J Glob Optim
DOI 10.1007/s10898-017-0521-1

A branch and bound algorithm for nonconvex quadratic optimization with ball and linear constraints

Amir Beck[1] · **Dror Pan**[1]

Received: 15 September 2016 / Accepted: 8 April 2017

Abstract We suggest a branch and bound algorithm for solving continuous optimization problems where a (generally nonconvex) objective function is to be minimized under nonconvex inequality constraints which satisfy some specific solvability assumptions. The assumptions hold for some special cases of nonconvex quadratic optimization problems. We show how the algorithm can be applied to the problem of minimizing a nonconvex quadratic function under ball, out-of-ball and linear constraints. The main tool we utilize is the ability to solve in polynomial computation time the minimization of a general quadratic under one Euclidean sphere constraint, namely the so-called *trust region subproblem*, including the computation of all local minimizers of that problem. Application of the algorithm on *sparse source localization* problems is presented.

Keywords Quadratically constrained quadratic problems · Nonconvex programming · Branch and bound · Sparse source localization · Trust region subproblem

1 Introduction

We consider a general nonlinear constrained optimization problems of the form

$$\text{(P)} \quad \begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & f_0(\mathbf{x}) \\ \text{s.t.} & f_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m, \end{array}$$

where the functions $f_0, f_1, \dots, f_m : \mathbb{R}^n \rightarrow \mathbb{R}$ are continuous. None of the functions is assumed to be convex. Nonconvex optimization problems are considered hard in general. Methods seeking for stationary points exist in the literature, such as gradient, Newton-based

✉ Amir Beck
becka@ie.technion.ac.il

Dror Pan
dror.pan@campus.technion.ac.il

[1] Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology, Haifa, Israel

Published online: 17 April 2017

algorithms, Lagrange multiplier based methods and trust region method; see e.g., the classical books of Bertsekas [6] and Nocedal and Wright [21]. However, general methods for computing a global optimal solution for the general case do not exist for these problems, except for some special classes of problems. In this paper we will develop a branch and bound type algorithm for solving a special class of problem (P). We focus on the implementation of the proposed method on *quadratically constrained quadratic problems* (QCQP). More specifically, the examples considered in this paper are special classes QCQPs of the form

$$\begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & q_0(\mathbf{x}) \equiv \frac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ \text{s.t.} & \|\mathbf{x}-\mathbf{w}_i\|^2 \le d_i^2, \quad i = 1,\ldots,m, \\ & \|\mathbf{x}-\mathbf{w}_i\|^2 \ge d_i^2, \quad i = m+1,\ldots,m+p, \\ & \boldsymbol{\alpha}_i^T\mathbf{x} \le \beta_i, \qquad i = m+p+1,\ldots,m+p+l, \end{array} \tag{1.1}$$

where $m \ge 1,\ p,l \ge 0,\ \mathbf{Q}_0 \in \mathbb{S}^n,\ \mathbf{b}_0 \in \mathbb{R}^n,\ \mathbf{w}_i, \boldsymbol{\alpha}_i \in \mathbb{R}^n, \beta_i \in \mathbb{R},\ d_i > 0\,\forall i$. Such a problem can be seen as an extension of the standard *trust region subproblem* (TRS), which is the case where only one ball constraint is involved ($m = 1,\ p,l = 0$).

The main difficulty to reach a global optimal solution of a general nonconvex problem of the described type is the large number of local minima it might harbor. Nevertheless, simple cases are known to be tractable, such as the TRS problem, where here we refer to the "TRS problem" as the problem consisting of minimizing a (possibly) nonconvex function over an inequality or equality Euclidean norm constraint. Gay [13], Sorensen [22], and Moré and Sorensen [20] showed that a TRS problem, though nonconvex, possesses necessary and sufficient optimality conditions, and that its global solution can be found by a simple root search procedure on a strictly monotone, strictly convex one-variable function.

Martínez showed in [18] that the local-non-global minimizers of the TRS can also be found utilizing a root search on a one-variable function with "good" properties. In Sect. 4 we utilize the ability to efficiently compute all the local minimizers of a TRS to show how our general algorithm can be applied for solving (1.1). A *generalized trust region subproblem* (GTRS), in which a general quadratic constraint appears instead of the norm constraint, is also known to be tractable. Moré formulated in [19] necessary and sufficient conditions for global minimizers of a GTRS. This result was utilized by Beck, Stoica and Li [3] in solving a squared least squares formulation of the source localization problem, which will be discussed in Sect. 5.

Some simple extensions of the standard TRS have also been shown to be tractable. The case of (1.1) with only one added linear constraint ($m = 1,\ p = 0,\ l = 1$) can be solved in polynomial computation time though its semidefinite relaxation is not tight in general; see Strurm and Zhang [23]. Ye and Zhang [24] also treated problem (1.1) for the case involving one ball ($m = 1,\ p = 0$) and two linear inequalities ($l = 2$) which are parallel (a two-sided linear inequality constraint), and showed a polynomial-time algorithm, based on a convex relaxation containing second order cone (SOC) and semidefinite programming (SDP) constraints. Burer and Anstreicher [9] improved the complexity result of [24] utilizing the existence of extreme point solutions of the relaxation. Burer and Yang [10] generalized the result to the setting $m = 1,\ p = 0$ and a general l, under the assumption of *non-intersecting faces*: for each two distinct i and j, the *face* defined by

$$\{\mathbf{x} \in \mathbb{R}^n : \quad \boldsymbol{\alpha}_i^T\mathbf{x} = \beta_i,\ \boldsymbol{\alpha}_j^T\mathbf{x} = \beta_j\}$$

does not intersect the feasible set. Under this assumption, they showed that the same convex relaxation in [9] still has no gap with the original problem (1.1).

Problem (1.1) with $m = 1$, $p = 0$ and a general l, that is, the *extended trust region subproblem*

$$\begin{aligned} \min \;& q_0(\mathbf{x}) \equiv \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ \text{s.t.} \;& \|\mathbf{x}-\mathbf{w}\|^2 \leq d^2, \\ & \boldsymbol{\alpha}_k^T\mathbf{x} \leq \beta_k, \qquad k = 1,\ldots,l \end{aligned} \tag{1.2}$$

has been recently studied by Jeyakumar and Li [17]. In general, it has neither a tight SDP-relaxation nor a strong duality property. They formulated a dimension condition (generalizing a previous result of Beck and Eldar [2, Section 4.3] for $l = 1$), under which the problem admits an exact SDP-relaxation, and thus can be solved in polynomial running time. Hsia and Sheu [16] recently improved that dimension condition. In addition, they showed that for a fixed number of linear constraints l, problem (1.2) can be solved in polynomial time in n. On the other hand, they have shown that the class of all the problems (1.2) where both n and l are arbitrary is NP-hard.

The case $m = 1$, $p = 0$ and a general l (that is, problem (1.2)) is important as it can be a building block in a type of trust region methods for constrained minimization problems. In such methods, a quadratic approximation is minimized at each iteration over the intersection between the trust region (ball) and the polyhedron defined by the linear inequalities gained by linear approximations of the constraints. An important simple case with just bound constraints on the variables is discussed and solved in [11, Section 5], a case to which we also refer in our numerical experiences of this paper.

Bienstock and Michalka [7] suggested a different algorithm for solving problem (1.1). Their algorithm enumerates candidates for the optimal solution on each face of the feasible domain. Here a face is defined by any subset of the linear constraints restricted to hold as equalities. They proved that their algorithm computes a global minimizer of (1.1) in polynomial running time in n, m, p, l and in the cardinality of $\mathcal{F}^*$, the pre-computed set of all the faces of the polyhedron defined by the linear inequalities which intersect the domain defined by the ball constraints. In their work, they also described some possible applications of problem (1.1) for solving combinatorial optimization problems. While their paper provides an important theoretical result concerning the polynomial complexity in special cases of the problem, the actual size $|\mathcal{F}^*|$ can be very large in many practical cases.

In this paper, we suggest an algorithm that solves problem (1.1) globally, as a special case of problem (P) which satisfies some specific properties we define. We do not assume that the number of intersecting faces is small. Unlike the method of [7] (and [16] for the case of problem (1.2)), our algorithm does not perform an exhaustive search of candidates through *all* the intersecting faces. Instead, we apply *branch and bound* (BB) scheme, in which tractable relaxations are solved to compute bounds on the optimal value. The principle of enumeration candidates for the global solution utilizes an approach similar to [7,16], as well as the technique of solving a subproblem. However, the local and the global solutions of the subproblem give effective lower bounds on the optimal value of (P), which enable to *fathom* some of the nodes and avoid further branching of such faces. That is, our algorithm can enumerate much less "faces" than the aforementioned algorithms in many practical cases, as numerical results show.

The rest of the paper is organized as follows. In Sect. 2 we describe the general model (P), including the assumptions that should hold to ensure the ability to implement the main algorithm we suggest. We also explain why those assumptions hold for problem (1.1). In Sect. 3 we describe the promised algorithm, based on a BB scheme, along with all the concepts and the required notations. Section 4 describes the techniques to be applied for solving the relaxation and the other subproblems appearing during the BB-based algorithm in the case of problem (1.1), including a review of the main results on the trust region subproblem. In Sect. 5 we suggest an application of our algorithm to the *sparse source localization* problem, a nonsmooth and nonconvex optimization problem, by applying the algorithm on several problems of the form of (1.1) with a simpler objective. Finally, numerical experiments are provided in Sect. 6 that demonstrate the effectiveness of our method on problem (1.1) with one ball constraint and several linear constraints. A numerical comparison with the algorithm of [7] is also given for that case. In addition, we show the utilization of the algorithm for some sparse source localization examples.

Notations We denote scalars in italic (e.g., $a, b, c, C, L, U, \ldots$), vectors in boldface lower case ($\mathbf{v}, \mathbf{x}, \ldots$) and matrices in boldface upper case ($\mathbf{A}, \mathbf{B}, \ldots$). In addition, sets are written in italic upper case (S, R). The space $\mathbb{S}^n$ is the subspace of $\mathbb{R}^{n\times n}$ comprising all symmetric matrices. For any two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{S}^n$, $\mathbf{A} \succeq \mathbf{B}$ ($\mathbf{A} \succ \mathbf{B}$) means that $\mathbf{A} - \mathbf{B}$ is positive semidefinite (positive definite). $\mathbf{e}$ denotes the all-ones column vector, $\mathbf{I}$ the identity matrix, and $\mathbf{0}$ the all-zeros vector or matrix (the dimensions are clear by the context). The notation $\|\cdot\|$ denotes the Euclidean l_2-norm on vectors in $\mathbb{R}^p$ and the spectral norm on matrices in $\mathbb{R}^{p\times q}$.

2 The model

The model we consider is the following general nonlinear optimization problem:

$$(\mathrm{P}) \quad \begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & f_0(\mathbf{x}) \\ \text{s.t.} & f_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m, \end{array}$$

where the functions $f_0, f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ are continuous.

For any two sets $E, I \subseteq \{1, \ldots, m\}$ satisfying $E \cap I = \emptyset$, we associate a problem, denoted by ($\mathrm{P}_{E,I}$), given by

$$(\mathrm{P}_{E,I}) \quad \begin{array}{lll} \min & f_0(\mathbf{x}) & \\ \text{s.t.} & f_i(\mathbf{x}) \le 0, & i \in I, \\ & f_i(\mathbf{x}) = 0, & i \in E. \end{array}$$

In addition, in the cases where $I = \emptyset$, we use the notation (P_E) rather than ($\mathrm{P}_{E,\emptyset}$), that is,

$$(\mathrm{P}_E) \quad \begin{array}{ll} \min & f_0(\mathbf{x}) \\ \text{s.t.} & f_i(\mathbf{x}) = 0, \quad i \in E. \end{array}$$

A constraint $f_i(\mathbf{x}) \le 0$ of problem (P) is called **regularizing** if for any two disjoint sets $E, I \subseteq \{1, \ldots, m\}$ satisfying $i \in E \cup I$, the problem ($\mathrm{P}_{E,I}$) is either **infeasible** or **solvable**. That is, any problem of the form ($\mathrm{P}_{E,I}$) with the regularizing constraint is involved either has an empty feasible domain, or it attains its minimum at a feasible point (a global minimizer). To assure the ability to develop our method, we further assume that the following properties hold.

(1) There exists $i_0 \in \{1, \ldots, m\}$ such that the constraint $f_{i_0}(\mathbf{x}) \le 0$ is regularizing.
(2) For any subset $E \subseteq \{1, \ldots, m\}$ there exists an oracle* which can compute all the local minimizers of the problem (P_E).
(3) For any subset $E \subseteq \{1, \ldots, m\}$ the set of all local minimizers of problem (P_E) is either path-connected with no local-non-global minimizers, or a finite set.

*By "oracle" we mean an algorithm that we can apply. Though we do not require that it would be polynomial, it should be practically efficient, as it is used as a sub-routine within the full method.

The assumptions (1), (2) and (3) restrict the discussion to specific types of problems on which our method could be applied. For any problem satisfying properties (1), (2) and (3), we assume without loss of generality that $i_0 = 1$ in assumption (1).

In the next section we describe the main construction: an algorithm based on the general approach of branch and bound (BB). During the execution of the algorithm, a tree of subproblems of the form ($\mathrm{P}_{E,I}$) is being built. The basic relaxation of (P), which is the root node of the BB tree, is the unconstrained minimization of f_0, namely ($\mathrm{P}_{\emptyset,\emptyset}$). At each branching step one constraint is added either as an equality or as an inequality, resulting in an addition of its index to either E or I. Throughout the proof of the correctness of the proposed algorithm in Sect. 3 we utilize assumptions (1), (2) and (3). We show that at each step of the algorithm, a lower bound on the relevant problem ($\mathrm{P}_{E,I}$) can be computed utilizing the ability to compute all the local minimizers of problems of the form ($\mathrm{P}_{E,I}$). Notice that assumption (2) in particular guarantees that even if f_0 does not admit a finite minimum over $\mathbb{R}^n$, all its unconstrained local minimizers are computable.

Although we suggest the algorithm for solving a general optimization problem in the form (P) under assumptions (1), (2) and (3), we actually focus in this paper only on the quadratic problem (1.1). The general model is provided only to show that our approach may be generalized to other families of challenging optimization problems, and to highlight the key properties of the class of problems that guarantee the validity of the analysis. It is important to notice that the question whether the above three assumptions hold or not can itself be difficult to answer for the general case of (P). However, for problem (1.1) with general $m \geq 1,\ p, l \geq 0$, the assumptions are satisfied, as we show in Sect. 4.

3 The main branch and bound algorithm

In this section we devise our algorithm for globally solving problem (P) under assumptions (1), (2) and (3). First, we assume a fixed and given order of the constraints (later on, in Sect. 3.1, we show a better strategy for ordering the constraints). The regularizing constraint in assumption (1) is assumed to be the first ($i = 1$). For any index $i \in \{1, \dots, m\}$ and set $E \subseteq \{1, \dots, i\}$ we associate a node $[i, E]$.

For each node $[i, E]$, we associate a problem of the form ($\mathrm{P}_{E,I}$) with $I = \{1, \dots, i\} \backslash E$. That is, each node $[i, E]$ is associated with a problem of the form ($\mathrm{P}_{E,I}$) derived form (P) by taking into account only its first i constraints, and enforcing the constraints whose indices reside in E to be active.

