Constraint Programming for Strictly Convex Integer Quadratically-Constrained Problems

Wen-Yang Ku and J. Christopher Beck

Department of Mechanical & Industrial Engineering
University of Toronto, Toronto, Ontario M5S 3G8, Canada
{wku,jcb}@mie.utoronto.ca

Abstract. Inspired by the geometric reasoning exploited in discrete ellipsoid-based search (DEBS) from the communications literature, we develop a constraint programming (CP) approach to solve problems with strictly convex quadratic constraints. Such constraints appear in numerous applications such as modelling the ground-to-satellite distance in global positioning systems and evaluating the efficiency of a schedule with respect to quadratic objective functions. We strengthen the key aspects of the DEBS approach and implement them as combination of a global constraint and variable/value ordering heuristics in IBM ILOG CP Optimizer. Experiments on a variety of benchmark instances show significant improvement compared to the default settings and state-of-the-art performance compared to competing technologies of mixed integer programming, semi-definite programming, and mixed integer nonlinear programming.

1 Introduction

The strictly convex integer quadratically-constrained problem (IQCP) is an optimization problem where the objective and/or some constraints are strictly convex quadratic functions. The IQCP is known to be NP-hard [1] and arises in a number of applications including global positioning systems, communications, cryptography, bioinformatics, scheduling and finance [2,3,4,5]. Given its theoretical challenge and practical value, it is of great interest to develop efficient algorithms to solve IQCPs. Despite the long history of CP, the techniques to solve quadratically constrained problems have not receive much attention. There are only a few dedicated global constraints that reason about quadratic terms. For example, the Spread constraint [6] enforces the standard quadratic relationship amongst a set of variables, their mean, and their standard deviation. General quadratic constraints [7,8] can be applied to both convex and nonconvex quadratic functions. However, they do not exploit the strictly convex nature of the IQCP.

It has been shown in Ku and Beck [9] that strictly convex IQCPs can be formulated as integer least squares (ILS) problems and solved with discrete ellipsoid-based search (DEBS), a specialized search used in the communications literature (e.g., see [4]). From a CP perspective, DEBS can be understood as

a form of CP search. First, the search strategy uses a static variable ordering heuristic and a dynamic value ordering heuristic based on the structure of the ellipsoid. Second, the geometry of the ellipsoid induces an interval domain for each variable. As a result, DEBS is essentially the enumeration of these domains under the prescribed variable and value orderings with some bounds pruning based on the radius of the hyper-ellipsoid. DEBS was originally formulated to solve only three types of ILS problems: unconstrained, box-constrained, and ellipsoid-constrained [10,3,11,9]. A recent work has extended DEBS for ILS problems with general linear constraints [12].

In this work, we aim at developing techniques that are both powerful and cover a broad range of problems. We introduce two novel techniques, inspired by DEBS, for solving strictly convex IQCPs with constraint programming. First, we propose the ELLIPSOID constraint, a global constraint that filters variable domains with respect to strictly convex quadratic functions. We derive a direct quadratically-constrained programming (QCP) formulation that achieves bounds consistency (BC), and two light-weight filtering algorithms that do not guarantee BC. Though it is natural to consider integer domains in CP, our filtering algorithm can be applied to variables with real domains, broadening its application to, for example, mixed integer programming solvers. Second, we propose a pair of variable/value selection rules. We implement the filtering algorithms and the branching heuristics in IBM ILOG CP Optimizer. We experiment with five problem classes and show orders of magnitude improvement compared to the default CP Optimizer. We then compare our new CP approach with the best known algorithms on the same problem sets. Our results demonstrate that the new CP approach is competitive to the best known approaches, and, for some problem classes, establishes a new state of the art.

The rest of the paper is organized as follows. We give the necessary background in Section 2. In Section 3 we define of our new global constraint. Section 4 and 5 present the filtering algorithms and the branching heuristics. Section 6 provides computational results and discussions. We conclude in Section 7.

2 Background

2.1 The Strictly Convex Integer Quadratically-Constrained Problem (IQCP)

The general IQCP problem has the following form:

$$\min_{\boldsymbol{x}\in\mathcal{C}} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x},$$

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{Z}^n : \frac{1}{2}\boldsymbol{x}^T\boldsymbol{M}_k\boldsymbol{x}+\boldsymbol{c}_k^T\boldsymbol{x} \leq \boldsymbol{b}_k, \forall k = 1,\ldots,m,\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n\}.$$

The IQCP is strictly convex if the quadratic matrices $\boldsymbol{H}$, $\boldsymbol{M}_k, \forall k$ are symmetric positive definite [13]. As the above form suggests, quadratic formulations can exist in the form of an objective function and/or as constraints.

In Operations Research, the common generic approaches to solving IQCPs exactly are the use of mixed integer nonlinear programming (MINLP) such as BARON [14,15] and ANTIGONE [16], and the application of MIP solvers such as CPLEX and Gurobi which have been extended to reason about quadratic constraints [17]. Another generic approach is semi-definite programming (SDP) based branch-and-bound [18]. The available SDP solvers, e.g., BiqCrunch [19], only solve problems with binary variables as opposed to general integer variables.

2.2 Discrete Ellipsoid-based Search (DEBS)

The DEBS method consists of two phases: reduction and search. The reduction is a preprocessing step that transforms $\boldsymbol{A}$ to an upper triangular matrix $\boldsymbol{R}$ using the "QRZ" factorization [3]:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 \rightarrow \min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\bar{\boldsymbol{y}}-\boldsymbol{R}\boldsymbol{z}\|_2^2\,, \tag{1}$$

where $\bar{\boldsymbol{y}}=\boldsymbol{Q}^T\boldsymbol{y}$, $\boldsymbol{z}=\boldsymbol{Z}^{-1}\boldsymbol{x}$, $\boldsymbol{Q}$ is orthogonal, $\boldsymbol{Z}$ is unimodular. The diagonal entries of $\boldsymbol{R}$ are approximately[1] ordered in non-decreasing order: $|r_{ii}| \leq |r_{i+1,i+1}|$. This ordering has been shown to increase the efficiency of the DEBS search by reducing the branching factor at the top of the search tree [10]. As we argue in Section 5 below, this ordering is approximately equivalent to a static smallest domain first variable ordering.

