be to look for (easy-to-find) feasible points to (7); as noted in Sect 2, it is not necessary to solve (7) optimally in order to compute a lower bound, since any feasible point of it yields a lower bound on $f^\star$. (See, e.g., [12].)

Acknowledgements We thank three anonymous referees for providing helpful comments and constructive remarks.

References

1. Agrell, E., Eriksson, T., Vardy, A., Zeger, K.: Closest point search in lattices. IEEE Trans. Inf. Theory **48**(8), 2201–2214 (2002)
2. Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The hardness of approximate optima in lattices, codes, and systems of linear equations. In: Proceedings of the 34th Annual Symposium on Foundations of Computer Science, pp. 724–733. IEEE Computer Society Press (1993)
3. Bienstock, D.: Eigenvalue techniques for convex objective, nonconvex optimization problems. In: Integer Programming and Combinatorial Optimization, pp. 29–42. Springer (2010)
4. Boyd, S., El Ghaoui, L., Feron, E., Balakrishnan, V.: Linear Matrix Inequalities in System and Control Theory, vol. 15. SIAM, Philadelphia (1994)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
6. Buchheim, C., Caprara, A., Lodi, A.: An effective branch-and-bound algorithm for convex quadratic integer programming. Math. Program. **135**(1–2), 369–395 (2012)
7. Buchheim, C., Hubner, R., Schobel, A.: Ellipsoid bounds for convex quadratic integer programming. SIAM J. Optim. **25**(2), 741–769 (2015)
8. Buchheim, C., Wiegele, A.: Semidefinite relaxations for non-convex quadratic mixed-integer programming. Math. Program. **141**(1–2), 435–452 (2013)
9. Chang, X.W., Yang, X., Zhou, T.: MLAMBDA: a modified LAMBDA method for integer least-squares estimation. J. Geod. **79**(9), 552–565 (2005)
10. Chang, X.W., Zhou, T.: MILES: MATLAB package for solving mixed integer least squares problems. GPS Solut. **11**(4), 289–294 (2007)
11. Dinur, I., Kindler, G., Raz, R., Safra, S.: Approximating CVP to within almost-polynomial factors is NP-hard. Combinatorica **23**(2), 205–243 (2003)
12. Dong, H.: Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. SIAM J. Optim. **26**(3), 1962–1985 (2016)

13. Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**(170), 463–471 (1985)
14. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
15. Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. In: V. Blondel, S. Boyd, H. Kimura (eds.) Recent Advances in Learning and Control, Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited (2008). http://stanford.edu/~boyd/graph_dcp.html
16. Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx (2014)
17. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, vol. 2. Springer, Heidelberg (2012)
18. Hassibi, A., Boyd, S.: Integer parameter estimation in linear models with applications to gps. IEEE Trans. Signal Process. **46**(11), 2938–2952 (1998)
19. Jaldén, J., Ottersten, B.: On the complexity of sphere decoding in digital communications. IEEE Trans. Signal Process. **53**(4), 1474–1484 (2005)
20. Knuth, D.E.: Seminumerical Algorithms, The Art of Computer Programming. Addison-Wesley, Boston (1997)
21. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
22. Luo, Z.Q., Ma, W.K., So, A.M.C., Ye, Y., Zhang, S.: Semidefinite relaxation of quadratic optimization problems. IEEE Signal Process. Mag. **27**(3), 20–34 (2010)
23. MOSEK ApS: the MOSEK optimization toolbox for MATLAB manual, version 7.1 (revision 28). http://docs.mosek.com/7.1/toolbox.pdf
24. Nguyen, P.Q., Stern, J.: The two faces of lattices in cryptology. In: Cryptography and Lattices, pp. 146–180. Springer (2001)
25. Schnorr, C.P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**(1–3), 181–199 (1994)
26. Ustun, B., Rudin, C.: Supersparse linear integer models for optimized medical scoring systems. Mach. Learn. **102**(3), 349–391 (2016)
27. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Rev. **38**(1), 49–95 (1996)