Mathematical Programming Computation (2019) 11:755–789
https://doi.org/10.1007/s12532-019-00160-8

FULL LENGTH PAPER

An active set algorithm for robust combinatorial optimization based on separation oracles

Christoph Buchheim[1] · **Marianna De Santis**[2]

Received: 4 April 2018 / Accepted: 15 April 2019 / Published online: 24 April 2019

Abstract
We address combinatorial optimization problems with uncertain coefficients varying over ellipsoidal uncertainty sets. The robust counterpart of such a problem can be rewritten as a second-oder cone program (SOCP) with integrality constraints. We propose a branch-and-bound algorithm where dual bounds are computed by means of an active set algorithm. The latter is applied to the Lagrangian dual of the continuous relaxation, where the feasible set of the combinatorial problem is supposed to be given by a separation oracle. The method benefits from the closed form solution of the active set subproblems and from a smart update of pseudo-inverse matrices. We present numerical experiments on randomly generated instances and on instances from different combinatorial problems, including the shortest path and the traveling salesman problem, showing that our new algorithm consistently outperforms the state-of-the art mixed-integer SOCP solver of Gurobi.

Keywords Robust optimization · Active set methods · SOCP

Mathematics Subject Classification 90C25 · 90C27 · 90C57

1 Introduction

We address combinatorial optimization problems given in the general form

$$\min_{x \in P \cap \mathbb{Z}^n} \; c^\top x \tag{CP}$$

✉ Marianna De Santis
marianna.desantis@uniroma1.it

Christoph Buchheim
christoph.buchheim@math.tu-dortmund.de

[1] Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany

[2] Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy

where $P \subseteq \mathbb{R}^n$ is a compact convex set, say $P \subseteq [l, u]$ with $l, u \in \mathbb{R}^n$, and the objective function vector $c \in \mathbb{R}^n$ is assumed to be uncertain. This setting appears in many applications where the feasible set is certain, but the objective function coefficients may have to be estimated or result from imprecise measurements. As an example, when searching for a shortest path in a road network, the topology of the network is usually considered fixed, but the travel times may vary depending on the traffic conditions.

A classical way of dealing with uncertain optimization problems is the strictly robust optimization approach, introduced in [3] for linear programming and in [2] for general convex programming; we also refer the reader to the book by Ben-Tal and Nemirovski [4]. In strictly robust optimization, we look for a worst-case solution, where the uncertain parameter c is assumed to belong to a bounded set $U \subseteq \mathbb{R}^n$, called the *uncertainty set*, and the goal of the robust counterpart is to compute the solution of the following min-max problem:

$$\min_{x \in P \cap \mathbb{Z}^n} \max_{c \in U} c^\top x \tag{RP}$$

A natural choice in this approach are ellipsoidal uncertainty sets, defined as

$$U = \left\{ c \in \mathbb{R}^n \mid (c - \bar{c})^\top M (c - \bar{c}) \leq 1 \right\},$$

where $M \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix and $\bar{c} \in \mathbb{R}^n$ is the center of the ellipsoid. Assuming that the uncertain vector c in (CP), considered as a random variable, follows a normal distribution, we can interpret the ellipsoid U as a confidence set of c; in this case, M is the inverse covariance matrix of c and $\bar{c}$ is its expected value. Unfortunately, for ellipsoidal uncertainty sets, the robust counterpart (RP) is usually much harder to solve than the original problem (CP): it is known that Problem (RP) is NP-hard in this case for the shortest path problem, for the minimum spanning tree problem, and for the assignment problem [11] as well as for the unconstrained binary optimization problem [6].

Even in the case of a diagonal matrix M, i.e., when ignoring correlations and only taking variances into account, no polynomial time algorithm for the robust shortest path problem is known. There exists however an FPTAS for the diagonal case whenever the underlying problem (CP) admits an FPTAS [17], and polynomial time algorithms for the minimum spanning tree problem and the unconstrained binary problem have been devised for the diagonal case.

For general ellipsoids U, most exact solution approaches for (RP) are based on solving SOCPs. In fact, it is easy to see that the optimal objective value of the inner maximization problem

$$\max_{c \in U} c^\top x$$

for fixed x is given by

$$\bar{c}^\top x + \sqrt{x^\top M^{-1} x}.$$

Therefore, Problem (RP) is equivalent to the integer non-linear problem

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

where $Q \in \mathbb{R}^{n \times n}$ is the symmetric and positive definite inverse of M and we replace $\bar{c}$ by c for ease of notation. Note that, when addressing so called value-at-risk models

$$\begin{array}{ll} \min & z \\ \text{s.t.} & \Pr(c^\top x \geq z) \leq \varepsilon \\ & x \in P \cap \mathbb{Z}^n, \end{array}$$

we arrive at essentially the same formulation (P), assuming normally distributed coefficients again; see, e.g., [17].

In the following, we assume that the convex set P is given by a separation algorithm, i.e., an algorithm that decides whether a given point $\bar{x} \in \mathbb{R}^n$ belongs to P or not, and, in the negative case, provides an inequality $a^\top x \leq b$ valid for P but violated by $\bar{x}$. Even in cases where the underlying problem (CP) is tractable, the polytope $\text{conv}(P \cap \mathbb{Z}^n)$ may have an exponential number of facets, so that a full linear description cannot be used efficiently. This is true, e.g., for the standard formulation of the spanning tree problem. However, we do not require that a complete linear description of $\text{conv}(P \cap \mathbb{Z}^n)$ be known; it suffices to have an integer linear description, i.e., we allow $P \neq \text{conv}(P \cap \mathbb{Z}^n)$. In particular, our approach can also be applied when the underlying problem is NP-hard, e.g., when (CP) models the traveling salesman problem.

As soon as P is given explicitly by linear constraints $Ax \leq b$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, the continuous relaxation of Problem (P) reduces to an SOCP of the form

$$\begin{array}{ll} \min & c^\top x + \sqrt{x^\top Q x} \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{R}^n. \end{array} \tag{R2}$$

Such SOCPs can be solved efficiently using interior point algorithms [16] and popular solvers for SOCPs such as Gurobi [10], SeDuMi [19] or MOSEK [15] are based on interior point methods. However, in our branch-and-bound algorithm, we need to address a sequence of related SOCPs. Compared with interior point methods, active set methods have the advantage to allow warmstarts.