Suppose the optimal solution $\boldsymbol{z}^*$ satisfies $\|\bar{\boldsymbol{y}}-\boldsymbol{R}\boldsymbol{z}^*\|_2^2 < \beta$, or equivalently $\sum_{k=1}^{n}(\bar{y_k}-\sum_{j=k}^{n} r_{kj}z_j)^2 < \beta$, where β is a constant that can be obtained by substituting any feasible integer solution to equation (1). This expression defines a hyper-ellipsoid with center $\boldsymbol{R}^{-1}\bar{\boldsymbol{y}}$. The search, then, systematically enumerates all the integer points in the bounded hyper-ellipsoid [21]. When an incumbent, i.e., new upper bound on β, is found, the hyper-ellipsoid is contracted resulting in reduction of the bounds of the decision variables.

In more detail, let $\boldsymbol{z}_i^n = [z_i, z_{i+1}, \ldots, z_n]^T$ be the vector of decision variables and define the so-far-unknown (apart from c_n) and usually non-integer variables:

$$c_n = \bar{y}_n/r_{nn}, \quad c_k : c_k(z_{k+1},\ldots,z_n) = (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj}z_j)/r_{kk}, \quad k = n-1,\ldots,1.$$

Note that c_k is a function of z_{k+1} to z_n, and it is fixed when z_{k+1} to z_n are fixed. The above equation can be rewritten as $\sum_{k=1}^{n} r_{kk}^2(z_k-c_k)^2 < \beta$, which defines the possible values that z_k can take on. This inequality is equivalent to

[1] Depending on the data, it is sometimes not possible to transform a matrix to exactly achieve this ordering [20].

the following n inequalities:

$$\begin{aligned}
\text{level } n: &\quad (z_n - c_n)^2 < \frac{1}{r_{nn}^2}\beta, \\
\text{level } n-1: &\quad (z_{n-1} - c_{n-1})^2 < \frac{1}{r_{n-1,n-1}^2}[\beta - r_{nn}^2(z_n - c_n)^2], \\
&\vdots \\
\text{level } k: &\quad (z_k - c_k)^2 < \frac{1}{r_{kk}^2}[\beta - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2], \\
&\vdots \\
\text{level } 1: &\quad (z_1 - c_1)^2 < \frac{1}{r_{11}^2}[\beta - \sum_{i=2}^{n} r_{ii}^2(z_i - c_i)^2].
\end{aligned}$$

The search starts at level n, heuristically assigning $z_n = \lfloor c_n \rceil$, the nearest integer to c_n. Given the value of z_n, c_{n-1} can be calculated from the above equation as $c_{n-1} = (\bar{y}_{n-1} - r_{n-1,n}z_n)/r_{n-1,n-1}$. From this value, we can set $z_{n-1} = \lfloor c_{n-1} \rceil$ and search continues. During the search process, z_k is determined at level k, where $z_n, z_{n-1}, \ldots, z_{k+1}$ have already been determined, but $z_{k-1}, z_{k-2}, \ldots, z_1$ are still unassigned. At some level $k-1$ in the search, it is likely that the inequality cannot be satisfied, requiring the search to backtrack to a previous decision. When we backtrack from level $k-1$ to level k, we choose z_k to be the next nearest integer to c_k.

After the optimal solution $\boldsymbol{z}^*$ to the reduced problem (right hand side of equation (1)) is found, the optimal solution, $\boldsymbol{x}^*$, to the original problem (left hand side of equation (1)) can be recovered with the relationship $\boldsymbol{x}^* = \boldsymbol{Z}\boldsymbol{z}^*$.

3 The Ellipsoid Constraint

We propose the ELLIPSOID constraint to reason about convex quadratic functions. It consists of a set of n variables $\{x_1, \ldots, x_n\}$, an $n \times n$ matrix $\boldsymbol{A}$ with full column rank, an n-dimensional vector $\boldsymbol{y}$, and a constant β. The definition is given as follows:

$$\texttt{ellipsoid}(\{x_1, \ldots, x_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta),$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{y} \in \mathbb{R}^n$, $\beta \in \mathbb{R}$. The constraint ensures the following condition:

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \beta. \tag{2}$$

Geometrically, the above expression defines a hyper-ellipsoid with center $\boldsymbol{A}^{-1}\boldsymbol{y}$. Equivalently, 2 can be written in its standard convex quadratic constraint form as

$$\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} \leq \bar{\beta}, \tag{3}$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, $\boldsymbol{f} \in \mathbb{R}^n$ is a vector, and $\bar{\beta} = (\beta - \boldsymbol{y}^T\boldsymbol{y})/2$. The transformation is obtained with the relationships $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{f} = -\boldsymbol{y}^T\boldsymbol{A}$.

The ELLIPSOID constraint can be applied to any formulation with a strictly convex quadratic function. For example, consider the following objective function: $\min \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{H}_1\boldsymbol{y} + \boldsymbol{f}_1^T\boldsymbol{y}$, where only $\boldsymbol{H}$ is symmetric positive definite. We can still apply the ELLIPSOID constraint to the first half of the objective function: $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}$, even if the second part is not strictly convex.

4 Filtering Algorithms for the Ellipsoid Constraint

In this section, we present a number of filtering algorithms that achieve or approximate bounds consistency of the ELLIPSOID constraint.

Let x_j be a finite-domain variable, $Dom(x_j)$ be the domain of x_j, which is a set of ordered values that can be assigned to x_j, and $I_D(x_j) = [l_j, u_j]$ be the interval domain of x_j.

Definition 1. *A* ELLIPSOID *constraint is bounds consistent [22] with respect to domains $Dom(x_j)$ if for all $j \in 1, \ldots, n$ and each value $v_j \in \{l_j, u_j\}$, there exists values $v_i \in I_D(x_i)$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ such that* `ellipsoid`$(\{x_1 = v_1, \ldots, x_n = v_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta)$ *holds.*

In the 2D example shown in Fig. 1, the ELLIPSOID constraint (on the two variables) is bounds consistent.

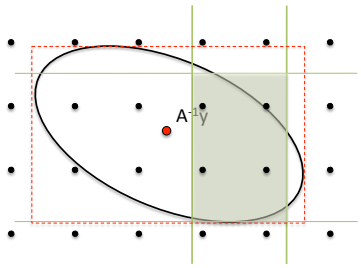


Fig. 1: A 2D example that shows the tangent box of the ellipsoid and bounds consistency of the ELLIPSOID constraint.