For given i and E, we refer to problem ($\mathrm{P}_{E,I}$) as problem ($[i, E]$). At each layer i, the nodes $[i, E]$ for all $E \subseteq \{1, \dots, i\}$ represent all the possibilities for choosing an active subset of the inequalities $\{1, \dots, i\}$, and thus, as we see next, solving them all enables to find an optimal solution of problem ($[i, \emptyset]$). Problem ($[i, \emptyset]$) can be considered as a relaxation of problem (P), considering only the first i constraints (with $E = \emptyset,\ I = \{1, \dots, i\}$, that is, no activeness enforcement). During the algorithm, we basically seek to solve such relaxations and other subproblems of the form ($\mathrm{P}_{E,I}$). Upper bounds on the optimal value of problem (P) are obtained at solutions of subproblems ($\mathrm{P}_{E,I}$) which are also feasible for (P). The full tree contains $m + 1$ layers, each containing 2^i nodes, where node $[0, \emptyset]$ is associated with the unconstrained minimization problem ($\mathrm{P}_{\emptyset,\,\emptyset}$), induced by removing all the constraints of (P).

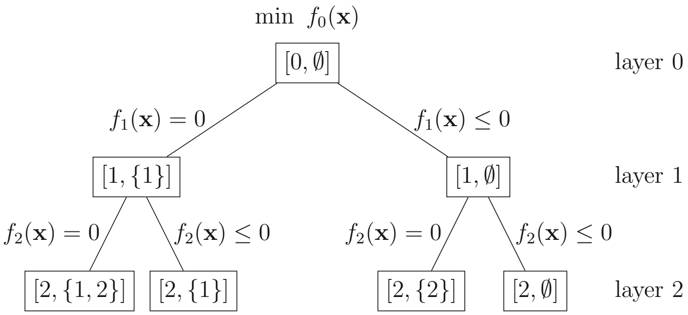

Fig. 1 A tree after branching on 2 constraints

The children of each node $[i, E]$ are the nodes $[i+1,\ E \cup \{i+1\}]$ and $[i+1,\ E]$, and they correspond to adding the $(i+1)$th constraint as $f_{i+1}(\mathbf{x}) = 0$ or $f_{i+1}(\mathbf{x}) \leq 0$, respectively.

Figure 1 provides an illustration of a tree after branching on 2 constraints.

The algorithm we are about to construct has to solve problems of the form ($\mathrm{P}_{E,I}$). Under assumption (1), starting from layer 1, all such problems are solvable or infeasible, since we require that the first chosen constraint ($f_1(\mathbf{x}) \leq 0$) is a regularizing constraint. We now explain how problem ($\mathrm{P}_{E,I}$) can be solved by utilizing all the local minimizers of problems of the form (P_E). The following key result enables us to compute all the candidates for solutions of problem ($\mathrm{P}_{E,I}$) by solving relaxed problems of the form (P_E).

Lemma 3.1 *Consider a general minimization problem of the form*

$$\begin{array}{lll} \text{(PX)} & \min & f(\mathbf{x}) \\ & \text{s.t.} & \mathbf{x} \in X, \\ & & h_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, q. \end{array}$$

where $X \subseteq \mathbb{R}^n$, $h_i : \mathbb{R}^n \to \mathbb{R}$ *are continuous functions for all* $i = 1, \ldots, q$, *and* $f : \mathbb{R}^n \to \mathbb{R}$.

Let $\mathbf{x}^*$ *be a local minimizer of (PX). Denote*

$$E = \{i \in \{1, \ldots, q\} : h_i(\mathbf{x}^*) = 0\}$$

and

$$I = \{i \in \{1, \ldots, q\} : h_i(\mathbf{x}^*) < 0\}.$$

Then $\mathbf{x}^*$ *is a local minimizer of the problem*

$$\begin{array}{lll} (\mathrm{PX}_E) & \min & f(\mathbf{x}) \\ & \text{s.t.} & \mathbf{x} \in X, \\ & & h_i(\mathbf{x}) = 0, \quad \forall i \in E. \end{array}$$

Proof Since $\mathbf{x}^*$ is a local minimizer of (PX), it follows that there exists $\delta_1 > 0$ such that

$$f(\mathbf{x}) \geq f(\mathbf{x}^*) \text{ for any } \mathbf{x} \in X \text{ satisfying } \|\mathbf{x} - \mathbf{x}^*\| < \delta_1 \text{ and } h_i(\mathbf{x}) \leq 0\ \forall i \in \{1, \ldots, q\}. \tag{3.1}$$

In addition, the function $g(\mathbf{x}) := \max_{i \in I} h_i(\mathbf{x})$ is continuous as a maximum of $|I|$ continuous functions. Therefore, since $g(\mathbf{x}^*) < 0$, there exists $\delta_2 > 0$ such that

$$g(\mathbf{x}) < 0 \text{ for any } \mathbf{x} \text{ satisfying } \|\mathbf{x} - \mathbf{x}^*\| < \delta_2. \tag{3.2}$$

Let $\mathbf{x} \in X$ be an arbitrary vector assumed to satisfy $\|\mathbf{x} - \mathbf{x}^*\| < \min\{\delta_1, \delta_2\}$ along with $h_i(\mathbf{x}) = 0$ for all $i \in E$. Then by (3.2), $\mathbf{x}$ satisfies $h_i(\mathbf{x}) \leq g(\mathbf{x}) < 0$ for all $i \in I$, and thus by (3.1), it follows that $\mathbf{x}$ satisfies $f(\mathbf{x}) \geq f(\mathbf{x}^*)$. We can thus conclude that $\mathbf{x}^*$ is a local minimizer of (PX$_E$). □

Corollary 3.1 *Any local minimizer* $\mathbf{x}^*$ *of problem (PX) in Lemma 3.1 is necessarily a local minimizer of a problem of the form* (PX_E) *for some* $E \subseteq \{1, \ldots, q\}$.

Proof Define E to be the set of all the indices of the active constraints in $\mathbf{x}^*$, and the result immediately follows by Lemma 3.1. □

Corollary 3.2 *Consider the problem*

$$\min\{f(\mathbf{x}) : \mathbf{x} \in X,\ h(\mathbf{x}) \leq 0\}, \tag{3.3}$$

where $X \subseteq \mathbb{R}^n$, $h : \mathbb{R}^n \to \mathbb{R}$ *is a continuous function, and* $f : \mathbb{R}^n \to \mathbb{R}$. *Let* $\mathbf{x}^*$ *be a local minimum of* (3.3). *Then one of the following cases occurs:*

(1) $\mathbf{x}^*$ *is a local minimizer of*

$$\min\{f(\mathbf{x}) : \mathbf{x} \in X\}$$

with $h(\mathbf{x}^*) < 0$, *or,*

(2) $\mathbf{x}^*$ *is a local minimizer of*

$$\min\{f(\mathbf{x}) : \mathbf{x} \in X,\ h(\mathbf{x}) = 0\}.$$

Proof The result follows immediately by applying Lemma 3.1 with $q = 1$. Suppose that $\mathbf{x}^*$ is a local minimizer of (3.3). If $h(\mathbf{x}^*) < 0$, then we have $E = \emptyset$ and $I = \{1\}$, and by Lemma 3.1, case (1) occurs. If $h(\mathbf{x}^*) = 0$, we have $E = \{1\}$ and $I = \emptyset$, and thus, case (2) occurs. □

In terms of problem (P$_{E,I}$) corresponding to a node $[i, E]$, Lemma 3.1 states that any local minimizer $\mathbf{x}^*$ of (P$_{E,I}$) with $I = \{1, \ldots, i\} \backslash E$ which satisfies $f_i(\mathbf{x}^*) < 0$ for all $i \in I$ is a local minimizer of the corresponding problem (P$_E$). More generally, any local minimizer $\mathbf{x}^*$ of (P$_{E,I}$) which satisfies $f_i(\mathbf{x}^*) < 0$ for all $i \in J$ for some $J \subseteq I$, and $f_i(\mathbf{x}^*) = 0$ for all $i \in I \backslash J$ is a local minimizer of the problem (P$_{\tilde{E}}$) with $\tilde{E} = E \cup (I \backslash J)$.

As a result, suppose we computed all the local minimizers of all the problems of the form (P$_{\tilde{E}}$) for all $\tilde{E}$ satisfying $E \subseteq \tilde{E} \subseteq E \cup I = \{1, \ldots, i\}$. Then by Corollary 3.1 all the local minimizers of (P$_{E,I}$) must reside among these minimizers. In addition, by Corollary 3.2, as a problem ($[i + 1,\ E]$) is obtained by adding one inequality ($f_{i+1}(\mathbf{x}) \leq 0$) to problem ($[i, E]$), its local minimizers must be either

- local minimizers of ($[i, E]$) which **strictly** satisfy the added inequality, or
- local minimizers of ($[i + 1,\ E \cup \{i + 1\}]$).

Our algorithm utilizes the above observations whenever it has to solve a problem of the form (P$_{E,I}$) at a given node.

The algorithm basically computes lower bounds on nodes of the tree that are used to decide which nodes can be *fathomed*, that is, closed for further branching. It also computes candidates for an optimal solution for each node's subproblem. The following notations are required to specify the previous information needed for computing candidates and bounds for a given node.

Definition 3.1 Given i and $E \subseteq \{1, \dots, i\}$, a vector $\mathbf{x}$ is called an **$([i, E])$-relaxed solution** if it is a local minimizer of problem (P_E).

Denote by $R_{i,E}$ the set of all $([i, E])$-relaxed solutions. Note that not all the $([i, E])$-relaxed solutions are necessarily $([i, E])$-feasible, as problem (P_E) does not include the inequality constraints of problem $([i, E])$, which is $(\mathrm{P}_{E,I})$ with $I = \{1, \dots, i\} \backslash E$. Under assumption (3), the sets $R_{i,E}$ are either finite or have a special property of being path-connected, with a constant objective value over the whole set $R_{i,E}$. To this end, we assume that all such sets are finite for the sake of simplicity of the described algorithm, and later on we will modify the algorithm to apply the more delicate case, where some relaxed-solutions sets are infinite (but assumption (3) still holds).

Denote by $F_{i,E} \subseteq \mathbb{R}^n$ the feasible set of problem $([i, E])$. The set $F_{i,E}$ is closed as an intersection of the closed level sets of the continuous functions f_i. Recall that in assumption (1) we require that problem $([i, E])$ is either solvable or infeasible for all i and $E \subseteq \{1, \dots, i\}$. Under this assumption we can solve problem $([i, E])$ using the following concept.

Definition 3.2 Let $i \in \{1, \dots, m\}$, $E \subseteq \{1, \dots, i\}$. If $i \in E$, we define

$$C_{i,E} \equiv \bigcup_{\tilde{E}:\, E \subseteq \tilde{E} \subseteq \{1,\dots,i\}} (R_{i,\tilde{E}} \cap F_{i,E})$$

and if $i \notin E$, we define

$$C_{i,E} \equiv C_{i,E\cup\{i\}} \cup \left(C_{i-1,\,E} \cap F_{i,E}\right).$$

In addition, $C_{0,\emptyset}$ is the set of all the local minimizers of the unconstrained problem associated with node $[0, \emptyset]$. For each node $[i, E]$, $E \subseteq \{1, \dots, i\}$, the set $C_{i,E}$ is called the **candidates set** of node $[i, E]$.

The following lemma is based on Lemma 3.1, and it shows that the candidate set always contains all the local minima corresponding with the associated node.

Lemma 3.2 *Let $i \in \{1, \dots, m\}$.*

(1) Any local minimizer $\mathbf{x}^*$ problem of $([i, E])$ for $E \subseteq \{1, \dots, i\}$ satisfies $\mathbf{x}^* \in C_{i,E}$.
(2) For $i \geq 1$ an optimal solution of problem $([i, E])$ is a vector with minimal objective value over the set $C_{i,E}$.

Proof (1) First assume that $i \in E$, and $I = \{1, \dots, i\} \backslash E$. There exist $2^{|I|}$ options where a local solution $\mathbf{x}^*$ of $([i, E])$ can reside; each option is defined by a subset $\tilde{E} \subseteq \{1, \dots, i\}$ containing E. Applying Corollary 3.1 on $([i, E])$, we conclude that $\mathbf{x}^*$ is an $([i, \tilde{E}])$-relaxed solution for some $E \subseteq \tilde{E} \subseteq \{1, \dots, i\}$. It is also $([i, E])$-feasible, and thus, $\mathbf{x}^* \in R_{i,\tilde{E}} \cap F_{i,E} \subseteq C_{i,E}$.

Now assume that $i \notin E$. We apply induction on $i = 1, \dots, m$, and show that any local minimizer of $([i, E])$ belongs to the set $C_{i,E}$ as defined. If $i = 1$, then $E = \emptyset$, and we should consider the set $C_{1,\emptyset} = C_{1,\{1\}} \cup \left(C_{0,\emptyset} \cap F_{1,\emptyset}\right)$. The set $C_{0,\emptyset}$ contains by definition all the local minimizers of the unconstrained problem. By Corollary 3.2, any local minimizer of $([1, \emptyset])$ is either a local minimizer of the unconstrained problem (in node $[0, \emptyset]$) that also satisfies the first constraint, and thus belongs to $C_{0,\emptyset} \cap F_{1,\emptyset}$; or a local minimizer of $([1, \{1\}])$, and thus, belongs to $C_{1,\{1\}}$, by the (already proven) result for the nodes with E containing the index i.

Assume now that all the local minimizers of problem ($[i-1,\ E]$) are included in the set $C_{i-1,\,E}$. Again, by Corollary 3.2, a local minimizer $\mathbf{x}^*$ of ($[i, E]$) can either be a local minimizer of ($[i, E \cup \{i\}]$), and thus $\mathbf{x}^* \in C_{i,E\cup\{i\}}$ by the result for nodes where $i \in E$; or a local minimizer of ($[i-1,\ E]$) that also satisfies the i-th inequality, and thus, by the induction assumption, $\mathbf{x}^* \in C_{i-1,\,E} \cap F_{i,E}$. In both cases $\mathbf{x}^* \in C_{i,E}$.

(2) A direct result of the facts that problem ($[i, E]$) is solvable (if feasible) by assumption (1), and that a global solution is also a local solution. □

We should note that $C_{i,E}$ is not necessarily the set of all local minimizers of problem ($[i, E]$), but it rather only guaranteed to contain all of those minimizers.

Example 3.1 Consider the case where problem (P) is given by

$$
\begin{aligned}
\text{(P)} \qquad \min_{\mathbf{x}\in\mathbb{R}^2} \quad & x_1^2 - x_2^2 \\
\text{s.t.} \quad & x_1^2 + x_2^2 \le 1, \\
& x_2 \le 0, \\
& -x_2 \le 0.8,
\end{aligned}
$$

where the quadratic constraint enables assumption (1) to hold. The point $\mathbf{x} = (0, 0)^T$ is a local minimizer of problem ($[2, \{2\}]$), which is the problem containing the quadratic inequality (ball) and the linear equality constraint $x_2 = 0$. Thus, by the definition of the candidates sets, it belongs to $C_{2,\emptyset}$. However, it is not a local minimizer of problem ($[2, \emptyset]$), which is the problem containing the first two original inequalities; it is rather a saddle point of it. However, $C_{2,\emptyset}$ also contains the candidate $\bar{\mathbf{x}} = (0, -1)^T$, which was included in $C_{1,\emptyset}$, and has a lower objective value than $\mathbf{x}$ has. Thus, the lower bound on $[2, \emptyset]$ is correctly computed. The optimal solution of (P) is $\mathbf{x}^* = (0, -0.8)^T$, and it is obtained at node $[3, \{3\}]$.

Lemma 3.2 particularly enables to solve nodes of the form $[i, \{1, \ldots, i\}]$, as at such nodes the only possible $\tilde{E}$ containing $E = \{1, \ldots, i\}$ is the set E itself. Solving such nodes resorts to solving problem (P_E) without using any previous computed solutions. As for a general $E \subseteq \{1, \ldots, i\}$, it seems that to compute the candidates sets $C_{i,E}$ where $i \in E$ we need to store all the sets $R_{i,\tilde{E}}$ for all $\tilde{E} \subseteq \{1, \ldots, i\}$ containing E. In addition, the number of such sets can have size exponential in $|E|$, and thus, each computation of $C_{i,E}$ seems to require looking at exponential number of sets.