For this reason, in order to solve the SOCP relaxations of Problem (RP), we devised the active set algorithm `EllAS`. It is applied to the Lagrangian dual of (R2) and exploits the fact that the active set subproblems can be solved by closed form expressions. For this, the main ingredient is the pseudo-inverse of $AQ^{-\frac{1}{2}}$. Since the matrix A is updated in each iteration of the active set method, an incremental update of the pseudo-inverse is crucial for the running time of `EllAS`. Altogether, we can achieve a running time of $O(n^2)$ per iteration. Combined with an intelligent embedding into the branch-and-bound scheme, we obtain an algorithm that consistently outperforms the MISOCP solver of Gurobi 7.5.1, where the latter is either applied to a full linear description

of P or, in case a compact linear description does not exist, uses the same separation oracle as EllAS.

The rest of the paper is organized as follows: the Lagrangian dual of (R2) is derived in Sect. 2. The closed-form solution of the resulting active set subproblems is developed in Sect. 3. The active set algorithm EllAS is detailed and analyzed in Sects. 4 and 5. In Sect. 6, we discuss how to embed EllAS into a branch-and-bound algorithm. Numerical results for random SOCP instances, random integer instances as well as instances of different combinatorial optimization problems are reported in Sect. 7. Section 8 concludes.

2 Dual problem

The algorithm we propose for solving Problem (RP) uses the Lagrangian dual of relaxations of the form (R2). Let $\mathscr{L}(x,\lambda) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be the Lagrangian function associated to (R2):

$$\mathscr{L}(x,\lambda) = c^\top x + \sqrt{x^\top Q x} + \lambda^\top (Ax - b).$$

The Lagrangian dual of Problem (R2) is then

$$\max_{\lambda \in \mathbb{R}^m_+} \inf_{x \in \mathbb{R}^n} \mathscr{L}(x,\lambda). \tag{1}$$

After applying the bijective transformation $z = Q^{\frac{1}{2}} x$, the inner minimization problem of (1) becomes

$$-b^\top \lambda + \inf_{z \in \mathbb{R}^n} \left(Q^{-\frac{1}{2}} \left(c + A^\top \lambda\right)\right)^\top z + \|z\|$$

for fixed $\lambda \in \mathbb{R}^m_+$. It is easy to see that

$$\inf_{z \in \mathbb{R}^n} \left(Q^{-\frac{1}{2}} \left(c + A^\top \lambda\right)\right)^\top z + \|z\| = \min_{z \in \mathbb{R}^n} \left(Q^{-\frac{1}{2}} \left(c + A^\top \lambda\right)\right)^\top z + \|z\| = 0$$

if $\|Q^{-\frac{1}{2}}(c + A^\top \lambda)\| \leq 1$ and $-\infty$ otherwise. Therefore, Problem (1) reduces to

$$\begin{aligned} \max\ & -b^\top \lambda \\ \text{s.t.}\ & \left(c + A^\top \lambda\right)^\top Q^{-1} \left(c + A^\top \lambda\right) \leq 1 \\ & \lambda \geq 0. \end{aligned} \tag{D}$$

Theorem 1 *For the primal-dual pair of optimization problems* (R2) *and* (D)*, strong duality holds as soon as one of the two problems is feasible. Moreover, if one of the problems admits an optimal solution, the same holds for the other problem.*

Proof This follows from the convexity of (R2) and from the fact that all constraints in (R2) are affine linear. □

In order to solve Problem (R2), we have devised the dual active set algorithm EllAS detailed in Sect. 4. Along its iterations, EllAS produces dual feasible solutions of Problem (D), converging to a KKT point of Problem (R2) and therefore producing also a primal optimal solution when terminating.

3 Solving the active set subproblem

At every iteration, the active set algorithm EllAS presented in the subsequent sections fixes certain dual variables to zero while leaving unconstrained the remaining variables. In the primal problem, this corresponds to choosing a set of valid linear constraints $Ax \leq b$ for P and replacing inequalities by equations. We thus need to solve primal-dual pairs of problems of the following type:

$$\begin{aligned} \min \quad & f(x) = c^\top x + \sqrt{x^\top Q x} \\ \text{s.t.} \quad & \hat{A}x = \hat{b} \\ & x \in \mathbb{R}^n \end{aligned} \tag{P-AS}$$

$$\begin{aligned} \max \quad & -\hat{b}^\top \lambda \\ \text{s.t.} \quad & \left(c + \hat{A}^\top \lambda\right)^\top Q^{-1} \left(c + \hat{A}^\top \lambda\right) \leq 1 \\ & \lambda \in \mathbb{R}^{\hat{m}} \end{aligned} \tag{D-AS}$$

where $\hat{A} \in \mathbb{R}^{\hat{m} \times n}$, $b \in \mathbb{R}^{\hat{m}}$. For the efficiency of our algorithm, it is crucial that this pair of problems can be solved in closed form. For this, the pseudo-inverse $(\hat{A}Q^{-\frac{1}{2}})^+$ of $\hat{A}Q^{-\frac{1}{2}}$ will play an important role. It can be used to compute orthogonal projections onto the kernel and onto the range of $Q^{-\frac{1}{2}}\hat{A}^\top$ as follows: we have

$$\operatorname{proj}_{\ker\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)}(y) = y - \hat{A}Q^{-\frac{1}{2}}\left(\hat{A}Q^{-\frac{1}{2}}\right)^+ y \tag{2}$$

and

$$\operatorname{proj}_{\operatorname{ran}\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)}(y) = \left(\hat{A}Q^{-\frac{1}{2}}\right)^+ \hat{A}Q^{-\frac{1}{2}} y, \tag{3}$$

see e.g. [13]. We later explain how to update the pseudo-inverse incrementally instead of computing it from scratch in every iteration, which would take $O(n^3)$ time; see Sect. 5.2.

In the following, we assume that the dual problem (D-AS) admits a feasible solution; this will be guaranteed in every iteration of our algorithm; see Lemma 1 below.