4.1 A Direct Quadratically-Constrained Programming (QCP) Formulation

Achieving bounds consistency for a variable x_j is equivalent to finding the lower bound l_j^{BC} and upper bound u_j^{BC} for x_j, given the ELLIPSOID constraint and the current bounds of the variables. Assume that no further reduction can be inferred on the domains of $x_i, \forall i \neq j$, the mathematical model for achieving bounds consistency for x_j can be defined as follows:

$$l_j^{BC} = \min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{e}_j^T\boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \sqrt{\beta},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \tag{4}$$

$$u_j^{BC} = \max_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{e}_j^T\boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2 \le \sqrt{\beta},\ \boldsymbol{l}\le\boldsymbol{x}\le\boldsymbol{u}. \tag{5}$$

Note that $\boldsymbol{e}_j \in \mathbb{R}^n$ is the unit vector in the j-th direction, i.e., the j-th column of an identity matrix with size n. The problems (4) and (5) are quadratically-constrained programming (QCP) optimization problems, which can be solved with QCP solvers such as CPLEX. However, it is computationally expensive, since at least $2n$ QCPs have to be solved in each iteration.

This approach is essentially a QCP version of optimization-based bound tightening (OBBT) as used in the MINLP literature [23]. While typically performed with linear constraints, OBBT is often only done at the root node of the search tree as it is too expensive to perform at every node. Convex QCPs can be solved in polynomial time using iterative approaches such as the interior point method [24].

4.2 Axis-Aligned Tangent Box Filtering (BOX)

The simplest way to tighten the domains of the variables is to compute the tangent box of the hyper-ellipsoid defined in Equation (2), where the edges of the box are parallel to the axes of the coordinate system. As a 2D example, the dotted box in Fig. 1 shows the tangent box. The lower bound $\boldsymbol{l}^b$ and the upper bound $\boldsymbol{u}^b$ that define the tangent box can be computed by solving the following problems:

$$l_j^b = \min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{e}_j^T\boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2 \le \sqrt{\beta}, \tag{6}$$

$$u_j^b = \max_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{e}_j^T\boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2 \le \sqrt{\beta}. \tag{7}$$

Chang & Golub [3] proposed an efficient way to solve the above problems. We first solve the problem in (7) for u_j^b, and the lower bound l_j^b can be obtained by using the symmetric property of an ellipsoid. Let $\boldsymbol{p} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}$, the problem (7) becomes

$$\begin{aligned} u_j^b &= \max_{\boldsymbol{p}} \boldsymbol{e}_j^T\boldsymbol{A}^{-1}(\boldsymbol{p}+\boldsymbol{y}) \\ &= \max_{\boldsymbol{p}} \boldsymbol{e}_j^T\boldsymbol{A}^{-1}\boldsymbol{p} + \boldsymbol{e}_j^T\boldsymbol{A}^{-1}\boldsymbol{y} \quad \text{subject to } \|\boldsymbol{p}\|_2 \le \sqrt{\beta}. \end{aligned} \tag{8}$$

By the Cauchy-Schwarz inequality, we have

$$\boldsymbol{e}_j^T\boldsymbol{A}^{-1}\boldsymbol{p} \le \left\|\boldsymbol{A}^{-T}\boldsymbol{e}_j\right\|_2 \|\boldsymbol{p}\|_2 \le \left\|\boldsymbol{A}^{-T}\boldsymbol{e}_j\right\| \sqrt{\beta}.$$

The first inequality becomes an equality if and only if $\boldsymbol{p} = c\boldsymbol{A}^{-T}\boldsymbol{e}_j$ for some non-negative scalar c. The second inequality becomes equality if and only if $\|\boldsymbol{p}\|_2 = \sqrt{\beta}$. Therefore, $\boldsymbol{p}$ is the minimizer for (8) when $\boldsymbol{p} = \sqrt{\beta}\boldsymbol{A}^{-T}\boldsymbol{e}_j / \left\|\boldsymbol{A}^{-T}\boldsymbol{e}_j\right\|_2$. Substituting $\boldsymbol{p}$ into (8), we have

$$u_j^b = \sqrt{\beta}\left\|\boldsymbol{A}^{-T}\boldsymbol{e}_j\right\|_2 + \boldsymbol{e}_j^T\boldsymbol{A}^{-1}\boldsymbol{y}. \tag{9}$$

From the symmetry property of an ellipsoid, we have

$$l_j^b = -\sqrt{\beta}\left\|\boldsymbol{A}^{-T}\boldsymbol{e}_j\right\|_2 + \boldsymbol{e}_j^T\boldsymbol{A}^{-1}\boldsymbol{y}. \tag{10}$$

Computing the Reduced Intersecting Ellipsoid $\mathcal{E}_{\mathcal{F}}$. When a variable is fixed during the search, e.g., $x_i = v_i$, the dimension of the ellipsoid is reduced by one. Geometrically, we need to find the one-dimension-smaller ellipsoid that intersects at $x_i = v_i$ and $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \beta$. We propose a general way to compute the reduced intersecting ellipsoid with any number of variables fixed.

Let $\mathcal{F}$ be the set of the variables that are fixed and let $\tilde{\boldsymbol{A}} = [\boldsymbol{A}_j], \forall j \neq \mathcal{F}$, $\bar{\boldsymbol{y}} = \boldsymbol{y} - \sum_{i\in\mathcal{F}} \boldsymbol{A}_i v_i$ and the QR factorization of $\tilde{\boldsymbol{A}}$ as: $\tilde{\boldsymbol{A}} = \begin{bmatrix}\tilde{\boldsymbol{Q}}_1 & \tilde{\boldsymbol{Q}}_2\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{R}} \\ \boldsymbol{0}\end{bmatrix}$, the reduced intersecting ellipsoid $\mathcal{E}_{\mathcal{F}}$ can be computed as follows:

$$\mathcal{E}_{\mathcal{F}} = \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}\right\|_2^2 \leq \tilde{\beta}, \tag{11}$$

where $\tilde{x}$ is the vector of the unknown variables of the reduced ellipsoid, $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{Q}}_1^T\bar{\boldsymbol{y}}$ and $\tilde{\beta} = \beta - \|\bar{\boldsymbol{y}}\|_2^2 + \|\tilde{\boldsymbol{y}}\|_2^2$. The derivations on $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{y}}$ are straightforward so we only explain $\tilde{\beta}$ as follows. We know that