However, Lemma 3.3 that follows gives an equivalent formula to compute the candidates sets $C_{i,E}$ where $i \in E$, which does not require knowledge of any of the sets $R_{i,\tilde{E}}$ except for $R_{i,E}$, that is, the relaxed solutions of the node $[i, E]$ itself. It recursively builds the sets $C_{i,E}$, based on adding one equality constraint to E at each recursive call. In that way, only a linear number (in i, $|E|$) of sets needs to be considered at each computation. For a fixed $i \in \{0, 1, \ldots, m\}$ denote for all $E \subseteq \{1, \ldots, i\}$ containing i the set

$$S^1_{i,E} \equiv \{\tilde{E} : \ \tilde{E} = E \cup \{k\} \text{ for some } k \in \{1, \ldots, i\}\backslash E\}.$$

An important note here is that if $\tilde{E} \in S^1_{i,E}$, then $\tilde{E}$ must contain i. In addition, for all $r \in \{1, \ldots, i - |E|\}$ we define

$$S^r_{i,E} \equiv \{\tilde{E} : \tilde{E} = E \cup J \text{ for some } J \subseteq \{1,\dots,i\}\backslash E \text{ with } |J| = r\}.$$

A simple combinatorial argument shows that

$$\bigcup_{\tilde{E}\in S^{r-1}_{i,E}} S^1_{i,\tilde{E}} = S^r_{i,E} \quad \forall r \in \{2,\dots,i-|E|\}. \tag{3.4}$$

We utilize (3.4) to prove the promised lemma.

Lemma 3.3 *For a fixed $i \in \{0,1,\dots,m\}$ and for all $E \subseteq \{1,\dots,i\}$ such that $i \in E$, it holds that*

$$C_{i,E} = \left(\bigcup_{\tilde{E}\in S^1_{i,E}} C_{i,\tilde{E}}\right) \cup (R_{i,E} \cap F_{i,E}).$$

In particular,

$$C_{i,\{1,\dots,i\}} = R_{i,\{1,\dots,i\}}.$$

Proof First we show that for all $\tilde{E} \subseteq \{1,\dots,i\}$ containing E it holds that

$$R_{i,\tilde{E}} \cap F_{i,\tilde{E}} = R_{i,\tilde{E}} \cap F_{i,E}. \tag{3.5}$$

Let us show (3.5) by showing two inclusions. The inclusion $\subseteq$ is valid since if $\tilde{E} \supseteq E$, then $F_{i,\tilde{E}} \subseteq F_{i,E}$. For the inclusion $\supseteq$, assume that $\mathbf{x} \in R_{i,\tilde{E}} \cap F_{i,E}$. Then $\mathbf{x}$ satisfies all the equalities in $\tilde{E}$, as an ($[i,\tilde{E}]$)-relaxed solution. In addition, it satisfies all the inequalities in $\tilde{I}{:=}\{i,\dots,i\}\backslash\tilde{E}$, since $\tilde{I} \subseteq I{:=}\{1,\dots,i\}\backslash E$. Thus $\mathbf{x} \in R_{i,\tilde{E}} \cap F_{i,\tilde{E}}$.

The equality (3.5) is in particular valid for any $\tilde{E} \in S^r_{i,E}$, $r \in \{1,\dots,i-|E|\}$ as such $\tilde{E}$ satisfy $E \subseteq \tilde{E} \subseteq \{1,\dots,i\}$. We now show the main result of the lemma by induction on $|I|$. If $|I| = 0$, then $E = \{1,\dots,i\}$. Therefore, as the only option for $\tilde{E}$ is E itself, by definition we get $C_{i,\{1,\dots,i\}} = R_{i,\{1,\dots,i\}} \cap F_{i,\{1,\dots,i\}} = R_{i,\{1,\dots,i\}}$, where the last equality follows from the fact that $R_{i,\{1,\dots,i\}} \subseteq F_{i,\{1,\dots,i\}}$. Assume that the lemma has been proven for all nodes $[i,E]$ satisfying $|I| \le k-1$ for some positive integer k. Then for a node $[i,E]$ having $|I| = k$ we get that $[i,\tilde{E}]$ has $|\tilde{I}| = k-1$ for all $\tilde{E} \in S^1_{i,E}$, where $\tilde{I} = \{1,\dots,i\}\backslash\tilde{E}$. Applying the induction's assumption on each set $C_{i,\tilde{E}}$ with $\tilde{E} \in S^1_{i,E}$ yields

$$\begin{aligned}\left(\bigcup_{\tilde{E}\in S^1_{i,E}} C_{i,\tilde{E}}\right) \cup (R_{i,E}\cap F_{i,E}) &= \bigcup_{\tilde{E}\in S^1_{i,E}} \left\{\left(\bigcup_{\tilde{E}'\in S^1_{i,\tilde{E}}} C_{i,\tilde{E}'}\right) \cup (R_{i,\tilde{E}} \cap F_{i,\tilde{E}})\right\} \cup (R_{i,E}\cap F_{i,E})\\ &= \left(\bigcup_{\tilde{E}\in S^2_{i,E}} C_{i,\tilde{E}}\right) \cup \left(\bigcup_{\tilde{E}\in S^1_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup (R_{i,E}\cap F_{i,E}),\end{aligned}$$

where the second equality holds due to (3.4) and (3.5).

Applying the induction's assumption on the elements $C_{i,\tilde{E}}$ with $\tilde{E} \in S^2_{i,E}$ implies that

$$\left(\bigcup_{\tilde{E}\in S^2_{i,E}} C_{i,\tilde{E}}\right) \cup \left(\bigcup_{\tilde{E}\in S^1_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup (R_{i,E} \cap F_{i,E})$$
$$= \left(\bigcup_{\tilde{E}\in S^3_{i,E}} C_{i,\tilde{E}}\right) \cup \left(\bigcup_{\tilde{E}\in S^2_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup \left(\bigcup_{\tilde{E}\in S^1_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup (R_{i,E} \cap F_{i,E}).$$

Repeating the above procedure $|I| - 1$ times (where $|I| = i - |E|$) finally yields that

$$\left(\bigcup_{\tilde{E}\in S^1_{i,E}} C_{i,\tilde{E}}\right) \cup (R_{i,E} \cap F_{i,E})$$
$$= \left(\bigcup_{\tilde{E}\in S^{|I|}_{i,E}} C_{i,\tilde{E}}\right) \cup \left(\bigcup_{\tilde{E}\in S^{|I|-1}_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup \cdots \cup \left(\bigcup_{\tilde{E}\in S^1_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right)$$
$$\cup (R_{i,E} \cap F_{i,E}).$$

Since $S^{|I|}_{i,E} = \{\{1,\ldots,i\}\}$, the union over all $\tilde{E} \in S^{|I|}_{i,E}$ is just $C_{i,\{1,\ldots,i\}}$. In addition,

$$\left\{\tilde{E} : E \subseteq \tilde{E} \subseteq \{1,\ldots,i\}\right\} = \left(\bigcup_{r=1}^{|I|} S^r_{i,E}\right) \cup \{E\}.$$

Thus, we obtain

$$\left(\bigcup_{\tilde{E}\in S^1_{i,E}} C_{i,\tilde{E}}\right) \cup (R_{i,E} \cap F_{i,E})$$
$$= C_{i,\{1,\ldots,i\}} \cup \left(\bigcup_{\tilde{E}\in S^{|I|-1}_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup \cdots \cup \left(\bigcup_{\tilde{E}\in S^1_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup (R_{i,E} \cap F_{i,E})$$
$$= (R_{i,\{1,\ldots,i\}} \cap F_{i,E}) \cup \left(\bigcup_{\tilde{E}\in S^{|I|-1}_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right) \cup \cdots \cup \left(\bigcup_{\tilde{E}\in S^1_{i,E}} (R_{i,\tilde{E}} \cap F_{i,E})\right)$$
$$\cup (R_{i,E} \cap F_{i,E})$$
$$= \bigcup_{\tilde{E}\in\left(\bigcup_{r=1}^{|I|} S^r_{i,E}\right)\cup\{E\}} (R_{i,\tilde{E}} \cap F_{i,E}) = \bigcup_{\tilde{E}:\, E\subseteq\tilde{E}\subseteq\{1,\ldots,i\}} (R_{i,\tilde{E}} \cap F_{i,E}) = C_{i,E}.$$

□

Lemmata 3.2 and 3.3 together enable us to solve any node $[i, E]$ with $i \in E$, given that all the candidates sets of the nodes $[i, \tilde{E}]$ for $\tilde{E} \in S^1_{i,E}$ are already computed. It basically states that to solve a node we have to compute the $([i, E])$-relaxed solutions, that is, to solve a problem of the form (P_E), and to compare the values of all the relevant candidates, including the ([i, E])-relaxed solutions. The importance of Lemma 3.3 is the ability to search for the best candidate in a linear number (in m) of $|S^1_{i,E}|$ previously solved nodes, rather than on an exponential number (of all the sets $\tilde{E}$ satisfying $E \subseteq \tilde{E} \subseteq \{1, \ldots, i\}$), and the sufficiency to store the candidates sets only, without storing all the relaxed solutions of all nodes (only the current's node relaxed solutions are considered when it is solved).

However, during the running of the algorithm, some nodes in the current layer might be unavailable due to fathoming of early layers nodes. Therefore, whenever the set $S^1_{i,E}$ is not fully available, our algorithm might not necessarily solve node [i, E]. It might neither compute an optimal solution of problem ([i, E]) nor its optimal value. However, the fathoming rules of the algorithm guarantee that in such cases the best value computed among the available candidates can still cast as a lower bound on the node in question, in the sense that no candidate better than those already been reached can be reached at that node or at any child-node of it. Whenever the algorithm seeks to compute a lower bound on a node [i, E] with $i \in E$, we define

$$S^{1,open}_{i,E} := \left\{ \tilde{E} : \tilde{E} \in S^1_{i,E} \text{ and } [i, \tilde{E}] \text{ is available} \right\}$$

(available-none of its father-nodes has been fathomed). Evidently, $S^{1,open}_{i,E} \subseteq S^1_{i,E}$. In addition, for all nodes [i, E] where $i \in E$ we define the **available candidates set** as

$$\tilde{C}_{i,E} = \left(\bigcup_{\tilde{E} \in S^{1,open}_{i,E}} C_{i,\tilde{E}} \right) \cup (R_{i,E} \cap F_{i,E}). \tag{3.6}$$

For nodes of the form [i, E] with $i \notin E$ we define

$$\tilde{C}_{i,E} = \tilde{C}_{i,E\cup\{i\}} \cup (\tilde{C}_{i-1,E} \cap F_{i,E}). \tag{3.7}$$

We describe now our main branch and bound algorithm for solving (P). In this algorithm, U denotes the current upper bound on the optimal value, and $L_{i,E}$ denotes the computed lower bound on all the subproblems obtained by adding more constraints to the subproblem of node [i, E] (i.e., on all its “child-nodes”). Recall that $i = 1$ corresponds to a regularizing constraint.

Let us represent a set $E \subseteq \{1, \ldots, i\}$ by an i-bit number $B_{i,E}$, where the leftmost bit in $B_{i,E}$ corresponds to the index 1 and the rightmost bit to the index i. For each index $j = 1, \ldots, i$ we set the corresponding bit to 0 if $j \in E$, and to 1 if $j \notin E$. For example, for a node [3, E] with $E = \{1, 3\}$ the corresponding bit-representation is $B_{3,E} = [0\,1\,0]$. For a given i the above representation induces a full **ascending order** over the sets E - the standard order of binary numbers. For example, for $i = 3$, the ascending order of sets E is $\{1, 2, 3\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1\}$, $\{2, 3\}$, $\{2\}$, $\{3\}$, $\emptyset$, because the corresponding sequence of binary numbers is $[0\,0\,0]$, $[0\,0\,1]$, $[0\,1\,0]$, $[0\,1\,1]$, $[1\,0\,0]$, $[1\,0\,1]$, $[1\,1\,0]$, $[1\,1\,1]$.

Algorithm BB

```
(1) Initialize $U \leftarrow \infty$, $L_{0,\emptyset} \leftarrow \infty$, and $i \leftarrow 0$. Open node $[0, \emptyset]$.
(2) In an ascending order of the sets $E \subseteq \{1, \dots, i\}$ (induced by the order of the binary
    representation of the sets), perform for each open node $[i, E]$ the following:
      • Compute the set $\tilde{C}_{i,E}$ containing all the available candidates for minimizers of
        problem ($[i, E]$) by the formulae (3.6) and (3.7). Store in memory this set. Take
        the minimal objective value over $\tilde{C}_{i,E}$ as a lower bound $L_{i,E}$.
      • Set $U \leftarrow \min\{U, f_0(\mathbf{x}^1), \dots, f_0(\mathbf{x}^l)\}$, where $\{\mathbf{x}^1, \dots, \mathbf{x}^l\}$ contains all the available
        candidates for minimizers of ($[i, E]$) which are also (P)-feasible ($l = 0$ is possible).
        If $U$ achieves a new value, say $f_0(\mathbf{x}^k)$, then set $Sol \leftarrow \mathbf{x}^k$.
(3) For each $E \subseteq \{1, \dots, i\}$, in an ascending order (induced as before):
    If an open node $[i, E]$ satisfies $L_{i,E} < U$, then open the child-nodes $[i + 1, E \cup \{i\}]$
    with $L_{i+1,E\cup\{i\}} \leftarrow \infty$ and $[i + 1, E]$ with $L_{i+1,E} \leftarrow \infty$.
    Otherwise, fathom node $[i, E]$.
(4) If some nodes in layer $i + 1$ were opened, clear from memory any candidates set stored
    in layer $i - 1$ if exists, set $i \leftarrow i + 1$, and return to step 3.
    Otherwise (i.e., if all existing open nodes in layer $i$ are fathomed), stop, and return
    "$Sol$" as a solution, and $U$ as an optimal value.
```

Remark We should explain why the sets $\tilde{C}_{i,E}$ defined by (3.6) or (3.7) are computable when step 2 is employed. Indeed, assume the algorithm is currently solving a node $[i, E]$ with $i \in E$. Then for all $k \in \{1, \dots, i\} \backslash E$, the node $[i, \tilde{E}]$ with $\tilde{E} = E \cup \{k\}$ has already been solved, as the bit representations satisfy $B_{i,\tilde{E}} < B_{i,E}$ (exactly one bit in $B_{i,E}$ has changed from 1 to 0 yielding $B_{i,\tilde{E}}$). Thus, for any node with a set belonging to $S^{1,open}_{i,E}$ the available candidate set is already known due to the ascending order in which the nodes in layer i are treated and thus $\tilde{C}_{i,E}$ can be computed via the formula (3.6). Similarly, in nodes where $i \notin E$, the set $\tilde{C}_{i,E\cup\{i\}}$ is known by the ascending order in layer i, and the set $\tilde{C}_{i-1,E}$ is known from the previous layer $i - 1$. The following lemma shows that the use of $\tilde{C}_{i,E}$ (rather than $C_{i,E}$) does not have an effect on the correctness of the fathoming rules in Algorithm BB.