3.1 Dual unbounded case

If $\hat{b} \notin \operatorname{ran}(\hat{A})$, or equivalently, if $\hat{b}$ is not orthogonal to $\ker(\hat{A}^\top) = \ker(Q^{-\frac{1}{2}}\hat{A}^\top)$, then the dual problem (D-AS) is unbounded, and the corresponding primal problem (P-AS) is infeasible. When this case occurs, `EllAS` uses an unbounded direction of (D-AS) to continue. The set of unbounded directions of (D-AS) is $\ker(Q^{-\frac{1}{2}}\hat{A}^\top)$. Consequently, the unbounded direction with steepest ascent can be obtained by projecting the gradient of the objective function $-\hat{b}$ to $\ker(Q^{-\frac{1}{2}}\hat{A}^\top)$. According to (2), this projection is

$$\operatorname{proj}_{\ker\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)}(-\hat{b}) = \left(\hat{A}Q^{-\frac{1}{2}}\right)\left(\hat{A}Q^{-\frac{1}{2}}\right)^{+}\hat{b} - \hat{b}.$$

3.2 Bounded case

If $\hat{b} \in \operatorname{ran}(\hat{A})$, we first consider the special case $\hat{b} = 0$. As we assume (D-AS) to be feasible, its optimum value is thus 0. Therefore, the corresponding primal problem (P-AS) admits $x^* = 0$ as optimal solution. In the following, we may thus assume $\hat{b} \neq 0$. The feasible set of problem (D-AS) consists of all $\lambda \in \mathbb{R}^{\hat{m}}$ such that

$$\left\| Q^{-\frac{1}{2}}\left(c + \hat{A}^\top\lambda\right)\right\| \leq 1,$$

i.e., such that the image of λ under $-Q^{-\frac{1}{2}}\hat{A}^\top$ belongs to the ball $B_1(Q^{-\frac{1}{2}}c)$. Consider the orthogonal projection of $Q^{-\frac{1}{2}}c$ to the subspace $\operatorname{ran}(Q^{-\frac{1}{2}}\hat{A}^\top)$, which by (3) is

$$q := \operatorname{proj}_{\operatorname{ran}\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)}\left(Q^{-\frac{1}{2}}c\right) = \left(Q^{-\frac{1}{2}}\hat{A}^\top\right)\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)^{+}Q^{-\frac{1}{2}}c.$$

If $||q - Q^{-\frac{1}{2}}c|| > 1$, then the intersection $B_1(Q^{-\frac{1}{2}}c) \cap \operatorname{ran}(Q^{-\frac{1}{2}}\hat{A}^\top)$ is empty, so that Problem (D-AS) is infeasible, contradicting our assumption. Hence, we have that this intersection is a ball with center q and radius

$$r := \sqrt{1 - ||q - Q^{-\frac{1}{2}}c||^2}$$

and $\lambda \in \mathbb{R}^{\hat{m}}$ is feasible for (D-AS) if and only if $-Q^{-\frac{1}{2}}\hat{A}^\top\lambda \in B_r(q)$. Since $\hat{b} \in \operatorname{ran}(\hat{A}Q^{-\frac{1}{2}})$, we have $(\hat{A}Q^{-\frac{1}{2}})(\hat{A}Q^{-\frac{1}{2}})^{+}\hat{b} = \hat{b}$. This allows us to rewrite the objective function $-\hat{b}^\top\lambda$ of (D-AS) in terms of $Q^{-\frac{1}{2}}\hat{A}^\top\lambda$ only, as

$$-\hat{b}^\top\lambda = -\hat{b}^\top\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)^{+}\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)\lambda.$$

We can thus first compute the optimal solution $v^* \in \text{ran}(Q^{-\frac{1}{2}}\hat{A}^\top)$ of

$$\begin{aligned} \max \quad & \hat{b}^\top \left(Q^{-\frac{1}{2}}\hat{A}^\top\right)^+ v \\ \text{s.t.} \quad & ||Q^{-\frac{1}{2}}c - v|| \leq 1, \end{aligned}$$

which is unique since $\hat{b} \neq 0$, and then solve $v^* = -(Q^{-\frac{1}{2}}\hat{A}^\top)\lambda$. We obtain

$$v^* = q + \frac{r}{||\left(\hat{A}Q^{-\frac{1}{2}}\right)^+ \hat{b}||}\left(\hat{A}Q^{-\frac{1}{2}}\right)^+ \hat{b}, \tag{4}$$

so that we can state the following

Proposition 1 *Let $\hat{b} \in \text{ran}(\hat{A})\backslash\{0\}$ and let v^* be defined as in (4). Then, the unique optimal solution of (D-AS) with minimal norm is*

$$\lambda^* := -\left(Q^{-\frac{1}{2}}\hat{A}^\top\right)^+ v^*.$$

From λ^*, it is possible to compute an optimal solution x^* of the primal problem (P-AS) as explained in the following result.

Theorem 2 *Let $\hat{b} \in \text{ran}(\hat{A})\backslash\{0\}$. Let λ^* be an optimal solution of (D-AS) and $\bar{x} := Q^{-1}(c + \hat{A}^\top\lambda^*)$.*

(a) *If $\hat{b}^\top\lambda^* \neq 0$, then the unique optimal solution of (P-AS) is $x^* = \alpha\bar{x}$, with*

$$\alpha := -\frac{\hat{b}^\top\lambda^*}{c^\top\bar{x} - \sqrt{\bar{x}^\top Q\bar{x}}}.$$

(b) *Otherwise, there exists a unique $\alpha < 0$ such that $\alpha\hat{A}\bar{x} = \hat{b}$. Then, $x^* = \alpha\bar{x}$ is the unique optimal solution of (P-AS).*

Proof Let (x^*, λ^*) be a primal-dual optimal pair for (P-AS) and (D-AS), which exists by the same reasoning as in the proof of Theorem 1. Since $\hat{b} \neq 0$ and $\hat{A}x^* = \hat{b}$, it follows that $x^* \neq 0$. The gradient equation yields

$$0 = \nabla_x \mathscr{L}(x^*, \lambda^*) = c + \frac{2Qx^*}{2\sqrt{(x^*)^\top Q(x^*)}} + \hat{A}^\top\lambda^*$$

which is equivalent to

$$\frac{Q^{\frac{1}{2}}x^*}{\|Q^{\frac{1}{2}}x^*\|} = -Q^{-\frac{1}{2}}\left(c + \hat{A}^\top\lambda^*\right)$$

and hence to

$$x^* = \beta Q^{-1}\left(c + \hat{A}^\top\lambda^*\right) = \beta\bar{x}$$

for some $\beta \neq 0$. Since $\beta = -\|Q^{\frac{1}{2}}x^*\|$, we have $\beta < 0$. By strong duality, we then obtain

$$-\hat{b}^\top \lambda^* = c^\top x^* + \sqrt{(x^*)^\top Q(x^*)} = \beta c^\top \bar{x} + |\beta|\sqrt{\bar{x}^\top Q\bar{x}} = \beta\big(c^\top \bar{x} - \sqrt{\bar{x}^\top Q\bar{x}}\big).$$

Now if $\hat{b}^\top \lambda^* \neq 0$, also the right hand side of this equation is non-zero, and we obtain β as claimed. Otherwise, it still holds that there exists $\beta < 0$ such that $\beta\bar{x}$ is optimal. In particular, $\beta\bar{x}$ is primal feasible and hence $\beta\hat{A}\bar{x} = \hat{A}(\beta\bar{x}) = \hat{b}$. As $\hat{b} \neq 0$, we derive $\hat{A}\bar{x} \neq 0$, as $\beta < 0$. This in particular shows that β is uniquely defined by $\beta\hat{A}\bar{x} = \hat{b}$. □

Note that the proof (and hence the statement) for case (b) in Theorem 2 are formally applicable also in case (a). However, in the much more relevant case (a), we are able to derive a closed formula for β in a more direct way.