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \left\|\bar{\boldsymbol{y}} - \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}\right\|_2^2 = \left\|\begin{bmatrix}\tilde{\boldsymbol{Q}}_1^T \\ \tilde{\boldsymbol{Q}}_2^T\end{bmatrix}\bar{y} - \begin{bmatrix}\tilde{\boldsymbol{R}} \\ \boldsymbol{0}\end{bmatrix}\tilde{\boldsymbol{x}}\right\|_2^2.$$

It follows that

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 - \left\|\tilde{\boldsymbol{Q}}_1^T\bar{\boldsymbol{y}} - \tilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}\right\|_2^2 = \left\|\tilde{\boldsymbol{Q}}_2^T\bar{\boldsymbol{y}}\right\|_2^2 = \|\bar{\boldsymbol{y}}\|_2^2 - \left\|\tilde{\boldsymbol{Q}}_1^T\bar{\boldsymbol{y}}\right\|_2^2.$$

We assume that β and $\tilde{\beta}$ constraints are satisfied at equality as this corresponds to the largest ellipsoids defined by our inequalities and therefore ensures that no valid values are pruned. Since $\beta = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ and $\tilde{\beta} = \left\|\tilde{\boldsymbol{Q}}_1^T\bar{\boldsymbol{y}} - \tilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}\right\|_2^2$, we have $\tilde{\beta} = \beta - \|\bar{\boldsymbol{y}}\|_2^2 + \|\tilde{\boldsymbol{y}}\|_2^2$.

At each node of the tree, we can first compute the reduced ellipsoid $\mathcal{E}_F$ w.r.t the variables that are already fixed with Equation (11), then apply Equations (9) and (10) to compute the axis-aligned tangent box. The algorithm propagates when variables are instantiated or β is reduced. As our CP search instantiates variables, the propagation is active at each node.

Complexity of the Filtering Algorithm. The filtering algorithm reduces the domains of all the variables based on β with $O(n^3)$ time-complexity. First, computing the reduced ellipsoid takes $O(n^3)$, as the QR factorization is required, Second, computing the tangent box takes $O(n^3)$, as the complexity is dominated by computing the matrix inverse $\boldsymbol{A}^{-1}$. Therefore the total time complexity for filtering the domain for all the variable is $O(n^3)$.

4.3 Approximate Bounds Consistency (ABC) Filtering

Before we introduce the next filtering algorithm, our notation is summarized as follows:

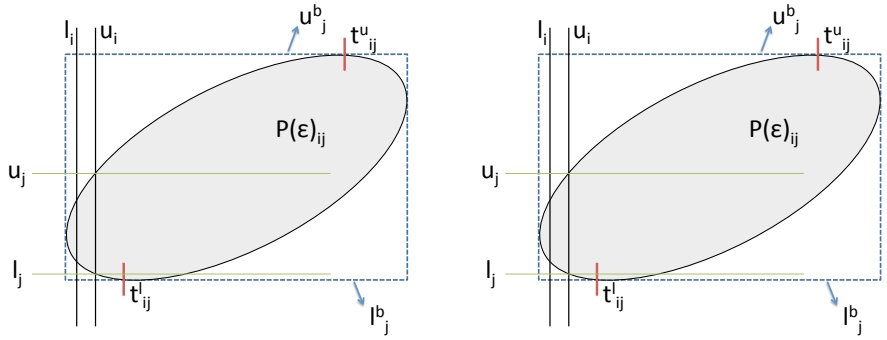
- $[l_j, u_j]$: The interval domains (local bounds) of variable x_j.
- $[l_j^b, u_j^b]$: The tangent box derived with Equation (9) and (10) of the ellipsoid $\mathcal{E}$ defined in Equation (2).
- v_j: A value that is within x_j's domain, i.e, $v_j \in I_D(x_j) = [l_j, u_j]$.

It is observed that if $\boldsymbol{l} \leq \boldsymbol{l}^b \leq \boldsymbol{u}^b \leq \boldsymbol{u}$, then the tangent box defines the bounds of the variables. However, a pair of bounds may lead to reductions in other variable domains. For example, in Fig. 2-a, the bounds l_i and u_i have the effect of increasing the lower bound l_j^b to l_j and decreasing the upper bound u_j^b to u_j.

To perform stronger domain reductions on the ellipsoid, first we need to determine the set of variables that can be used to infer reductions on the lower bounds or upper bounds of the variables. We explain the propagation algorithm below for the lower bound only since the propagation on the upper bound can be derived in a symmetric manner.

Proposition 1. *Let $P(\mathcal{E})_{ij}$ be the ellipse defined by the projection of the hyper-ellipsoid (Equation 2) onto the x_ix_j plane. Then the variable x_i can be used to infer domain reductions on x_j's lower bound if and only if $l_i \leq u_i \leq t_{ij}^l$ or $t_{ij}^l \leq l_i \leq u_i$, where t_{ij}^l is the x_i value at the intersection of $x_j = l_j^b$ and the projected ellipse $P(\mathcal{E})_{ij}$.*

Proof. If $l_i \leq t_{ij}^l \leq u_i$ (Fig. 2-b), we can set $x_i = t_{ij}^l$, so that $x_j = l_j^b$, thus no domain reduction can be inferred to x_j's lower bound. In the other two cases where $l_i \leq u_i \leq t_{ij}^l$ (Fig. 2-a) or $t_{ij}^l \leq l_i \leq u_i$, since x_j is forced to take a value that is greater than l_j^b, we can increase x_j's lower bound.



(a) The case that $l_i \leq u_i \leq t_{ij}^l$. (b) The case that $l_i \leq t_{ij}^l \leq u_i$.

Fig. 2: The 2D projection of the hyper-ellipsoid onto the x_ix_j plane.

We refer to t_{ij}^l and t_{ij}^u as the touching points.

Computing the Touching Points. The touching points can be computed easily as a by-product of computing the axis-aligned tangent box (9) and (10). Since $\boldsymbol{p} = \sqrt{\beta}\boldsymbol{A}^{-T}\boldsymbol{e}_j / \left\|\boldsymbol{A}^{-T}\boldsymbol{e}_j\right\|_2$ uniquely defines the minimizer for the upper bound u_j^b, let $\boldsymbol{x}^*$ be the solution to the equation $\boldsymbol{p} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}$, we have:

$$(\boldsymbol{t}_j^u)^T = [t_{1j}^u, \ldots, t_{j-1,j}^u, t_{j+1,j}^u, \ldots, t_{nj}^u]^T = [x_1^*, \ldots, x_{j-1}^*, x_{j+1}^*, \ldots, x_n^*]^T$$

Note that $\boldsymbol{t}_j^u$ is a $n-1$ dimensional vector and $x_j^* = u_j^b$. Using the symmetry property of the ellipsoid, $\boldsymbol{t}_j^l$ can be computed by reflecting t_j^u about the center of the ellipsoid.