Lemma 3.4 *Let $\mathbf{x}^*$ be a vector with the minimal objective value over the set $\tilde{C}_{i,E}$ (rather than over $C_{i,E}$) for $i \in E$, and plug its value into $L_{i,E}$. Then either $\mathbf{x}^*$ is an optimal solution of problem ($[i, E]$), or it is guaranteed that node $[i, E]$ is going to be correctly fathomed (despite the wrong value plugged into $L_{i,E}$).*

Proof Denote the optimal value of problem ($[i, E]$) by V. Assume that $L_{i,E}$ is computed as the minimal value among the candidates in $\tilde{C}_{i,E}$, and that $V < L_{i,E}$, assuming that the chosen candidate, $\mathbf{x}^*$, is not an optimal solution of ($[i, E]$), and thus, $f_0(\mathbf{x}^*) = L_{i,E} > V$. By part (2) of Lemma 3.2, the value V must be obtained at a vector in $C_{i,E}$. Since it is not obtained in $\tilde{C}_{i,E}$, it must be obtained at a node $[i, \tilde{E}]$ with $\tilde{E} \in S^1_{i,E}$, which has not been opened. The only reason for not opening this node is that one of its father-nodes gained a lower bound, L, which satisfied $L \geq U$, where U is an upper bound on the optimal value of (P). That is, $L_{i,E} > V \geq L \geq U$, where the middle inequality is because L is a lower bound on all child-nodes of the node it is gained at. Therefore, although $L_{i,E}$ might not be a lower bound on all the child-nodes of node $[i, E]$, it is guaranteed that the node will be fathomed by the algorithm, as $L_{i,E} > U$. □

Lemma 3.4 guarantees that even if $L_{i,E}$ for $i \in E$ is not equal to the optimal value of problem ($[i, E]$), the decision whether to fathom node $[i, E]$ or not, would be exactly the same as if it was. As a consequence, we can assume that at any node we can compute a valid lower bound $L_{i,E}$ ensuring correct fathoming decisions. The above assumption is essential when we seek to establish the validity of Algorithm BB; see Theorem 3.1 below.

By now we assumed that all the sets $R_{i,E}$ are finite. Since $|R_{i,\tilde{E}}|$ is finite, it follows by Lemma 3.2 that the number of candidates at each node is also finite, and the optimal solution can be found by simple value comparisons. The upper bound on $|R_{i,\tilde{E}}|$ can effect the efficiency of the algorithm, as it is highly connected with the number of candidates $|C_{i,\tilde{E}}|$ being considered at each node, as follows by the same lemma. An upper bound on $|C_{i,\tilde{E}}|$ is calculated for the classes of problems presented in Sect. 4.2. When some sets of the form $R_{i,E}$ are infinite, a slight modification is required in Algorithm BB:

Algorithm BB - a Modification:

- Whenever there exist infinitely many ($[i, E]$)-relaxed solutions change the definition of $R_{i,E}$ such that it would be a singleton containing only one arbitrary ($[i, E]$)-relaxed solution (rather than an infinite set).
- Whenever the number of ($[i, E]$)-relaxed solutions is finite, $R_{i,E}$ is defined as before.
- Instead of computing *all* candidates of $\tilde{C}_{i,E}$ in step (3), use the same formulae (3.6) and (3.7) to evaluate those sets, but use the above modified sets $R_{i,E}$ (now always finite).

From now on, we consider only the modified version as "Algorithm BB". The next theorem is the main result of this section, and it establishes the validity of the algorithm. Though it might seem incorrect to ignore an infinite number of relaxed solutions (as the modified $R_{i,E}$ contains only one when there exist infinitely many) this issue is also resolved.

Theorem 3.1 *Algorithm BB is finite and correct; that is, if problem (P) is feasible and satisfies assumptions (1), (2) and (3), the algorithm computes a global optimal solution after solving a finite number of nodes.*

Proof The tree has at most $m + 1$ layers, where for each $i \in \{0, 1, \dots, m\}$, layer i contains at most 2^i open nodes. Therefore, the maximal number of nodes in a tree is

$$\sum_{i=0}^{m} 2^i = 2^{m+1} - 1,$$

and thus, it is finite.

We now prove that an optimal solution of (P) is always returned, assuming that problem (P) is feasible. For any feasible vector $\mathbf{x}$ denote

$$E(\mathbf{x}) := \{k : f_k(\mathbf{x}) = 0\},$$

and

$$I(\mathbf{x}) := \{1, \dots, m\} \backslash E(\mathbf{x}).$$

As its domain is compact, problem (P) attains its minimum by Weierstrass's Theorem. Similarly, any subproblem corresponding to a node is either infeasible, or has a compact nonempty domain, and thus, attains its minimum. Let $\mathbf{x}^*$ be an optimal solution of (P) with a value U. Then, by Lemma 3.1 $\mathbf{x}^*$ must be a local minimizer of (P_E) for $E = E(\mathbf{x}^*)$. Let i denote the maximal index in E. The choices of i and E define a specific node $[i, E]$ in the tree.

We shall prove that Algorithm BB either returns $\mathbf{x}^*$ as an optimal solution with the optimal value U, or some other optimal solution of (P). Assume that the algorithm does not return $\mathbf{x}^*$. Then the two options are

(A) the algorithm does not reach node $[i, E]$, or
(B) it does, but misses $\mathbf{x}^*$ due to the arbitrary choice of a candidate.

Option (A) can occur only when an early "father-node", $[\tilde{i},\ \tilde{E}]$, $\tilde{i} < i$, from which this node stems (not necessarily directly), was fathomed. In such a case, when layer $\tilde{i}$ was treated, we already had an upper bound $\tilde{U} \geq U$, attained at some feasible solution, $\tilde{\mathbf{x}}$, with $L_{\tilde{i},\,\tilde{E}} \geq \tilde{U}$. Since $L_{\tilde{i},\,\tilde{E}}$ is a lower bound on all child-nodes of $[\tilde{i},\ \tilde{E}]$, including $[i, E]$, we have

$$U \geq L_{i,E} \geq L_{\tilde{i},\,\tilde{E}} \geq \tilde{U},$$

where the left inequality is valid since U is an upper bound, and $L_{i,E}$ is a lower bound on problem ($[i, E]$). If a strict inequality holds at least in one of the above inequalities, it is a contradiction to the optimality of $\mathbf{x}^*$, and if only equalities hold, then $\tilde{\mathbf{x}}$ is also optimal, and is already obtained. Note that the algorithm returns only one optimal solution (in the last case it was $\tilde{\mathbf{x}}$).

According to assumption (3), option (B) can occur only if a problem (P_E) of the form (P_E) with $E = E(\mathbf{x}^*)$ has infinitely many global minimizers forming a path-connected set S^*. In this case, we have to show that an optimal solution of (P), possibly different than $\mathbf{x}^*$, is returned by the algorithm, even if the arbitrary candidate chosen by the algorithm at node $[i, E]$ is infeasible for (P). Indeed, the inequalities

$$f_k(\mathbf{x}^*) < 0$$

hold for all $k \in I(\mathbf{x}^*)$, since $\mathbf{x}^*$ is a feasible solution. Denote

$$\alpha(\mathbf{x}) := \max_{k \in I(\mathbf{x}^*)} \{f_k(\mathbf{x})\}.$$

By the feasibility of $\mathbf{x}^*$ it holds that $\alpha(\mathbf{x}^*) < 0$. If for any $\mathbf{x} \in S^*$ it holds that $\alpha(\mathbf{x}) \leq 0$, then the whole set S^* is also optimal for (P), as any vector in it attains the optimal value (the same as at $\mathbf{x}^*$) and it is feasible. In this case, an arbitrary candidate on S^* is marked as a candidate, and finally returned by the algorithm (unless another optimal solution has been discovered earlier). Otherwise, there exists an index $k \in I(\mathbf{x}^*)$ such that the surface $f_k(\mathbf{x}) = 0$ intersects S^* at a feasible point $\tilde{\mathbf{x}}$. The last claim follows by the continuity of the function α, and by the path-connectedness of S^*: since we are assuming that there exists an $\mathbf{x} \in S^*$ such that $\alpha(\mathbf{x}) > 0$, and since $\alpha(\mathbf{x}^*) < 0$, it follows that there exists a vector $\tilde{\mathbf{x}} \in S^*$ such that $\alpha(\tilde{\mathbf{x}}) = 0$, so $\tilde{\mathbf{x}}$ is feasible, and satisfies $f_k(\tilde{\mathbf{x}}) = 0$ for some $k \in I(\mathbf{x}^*)$.

Therefore, $\tilde{\mathbf{x}}$ is also an optimal solution of (P), with $E(\tilde{\mathbf{x}})$ strictly containing $E(\mathbf{x}^*)$. We can repeat the same argument on $\tilde{\mathbf{x}}$, and conclude that it is either marked as a candidate of some node with a finite number of candidates, whose equality constraints set is $E(\tilde{\mathbf{x}})$, or, again, it resides on an optimal path-connected set $\tilde{S}$ of that node, and has the same objective value as some arbitrary chosen candidate of it. The same process of moving from $\tilde{\mathbf{x}}$ to another vector which is feasible and satisfies one more equality can be repeated until either we obtain a finite number of solutions for the problem (P_E) corresponding to the relevant set E, or we reach the situation where $E = \{1, \ldots, m\}$. In the latter, any candidate (local minimizer) obtained in problem (P_E) is necessarily feasible for (P), as all the constraints are considered. Thus, such an optimal solution is discovered by the algorithm when a corresponding node is solved (unless another optimal solution has already been discovered). □

We should note that the total number of nodes solved during the algorithm is always finite. However, in the worst case, it can be equal to 2^{m+1}, meaning that all the possible nodes are opened and evaluated. Such cases are likely to occur when the number of variables is higher than the number of constraints. In cases where $n < m$, we are more likely to achieve the optimal solution after much fewer nodes evaluations, as many possibilities for choosing a subset of equalities constraints (E) are infeasible. In addition, if the constraints that are satisfied as equalities at optimality appear at the first i layers of the tree, the nodes number is expected to be no more than 2^{i+1}.

Remark The algorithm of Bienstock and Michalka [7] has similar properties, as it also enumerates candidates attained as local solutions of subproblems defined similarly to ($\mathrm{P}_{E,I}$). However, it does not apply a branch-and-bound approach, but rather a breadth-first search, which finds all such feasible subproblems (only infeasible subproblems can be fathomed), and then computes the best candidate by solving each of the feasible subproblems. While [7] focuses on obtaining a polynomial time algorithm for a special case, we suggest Algorithm BB as a practical algorithm. While the number of feasible subproblems might be large, our algorithm might solve much fewer such subproblems, as it applies also effective lower bounds on currently solved nodes, obtained on former solved nodes. For the sake of comparison with our algorithm, we provide a brief description of theirs in Sect. 6, where the numerical performances are compared.

3.1 Heuristics

Algorithm BB can yield better results with respect to total running time and memory if the number of nodes being opened during the procedure is small. In particular, we would like the total number of layers to be treated to be small. Such a scenario is most probable when feasible solutions of (P) with nearly optimal objective values are attained at the earliest layers (the constraints with the lowest i indices). The constraints which are active at optimality should then be involved as early as possible. In fact, if those constraints appear at the earliest i indices, all the other layers will not be involved, and the algorithm will stop after a few layers.

An open question is how we can "predict" which and how many constraints are to be satisfied as equalities, besides the obvious limitation of n constraints. We propose the following heuristics regarding the order in which we add the constraints during Algorithm BB.

Heuristics: The ith constraint to be added is the constraint violated by the largest number of candidates computed at all nodes of layer $i-1$.

When applying this heuristics, we do not fix the order of the constraints in advance, rather the order is built layer by layer. As this choice of ordering is just a heuristics, it does not theoretically guarantee that the best order is taken, but as will be shown in Sect. 6, it does make significant improvement from a practical point of view.

4 The inner oracle in the quadratic problem (1.1)

In this section we show that the quadratic problem (1.1) satisfies the assumptions from Sect. 2, enabling to apply Algorithm BB to solve it. The main tool we utilize is the well known trust region subproblem.

4.1 The trust-region subproblem

In the case of problem (1.1), the subproblems (P_E) may contain sphere constraints and linear equalities. We first summarize the main result enabling to find all the local minimizers of the simplest case where only one sphere constraint is involved. In this case (P_E) contains only one sphere constraint, and without loss of generality we can assume that the sphere is centered at the origin (otherwise, a simple translation of the variables can transform the problem into that form). We are therefore interested in computing all the local minimizers of the problem

$$\begin{aligned} &\min \ q_0(\mathbf{x}) \equiv \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ &\text{s.t.} \ \ \|\mathbf{x}\|^2 = r^2. \end{aligned} \tag{4.1}$$

In addition, in the following review (and in one numerical experiment in Sect. 6) we may refer also to the inequality constraint (ball), namely the problem

$$\begin{aligned} &\min \ q_0(\mathbf{x}) \equiv \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ &\text{s.t.} \ \ \|\mathbf{x}\|^2 \leq r^2. \end{aligned} \tag{4.2}$$

The following two lemmata, proved in [22, Lemma 2.4, 2.8], give necessary and sufficient optimality conditions for a global minimizer of problems (4.1) and (4.2).

Lemma 4.1 $\mathbf{x}^*$ *is a global minimizer of* (4.1)*, if and only if* $\|\mathbf{x}^*\| = r$*, and there exists* $\mu^* \in \mathbb{R}$ *such that*

(1) $(\mathbf{Q}_0 + \mu^*\mathbf{I})\mathbf{x}^* = \mathbf{b}_0$,
(2) $\mathbf{Q}_0 + \mu^*\mathbf{I} \succeq \mathbf{0}$.

Lemma 4.2 $\mathbf{x}^*$ *is a global minimizer of* (4.2)*, if and only if* $\|\mathbf{x}^*\| \leq r$*, and there exists* $\mu^* \geq 0$ *such that*

(1) $(\mathbf{Q}_0 + \mu^*\mathbf{I})\mathbf{x}^* = \mathbf{b}_0$,
(2) $\mathbf{Q}_0 + \mu^*\mathbf{I} \succeq \mathbf{0}$,
(3) $\mu^*(\|\mathbf{x}^*\| - r) = 0$.

For the computation of such an optimal solution, we may apply a one-dimensional root-search procedure on a monotone *secular* function of μ in the domain where $\mathbf{Q}_0+\mu\mathbf{I}$ is positive definite. The procedure computes a multiplier μ^* satisfying (1) and $\|\mathbf{x}^*\| = r$. Details can be found in [20]. By Lemmata 4.1 and 4.2, once a solution μ^* for which $\mathbf{Q}_0+\mu^*\mathbf{I}$ is non-singular is found, an optimal solution of (4.1) or (4.2) respectively, is given by

$$\mathbf{x}^* = (\mathbf{Q}_0 + \mu^*\mathbf{I})^{-1}\mathbf{b}_0. \tag{4.3}$$

When condition (1) in the above lemmata is satisfied only when $\mathbf{Q}_0 + \mu^*\mathbf{I}$ is singular, we call it a *hard case*. In that case, it holds that $\mu^* = -\lambda_1$, where λ_1 is the smallest eigenvalue of $\mathbf{Q}_0$ and $\mathbf{b}_0 \in \mathrm{Range}(\mathbf{Q}_0 - \lambda_1\mathbf{I})$. Problems (4.1) and (4.2) posses in this case infinitely may global solutions of the form

$$\mathbf{x}^* = (\mathbf{Q}_0 - \lambda_1\mathbf{I})^{\dagger}\mathbf{b}_0 + \mathbf{v}, \tag{4.4}$$

with $\mathbf{v}$ being any eigenvector of $\mathbf{Q}_0$ associated with λ_1. Here, for a given matrix $\mathbf{M}$, the notation $\mathbf{M}^{\dagger}$ stands for the Moore-Penrose pseudo-inverse of $\mathbf{M}$. As (4.1) and (4.2) are not convex, they might also posses local-non-global[1] minimizers. The following results of [18] characterize these solutions.