4 The dual active set method EllAS

As all active set methods, our algorithm EllAS tries to forecast the set of constraints that are active at the optimal solution of the primal-dual pair (R2) and (D), adapting this forecast iteratively: starting from a subset of primal constraints $A^{(1)}x \leq b^{(1)}$, where $A^{(1)} \in \mathbb{R}^{m^{(1)}\times n}$ and $b^{(1)} \in \mathbb{R}^{m^{(1)}}$, one constraint is removed or added per iteration by performing a dual or a primal step; see Algorithm 1. We assume that a corresponding dual feasible solutions $\lambda^{(1)} \geq 0$ is given when starting the algorithm; we explain below how to obtain this initial solution.

Algorithm 1 Ellipsoidal Active Set algorithm EllAS

Input: $Q \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$, $A^{(1)} \in \mathbb{R}^{m^{(1)}\times n}$, $b^{(1)} \in \mathbb{R}^{m^{(1)}}$;
$\lambda^{(1)} \geq 0$ with $(c + (A^{(1)})^\top\lambda^{(1)})^\top Q^{-1}(c + (A^{(1)})^\top\lambda^{(1)}) \leq 1$;
pseudo-inverse $(A^{(1)}Q^{-\frac{1}{2}})^+$
Output: optimal solutions of (R2) and (D)

1: **for** $k = 1, 2, 3, \ldots$ **do**
2: **solve** (D-ASk) and obtain optimal $\tilde{\lambda}^{(k)}$ with minimal norm
3: **if** problem (D-ASk) is bounded and $\tilde{\lambda}^{(k)} \geq 0$ **then**
4: **set** $\lambda^{(k)} := \tilde{\lambda}^{(k)}$
5: **perform** the primal step (Algorithm 3) and **update** $x^{(k)}$, $A^{(k)}$, $b^{(k)}$
6: **else**
7: **perform** the dual step (Algorithm 2) and **update** $\lambda^{(k)}$, $A^{(k)}$, $b^{(k)}$
8: **end if**
9: **end for**

At every iteration k, in order to decide whether to perform the primal or the dual step, the dual subproblem is addressed, namely Problem (D) where only the subset of active constraints is taken into account. This leads to the following problem:

$$\begin{aligned} \max\ & -b^{(k)\top}\lambda \\ \text{s.t.}\ & \left(c + A^{(k)\top}\lambda\right)^{\top} Q^{-1}\left(c + A^{(k)\top}\lambda\right) \leq 1 \\ & \lambda \in \mathbb{R}^{m^{(k)}}. \end{aligned} \tag{D-ASk}$$

The solution of Problem (D-ASk) has been explained in Sect. 3. Note that formally Problem (D-ASk) is defined in a smaller space with respect to Problem (D), but its solutions can also be considered as elements of $\mathbb{R}^m$ by setting the remaining variables to zero.

In case the dual step is performed, the solution of Problem (D-ASk) gives an ascent direction $p^{(k)}$ along which we move in order to produce a new dual feasible point with better objective function value. We set

$$\lambda^{(k)} = \lambda^{(k-1)} + \alpha^{(k)} p^{(k)},$$

where the steplength $\alpha^{(k)}$ is chosen to be the largest value for which non-negativity is maintained at all entries. Note that the feasibility with respect to the ellipsoidal constraint in (D), i.e.,

$$\left(c + A^{\top}\lambda\right)^{\top} Q^{-1}\left(c + A^{\top}\lambda\right) \leq 1,$$

is guaranteed from how p^k is computed, using convexity. Therefore, $\alpha^{(k)}$ can be derived by considering the negative entries of $p^{(k)}$. In order to maximize the increase of $-b^{\top}\lambda$, we ask $\alpha^{(k)}$ to be as large as possible subject to maintaining non-negativity; see Steps 9–10 in Algorithm 2.

The constraint index j computed in Step 9 of Algorithm 2 corresponds to the primal constraint that needs to be released from the active set. The new iterate $\lambda^{(k+1)}$ is then obtained from $\lambda^{(k)}$, by dropping the jth entry.

Algorithm 2 Dual Step

1: **if** problem (D-ASk) is bounded **then**
2: **set** $p^{(k)} := \tilde{\lambda}^{(k)} - \lambda^{(k-1)}$
3: **else**
4: **let** $p^{(k)}$ be an unbounded direction of (D-ASk) with steepest ascent
5: **if** $p^{(k)} \geq 0$ **then**
6: **STOP:** primal problem is infeasible
7: **end if**
8: **end if**
9: **choose** $j \in \operatorname{argmin}\{-\lambda_i^{(k-1)}/p_i^{(k)} \mid i = 1, \ldots, m^{(k)},\ p_i^{(k)} < 0\}$
10: **set** $\alpha^{(k)} := -\lambda_j^{(k-1)}/p_j^{(k)}$
11: **set** $\lambda^{(k)} := \lambda^{(k-1)} + \alpha^{(k)} p^{(k)}$
12: **compute** $(A^{(k+1)}, b^{(k+1)})$ by removing row j in $(A^{(k)}, b^{(k)})$
13: **compute** $\lambda^{(k+1)}$ by removing entry j in $\lambda^{(k)}$
14: **set** $m^{(k+1)} := m^{(k)} - 1$
15: **update** $(A^{(k+1)}Q^{-\frac{1}{2}})^{+}$ from $(A^{(k)}Q^{-\frac{1}{2}})^{+}$