The complexity of computing the touching points for all the variables, i.e., $\boldsymbol{t}_j^l, \boldsymbol{t}_j^u, \forall j$, is $O(n^3)$, as $\boldsymbol{x}^*$ can be computed in $O(n^2)$ for each variable, given that $\boldsymbol{A}^{-1}$ is known.

Proposition 2. *If x_i can be used to increase x_j's lower bound l_j according to Proposition 1, the value v_i^d that should be used to increase l_j is defined as*
$v_i^d = \begin{cases} l_i, \text{ if } (t_{ij}^l - l_i)^2 \leq (t_{ij}^l - u_i)^2. \\ u_i, \qquad \text{otherwise.} \end{cases}$

Proof. Since $P(\mathcal{E})_{ij}$ is convex and x_j achieves its minimum l_j^b at $x_i = t_{ij}^l$, for any point x_j in $P(\mathcal{E})_{ij}$, we have x_j strictly larger than l_j^b if x_i takes any value other than t_{ij}^l. That is, x_j increases strictly when x_i moves away from t_{ij}^l. Therefore, the value (l_i or u_i) that achieves the minimum of the expression $\min((t_{ij}^l - u_i)^2, (t_{ij}^l - l_i)^2)$ determines x_j's lower bound.

As Fig. 2-a depicts, we choose u_i in this example, as using l_i removes the valid value l_j.

Using Proposition 1 and 2, we can identify the set of variables and their values that can be used to increase the lower bound of a variable.

Definition 2. *For each variable x_j, let S_j^l (S_j^u) be the set of all the variables that can be used to infer domain reductions on x_j's lower (upper) bound.*

Definition 3. *An assignment A: $x_i \mapsto I_D(x_i)$, $i \in S_j^l$ or S_j^u is said to be a determining assignment when $A(x_i) = v_i^d$.*

When a variable takes a determining assignment, we can compute the reduced intersecting ellipsoid using the method in Section 4.2. The complete filtering algorithm for pruning the lower bound of a variable is presented in Algorithm 1. The upper bound pruning can be derived in a symmetric manner.

Complexity of the Filtering Algorithm The filtering algorithm reduces the domain of a single variable in $O(n^3)$ time-complexity. First, computing the tangent box takes $O(n^3)$ (see Section 4.2). The touching points require $O(n^2)$ as explained previously. In line seven, the sets S_j^l can be computed in $O(n)$. The for-loop at Line 8 requires $O(n^3)$, as Line 9 and 10 require $O(n^2)$ for computing the reduced ellipsoid (by updating the QR factorization) and the tangent plane for

Algorithm 1 $Prune(l_j)$

Data: The local bounds: $l_1, \ldots, l_n$, $u_1, \ldots, u_n$, the tangent box for the ellipsoid $\mathcal{E}$: $l_1^b, \ldots, l_n^b$, $u_1^b, \ldots, u_n^b$, the touching points t_{ij}^l, β
Results: The filtered lower bound l'_j
Initialization: Set $l'_j = -\infty$, $\mathcal{F} = \{\}$
if $u_j < l_j^b$ **then**
  The constraint is not satisfiable
else
  Compute the set S_j^l and the associated assignments $A(x_i)$, $\forall i \in S_j^l$
  for each $i \in S_j^l$ **do**
    Let $\mathcal{F} = \{i\}$, compute the reduced intersecting ellipsoid $\mathcal{E}_F$
    Compute the tangent box $(l_j^F)^b$ of $\mathcal{E}_F$
    Set $l'_j = \max((l_j^F)^b, l'_j)$
  end for
  if $u_i < l'_j$ **then**
    The constraint is not satisfiable
  else
    Set $l'_j = \max((l'_j, l_j)$
  end if
end if

each of the i in S_j^l. Therefore the total time-complexity for filtering the domain for one variable is $O(n^3)$.

The complexity for filtering all the variables is therefore $O(n^4)$, which is the complexity when $S_j^l, \forall j$, contains $n-1$ variables. However, it is beneficial to have more variables in the set, as more and stronger pruning might be done.

4.4 Relative Strength of the Three Filtering Algorithms

It is clear that the ABC filtering algorithm is at least as strong as the BOX filtering algorithm, since ABC uses the tangent box as the starting point. The QCP filtering is at least as strong as ABC. ABC only considers the 2D projection of the hyper-ellipsoid onto each $x_i x_j$ plane, i.e., x_j is only tightened using the bounds of each x_i, independently. However, it is also possible to perform a higher dimension projection of the hyper-ellipsoid and use the bounds of more than one variable together to do bound tightening. Consider the 3D projection of the hyper-ellipsoid and the reasoning among x_i, x_j, x_k. It is possible to use x_i and x_k, together, to tighten x_j, given that x_i and x_k intersects inside the hyper-ellipsoid. For this reason, ABC only achieves BC when $|S_j^l| = 1, |S_j^u| = 1, \forall j$, and the tangent box only achieves BC when $|S_j^l| = 0, |S_j^u| = 0, \forall j$.

5 Branching Rules

We propose variable and value ordering rules inspired by the search strategy of DEBS. Recall that the static variable ordering in DEBS tries to minimize

the branching factor at the top of the tree so that it is easier to find feasible solutions. In CP, this is the same as choosing a variable with minimum domain size. We therefore use the standard dynamic variable selection rule that chooses a variable with the smallest domain. The value ordering rule in DEBS always assigns a variable to the integer value closest to the center of the ellipsoid of the objective function so that the search greedily chooses the best integer value at the current node with the hope of finding good feasible solution quickly. In our implementation, suppose x_j is the variable chosen for branching, we first compute the center of the ellipsoid c_j in j-th dimension given the reduced ellipsoid, and then round it to the nearest integer. When the search backtracks, we assign x_j to the next nearest integer to c_j, and so on.

6 Experimental Results

Experimental Setup. We design two experiments. The goal of the first experiment is to evaluate the impact of our filtering algorithms and the branching heuristics compared to a default CP model. The second experiment compares our new CP approach to the best known exact approaches for IQCPs.