Lemma 4.3 *Let* $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ *denote the eigenvalues of* $\mathbf{Q}_0$. *If either* $\lambda_1 = \lambda_2$ *or* $\mathbf{b}_0^T\mathbf{v} = 0$ *for any eigenvector* $\mathbf{v}$ *associated with* λ_1, *then problems* (4.1) *and* (4.2) *have no local-non-global minimizer.*

Proof It is a direct result of Lemmata 3.2 and 3.3 in [18]. □

Lemma 4.3 readily implies that in the hard case, no local-non-global points exist since $\mathbf{b}_0 \in \text{Range}(\mathbf{Q}_0 - \lambda_1\mathbf{I}) = \text{Null}(\mathbf{Q}_0 - \lambda_1\mathbf{I})^{\perp}$. Here for a given matrix $\mathbf{M}$ the notation Range$(\mathbf{M})$ stands for the linear space spanned by the columns of $\mathbf{M}$, and Null$(\mathbf{M})$ denotes the space of all vectors $\mathbf{v}$ satisfying $\mathbf{Mv} = \mathbf{0}$.

Theorem 4.1 [18, Theorem 3.1] *Let* $\varphi(\mu) \equiv \|(\mathbf{Q}_0 + \mu\mathbf{I})^{-1}\mathbf{b}_0\|^2 - r^2$ *defined for all* $\mu \in \mathbb{R}$ *such that* $\mathbf{Q}_0 + \mu\mathbf{I}$ *is non-singular.*

(1) If $\mathbf{x}^*$ *is a local-non-global minimizer of* (4.1) *or* (4.2), *then the equation*

$$(\mathbf{Q}_0 + \mu^*\mathbf{I})\mathbf{x}^* = \mathbf{b}_0 \tag{4.5}$$

holds for some $\mu^* \in (-\lambda_2, -\lambda_1)$, *with* $\varphi'(\mu^*) \geq 0$. *If* $\mathbf{x}^*$ *is a local-non-global minimizer of* (4.2), *then* $\mu^* \geq 0$.

(2) There exists at most one local-non-global minimizer of (4.1) *or* (4.2).

(3) If $\|\mathbf{x}^*\| = r$, *and equation* (4.5) *holds true for some* $\mu^* \in (-\lambda_2, -\lambda_1)$ *with* $\varphi'(\mu^*) > 0$, *then* $\mathbf{x}^*$ *is a strict local-non-global minimizer of* (4.1). *If, in addition,* $\mu^* > 0$, *then* $\mathbf{x}^*$ *is also a strict local minimizer of* (4.2).

To compute the (unique) local-non-global solution, a more complicated root search algorithm can be applied on a similar one-variable equation in the domain $(-\lambda_2, -\lambda_1)$ (see [18, Algorithm 4.1]). It either computes a number $\mu_L^* \in (-\lambda_1, -\lambda_2)$ such that the necessary conditions in part (1) of Theorem 4.1 are satisfied, or detects that such a number does not exist. Once such μ_L^* is computed, a local-non-global minimizer of (4.1) is given by $\mathbf{x}^* = (\mathbf{Q}_0 + \mu_L^*\mathbf{I})^{-1}\mathbf{b}_0$. If $\mu_L^* > 0$, then $\mathbf{x}^*$ is also a local-non-global minimizer of (4.2).

In particular, the above results imply that in the hard case no local-non-global minimizers exist, and the global minimizers comprise a complete sphere whose dimension is the multiplicity of λ_1 as an eigenvalue of $\mathbf{Q}_0$. If that multiplicity is 1, there exist exactly two global minimizers. In any non-hard case, there exists at most one local-non-global minimizer and exactly one global minimizer.

4.2 Solving relaxed subproblems

The subproblems $(\mathrm{P}_{E,I})$ where $E \subseteq \{1, \ldots, i\}$ and $I = \{1, \ldots, i\} \backslash E$ to be solved during Algorithm BB contain some of the inequality constraints of (1.1), along with some equality constraints (spheres and hyperplanes). Their relaxations of the form (P_E) contain only spheres and hyperplanes, and by repeatedly reducing the dimension, they can be reformulated as TRS problems of the form (4.1).

[1] That is, vectors which minimize the objective over a small neighborhood of the feasible domain, but are not global minimizers of (4.1) or (4.2).

The reduction procedure utilizes the fact that the intersection of two given spheres can be represented as an intersection of a sphere and a hyperplane (see [23, Section 6]). Thus, all the spheres except for one can be replaced by hyperplanes, and finally, a standard dimension reduction can be performed based on a null space representation of the solutions of the linear system, and a trust region problem in a lower dimension is obtained. As described in the previous subsection, all the local minimizers of (4.1) can be efficiently computed. Thus, assumption (2) is satisfied. In addition, the change of variables from (P_E) into a lower dimensional problem of the form (4.1) is affine and non-singular, and thus, the set of all local minimizers of the two problems are homeomorphic. In particular, they have the same cardinality and the same topological properties, such as path-connectedness. The set of all the local minimizers of a problem of the form (P_E) contains either one global minimizer, two local minimizers (at least one of which is global) or a complete sphere in dimension 2 or higher. Thus, the sets $R_{i,E}$ for $i \in E$ either contain 1 or 2 elements, or comprise a sphere which is path-connected, and property (3) holds too.

Note that assumption (1) from Sect. 2 also holds true in problems of the form ($\mathrm{P}_{E,I}$) assuming that $m \geq 1$, that is, there exists at least one ball constraint, say $\|\mathbf{x} - \mathbf{w}\|^2 \leq d^2$. This constraint is indeed a regularizing constraint. Any subproblem of the form ($\mathrm{P}_{E,I}$) for which $1 \in E \cup I$ is guaranteed to have a compact feasible set, given by an intersection of a ball (if $1 \in I$) or a sphere (if $1 \in E$) and other closed sets (all the constraints are given by equalities of weak inequalities of continuous functions). By Weierstrass Theorem any continuous function attains a global minimum over a nonempty compact set. Thus, unless infeasible, a subproblem of the form ($\mathrm{P}_{E,I}$) with $1 \in E \cup I$ attains its minimum, and the latter are exactly the problems ($\mathrm{P}_{E,I}$) in assumption (1). In our BB algorithm, the first branching step restricts the feasible domain to reside within that ball. Thus, all the feasible subproblems which stem from the root node are guaranteed to admit a finite minimum obtained at an optimal solution.

Since each candidate has to be checked for feasibility and for the comparison of objective values, the exact number of such candidates (which is $|\tilde{C}_{i,E}|$ by Lemma 3.2) has an effect on the efficiency of the algorithm. In addition, the algorithm can treat only finite specific candidates at each node, so the case where infinitely many candidates exist has to be treated according to the new definition of $R_{i,E}$ for $i \in E$, provided in the modification of Algorithm BB. That is, if $R_{i,E}$ is infinite, we take an arbitrary vector from it, and the new set is defined as the corresponding singleton. Thus, in the new definition $|R_{i,E}| \leq 2$ always holds.

5 Application to sparse source localization problem

In this section we will show how Algorithm BB can be used to solve a class of nonsmooth and nonconvex source localization problems. In a source localization problem (see [3]), the decision variable $\mathbf{x} \in \mathbb{R}^n$ (n usually being 2 or 3) denotes the unknown location of a radiating source. In addition, m sensors in locations $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m \in \mathbb{R}^n$ are given, and each of them reports a noisy distance $d_i > 0$ to the source:

$$\|\mathbf{x} - \mathbf{a}_i\| \approx d_i, \quad i = 1, \ldots, m.$$

Two known optimization models of this problem are the *least squares* (LS) and the *squared least squares* (SLS) formulations. The LS formulation is given by

$$\text{(LS)} \qquad \min_{\mathbf{x} \in \mathbb{R}^n} \sum_{i=1}^{m} (\|\mathbf{x} - \mathbf{a}_i\| - d_i)^2. \tag{5.1}$$

As a mean squared error approximation, the optimal solution of (LS) is in fact a maximum likelihood estimator for the true location $\mathbf{x}$ whenever the noise values of the measurements are independent and each has a Gaussian distribution with the same standard deviation; see [12], as well as [5] for solution methodologies. However, the model (LS) has a nonsmooth and nonconvex objective, and it is thus considered as a hard problem, where most known algorithms are only guaranteed to generate a subsequence converging to a stationary point (see [5]), or to find a semidefinite relaxation-based approximation [12].
Another approach is to consider the model (SLS) given by

$$\text{(SLS)} \qquad \min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} (\|\mathbf{x}-\mathbf{a}_i\|^2 - d_i^2)^2. \tag{5.2}$$

The SLS formulation, suggested in [3] has a smooth (but still nonconvex) objective, and can be globally solved efficiently by rewriting it as a GTRS.

A common situation is when outliers in the measurements are present, meaning that some measurements are extremely noisy, and inconsistent with the others. The SLS formulation is sensitive to exceptional measurements, and it therefore might lead to poor results. The formulation we suggest in this paper is designed to handle such situations. Inspired by the compressed sensing/sparsity literature (see for example the review [8] and references therein), we will consider an l_1-based model in which the sum of absolute values of the errors is minimized:

$$\text{(SSL)} \qquad \min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} \left| \|\mathbf{x}-\mathbf{a}_i\|^2 - d_i^2 \right|. \tag{5.3}$$

The acronym SSL stands for "sparse source localization", and the word "sparse" comes to emphasize that we assume that most of the errors measurements are close to zero, while only a few have a substantial magnitude. Problem (5.3) can be divided into 2^m cases in the following way. For each vector $\boldsymbol{\sigma} \in \{-1, 1\}^m$, we can solve the following subproblem:

$$\begin{aligned} \min \quad & -\sum_{i=1}^{m} \sigma_i \left(\|\mathbf{x}-\mathbf{a}_i\|^2 - d_i^2\right) \\ \text{s.t.} \quad & \sigma_i \left(\|\mathbf{x}-\mathbf{a}_i\|^2 - d_i^2\right) \le 0, \qquad i = 1, \ldots, m. \end{aligned} \tag{5.4}$$

To solve problem (5.3), one can solve each of the problems (5.4) – one problem for each option for the choice of the signs vector $\boldsymbol{\sigma}$, and finally, take the solution having the best objective value among all the solutions. Each case of problem (5.4) is in fact a special case of problem (1.1), whose objective function has the parameters

$$\mathbf{Q}_0 = -2\left(\sum_{i=1}^{m} \sigma_i\right)\mathbf{I}, \qquad \mathbf{b}_0 = -2\left(\sum_{i=1}^{m} \sigma_i \mathbf{a}_i\right),$$

with a constant $C_0 = -\sum_{i=1}^{m} \sigma_i \left(\|\mathbf{a}_i\|^2 - d_i^2\right)$ added. As shown in Sect. 4, problem (1.1), having the form of problem (P), can be treated by Algorithm BB, provided that assumptions (1)–(3) are satisfied, which is the case for any option of $\boldsymbol{\sigma}$. Indeed, if $\boldsymbol{\sigma} \neq -\mathbf{e}$, then problem (5.4) contains at least one ball constraint, and the feasible domain is compact. Thus an optimal solution exists for each subproblem of the form ($\mathrm{P}_{E,I}$) (unless infeasible). If $\boldsymbol{\sigma} = -\mathbf{e}$, then the objective function becomes a strictly convex quadratic (hence coercive) over the closed nonempty feasible domain, and thus each subproblem also has a global minimizer. That establishes the validity of assumption (1). In addition, each subproblem of the form (P_E) consists of minimizing a quadratic function over several sphere constraints. As described in

Sect. 4, if feasible, problem (P$_E$) can be converted into a lower dimension problem of the form (4.1).

In this particular case, the equivalent TRS problem is even simpler than in the general case. The objective can also be made linear by subtracting the equality constraint (sphere) from it. That is, problem (P$_E$) is equivalent to a problem of the form

$$\begin{aligned} &\min_{\mathbf{y}\in\mathbb{R}^k} \tilde{q}_0(\mathbf{y}) \equiv \mathbf{g}_0^T\mathbf{y} + E_0 \\ &\text{s.t.} \quad \|\mathbf{y}-\mathbf{y}_0\|^2 = r^2. \end{aligned} \tag{5.5}$$

The global optimal solution is given by

$$\mathbf{y}^* = \mathbf{y}_0 - \frac{r}{\|\mathbf{g}_0\|}\mathbf{g}_0$$

for $\mathbf{g}_0 \neq \mathbf{0}$. Otherwise, any point on the sphere is optimal. If $k = 1$, the sphere contains only two points $y_0 + r$, $y_0 - r$, both are local minimizers by definition. Assume now $k \geq 2$. Since $\tilde{\mathbf{Q}}_0 = \nabla^2\tilde{q}_0 = \mathbf{0}$, in particular $\lambda_1(\tilde{\mathbf{Q}}_0) = \lambda_2(\tilde{\mathbf{Q}}_0) = 0$. Thus, by Lemma 4.3 it follows that problem (5.5) has no local-non-global minimizer. In addition, when $\mathbf{g}_0 \neq \mathbf{0}$, (5.5) has exactly one local minimizer (the global). When $\mathbf{g}_0 = \mathbf{0}$, it has a path-connected global optimal set – the whole sphere – and no local-non-global minimizers.

The above description basically explains how problem (5.3) can be solved based on Algorithm BB. However, this approach results in an algorithm that requires to solve 2^m nonconvex optimization problems, which seems to be too computationally expensive. Therefore, we suggest a preprocess procedure to reduce the number of problems of the form (5.4) that need to be solved. The procedure consists of two phases, where in each one we seek to rule out as many options as possible for the signs vector $\boldsymbol{\sigma}$. For each $\boldsymbol{\sigma} \in \{-1, 1\}^m$ we define $J_{\boldsymbol{\sigma}} \subseteq \{1, \ldots, m\}$ to be the set of all indices j for which $\sigma_j = 1$. In phase 1 the procedure performs a feasibility test on the set of inequalities

$$\|\mathbf{x} - \mathbf{a}_i\|^2 - d_i^2 \leq 0, \quad i \in J \tag{5.6}$$

with $J = J_{\boldsymbol{\sigma}}$ for some $\boldsymbol{\sigma} \in \{-1, 1\}^m$, and rule out problems detected as infeasible. In phase 2 the procedure proceeds with a similar feasibility test on the nonconvex system of inequalities

$$\begin{aligned} &\|\mathbf{x} - \mathbf{a}_i\|^2 - d_i^2 \leq 0, \quad i \in J, \\ &\|\mathbf{x} - \mathbf{a}_i\|^2 - d_i^2 \geq 0, \quad i \in \{1, \ldots, m\}\backslash J, \end{aligned} \tag{5.7}$$

for all the subsets $J = J_{\boldsymbol{\sigma}}$ that were not ruled out in phase 1. In any case, the systems of inequalities that we consider are of the form

$$\delta_i(\|\mathbf{x} - \mathbf{a}_i\|^2 - d_i^2) \leq 0, \quad i \in \{1, 2, \ldots, m\}, \tag{5.8}$$

where $\boldsymbol{\delta} \in \{-1, 1\}^m$. The system (5.6) corresponds to the choice $\delta_i = \frac{\sigma_i+1}{2}$, while the system (5.7) corresponds to $\boldsymbol{\delta} = \boldsymbol{\sigma}$. The screening in both phases is based on the following lemma that provides a sufficient condition for infeasibility of the system (5.8). The argument relies on the weak duality theorem.