Proposition 2 *The set considered in Step 9 of Algorithm 2 is non-empty.*

Proof If Problem (D-ASk) is bounded, there is an index i such that $\tilde{\lambda}_i^{(k)} < 0$, since $\tilde{\lambda}^{(k)}$ is dual infeasible. As $\lambda^{(k-1)} \geq 0$, we derive $p_i^{(k)} = \tilde{\lambda}_i^{(k)} - \lambda_i^{(k-1)} < 0$. If Problem (D-ASk) is unbounded, we explicitly check whether $p^{(k)} \geq 0$ and only continue otherwise. □

The primal step is performed in case the solution of Problem (D-ASk) gives us a dual feasible solution. Starting from this dual feasible solution, we compute a corresponding primal solution $x^{(k)}$ according to the formula in Theorem 2. If $x^{(k)}$ belongs to P we are done: we have that $(x^{(k)}, \lambda^{(k)})$ is a KKT point of Problem (R2) and, by convexity of Problem (R2), $x^{(k)}$ is its global optimum. Otherwise, we compute a cutting plane violated by $x^{(k)}$ that can be considered active and will be then taken into account in defining the dual subproblem (D-ASk) at the next iteration. The new iterate $\lambda^{(k+1)}$ is obtained from $\lambda^{(k)}$ by adding an entry to $\lambda^{(k)}$ and setting this additional entry to zero.

Algorithm 3 Primal Step

if $b^{(k)} = 0$ **then**
  STOP: $(0, \lambda^{(k)})$ is an optimal primal-dual solution
else
  compute $x^{(k)}$ from $\lambda^{(k)}$ according to Theorem 2
  if $x^{(k)} \in P$ **then**
    STOP: $(x^{(k)}, \lambda^{(k)})$ is an optimal primal-dual solution
  else
    compute a cutting plane $a^\top x \leq b$ violated by $x^{(k)}$
    compute $(A^{(k+1)}, b^{(k+1)})$ by appending $(a^\top, b)$ to $(A^{(k)}, b^{(k)})$
    compute $\lambda^{(k+1)}$ by appending zero to $\lambda^{(k)}$
    set $m^{(k+1)} := m^{(k)} + 1$
    update $(A^{(k+1)} Q^{-\frac{1}{2}})^+$ from $(A^{(k)} Q^{-\frac{1}{2}})^+$
  end if
end if

Theorem 3 *Whenever Algorithm* `EllAS` *terminates, the result is correct.*

Proof If Algorithm `EllAS` stops at the primal step, the optimality of the resulting primal-dual pair follows from the discussion in Sect. 3. If Algorithm `EllAS` stops at the dual step, it means that the ascent direction $p^{(k)}$ computed is a feasible unbounded direction for Problem (D), so that Problem (D) is unbounded and hence Problem (R2) is infeasible. □

It remains to describe how to initialize `EllAS`. For this, we use the assumption of boundedness of P and construct $A^{(1)}$, $b^{(1)}$, and $\lambda^{(1)}$ as follows: for each $i = 1, \ldots, n$, we add the constraint $x_i \leq u_i$ if $c_i < 0$, with corresponding $\lambda_i := -c_i$, and the constraint $-x_i \leq -l_i$ otherwise, with $\lambda_i := c_i$. These constraints are valid since we assumed $P \subseteq [l, u]$ and it is easy to check that $(A^{(1)})^\top \lambda^{(1)} = -c$ by construction, so that $\lambda^{(1)}$ is dual feasible for (D). Moreover, we can easily compute $(A^{(1)} Q^{-\frac{1}{2}})^+$ in this case, as $A^{(1)}$ is a diagonal matrix with ± 1 entries: this implies $(A^{(1)} Q^{-\frac{1}{2}})^+ = Q^{\frac{1}{2}} A^{(1)}$.

To conclude this section, we shortly discuss how to deal with linear equations in EllAS. While it is perfectly feasible to replace each equation by two linear inequalities and then apply the algorithm exactly as described above, its performance can be improved by dealing with equations directly. For an equation, the corresponding dual variable in (D) is not restricted to be non-negative any more. In particular, in Step 9 of Algorithm 2 we only need to take the minimum over those indices corresponding to bounded variables. In other words, dual variables corresponding to equations will never leave the active set again, so that the presence of equations does not significantly increase the number of active set iterations.

5 Analysis of the algorithm

In this section, we show that Algorithm EllAS converges in a finite number of steps if cycling is avoided. Moreover, we prove that the running time per iteration can be bounded by $O(n^2)$, if implemented properly.

5.1 Convergence analysis

Our convergence analysis follows similar arguments to those used in [18] for the analysis of primal active set methods for strictly convex quadratic programming problems. In particular, as in [18], we assume that we can always take a nonzero steplength along the ascent direction. Under this assumption we will show that Algorithm EllAS does not undergo cycling, or, in other words, this assumption prevents from having $\lambda^{(k)} = \lambda^{(l)}$ and $(A^{(k)}, b^{(k)}) = (A^{(l)}, b^{(l)})$ in two different iterations k and l. As for other active set methods, it is very unlikely in practice to encounter a zero steplength. However, there are techniques to avoid cycling even theoretically, such as perturbation or lexicographic pivoting rules in Step 9 of Algorithm 2.

Lemma 1 *At every iteration k of Algorithm EllAS, Problem* (D-ASk) *admits a feasible solution.*

Proof It suffices to show that the ellipsoidal constraint

$$\left(c + {A^{(k)}}^\top \lambda^{(k)}\right)^\top Q^{-1} \left(c + {A^{(k)}}^\top \lambda^{(k)}\right) \leq 1 \tag{5}$$

is satisfied for each k. For $k = 1$, this is explicitely required for the input of Algorithm EllAS. Let $\lambda^{(k)}$ be computed from $\lambda^{(k-1)}$ by moving along the direction $p^{(k)}$. The feasibility of $\lambda^{(k)}$ with respect to (5) then follows from the definition of $p^{(k)}$ and from the convexity of the ellipsoid, taking into account that $\alpha \leq 1$, since otherwise we would not enter the dual step. □

Proposition 3 *At every iteration k of Algorithm EllAS, the vector $\lambda^{(k)}$ is feasible for* (D).

Proof Taking into account the proof of Lemma 1, it remains to show nonnegativity of $\lambda^{(k)}$, which is guaranteed by the choice of the steplength $\alpha^{(k)}$. □

Proposition 4 *Assume that the steplength α^k is always non-zero in the dual step. If Algorithm* `EllAS` *does not stop at iteration k, then one of the following holds:*

(i) $-b^{(k+1)\top}\lambda^{(k+1)} > -b^{(k)\top}\lambda^{(k)}$;
(ii) $-b^{(k+1)\top}\lambda^{(k+1)} = -b^{(k)\top}\lambda^{(k)}$ *and* $\|\lambda^{(k+1)}\| < \|\lambda^{(k)}\|$.