For the first experiment, we use IBM ILOG CP Optimizer v12.6.3 with its default settings. The three filtering algorithms, e.g., BOX, ABC, and QCP, are implemented in CP Optimizer as customized global constraints. The branching rules are implemented as customized variable and value choosers (denoted with the symbol "+b" in our results). We use CPLEX v12.6.3 for solving the QCPs. We report the arithmetic mean CPU time "time" in seconds, and the arithmetic mean number of choice points "chpts" to find and prove optimality for each problem set.

For the second experiment, we use CPLEX v12.6.3,[2] BARON v16.4.7 (using CPLEX v12.6.3 as its LP/MIP solver) and the SDP solver BiqCrunch downloaded from the website [19] for comparison. All solvers are executed with their default settings.[3] The DEBS algorithm is written in C.

The CPU time limit for each run on each problem instance is 3600 seconds. All experiments were performed on a Intel(R) Xeon(R) CPU E5-1650 v2 3.50GHz machine (in 64 bit mode) with 16GB memory running MAC OS X 10.9.2 with one thread.

6.1 Problem Sets

We experiment on medium size problems in five problem classes. The problem size of each set is chosen with the aim for a mix of solvable and non-solvable instances across the default CP Optimizer and the three filtering algorithms.

[2] A major improvement was made in solving IQCPs in CPLEX v12.6.3 [25].

[3] There are four versions of the SDP solver that deal with problem-specific structures. The SDP results presented are the best version for each individual problem *instance*, representing the "virtual best" SDP solver.

Binary Quadratic Programming (BQP) Problem. The BQP problem is defined as: $\min_{\boldsymbol{x}\in\{0,1\}} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}$, where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{f} \in \mathbb{R}^n$. BQPs arise in many combinatorial optimization problems such as task allocation [26], quadratic assignment [27], and max-cut problems [18]. We experiment on the Carter type problems [28] divided into four sub-sets of instances with size 40 ($p = 0.2$), 40 ($p = 0.3$), 50 ($p = 0.2$) and 50 ($p = 0.3$), respectively, where p is a problem generation parameter.

Exact Quadratic Knapsack Problem (EQKP). The EQKP [29] is defined as: $\min_{\boldsymbol{x}\in\mathcal{C}} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}, \quad \mathcal{C} = \{\boldsymbol{x} \in \{0,1\} : \boldsymbol{c}_1^T\boldsymbol{x} = K,\ \boldsymbol{c}_2^T\boldsymbol{x} \leq B\}$, where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{f} \in \mathbb{R}^n$, $\boldsymbol{c}_1 \in \mathbb{R}^n$ is a vector equal to ones, $\boldsymbol{c}_2 \in \mathbb{R}^n_+$, $K \in \mathbb{Z}_+$, $B \in \mathbb{R}^+$. The objective is to minimize a quadratic function subject to a cardinality constraint and a knapsack constraint. The EQKP is a extension of the BQP, maximum diversity problem [30], quadratic knapsack problem [31], and exact linear knapsack problem [32]. EQKPs arise in a wide range of real world applications such as wind farm optimization [33,34]. We experiment on the EQKP and a variation from Ku & Beck [12] with binary domains $x_j \in \{0, 1\}$ relaxed to $x_j \in \{0, 1, 2\}$. We use three sub-sets of the instances with size 10, 20 and 30.

Box-constrained ILS (BILS) Problem. The BILS problem can be defined as: $\min_{\boldsymbol{x}\in\mathcal{C}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{C} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n\}$. The problem minimizes a least squares expression subject to general integer bounds. Such problems exist in elevator scheduling [5] and signal processing [10]. We generate problems the same way as Chang et al. [10], with medium size variable domains ($0 \leq x_i \leq 10, \forall i$) and medium level of noise ($\sigma = 0.05$). We use five sub-sets of the instances with size 10, 20, 30, 40 and 50.

Box-constrained and Ellipsoid-constrained ILS (BEILS) Problem. The BEILS problem can be defined as: $\min_{\boldsymbol{x}\in\mathcal{C}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{C} = \{\boldsymbol{x} \in \mathbb{Z}^n : \|\boldsymbol{A}\boldsymbol{x}\| \leq \alpha^2, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n\}$, where α is a constant. In addition to the least squares objective function, the BEILS problem is also subject to a least squares constraint. Such problem can exist in signal processing [35]. We generate problems the same way as those in Chang et al. [3] on the ellipsoid-constrained ILS problem then add medium size variable domains ($-10 \leq x_i \leq 10, \forall i$). We use five sub-sets of the instances with size 10, 20, 30, 40 and 50.

Quadratic Lateness Scheduling Problem (QLSP) The QLSP can be defined as: $\min \sum_{j=1}^{n}(S_j + p_j - d_j)^2$ s.t. $\texttt{disjunctive}(\{S_1, ..., S_n\}, \{p_1, ..., p_n\})$, $S_j \geq 0, \forall j$, where S_j, p_j, and d_j are the start time, processing time, and due date of job j. The QLSP is a single machine scheduling problem with the goal of minimizing the sum of the quadratic lateness of the jobs. We generate problems the same way as those in Schaller [36]. We use four sub-sets of the instances with size 5, 10, 15 and 20.

Each problem set includes 10 instances of each size, e.g., the BQP problems includes 4 sub-sets of size 10 for 40 instances.

6.2 Results of Experiment 1

We present the results of Experiment 1 in Tables 1 and 2. From Table 1, it is clear that the ELLIPSOID constraint significantly improves the performance of

the default CP Optimizer for the BQPs, the BILS problems, and the BEILS problems both in terms of running time and number of choice points. Without the reasoning from the ELLIPSOID constraint, the default CP Optimizer cannot prove optimality for any instances of these three problem types. For the EQKPs, the ELLIPSOID constraint (BOX) is able to decrease the number of choice points by a factor of 1.5 for the {0,1} problems and almost a factor of 2 for the {0,1,2} problems. But the extra computation makes the running time worse than that of the default CP Optimizer.

For QLSPs, CPO achieves the best average running time. Interestingly, for the instances where all the filtering algorithms are able to prove optimality, the number of choice points is the same for all the filtering algorithms. Our preliminary investigation shows that the `disjunctive` constraint has a significant impact on reducing variable domains for the QLSP, therefore determining the number of choice points regardless of the strength of propagation of the ellipsoid constraint. It is worth pointing out that the convex ellipsoid structure of the QLSP has a simple *axis-aligned* structure, i.e., the off-diagonal entries of H in Equation (3) are all equal to zero. Our results show that the default CPO propagation does not achieve BC for axis-aligned ellipsoids and that the strength of pruning of our three inference algorithm follows the same pattern for axis-aligned ellipsoids as for the general case. As future work, we would like to exploit the axis-aligned property to achieve more efficient filtering.