Lemma 5.1 (sufficient condition for infeasibility of (5.6)). *Consider the system* (5.8) *for a given* $\boldsymbol{\delta} \in \{-1, 1\}^m$. *Assume that there exists* $\tilde{\boldsymbol{\lambda}} \in \mathbb{R}^m_+$ *for which*

$$\sum_{i=1}^m \delta_i\tilde{\lambda}_i = 1, \tag{5.9}$$

$$-\|\sum_{i=1}^m \delta_i\tilde{\lambda}_i\mathbf{a}_i\|^2 + \sum_{i=1}^m \delta_i\tilde{\lambda}_i\left(\|\mathbf{a}_i\|^2 - d_i^2\right) > 0. \tag{5.10}$$

Then the system (5.8) *is infeasible.*

Proof Suppose that there exists $\tilde{\boldsymbol{\lambda}} \in \mathbb{R}^m_+$ satisfying (5.9) and (5.10). Assume in contradiction that there exists $\tilde{\mathbf{x}} \in \mathbb{R}^n$ satisfying the system (5.8). Consider the problem

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & \delta_i(\|\mathbf{x} - \mathbf{a}_i\|^2 - d_i^2) \leq 0, \quad i \in \{1, 2, \ldots, m\}. \end{array} \tag{5.11}$$

For any $\boldsymbol{\lambda} \in \mathbb{R}^m_+$, a Lagrangian of problem (5.11) is

$$L(\mathbf{x}; \boldsymbol{\lambda}) = \sum_{i=1}^m \delta_i \lambda_i \left(\|\mathbf{x} - \mathbf{a}_i\|^2 - d_i^2\right), \qquad \boldsymbol{\lambda} \in \mathbb{R}^m_+,$$

and a dual of problem (5.11) is given by

$$\max\{q(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \mathbb{R}^m_+\},$$

where $q(\boldsymbol{\lambda}) \equiv \min_{\mathbf{x} \in \mathbb{R}^n} L(\mathbf{x}; \boldsymbol{\lambda})$. Since the primal problem is assumed (by contradiction) to be feasible, it follows by the weak duality theorem that

$$q(\boldsymbol{\lambda}) \leq 0 \text{ for any } \boldsymbol{\lambda} \in \mathbb{R}^m_+. \tag{5.12}$$

We will show that the above does not hold for the choice $\boldsymbol{\lambda} = \tilde{\boldsymbol{\lambda}}$. Using the fact that $\sum_{i=1}^m \delta_i \tilde{\lambda}_i = 1$, it follows that the minimizer of $L(\mathbf{x}; \tilde{\boldsymbol{\lambda}})$ is

$$\mathbf{x}_{\tilde{\boldsymbol{\lambda}}} = \sum_{i=1}^m \delta_i \tilde{\lambda}_i \mathbf{a}_i. \tag{5.13}$$

Plugging this expression back into the Lagrangian, we obtain that

$$q(\tilde{\boldsymbol{\lambda}}) = L(\mathbf{x}_{\tilde{\boldsymbol{\lambda}}}; \tilde{\boldsymbol{\lambda}}) = -\| \textstyle\sum_{i=1}^m \delta_i \tilde{\lambda}_i \mathbf{a}_i \|^2 + \sum_{i=1}^m \delta_i \tilde{\lambda}_i \left(\|\mathbf{a}_i\|^2 - d_i^2\right) > 0,$$

which is a contradiction to (5.12). □

Checking the sufficient condition of Lemma 5.1 is a tractable task since it can be validated by solving the convex problem

$$\begin{array}{ll} \max_{\boldsymbol{\lambda}} & -\| \sum_{i=1}^m \delta_i \lambda_i \mathbf{a}_i \|^2 + \sum_{i=1}^m \delta_i \lambda_i \left(\|\mathbf{a}_i\|^2 - d_i^2\right) \\ \text{s.t.} & \sum_{i=1}^m \delta_i \lambda_i = 1, \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{array} \tag{5.14}$$

The sufficient condition is satisfied if and only if the optimal value of the convex problem (5.14) is positive. Having established a simple criterion for ruling out infeasible problems, we are now ready to describe the overall two-phase procedure of ruling out infeasible problems.

Phase 1 We first apply a breadth-first-search (BFS) algorithm to check which of the problems of the form (5.6) for $J \subseteq \{1, \ldots, m\}$ are infeasible. The problems detected as infeasible are ruled out, and the remaining are considered as "admissible". For each J we utilize Lemma 5.1 to rule out infeasible cases of (5.6). As problem (5.14) is convex, we can apply a *fast gradient projection* (FGP) method which can be seen as a special (smooth) case of the *fast proximal gradient* method, also referred as *fast iterative shrinkage-thresholding algorithm* (FISTA); see [4] and references therein.

FGP-test: Run the FGP method till either: **I.** a vector with a positive objective value is reached, or, **II.** an optimality criterion is reached at a solution with a non-positive objective value. A subproblem is called *admissible* if applying the FGP-test on it yields option II.

Lemma 5.1 implies that if the FGP-test on a subproblem yields option I, it is infeasible and should be ruled out, and otherwise, we call that subproblem "admissible", as it may be feasible.

The BFS runs over a graph whose nodes are all the subsets $J \subseteq \{1, \ldots, m\}$, with edges between any two nodes J, J' with $J' = J \cup \{j\}$ for some $j \in \{1, \ldots, m\} \backslash J$. The output is a list of all the admissible subsets – the sets J for which the corresponding problem (5.6) could not be ruled out by Lemma 5.1. The algorithm reads as follows.

A BFS algorithm for eliminating infeasible cases of problem (5.6)

```
(1) Initialization: Set $L = \{\emptyset\}$,  $Blacklist = \emptyset$; Mark node $J = \emptyset$.
(2) Pick a node $J \in L$, with minimal $|J|$; delete it from $L$;
    Apply the FGP-test on $J \cup \{j\}$ for all $j \notin J$ such that $J \cup \{j\} \notin L \cup Blacklist$;
    If $J \cup \{j\}$ is admissible, mark it, and add it to $L$;
    If not, add $J \cup \{j\}$ and all its immediate child-nodes to $Blacklist$.
(3) Return to step (5) unless $L = \emptyset$;
    If the latter occurs, return the set of all marked nodes as "admissible".
```

No node is being checked more than once, as L cannot contain nodes of more than two successive numbers of constraints $|J|$ and $|J| + 1$. In addition, the black list ensures that no immediate child-node of a node detected as infeasible is being checked. It avoids rechecking any face already deleted from L that was infeasible.

Phase 2 For each subset $J \subseteq \{1, \ldots, m\}$ with the corresponding problem (5.6) detected as admissible after phase 1, we now consider the feasibility issue of the corresponding problem (5.7). That is, we check whether the addition of the constraints of being outside an Euclidean ball yields an infeasible problem, or still an admissible one. At this phase, we go over each admissible subset J without any special search algorithm such as BFS. Again, each feasibility test is performed by the FGP-test based on Lemma 5.1. There might be cases in which admissible cases of problem (5.7) are in fact infeasible. This is even more likely than in phase 1, as problem (5.7) is not convex. Note that problem (5.14) can be solved by the FGP method, except for the case $\boldsymbol{\sigma} = \boldsymbol{\delta} = -\mathbf{e}$, in which (5.14) is infeasible. However, in the latter case, (5.7) is certainly feasible, so it can be automatically marked as "admissible" without any test.

Applying Algorithm BB. The cases which remained in the status "admissible" after the above two phases should be solved by Algorithm BB. Their number was usually significantly smaller than 2^m in our numerical experiments. In general, all the remaining cases are considered as admissible, and in order to obtain an optimal solution of problem (5.3), each of those cases should be considered. However, as described in phase 2, our feasibility test might not rule out all the infeasible cases. Algorithm BB solves each of the admissible cases and returns the solution with the minimal objective value among them. Infeasible cases do not affect the results, as the algorithm can detect them as well, and they achieve the value ∞.

6 Numerical experiments

In this section we present the performances of Algorithm BB on a variety of instances of problem (1.1). All implementations were coded in MATLAB R2014a, and run on a PC with processor ~3.4 GHz and 16.0 GB RAM.

6.1 Minimizing a general quadratic over the intersection of a ball and affine half-spaces

We started by solving a special setting of problem (1.1) with $p = 0$ and $m = 1$, including one Euclidean ball constraint and l linear inequalities.

$$\begin{aligned} \min\ & q_0(\mathbf{x}) \equiv \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ \text{s.t.}\ & \|\mathbf{x}-\mathbf{w}\|^2 \le d^2, \\ & \boldsymbol{\alpha}_k^T\mathbf{x} \le \beta_k, \qquad k = 1,\dots,l. \end{aligned} \tag{6.1}$$

On this setting we applied the algorithm with a slight improvement: the first constraint (the Euclidean ball) was included in our root-node problem. That is, in node $[0, 0]$ we already solved and found all the local minimizers of the inequality constrained version of the TRS problem (4.2). When starting the algorithm directly from problem (4.2) rather than from the unconstrained minimization, we avoid one layer in the branch and bound tree and expect to reduce the number of nodes by a factor of about 2. The subproblems to be solved (for local and global minimizers) at each node $[i, E]$ with $i \in E$ is equivalent to a problem having the form of (4.2) as well (in a lower dimension).

The analysis of the candidates sets is the same as in problem (4.1), except for the requirement $\mu_L^* \ge 0$ for the local-non-global solution if exists, and the treatment of the case where the objective function is also convex ($\mathbf{Q}_0 \succeq \mathbf{0}$). As its feasible set is a ball, problem (4.2) can posses a full ball as an optimal set in some degenerate cases (e.g., if the objective is constant). However, such a set is also path-connected as required by assumption (3). In this setting, for any node $[i, E]$ the set $F_{i,E}$ is a closed convex set as an intersection of a closed ball, $|E|$ hyperplanes, and $|I| = i - |E|$ half-spaces. The problems defining the candidates sets and the relaxed solutions are similar to ($\mathrm{P}_{E,I}$) and (P_E) with the change that each contains the inequality constraint $\|\mathbf{x} - \mathbf{w}\|^2 \le d^2$ and the subsets E and I are subsets of the index set of the linear constraints only. In this subsection the name Algorithm BB refers to the above improvement.

We compare our computational effort results of Algorithm BB with those of the method of Bienstock and Michalka [7]. While the method of [7] is applied to the more general model (1.1), we only briefly describe our modified implementation of it on the special case (6.1). The method of [7] first computes all the *intersecting faces*; that is, all the elements in the set

$$\mathcal{F}^* \equiv \left\{F^E : F^E \cap B[\mathbf{w}, d] \neq \emptyset,\ E \subseteq \{1,\dots,l\}\right\},$$

where for any $E \subseteq \{1,\dots,l\}$ the corresponding *face* is defined by

$$F^E \equiv \left\{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\alpha}_i^T\mathbf{x} \le \beta_i\ \forall i \in \{1,\dots,l\},\ \boldsymbol{\alpha}_i^T\mathbf{x} = \beta_i\ \forall i \in E\right\}.$$

Once $\mathcal{F}^*$ is computed, the method processes its elements one after the other. For each $F^E \in \mathcal{F}^*$ the algorithm considers the problem of minimizing the original objective of (6.1) over the nonempty intersection $F^E \cap B[\mathbf{w}, d]$, computes all the local minimizers of the relaxed problem

$$\begin{aligned} \min\ & q_0(\mathbf{x}) \equiv \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ \text{s.t.}\ & \|\mathbf{x}-\mathbf{w}\|^2 \le d^2, \\ & \boldsymbol{\alpha}_k^T\mathbf{x} = \beta_k, \qquad k \in E, \end{aligned} \tag{6.2}$$

and stores as a *candidate* any local minimizer $\mathbf{x}^*$ satisfying $\boldsymbol{\alpha}_i^T\mathbf{x}^* \le \beta_i\ \forall i \in \{1,\dots,l\}$, that is, $\mathbf{x}^* \in F^E$. The computation of such minimizers is performed by the techniques described in Sect. 4. As proved in [7], it is guaranteed that the minimal objective candidates among all the intersecting faces $F^E \in \mathcal{F}^*$ is an optimal solution of (6.1). A very similar approach for solving problem (6.1) appears in [16], where a sequence of problems reduced to TRS problems are solved. The number of such TRS problems required to be solved is in fact identical to the number of intersecting faces $|\mathcal{F}^*|$.

Unlike Algorithm BB, the method of [7] must process all the intersecting faces, and it does not apply fathoming rules implied by lower or upper bounds on the optimal value. In addition, as opposed to Algorithm BB, the method of [7] requires feasibility checks of the intersection of a system of linear equalities and inequalities with a ball. On the other hand, by utilizing proper data structures, the general method described in [7] was proved to have polynomial running time in $|\mathcal{F}^*|$ and in the inputs of (1.1). Thus, it may be superior over Algorithm BB if $|\mathcal{F}^*|$ is not too large. In practice, however, as we see next, the set $\mathcal{F}^*$ could be much larger than the number of nodes required in Algorithm BB. In addition, the computation of that set should be done in advance, and rarely can one preestimate its size. The set $\mathcal{F}^*$ is computed by a breadth-first search (BFS) algorithm.

A BFS algorithm for computing $\mathcal{F}^*$ for problem (6.1)

```
(1) Initialization: Set $L = \{F^{\emptyset}\}$, $Blacklist = \emptyset$. Mark the face $F^{\emptyset}$.
(2) Pick $F^E \in L$, with minimal $|E|$; delete it from $L$;
    For all $E \cup \{j\}$ such that $j \notin E$ and $F^{E\cup\{j\}} \notin L \cup Blacklist$ :
    check if $F^{E\cup\{j\}} \cap B[\mathbf{w}, d] \neq \emptyset$.   (*)
    If (*) holds - mark $F^{E\cup\{j\}}$, and add it to $L$;
    If not - put $F^{E\cup\{j\}}$ and all its immediate "child-nodes" in $Blacklist$.
(3) Return to step (6.1) unless $L = \emptyset$;
    If the latter occurs, terminate, and return the set of all marked faces as the output $\mathcal{F}^*$.
```

We first solved some simple cases (Table 1) by Algorithm BB, where the number of variables n was small. We created these cases manually, and the input parameters are provided as MATLAB files in the library in https://drive.google.com/file/d/0B9PeyTrETyApSDY2bllLNUY2cUE/view?usp=sharing. For each, we provide the number of nodes evaluated and the run times, where the heuristics of choosing the order of constraints with respect to number of violations is applied (under H) as described in Sect. 3.1, or not applied (bracketed under (nH)). Recall that in Algorithm BB, only about half of the evaluated nodes ($H/2$) require solving an optimization problem (equivalent to a TRS problem), while in the others only values comparisons are involved. The last two columns in Table 1 refer to the number of intersecting faces $|\mathcal{F}^*|$ involved in each case, and the total number of faces ("Tot") for which the BFS procedure went over and checked if they were in $\mathcal{F}^*$. Recall that for each element in "Tot", a convex feasibility problem was solved, and in addition, for each element in $\mathcal{F}^*$ a problem equivalent to a TRS was solved. In addition, for the sake of comparison, we provided run times of solving the same instances by the mature solver SCIP (see [1] and references therein) using the input format '.pip' and the default parameters.

As one could see, the value $H/2$ is in most cases was much smaller than Tot, and in some cases smaller than $|\mathcal{F}^*|$ as well. We do not provide CPU times for Bienstock and Michalka's algorithm, as its implementation was affected by the way we implemented the intersecting test of each face (we just solve a convex feasibility problem utilizing CVX—a package for specifying and solving convex programs; see [14,15]), which is probably different from their implementation. It should be mentioned that while SCIP could find the optimal solution in

Table 1 Examples of (6.1) with n variables and l linear constraints-BB (H—with heuristics, nH—without) compared with the sizes involved in [7] and with run times of SCIP, where ‘InSol’ denotes cases where SCIP could not compute a (reasonable) approximate upper bound

Case name	n	l	Nodes evaluated H (nH)	Time (s) H (nH)	Time (s) SCIP	$\|\mathcal{F}^*\|$	Tot
Data_lin	6	4	15 (31)	0.026 (0.05)	63.38	16	16
Data_lin_5_10	5	10	3 (47)	0.01 (0.05)	0.33	18	69
Data_lin_10_11	10	11	155 (831)	0.238 (1.31)	>3600	2016	2032
Data_lin_5	5	15	613 (2429)	0.50 (1.62)	0.22	201	901
Data_lin_5_extended	5	25	641 (2831)	0.50 (1.93)	0.17	93	938
Data_lin_5_20	5	20	533 (3351)	0.34 (1.97)	0.08	537	3245
Data_lin_5_20_c01	5	20	31 (81)	0.023 (0.055)	0.70	16	144
Data_lin_5_20_c001	5	20	31 (81)	0.024 (0.057)	0.68	16	144
Data_lin_5_20_c100	5	20	1513 (3759)	0.91 (2.17)	0.08	537	3245
Data_lin_5_30	5	30	1931 (6459)	1.50 (6.26)	0.05	483	5367
Data_lin_8_20	8	20	3471 (17,507)	3.56 (18.78)	0.17	13,750	46,740
Data_lin_20_8	20	8	7 (43)	0.03 (0.063)	>3600	256	256
Data3_100_10	100	10	93 (877)	0.29 (2.73)	InSol	1024	1024
Data3_100_15	100	15	6439 (14,357)	21.11 (48.87)	>3600	32,768	32,768
Data_200_10	200	10	255 (1023)	0.96 (3.99)	>3600	1024	1024
Data_200_15	200	15	2047 (18,063)	7.93 (83.6)	>3600	32,768	32, 768
Data3_300_10	300	10	287 (1067)	3.75 (13.69)	>3600	1024	1024
Data_300_15	300	15	8959 (9887)	128.03 (136.82)	>3600	32,768	32,768

all the small n value instances above, in some cases it needed a considerably long times to close the gap between upper and lower bound, and in the larger cases it did not stop in reasonable times (less than 1h.). The results in Table 1 may suggest that Algorithm BB should be preferred over the method introduced in [7]. The above conclusion is strengthened in the next set of results.