Proof In the primal step, suppose that $\tilde{\lambda}^{(k)} \geq 0$ solves Problem (D-ASk) and that the corresponding unique primal solution satisfies $x^{(k)} \notin P$. After adding a violated cutting plane, the optimal value of Problem (P-AS) strictly increases and the same is true for the optimal value of Problem (D-AS) by strong duality. Then,

$$p^{(k+1)} = \tilde{\lambda}^{(k+1)} - \lambda^{(k)} = \tilde{\lambda}^{(k+1)} - \tilde{\lambda}^{(k)}$$

is a strict ascent direction for $-b^\top\lambda$ and case (i) holds.

In the dual step, if $p^{(k+1)}$ is an unbounded direction, case (i) holds again. Otherwise, observe that $\lambda^{(k)} \neq \tilde{\lambda}^{(k+1)}$, as $\tilde{\lambda}^{(k+1)}$ is not feasible with respect to the nonnegativity constraints. Then, since $\tilde{\lambda}^{(k+1)}$ is the unique optimal solution for Problem (D-ASk) with minimal norm, $p^{(k+1)} = \tilde{\lambda}^{(k+1)} - \lambda^{(k)}$ is either a strict ascent direction for $-b^\top\lambda$, or $-b^\top p^{(k+1)} = 0$ and $p^{(k+1)}$ is a strict descent direction for $\|\lambda\|$, so that case (ii) holds. □

Lemma 2 *At every iteration k of Algorithm* `EllAS`*, we have $m^{(k)} \leq n+1$. Furthermore, if Algorithm* `EllAS` *terminates at iteration k with an optimal primal-dual pair, then $m^{(k)} \leq n$.*

Proof As only violated cuts are added, the primal constraints $A^{(k)}x = b^{(k)}$ either form an infeasible system or are linearly independent. If $m^{(k)} = n+1$, the primal problem is hence infeasible. Thus Problem (D-ASk) is unbounded, so that at iteration k a dual step is performed and a dependent row of $(A^{(k)}, b^{(k)})$ is deleted, leading to an independent set of constraints again. □

Theorem 4 *Assume that whenever a dual step is performed, Algorithm* `EllAS` *takes a non-zero steplength α^k. Moreover, assume that P is a polytope and that the separation oracle can produce at most m different inequalities. Then, after at most $n2^m$ iterations, Algorithm* `EllAS` *terminates with a primal-dual pair of optimal solutions for* (R2) *and* (D)*.*

Proof First note that, by Lemma 2, at most n dual steps can be performed in a row. Hence, it is enough to show that in any two iterations $k \neq l$ where a primal step is performed, we have $(A^{(k)}, b^{(k)}) \neq (A^{(l)}, b^{(l)})$. Otherwise, assuming $(A^{(k)}, b^{(k)}) = (A^{(l)}, b^{(l)})$, we obtain $\tilde{\lambda}^{(k)} = \tilde{\lambda}^{(l)}$ and hence $\lambda^{(k)} = \lambda^{(l)}$. This leads to a contradiction to Proposition 4. □

5.2 Running time per iteration

After computing Q^{-1} in the preprocessing, the running time in iteration k of `EllAS` is $O(n^2 + m^{(k)}n)$ and hence linear in the size of the matrices Q and $A^{(k)}$, if implemented properly. The main work is to keep the pseudo-inverse $(A^{(k)}Q^{-\frac{1}{2}})^+$ up-to-date.

Since $A^{(k)}Q^{-\frac{1}{2}}$ is only extended or shrunk by one row in each iteration, an update of $(A^{(k)}Q^{-\frac{1}{2}})^+$ is possible in $O(m^{(k)}n)$ time by a generalization of the Sherman–Morrison-formula [12]. Exploiting the fact that the matrix $A^{(k)}$ has full row rank in most iterations, we can proceed as follows. If $A^{(k+1)}$ is obtained from $A^{(k)}$ by adding a new row a, we first compute the row vectors

$$h := aQ^{-\frac{1}{2}}\left(A^{(k)}Q^{-\frac{1}{2}}\right)^+, \quad v := aQ^{-\frac{1}{2}} - hA^{(k)}Q^{-\frac{1}{2}}.$$

Now $v \neq 0$ if and only if $A^{(k+1)}$ has full row rank, and in the latter case

$$\left(A^{(k+1)}Q^{-\frac{1}{2}}\right)^+ = \left((A^{(k)}Q^{-\frac{1}{2}})^+ \mid 0\right) - \tfrac{1}{||v||^2}v^\top(h \mid -1).$$

Otherwise, if $v = 0$, we are adding a linearly dependent row to $A^{(k)}$, making the primal problem (P-AS) infeasible. In this case, an unbounded direction of steepest ascent of (D-AS) is given by $(-h \mid 1)^\top$ and the next step will be a dual step, meaning that a row will be removed from $A^{(k+1)}$ and the resulting matrix $A^{(k+2)}$ will have full row rank again. We can thus update $(A^{(k)}Q^{-\frac{1}{2}})^+$ to $(A^{(k+2)}Q^{-\frac{1}{2}})^+$ by first removing and then adding a row, in both cases having full row rank.

It thus remains to deal with the case of deleting the rth row a of a matrix $A^{(k)}$ with full row rank. Here we obtain $(A^{(k+1)}Q^{-\frac{1}{2}})^+$ by deleting the rth column in

$$\left(A^{(k)}Q^{-\frac{1}{2}}\right)^+ - \tfrac{1}{||w||^2}ww^\top\left(A^{(k)}Q^{-\frac{1}{2}}\right)^+,$$

where w is the rth column of $(A^{(k)}Q^{-\frac{1}{2}})^+$.

Theorem 5 *The running time per iteration of Algorithm* `EllAS` *is* $O(n^2)$.

Proof This follows directly from Lemma 2 and the discussion above. □

Clearly, the incremental update of the pseudo-inverse $(A^{(k)}Q^{-\frac{1}{2}})^+$ may cause numerical errors. This can be avoided by recomputing it from scratch after a certain number of incremental updates. Instead of a fixed number of iterations, we recompute $(A^{(k)}Q^{-\frac{1}{2}})^+$ whenever the primal solution computed in a primal step is infeasible, i.e., violates the current constraints, where we allow a small tolerance.