Among the three filtering algorithms, BOX performs the best in terms of running time to prove optimality. ABC and QCP both find better primal solutions in fewer nodes but, BOX finds better solutions in less time. On problems where all three algorithms are able to prove optimality, the tree size of BOX is about 15% larger than that of ABC and QCP. This is somewhat surprising and suggests that variable fixing leads to strong inference. We observe that the reduced ellipsoid obtained after fixing a variable often has a much tighter tangent box on the unfixed variables compared to that of the original ellipsoid. However, we also observe that ABC can sometimes achieve one order of magnitude improvement compared to BOX in tree size on some larger instances. We would like to investigate the problem characteristics that result in such difference. Note that the lower number of choice points of ABC and QCP are misleading because neither prove optimality for all instances within the time limit. However they are good indicators on the number of nodes that can be visited when these two algorithms are applied.

From Table 2, it is observed that the new branching rules greatly improve the number of choice points, the running time, and the percentage of optimal solutions found for most approaches. However CPO still cannot prove optimality for the BQPs, the BILS problems, and the BEILS problems. The most significant reduction is observed on the BILS problem and the BEILS problem, followed by the QLSP, where the variable domains are much larger than the other two types of problems: a good branching strategy apparently is especially important for problems with large domains. It is particularly striking to see that the BILS

problems are solved at least five orders of magnitude faster by CP through the use of the ellipsoid constraint and the branching rule.

	CPO		BOX		ABC		QCP	
Problem	time	chpts	time	chpts	time	chpts	time	chpts
BQP	-	-	**11.12**	**17169**	1204.89^{85}	10565	-	-
BILS	-	-	**44.61**	**61552**	871.80^{84}	19229	2810.01^{40}	3836
BEILS	-	-	**362.14**^{94}	**167162**	1480.68^{64}	10301	3092.14^{38}	1637
EQKP{0,1}	**23.48**	667740	44.09	**426255**	745.79^{87}	60789	2494.77^{67}	2494
EQKP{0,1,2}	**83.98**	1803982	99.70	**932035**	481.44^{90}	63177	2256.95^{57}	2273
QLSP	**136.34**	**962400**	166.16	**962400**	754.07^{85}	438394	2432.98^{50}	8631

Table 1: A comparison of default CP Optimizer and the three filtering algorithms. Bold numbers indicate the best approach for a given problem set. The symbol '-' means that no problem instances were solved to optimality within 3600 seconds. The superscripts indicate the percentage of instances solved to optimality within 3600 seconds. If no superscript is indicated, all of the instances are solved.

	CPO+b		BOX+b		ABC+b		QCP+b	
Problem	time	chpts	time	chpts	time	chpts	time	chpts
BQP	-	-	**8.28**	**12611**	1099.82^{92}	12575	3595.85^{3}	1069
BILS	-	-	**0.06**	225	5.21	228	314.16	**221**
BEILS	-	-	**0.47**	426	213.47	**394**	1713.89^{82}	360
EQKP{0,1}	**16.40**	247896	24.84	**193269**	578.74^{90}	72134	1868.71^{57}	2550
EQKP{0,1,2}	50.04	521037	**46.77**	**269937**	407.59^{90}	39076	2304.01^{60}	2365
QLSP	**20.63**	**347665**	31.47	**347665**	602.77^{90}	265252	2249.57^{50}	8516

Table 2: A comparison of default CP Optimizer and the three filtering algorithms with the branching rules. All notations are the same as in Table 1.

6.3 Results of Experiment 2

In this section, we compare our best CP results (using the BOX filtering with the branching rules) with the best known exact approaches. From Table 3, it is observed that our new CP approach significantly outperforms the general MINLP solver BARON and it is competitive with state-of-the-art MIP solver CPLEX, running at the same order of magnitude as CPLEX for BQPs and BILS problems. While our CP approach is one order of magnitude slower than CPLEX for EQKPs, it is almost two orders of magnitude faster for BEILS problems and it is significantly better for QLSPs. The reason that CPLEX performs particularly poorly on QLSPs is that the modeling of the disjunctive relationship among the jobs involve big-M constraints, which result in weak dual bounds. As future work, we would like to apply our CP approach to even more complicated scheduling problems, where quadratic component is only one of many components.

The SDP approach, while being the state-of-the-art for the BQPs, is limited to problems with binary domains. Similarly, DEBS cannot be applied to all the problem classes due to its specialized nature. In contrast, BARON is the most

general solver tested here (along with CPO) and these results on strictly convex IQCPs do not reflect its more general problem solving power [15].

	CPO	BOX+b	CPLEX	BARON	DEBS	SDP
Problem	time	time	time	time	time	time
BQP	-	8.28	37.06	38.03	**0.24**	0.69
BILS	-	0.06	0.02	2715.04^{26}	**0.01**	N/A
BEILS	-	**0.47**	14.79	3248.25^{16}	N/A	N/A
EQKP{0,1}	23.48	24.84	4.49	6.23	**0.24**	114.01
EQKP{0,1,2}	83.98	46.77	4.27	429.62^{93}	**0.34**	N/A
QLSP	136.34	**31.47**	1588.15^{65}	1855.02^{50}	N/A	N/A

Table 3: A comparison of default CP Optimizer and our best setting: BOX+b. All notations are the same as in Table 1. The additional symbol "N/A" indicates that the problem cannot be solved with the approach.

7 Conclusion

We propose a CP-based approach to solve strictly convex IQCPs via a novel ELLIPSOID constraint with three different filtering algorithms and variable/value ordering heuristics. The constraint and branching heuristics are based on the geometry of the strictly convex quadratic function. We experiment with a variety of problems and show significant improvement over the default CP Optimizer and competitive results to general state-of-the-art solvers CPLEX and BARON.

For future work, it is interesting to experiment with our filtering algorithms on other problem types. Although ABC and QCP perform worse than BOX for the problem types tested here, it is possible that ABC and QCP can perform better for problems where variable fixings do not take place frequently. For the same reason it is also interesting to implement the filtering algorithms in a MIP-based solver such as SCIP [37] where branching is typically done by tightening variable bounds instead of fixing variables. In general, our technique can be integrated with other solvers provided they represent bounds on variables and have an"inference loop".