In the next experiments, we considered some larger size instances with a specified structure (Table 2). The instances with names starting with “spar” were also treated by Chen and Burer [11, Section 5]; all were taken from the libraries boxqp and trbox to which their paper referred. Each instance contains $n = 20$ to $n = 40$ variables, the ball center is $\mathbf{w} = \mathbf{0}$, and all its linear constraints are given by

$$l_i \leq x_i \leq u_i, \;\; i = 1, \ldots, n, \tag{6.3}$$

where we assume $l_i < 0 < u_i$ for all $i = 1, \ldots, n$. Indexing the upper bound inequalities in $i = 1, \ldots, n$ and the lower bound inequalities in $i = n+1, \ldots, 2n$, we overall have $l = 2n$ linear inequalities. In all instances of problem (6.1) we took $d = 1$. In addition, we generated two more instances (also included in the link we provided for Table 1) with larger values of n, where only a portion of the variables are restricted by the two-sided bound constraints.

To compare the performances of Algorithm BB with the method of [7] we first estimated the values of $|\mathcal{F}^*|$ for the specific setting (6.3) of the linear constraints in problem (6.1). We note that to ensure that $F^E \neq \emptyset$, the following must hold: if $i \leq n$ satisfies $i \in E$, then $i + n \notin E$, and if $i \geq n + 1$ satisfies $i \in E$, then $i - n \notin E$. Otherwise the constraints defining the face F^E include both $x_i = l_i$ and $x_i = u_i$, which is impossible. In this case, there exists an analytic criterion for checking whether for a given set $E \subseteq \{1, \ldots, 2n\}$ the intersection $F^E \cap B[\mathbf{0}, 1]$ is empty or not, formulated as follows.

Lemma 6.1 *Consider problem* (6.1) *with the linear constraints* (6.3)*. Define* $\mathbf{b} = (u_1, \ldots, u_n, l_1, \ldots, l_n)^T$*. Then for a subset* $E \subseteq \{1, \ldots, 2n\}$ *for which the face* F^E *is not empty, it holds that* $F^E \cap B[\mathbf{0}, 1] = \emptyset$ *if and only if* $\sum_{i \in E} b_i^2 > 1$.

Proof "If": Assume $\sum_{i \in E} b_i^2 > 1$. Then for all $\mathbf{x} \in F^E$, as E can contain only one index of each couple $\{i, i+n\}$ for $i = 1, \ldots, n$, it follows that

$$\|\mathbf{x}\|^2 = \sum_{i=1}^{n} x_i^2 \geq \sum_{i \in E \cap \{1,\ldots,n\}} x_i^2 + \sum_{i \in E \cap \{n+1,\ldots,2n\}} x_{i-n}^2$$
$$= \sum_{i \in E \cap \{1,\ldots,n\}} u_i^2 + \sum_{i \in E \cap \{n+1,\ldots,2n\}} l_{i-n}^2 = \sum_{i \in E} b_i^2 > 1,$$

and thus, F^E does not intersect $B[\mathbf{0}, 1]$.

"Only if": Assume $\sum_{i \in E} b_i^2 \leq 1$. Define a vector $\tilde{\mathbf{x}}$ as follows:

$$\tilde{x}_i = \begin{cases} 0 & i, i+n \notin E, \\ u_i & i \in E, \\ l_i & i+n \in E. \end{cases}$$

Then $\tilde{\mathbf{x}}$ is well defined as E contains at most one index of each couple $\{i, i+n\}$, and it satisfies

$$\|\tilde{\mathbf{x}}\|^2 = \sum_{i=1}^{n} \tilde{x}_i^2 = \sum_{i \in E \cap \{1,\ldots,n\}} \tilde{x}_i^2 + \sum_{i \in E \cap \{n+1,\ldots,2n\}} \tilde{x}_{i-n}^2 + 0$$
$$= \sum_{i \in E \cap \{1,\ldots,n\}} u_i^2 + \sum_{i \in E \cap \{n+1,\ldots,2n\}} l_{i-n}^2 = \sum_{i \in E} b_i^2 \leq 1,$$

and $\tilde{\mathbf{x}} \in F^E$, as we assumed $l_i < 0 < u_i$, so $\tilde{\mathbf{x}} \in F^E \cap B[\mathbf{0}, 1]$. □

The implementation of the BFS on the instances described in Table 2 was done with no "black list". Though the black list had been intended to avoid some tests on immediate child-nodes of nodes (faces) already detected as infeasible, the test whether a face was in the black list practically required significant computational effort when the total numbers of faces checked became large. Thus, the results achieved without applying a black list were superior when the method was applied on the instances in Table 2. Whenever in step 2 of the algorithm we reached a face not in L, that face was checked to decide whether it intersected the ball. The simple criterion formulated in Lemma 6.1 was applied at each intersection check. It was run till a time limitation of 12 hours was reached. In some cases (where $n = 20$) the exact size of $\mathcal{F}^*$ was reached after running time of 6 minutes or less.

In the experiments described in Table 2, Algorithm BB found the optimum after evaluating only few nodes, sometimes even after just one node. The solver SCIP was not able to stop in reasonable running times for any of those instances, though in all the instances with the prefix "spar" it did compute the correct optimal solution (that is, the best upper bound till the run was interrupted manually indeed contained the optimal value, and the best feasible solution obtained by then was indeed optimal). Nevertheless, in the last two instances in Table 2 SCIP was unable to correctly detect feasibility of those problems. The results achieved by Algorithm BB clearly demonstrate the superiority of Algorithm BB over the method in [7], at

Table 2 Instances from [11], with $d = 1$ and all the constraints are of the form $l_i \le x_i \le u_i$

Case name	n	l	U	Nodes H (nH)	Time (s) H (nH)	$\vert\mathcal{F}^*\vert$ or a lower bound
spar020-100-1.mat	20	40	-180.74	3 (191)	0.01 (0.48)	$\vert\mathcal{F}^*\vert = 16{,}259$
spar020-100-2.mat	20	40	-152.32	3 (59)	0.02 (0.125)	$\vert\mathcal{F}^*\vert = 12{,}284$
spar020-100-3.mat	20	40	-181.28	1 (1)	0.016 (0.004)	$\vert\mathcal{F}^*\vert = 20{,}663$
spar030-060-1.mat	30	60	-143.14	1 (1)	0.02 (0.004)	$\vert\mathcal{F}^*\vert \ge 171{,}679$
spar030-070-1.mat	30	60	-159.18	3 (1875)	0.01 (4.22)	$\vert\mathcal{F}^*\vert \ge 147{,}079$
spar030-070-3.mat	30	60	-182.87	15 (421)	0.029 (1.04)	$\vert\mathcal{F}^*\vert \ge 132{,}278$
spar030-100-1.mat	30	60	-182.51	7 (3839)	0.03 (10.25)	$\vert\mathcal{F}^*\vert \ge 183{,}020$
spar040-080-1.mat	40	80	-224.67	3 (21)	0.02 (0.055)	$\vert\mathcal{F}^*\vert \ge 152{,}750$
spar040-100-1.mat	40	80	-234.81	3 (109)	0.02 (0.24)	$\vert\mathcal{F}^*\vert \ge 164{,}138$
spar040-100-3.mat	40	80	-264.96	1 (1)	0.02 (0.004)	$\vert\mathcal{F}^*\vert \ge 130{,}880$
Data_30of90_ers.mat	90	60	-195.485	3 (1081)	0.034 (2.51)	$\vert\mathcal{F}^*\vert \ge 148, 011$
Data_40of100_ers.mat	100	80	-281.98	15 (3757*)	0.056 (10.38*)	$\vert\mathcal{F}^*\vert \ge 270, 948$

The last two instances include two-sided bound constraints only on the first 30 or 40 variables, respectively.
* The non-heuristic BB yielded an approximate solution with $U = -280.62$

least for problem (6.1) with the bound constraints (6.3). We chose the setting (6.3), as it was tested and solved by Chen and Burer [11] and thus, it could be a good comparison between our algorithm and theirs. The paper [11] focuses on a different branch and bound method they developed for solving the problem

$$\begin{array}{ll}\min\ q_0(\mathbf{x}) \equiv \frac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} \\ \text{s.t.}\quad \mathbf{A}\mathbf{x} \le \mathbf{b}.\end{array} \tag{6.4}$$

The method for solving (6.1) presented in [11, Section 5] is an extension for the main branch and bound method presented at the same paper. The extended method for solving (6.1) solves problems of the form (6.4) by a branch and bound algorithm, and utilizes their optimal values to evaluate the function

$$L(\lambda) \equiv \min_{\mathbf{x}} \left\{ \frac{1}{2}\mathbf{x}^T\mathbf{Q}_0\mathbf{x} - \mathbf{b}_0^T\mathbf{x} + \lambda(\mathbf{x}^T\mathbf{x} - 1) : l_i \le x_i \le u_i\ \forall i \right\}$$

for any given $\lambda \ge 0$. They dual problem

$$\max_{\lambda \ge 0} L(\lambda) \tag{6.5}$$

is solved by a bisection procedure, which is possible as L is a concave function. Each evaluation of L requires solving a problem of the form (6.4). For each λ, a corresponding primal solution $\mathbf{x}$ is computed as well, as an optimal solution of the problem (6.4) for the relevant λ. Finally, when the maximizer of (6.5) λ^* is obtained, if a primal-dual solution $(\lambda^*, \mathbf{x}^*)$ satisfies $\lambda^*(1 - (\mathbf{x}^*)^T\mathbf{x}^*) = 0$, then $\mathbf{x}^*$ is an optimal solution of (6.1). Otherwise, $\lambda^*(1-(\mathbf{x}^*)^T\mathbf{x}^*)$ is the associated duality gap. The bounds l_i and u_i were randomly generated, independently, with distributions $l_i \sim U(-1, 0)$ and $u_i \sim U(0, 1)$. The ball's radius was taken as $d = 1$ in all instances. However, it is important to note that the method is not guaranteed to reach an optimal solution, as the problem (6.1) is not convex. Even in the simple case where only one linear constraint appears, there exist examples in which the duality gap is nonzero; see e.g., [17, Example 3.3].

For the sake of comparison of our results with the reported results in [11, Section 5] regarding the application of Chen and Burer's method on the above instances of (6.1) we cite their findings. Only 2 cases out of 33 were not solved within one hour by their method. They found an average running time of 233 seconds for their method applied to all other 31 instances including the 10 from Table 2. The average duality gap on those runs was 0.026%.

The performances of Algorithm BB on the example cases of [11] was superior to both methods described in [11] and [7]. It solved only few nodes and required 0.03 seconds or less, in contrast to few minutes reported in [11], and very large values of $|\mathcal{F}^*|$ in Table 2. Even in the general case of problem (6.1) the numbers of nodes being evaluated were superior to $|\mathcal{F}^*|$ in some cases, and to the total number of intersection checks performed by the BFS of [7]. As for the CPU times, it should be noted that the heuristics that chooses the order of the constraints to be involved requires an additional computation effort to compute the number of violations for a given candidate, but as it reduces (sometimes by an order of magnitude) the total number of nodes to evaluate, it indeed improves the times except for cases where only few nodes are evaluated anyway.

6.2 Sparse source localization example

In this subsection we describe results of experiments testing our approach applied to the SSL problem (5.3). Our experiments considered different settings of problem (5.3) with $n = 2, 3$ variables and $m = 6$ to $m = 13$ sensors. For each setting (m, n) and for each value of $\sigma \in \{0, 0.1, 1, 5\}$, we generated 100 random realizations of the data with the following properties:

- The sensors were located at m independent random vectors generated by
$$\mathbf{a}_i \sim U([-49, 51]^n),\ i = 1, \dots, m.$$
- The "true" source location was set at $\mathbf{x}_{true} = -50\mathbf{e} \in \mathbb{R}^n$, and the distances were noisy measurements given by
$$d_i = \|\mathbf{x}_{true} - \mathbf{a}_i\| + \varepsilon_i,\ \ i = 1, \dots, m,$$
with random noise comprised of independent normally distributed components
$$\varepsilon_i \sim N(0, \sigma),\ \ i = 1, \dots, m.$$
- To ensure positivity and reasonable values (not extremely small) of d_i, we actually took $\max\{d_i, 1\}$ instead of d_i.
- One measurement, d_1 was at first generated as the others $d_1^0 = \|\mathbf{x}_{true} - \mathbf{a}_1\| + \varepsilon_1$, but then we added another independent noise component $\delta \sim N(0, 300)$, and took the value $d_1 = \max\{d_1^0 + \delta, 1\}$.

For the small noise values ($\sigma = 0$ and $\sigma = 0.1$) we only took instances with $m = 6$, whereas for the large noise values ($\sigma = 1$ and $\sigma = 5$) we also generated some cases with $m = 8, 10, 13$ sensors. In addition, for the large noise values we also considered instances in which d_1 was not extremely noisy, and its distribution was the same as the other d_i components.