In order to avoid wrong solutions even when pseudo-inverses are not precise, we make sure in our implementation that the dual solution $\lambda^{(k)}$ remains feasible for (D-AS) in each iteration, no matter how big the error of $(A^{(k)}Q^{-\frac{1}{2}})^+$ is. For this, we slightly change the computation of $\tilde{\lambda}^{(k)}$: after computing $\tilde{\lambda}^{(k)}$ exactly as explained, we determine the largest $\delta \in \mathbb{R}$ such that $(1-\delta)\lambda^{(k-1)} + \delta\tilde{\lambda}^{(k)}$ is dual feasible. Such δ must exist since $\lambda^{(k-1)}$ is dual feasible, and it can easily be computed using the midnight formula. We then replace $\tilde{\lambda}^{(k)}$ by $(1-\delta)\lambda^{(k-1)} + \delta\tilde{\lambda}^{(k)}$ and go on as before.

6 Branch-and-bound algorithm

For solving the integer Problem (RP), the method presented in the previous sections must be embedded into a branch-and-bound scheme. The dual bounds are computed by Algorithm EllAS and the branching is done by splitting up the domain $[l_i, u_i]$ of some variable x_i. Several properties of Algorithm EllAS can be exploited to improve the performance of such a branch-and-bound approach.

Warmstarts Clearly, as branching adds new constraints to the primal feasible region of the problem, while never extending it, all dual solutions remain feasible. In every node of the branch-and-bound-tree, the active set algorithm can thus be warm started with the optimal set of constraints of the parent node. As in [7,8], this leads to a significant reduction of the number of iterations compared to a cold start. Moreover, the newly introduced bound constraint is always violated and can be directly added as a new active constraint, which avoids resolving the same dual problem and hence saves one more iteration per node. Finally, the data describing the problem can either be inherited without changes or updated quickly; this is particularly important for the pseudo-inverse $(AQ^{-\frac{1}{2}})^+$.

Early pruning Since we compute a valid dual bound for Problem (RP) in every iteration of Algorithm EllAS, we may prune a subproblem as soon as the current bound exceeds the value of the best known feasible solution.

Avoiding cycling or tailing off Last but not least, we may also stop Algorithm EllAS at every point without compromising the correctness of the branch-and-bound algorithm. In particular, as soon as an iteration of Algorithm EllAS does not give a strict (or a significant) improvement in the dual bound, the active set algorithm is stopped and we resort to branching. This excludes any potential cycling of the algorithm.

7 Numerical results

To test the performance of our algorithm EllAS, we considered convex SOCP instances (Sect. 7.1), random binary instances with up to one million constraints (Sect. 7.2), and combinatorial instances of Problem (RP), where the underlying problem is the Shortest Path problem (Sect. 7.3), the Assignment problem (Sect. 7.4), the Spanning Tree problem (Sect. 7.5), and the Traveling Salesman problem (Sect. 7.6). Concerning our approach, these combinatorial problems have different characteristics: while the first two problems have compact and complete linear formulations, the standard models for the latter problems use an exponential number of constraints that can be separated efficiently. In the case of the Spanning Tree problem, this exponential set of constraints again yields a complete linear formulation, while this is not the case for the NP-hard Traveling Salesman problem. In the latter case, however, we still have a complete integer programming formulation, which suffices for the correctness of our approach.

For all problems, we consider instances where the positive definite matrix $Q \in \mathbb{R}^{n\times n}$ is randomly generated. For this, we chose n eigenvalues λ_i uniformly at random

from [0, 1] and orthonormalized n random vectors v_i, each entry of which was chosen uniformly at random from $[-1, 1]$. Setting $\bar{Q} = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$, the entries of Q are given as $Q_{ij} = c_i c_j \bar{Q}_{ij}$, where c is the vector defining the linear term. For the random instances, the entries of c were chosen uniformly at random from $[-1, 0]$, while for all remaining instances the entries of c were chosen uniformly at random from [0.5, 1.5]. In the (very few) cases where the smallest eigenvalue of Q turned out to be smaller than 10^{-8}, we discarded the instance and produced a new one.

In the following, we present a comparison of `BB-EllAS`, an implementation of the branch-and-bound-algorithm based on `EllAS` in C++, with the MISOCP solver of Gurobi 7.5.1 [10]. According to the latest benchmark results of Hans D. Mittelmann [14], Gurobi is currently the fastest solver for SOCPs and MISOCPs. Furthermore, in order to analyze the performance of `EllAS` as an algorithm for convex SOCPs, we report in Sect. 7.1 a comparison between `EllAS` and the SOCP solver of Gurobi 7.5.1 [10].

For `BB-EllAS`, we use a simple home-made branch-and-bound framework using a straightforward depth first search. The most fractional variable is chosen for branching. We use an optimality tolerance of 10^{-4}. The same tolerance is applied when deciding whether a constraint is violated in a separation procedure. For Gurobi, we use standard settings, except that we apply the same optimality tolerance as in `BB-EllAS`, setting the absolute optimality tolerance `MIPGapAbs` to 10^{-4}. All other standard parameters are unchanged. In particular, Gurobi uses presolve techniques that decrease the solution times significantly. In case of the Spanning Tree problem and the Traveling Salesman problem, we apply dynamic separation algorithms using a callback adding lazy constraints. Again, constraints are added if the violation exceeds 10^{-4}.

All our experiments were carried out on Intel Xeon processors running at 2.60 GHz. All running times were measured in CPU seconds (not including the time needed for instance generation) and the time-limit was set to one CPU hour for each individual instance. All tables presented in this section include the following data for the comparison between `BB-EllAS` and Gurobi: the number of instances solved within the time limit (# sol), the average running time, and the average number of branch-and-bound nodes where applicable. For `BB-EllAS`, we also report the average total number of active set iterations (iter) and the average number of times the pseudo-inverse $(A^{(k)} Q^{-\frac{1}{2}})^+$ is recomputed from scratch, the latter in percentage with respect to the number of iterations (% ps). All averages are taken over the set of instances solved to optimality within the time limit. For all applications, we also present performance profiles, as proposed in [9]. Given our set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, we compare the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ against the best performance obtained by any solver in $\mathcal{S}$ on the same problem. To this end we define the performance ratio $r_{p,s} = t_{p,s} / \min\{t_{p,s'} : s' \in \mathcal{S}\}$, where $t_{p,s}$ is the computational time, and we consider a cumulative distribution function $\rho_s(\tau) = |\{p \in \mathcal{P} : \ r_{p,s} \leq \tau\}|/|\mathcal{P}|$. The performance profile for $s \in S$ is the plot of the function ρ_s.