More broadly, we propose that it may be possible to develop CP as a basis for MINLP. MINLPs are challenging optimization problems that arise in many industrial applications. As CP is not dependent on a strong linear relaxation to bound the search, it may be valuable to study inferences that can be made for common non-linear constraints as we have done here. We hope that this paper might serve as a step in this direction.

8 Acknowledgement

We would like to thank Nick Sahinidis for the BARON license and Felipe Serrano and Benjamin Müller for valuable discussions. This research has been supported by the Natural Sciences and Engineering Research Council of Canada and the University of Toronto School of Graduate Studies Doctoral Completion Award.

References

1. van Emde-Boas, P.: Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Mathematisch Instituut, Amsterdam, The Netherlands (1981)
2. Agrell, E., Eriksson, T., Vardy, A., Zeger, K.: Closest point search in lattices. Information Theory, IEEE Transactions on **48**(8) (2002) 2201–2214
3. Chang, X.W., Golub, G.H.: Solving ellipsoid-constrained integer least squares problems. SIAM Journal on Matrix Analysis and Applications **31**(3) (2009) 1071–1089
4. Teunissen, P.J., Kleusberg, A., Teunissen, P.: GPS for Geodesy. Volume 2. Springer Berlin (1998)
5. Kuusinen, J.M., Sorsa, J., Siikonen, M.L.: The elevator trip origin-destination matrix estimation problem. Transportation Science **49**(3) (2014) 559–576
6. Pesant, G., Régin, J.C.: Spread: A balancing constraint based on statistics. In: Principles and Practice of Constraint Programming-CP 2005. Springer (2005) 460–474
7. Domes, F., Neumaier, A.: Constraint propagation on quadratic constraints. Constraints **15**(3) (2010) 404–429
8. Lebbah, Y., Michel, C., Rueher, M.: A rigorous global filtering algorithm for quadratic constraints. Constraints **10**(1) (2005) 47–65
9. Ku, W.Y., Beck, J.C.: Combining discrete ellipsoid-based search and branch-and-cut for binary quadratic programming problems. In: Integration of AI and OR Techniques in Constraint Programming. Springer (2014) 334–350
10. Chang, X.W., Han, Q.: Solving box-constrained integer least squares problems. Wireless Communications, IEEE Transactions on **7**(1) (2008) 277–287
11. Ku, W.Y., Beck, J.C.: Combining discrete ellipsoid-based search and branch-and-cut for integer least squares problems. Submitted to IEEE Transactions on Wireless Communications (2014)
12. Ku, W.Y., Beck, J.C.: Combining constraint propagation and discrete ellipsoid-based search to solve the exact quadratic knapsack problem. In: Integration of AI and OR Techniques in Constraint Programming. Springer (2015) 231–239
13. Golub, G.H., Van Loan, C.F.: Matrix computations. Volume 3. JHU Press (2012)
14. Sahinidis, N.V.: BARON 14.3.1: Global Optimization of Mixed-Integer Nonlinear Programs, *User's Manual*. (2014)
15. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming **103** (2005) 225–249
16. Misener, R., Floudas, C.A.: ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations. Journal of Global Optimization (2014) DOI: 10.1007/s10898-014-0166-2.
17. Bussieck, M.R., Vigerske, S.: MINLP solver software. Wiley Encyclopedia of Operations Research and Management Science, Wiley, Chichester (2010)
18. Krislock, N., Malick, J., Roupin, F.: Improved semidefinite bounding procedure for solving max-cut problems to optimality. Mathematical Programming (2012) 1–26
19. Krislock, N., Malick, J., Roupin, F.: BiqCrunch solver. http://lipn.univ-paris13.fr/BiqCrunch/download (Retrieved: 04/12/2016)
20. Borno, M.A.: Reduction in solving some integer least squares problems. arXiv preprint arXiv:1101.0382 (2011)

21. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming **66**(1) (1994) 181–199
22. Dechter, R.: Constraint processing. Morgan Kaufmann (2003)
23. Gleixner, A.M.: Exact and fast algorithms for mixed-integer nonlinear programming. PhD thesis, Technische Universität Berlin (2015)
24. Nesterov, Y., Nemirovskii, A., Ye, Y.: Interior-point polynomial algorithms in convex programming. Volume 13. SIAM (1994)
25. Bonami, P., Tramontani, A.: Advances in CPLEX for mixed integer nonlinear optimization. Presented at ISMP 2015, Pittsburgh, PA (2015)
26. Lewis, M., Alidaee, B., Kochenberger, G.: Using xqx to model and solve the uncapacitated task allocation problem. Operations research letters **33**(2) (2005) 176–182
27. FlNKE, G., Burkard, R.E., Rendl, F.: Quadratic assignment problems. Surveys in combinatorial optimization **132** (2011) 61–82
28. Carter, M.W.: The indefinite zero-one quadratic problem. Discrete Applied Mathematics **7**(1) (1984) 23–44
29. Létocart, L., Plateau, M.C., Plateau, G.: An efficient hybrid heuristic method for the 0-1 exact k-item quadratic knapsack problem. Pesquisa Operacional **34**(1) (2014) 49–72
30. Martí, R., Gallego, M., Duarte, A.: A branch and bound algorithm for the maximum diversity problem. European Journal of Operational Research **200**(1) (2010) 36–44
31. Caprara, A., Pisinger, D., Toth, P.: Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing **11**(2) (1999) 125–137
32. Caprara, A., Kellerer, H., Pferschy, U., Pisinger, D.: Approximation algorithms for knapsack problems with cardinality constraints. European Journal of Operational Research **123**(2) (2000) 333–345
33. Turner, S., Romero, D., Zhang, P., Amon, C., Chan, T.: A new mathematical programming approach to optimize wind farm layouts. Renewable Energy **63** (2014) 674–680
34. Zhang, P.Y., Romero, D.A., Beck, J.C., Amon, C.H.: Solving wind farm layout optimization with mixed integer programs and constraint programs. EURO Journal on Computational Optimization **2**(3) (2014) 195–219
35. Damen, M.O., El Gamal, H., Caire, G.: On maximum-likelihood detection and the search for the closest lattice point. Information Theory, IEEE Transactions on **49**(10) (2003) 2389–2402
36. Schaller, J.: Single machine scheduling with early and quadratic tardy penalties. Computers & Industrial Engineering **46**(3) (2004) 511–532
37. Achterberg, T.: SCIP: solving constraint integer programs. Mathematical Programming Computation **1**(1) (2009) 1–41