Implementation For each realization we solved the SLS problem (5.2) and the SSL problem (5.3). The solution of (SLS) was computed by applying the solution of [3] utilizing the GTRS formulation. As described in Sect. 5, the procedure for solving (SSL) comprises phases 1 and 2 for predetermining for each option of the signs vectors $\boldsymbol{\sigma} \in \{-1, 1\}^m$, whether the subproblem (5.4) is "admissible" or infeasible. In each realization of (SSL) we applied the BFS algorithm describe in phase 1, and then, on the subproblems detected as admissible, we

Table 3 Averages, min and max over 100 runs of each setting. ext/reg refer to whether d_1 contains extreme noise (ext) or not (reg)

ext/reg	n	m	σ	Admissible Ave./min/max	NodesTot Ave./min/max	Time phases $1+2$ (s) Ave./min/max	Time BB Ave./min/max
ext	2	6	0	18.3/17/29	271.1/139/838	1.2/0.5/16.9	0.1/0.1/0.4
ext	2	6	0.1	22.5/16/34	525.0/209/1019	1.4/0.5/12.7	0.3/0.11/0.5
ext	2	6	1	21.7/14/33	503.4/177/970	1.2/0.4/8.9	0.3/0.1/0.5
ext	2	8	1	47.1/30/78	1821.1/769/3641	5.0/1.3/28.8	1.0/0.4/2.1
ext	2	10	1	86.5/49/125	4678.5/1767/7863	14.4/4.1/42.9	2.4/0.9/3.9
ext	2	13	1	183.9/136/272	14147.5/6577/27,847	96.5/24.7/272.3	7.3/3.4/14.5
ext	2	6	5	20.3/12/33	456.8/122/998	0.9/0.4/3.4	0.2/0.1/0.5
ext	2	8	5	42.5/16/68	1575.1/280/3604	3.5/0.9/22.3	0.77/0.1/1.8
ext	2	13	5	160.8/103/232	12094.9/5443/25,455	78.8/16.1/193.7	6.4/2.9/13.3
reg	2	6	1	29.4/18/41	700.7/286/1208	1.6/0.6/8.9	0.3/0.1/0.6
reg	2	8	1	59.8/37/85	2143.9/968/3463	5.7/1.8/28.8	1.0/0.5/1.7
reg	2	10	1	107.7/64/156	5282.3/2506/9214	20.0/5.4/330.7	2.8/1.4/4.8
reg	2	13	1	217.9/114/292	15178.5/4394/23,928	72.0/26.9/204.7	7.9/2.3/12.2
reg	2	6	5	26.8/15/40	618.0/183/1330	1.2/0.5/8.1	0.3/0.1/0.6
reg	2	8	5	53.1/19/84	1875.7 /335/3554	3.7/1.0/22.7	0.9/0.2/1.8
reg	2	13	5	188.9/101/278	13,051.7/3775/22,084	41.9/18.3/162.3	6.5/1.9/10.8
ext	3	6	0	29.4/27/51	791.1/471/2219	1.6/0.8/18.3	0.5/0.3/1.4
ext	3	6	0.1	30.4/21/52	1030.6/509/2448	1.5/0.7/9.7	0.6/0.3/1.5
ext	3	6	1	29.3/18/51	972.9/392/2448	1.2/0.6/2.8	0.6/0.2/1.5
ext	3	8	1	81.8/57/145	5599.5/2399/12,101	6.2/2.4/18.5	3.8/1.7/7.81
ext	3	10	1	182.6/102/242	19,355.6/5993/36,055	22.3/6.5/136.6	12.0/3.7/22.5
ext	3	13	1	519.1/346/720	91205.5/38,771/157,457	130.4/44.4/321.8	58.1/24.8/100.3
ext	3	6	5	26.8/15/47	827.7/267/2585	1.1/0.5/1.9	0.5/0.2/1.5
ext	3	8	5	71.7/37/135	4585.5/1371/11,202	4.2/1.7/9.6	2.8/0.8/6.8
reg	3	6	1	46.0/31/57	1778.7/790/2935	2.1/1.1/6.4	1.2/0.5/1.9
reg	3	8	1	119.3/80/159	8174.2/3704/14,499	8.6/3.7/61.5	5.0/2.3/8.8
reg	3	10	1	253.9/152/368	27527.0/9590/46,432	29.9/11.2/145.2	15.9/5.9/28.3
reg	3	13	1	663.6/471/893	108897.5/42,897/190,969	136.5/60.6/368.0	69.3/27.5/121.2
reg	3	6	5	40.1/21/57	1443.0/377/2971	1.5/0.8/2.8	0.9/0.2/1.8
reg	3	8	5	100.1/56/156	6386.7/2286/14,806	5.3/2.0/13.2	3.9/1.4/9.0

applied phase 2. In both phases we applied the FGP-test as explained in Sect. 5. The orthogonal projection on the feasible set of (5.14) was computed at each iteration of the FGP method by applying a one-dimensional bisection procedure on a monotonically decreasing function. Each of the remaining admissible subproblems after phase 2 was solved by Algorithm BB. The heuristics of ordering the constraints with respect to the number of violations by candidates was implemented in all the experiments. Finally, we took the solution with the best (minimal) objective value among all the solutions of admissible subproblems.

The results Table 3 presents the numbers of subproblems that remained "admissible" after phases 1+2, the total number of nodes required for solving these subproblems by Algorithm

Table 4 Average, max and min error values $\|\mathbf{x} - \mathbf{x}_{true}\|$ for the same 100 realizations of Table 3

ext/reg	n	m	σ	*SLS* *Ave.*	SLS min	*SLS* *max*	*SSL* *Ave.*	SSL min	*SSL* *max*
ext	2	6	0	74.11	0.01	338.9	5.11	0	190.49
ext	2	6	0.1	74.11	0.12	339.0	5.3	0.02	190.40
ext	2	6	1	74.15	0.30	339.56	7.7	0.19	189.61
ext	2	8	1	66.35	0.67	274.63	1.45	0.09	7.10
ext	2	10	1	64.66	0.36	226.05	1.17	0.05	7.97
ext	2	13	1	46.35	0.24	249.13	0.97	0.16	3.25
ext	2	6	5	76.13	1.22	342.03	15.6	0.96	186.04
ext	2	8	5	67.27	1.17	275.88	9.69	0.44	170.18
ext	2	13	5	46.75	0.72	249.88	4.79	0.81	16.03
reg	2	6	1	1.04	0.13	3.70	1.33	0.10	5.73
reg	2	8	1	0.99	0.08	3.21	1.15	0.09	4.96
reg	2	10	1	0.82	0.04	3.29	0.99	0.12	2.73
reg	2	13	1	0.76	0.05	2.93	0.95	0.06	2.88
reg	2	6	5	5.16	0.69	18.29	6.25	0.51	28.06
reg	2	8	5	4.88	0.39	17.13	5.51	0.44	13.07
reg	2	13	5	3.76	0.14	13.55	4.63	0.29	14.82
ext	3	6	0	93.20	1.82	321.45	20.35	0.00	206.79
ext	3	6	0.1	93.22	1.74	321.48	20.62	0.03	206.73
ext	3	6	1	93.44	1.54	321.73	23.24	0.26	206.23
ext	3	8	1	82.25	2.07	282.98	5.66	0.27	156.49
ext	3	10	1	81.43	1.02	237.89	2.31	0.43	11.21
ext	3	13	1	61.22	0.93	311.93	2.03	0.47	8.69
ext	3	6	5	98.35	4.27	322.85	39.14	1.29	203.34
ext	3	8	5	82.93	5.71	284.45	18.58	1.35	182.66
reg	3	6	1	2.33	0.38	10.4	2.72	0.26	15.57
reg	3	8	1	1.91	0.35	6.84	2.25	0.40	6.82
reg	3	10	1	1.55	0.26	4.10	1.8	0.27	3.94
reg	3	13	1	1.46	0.28	4.84	1.66	0.31	4.76
reg	3	6	5	14.97	1.92	178.91	15.73	1.29	177.78
reg	3	8	5	9.57	1.76	34.19	10.97	2.00	28.29

BB, and CPU times of phases 1+2 and of Algorithm BB. Table 4 presents the error values $\|\mathbf{x} - \mathbf{x}_{true}\|$ where $\mathbf{x}$ is the obtained solution of (SLS) or (SSL). In both tables, the given values are average, minimal and maximal values over the 100 realizations. Instances with no extreme noise in d_1 are marked by "reg" in Tables 3 and 4; the cases with an extremely noisy d_1 are marked by "ext". In addition, Fig. 2 includes 4 diagrams, corresponding to noise levels $\sigma = 0, 0.1, 1, 5$. It describes the error values $\|\mathbf{x} - \mathbf{x}_{true}\|$ of the solution of (SSL) compared with the same error in (SLS) for each of the 100 realizations of the data parameters with $n = 2$, $m = 6$, and an extreme noise ("ext") in d_1.

As one could observe in Table 4, when most of the measurements contained only a small noise (or no noise), but one measurement (d_1) was extremely noisy (the case "ext"), the

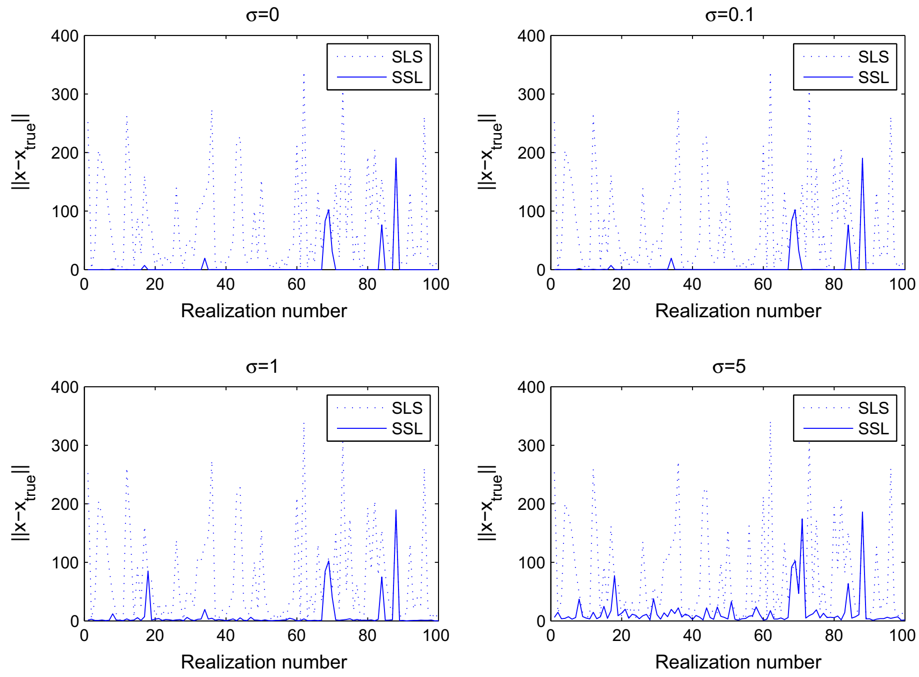

Fig. 2 Error values $\|\mathbf{x} - \mathbf{x}_{true}\|$ of the solutions of problems (SLS) and (SSL) computed for 100 individual realizations of the "ext" setting, $n = 2$ and $m = 6$

solution of (SSL) obtained much better (smaller) average error value than the solution of (SLS). In Fig. 2, the same observation can be seen for most of the individual realizations even more significantly than in average. Further notice in Fig. 2 could reveal that when most of the measurements were exact ($\sigma = 0$) we even got $\mathbf{x} = \mathbf{x}_{true}$ in many instances (although one measurement was extremely noisy), unlike the case of problem (SLS), whose solution was a very poor approximation in many cases. By the results in Table 4 it also follows that when the noise level of all the measurements was larger ($\sigma = 1$ or 5), the error of the solution of (SSL) was much lower in some cases, but much higher in others. When no extremely noisy measurement was involved ("reg"), the solution to (SSL) had no advantage (in average) over the solution to (SLS) with respect to error. The SSL formulation is therefore worthwhile only when most of the measurements are exact or very accurate, and some exceptional measurements are involved.

Table 3 provides information on the efficiency of Algorithm BB in solving subproblems of the form (5.4), including the ordering heuristics. The running times of Algorithm BB were usually a few seconds for totally solving all the "admissible" subproblems when the number of sensors was 13 with 2 or 3 variables. However, it is important to notice that the pre-process phases 1 + 2 should also be implemented efficiently, as they require much longer times than the total run of Algorithm BB on the remaining subproblems. The results described in Table 3 also reveal that the times of the implementation of Algorithm BB over all the admissible subproblems are quite stable, while the CPU times of the implementation of phases 1 + 2 admits extreme variations and large values in general. Nevertheless, phases 1 + 2 significantly reduce the number of subproblems to be solved by Algorithm BB.

It is also worthwhile to mention that in practice, at phase 1, it is very rare that a problem detected as admissible is in fact infeasible. In our simulations, such a case never occurred. That fact was checked out by a comparison with the solutions of the feasibility problem (5.6) using a standard convex optimization solver - CVX. In addition, the primal solution $\mathbf{x}^*$ obtained by the primal-dual relation (5.13) with respect to the optimal $\boldsymbol{\lambda}$ was feasible for (5.6) in all our simulations with non-positive dual optimal value. In phase 2, however, it is not unlikely that the remaining admissible subproblems contain some infeasible ones. Such cases indeed occurred in our experiments.

Concluding remarks For the lower dimensions $n = 2, 3$ usually involved in the SSL problem, with no more than $m = 13$ sensors, the method could be used to obtain a global optimal solution within about 10-15 minutes. For a non-smooth nonconvex problem like (SSL), it may be sufficient from a practical point of view. When one distance measurement is extremely noisy, the solution of (SSL) admits a low and stable error value $\|\mathbf{x} - \mathbf{x}_{true}\|$, unlike the solution of (SLS).

Acknowledgements The research of Amir Beck is partially supported by the Israel Science Foundation (ISF) Grant No. 1821/16.

References

1. Achterberg, T., Berthold, T., Koch, T., Wolter, K.: Constraint integer programming: a new approach to integrate cp and mip. ZIB-Report 08-01 (2008)
2. Beck, A., Eldar, Y.C.: Strong duality in nonconvex quadratic optimization with two quadratic constraints. SIAM J. Optim. **17**(3), 844–860 (2006)
3. Beck, A., Stoica, P., Li, J.: Exact and approximate solutions of source localiztaion problems. IEEE Trans. Signal Process. **56**(5), 1770–1778 (2008)
4. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear invers problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009)
5. Beck, A., Teboulle, M., Chikishev, Z.: Exact and approximate solutions of source localiztaion problems. IEEE Trans. Signal Process. **56**(5), 1770–1778 (2008)
6. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont (1999)
7. Bienstock, D., Michalka, A.: Polynomial solvability of variants of the trust-region subproblem. In: SODA, pp. 380–390 (2014)
8. Bruckstein, A.M., Donoho, D.L., Elad, M.: From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Rev. **51**(1), 34–81 (2009)
9. Burer, S., Anstreicher, K.: Second-order cone constraints for extended trust-region subproblems. Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA 52242, (2011)
10. Burer, S., Yang, B.: The Trust Region Subproblem with Non-intersecting Linear Constraints, Manuscript. University of Iowa, Iowa City (2013)
11. Chen, J., Burer, S.: Globally solving nonconvex quadratic programming problems via completely positive programming. Math. Program. Comput. **4**, 33–52 (2012). doi:10.1007/s12532-011-0033-9
12. Cheung, K.W., Ma, W.K., So, H.C.: Accurate approximation algorithm for toa-based maximum likelihood mobile location using semidefinite programming. In: Proceedings of ICASSP, vol. 2, pp. 145–148 (2004)
13. Gay, D.M.: Computing optimal locally constrained steps. SIAM J. Sci. Stat. Comput. **2**(2), 186–197 (1981)
14. Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. In: Blondel, V., Boyd, S., Kimura, H. (eds.) Recent Advances in Learning and Control. Lecture Notes in Control and Information Sciences, pp. 95–110. Springer, Berlin (2008). http://stanford.edu/~boyd/graph_dcp.html
15. Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 2.1, (March 2014) http://cvxr.com/cvx
16. Hsia, Y., Sheu, R.L.: Trust region subproblem with a fixed number of additional linear inequality constraints has polynomial complexity (2013). arXiv:1312.1398
17. Jeyakumar, V., Li, G.Y.: Trust-region problems with linear inequality constraints: exact sdp relaxation, global optimality and robust optimization. Math. Program. Ser. A **147**, 171–206 (2014)

18. Martínez, J.M.: Local minimizers of quadratic functions on euclidean balls and spheres. SIAM J. Optim. **4**(1), 159–176 (1994)
19. Moré, J.J.: Generalizations of the trust region subproblem. Optim. Methods Softw. **2**, 189–209 (1993)
20. Moré, J.J., Sorensen, D.C.: Computing a trust region step. SIAM J. Sci. Stat. Comput. **4**(3), 553–572 (1983)
21. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Berlin (2000)
22. Sorensen, D.C.: Newton's method with a model trust region modification. SIAM J. Numer. Anal. **19**(2), 409–426 (1982)
23. Sturm, J., Zhang, S.: On cones of nonnegative quadratic functions. Math. Oper. Res. **288**, 246–267 (2003)
24. Ye, Y., Zhang, S.: New results on quadratic minimization. SIAM J. Optim. **14**(1), 245–267 (2003)