7.1 Random SOCP instances

We first consider continuous relaxations of Problem (P), where the vector $c \in \mathbb{R}^n$ and the positive definite matrix $Q \in \mathbb{R}^{n \times n}$ are randomly generated as described above. The objective function is of the form $c^\top x + \beta \sqrt{x^\top Q x}$, where we use $\beta = \sqrt{(1-\epsilon)/\epsilon}$ as proposed in [5]. In order to study the effect of changing the weight β of the non-linear term, as done in [1], we consider instances with $\epsilon \in \{0.5, 0.2, 0.1, 0.05, 0.02, 0.01\}$, corresponding to values of β between 1 and 7. The set P is explicitely given as $\{x \in [0,1]^n \mid Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are also randomly generated: the entries of A were chosen uniformly at random from the integers in the range $[0, 10]$ and we set $b_i = \lfloor 0.5 \sum_{j=1}^n a_{ij} \rfloor$ for $i = 1, \ldots, m$. Altogether, we generated 480 different instances: for each combination of ϵ, $n \in \{25, 50\}$, and $m \in \{10^3, 10^4, 10^5, 10^6\}$, we generated 10 instances. Since the set P is explicitely given here, the linear constraints are separated by enumeration in `BB-EllAS`. More precisely, at Step 8 of Algorithm 3, we pick the linear constraint most violated by $x^{(k)}$.

In Table 1, we report the comparison between `EllAS` and the SOCP solver of Gurobi. Gurobi is not able to solve any instance with $n = 50$ variables and $m = 10^6$ constraints within the time limit. On the other hand, `EllAS` is in general very fast, but there are some instances where it stops without reaching optimality (these instances are counted as unsolved). Our implementation of `EllAS` is in fact tuned to compute valid lower bounds for Problem (P) quickly: we stop our algorithm as soon as an iteration does not give us a strict (or significant) improvement in the dual bound. Nevertheless, `EllAS` is able to compute the optimum for 444 out of 480 instances and the average running time is below four CPU seconds for all instance types. In Fig. 1, we report the performance profiles for all the instances.

7.2 Random binary instances

We next consider instances of Problem (P) where the vector $c \in \mathbb{R}^n$ and the positive definite matrix $Q \in \mathbb{R}^{n \times n}$ as well as the polyhedron P are created exactly as described in the previous section, except that $b_i = \lfloor q \sum_{j=1}^n a_{ij} \rfloor$ for $i = 1, \ldots, m$, with $q \in \{0.1, 0.2, 0.5\}$. However, we now allow only binary solutions, i.e., we require $x \in P \cap \{0,1\}^n$. Altogether, we generated 1440 different problem instances for (P): for each combination of q, ϵ, $n \in \{25, 50\}$, and $m \in \{10^3, 10^4, 10^5, 10^6\}$, we generated 10 instances.

In Tables 2, 3 and 4, we report the results for $q = 0.5$, $q = 0.2$, and $q = 0.1$, respectively. Besides the results of `BB-EllAS` and Gurobi, we report the results obtained by `BB-EllAS` without using warmstarts (`BB-EllAS NW`) and by Gurobi where all constraints are defined as lazy constraints (Gurobi LC). The latter has been considered in order to improve Gurobi's performance: from the results shown, it is clear that Gurobi benefits from the use of lazy constraints. It is also evident how the use of warmstarts improves the performance of `BB-EllAS`: the average number of iterations decreases up to a factor of 100. For `BB-EllAS`, smaller values of q lead to shorter running times in general. For Gurobi, instances with $q = 0.2$ are the hardest to solve, while instances with $q = 0.1$ are the easiest for both `BB-EllAS`

Table 1 Comparison on random SOCP instances

ε	n	m	EllAS				Gurobi	
			#sol	Time	iter	%ps	#sol	Time
0.5	25	10^3	10	0.00	1.4e+2	0.84	10	0.04
	25	10^4	10	0.01	2.0e+2	0.71	10	0.37
	25	10^5	10	0.15	2.3e+2	0.74	10	6.32
	25	10^6	10	1.92	2.6e+2	0.65	10	90.18
	50	10^3	9	0.01	2.5e+2	1.35	10	0.09
	50	10^4	10	0.03	3.4e+2	1.91	10	1.20
	50	10^5	9	0.29	4.1e+2	0.82	10	44.45
	50	10^6	9	3.85	5.1e+2	0.65	0	–
0.2	25	10^3	10	0.00	1.3e+2	0.32	10	0.03
	25	10^4	10	0.01	1.5e+2	0.19	10	0.38
	25	10^5	10	0.13	1.9e+2	0.16	10	5.99
	25	10^6	8	1.61	2.1e+2	0.12	10	79.70
	50	10^3	9	0.01	2.1e+2	0.74	10	0.08
	50	10^4	9	0.03	2.8e+2	1.03	10	1.21
	50	10^5	9	0.28	3.6e+2	0.92	10	48.23
	50	10^6	7	3.43	4.2e+2	0.38	0	–
0.1	25	10^3	10	0.00	1.1e+2	0.09	10	0.04
	25	10^4	10	0.01	1.5e+2	0.00	10	0.39
	25	10^5	9	0.12	1.7e+2	0.00	10	6.32
	25	10^6	10	1.48	1.9e+2	0.10	10	81.80
	50	10^3	10	0.01	2.0e+2	0.59	10	0.09
	50	10^4	8	0.02	2.7e+2	0.28	10	1.17
	50	10^5	8	0.26	3.2e+2	0.56	10	46.04
	50	10^6	9	3.06	3.8e+2	0.61	0	–
0.05	25	10^3	10	0.00	1.1e+2	0.28	10	0.04
	25	10^4	9	0.01	1.3e+2	0.18	10	0.40
	25	10^5	10	0.10	1.5e+2	0.14	10	5.39
	25	10^6	10	1.23	1.6e+2	0.12	10	67.40
	50	10^3	10	0.01	1.9e+2	0.21	10	0.09
	50	10^4	10	0.02	2.5e+2	0.67	10	1.12
	50	10^5	8	0.25	3.0e+2	0.63	10	43.19
	50	10^6	9	2.80	3.4e+2	0.69	0	–
0.02	25	10^3	9	0.00	8.9e+1	0.00	10	0.04
	25	10^4	10	0.01	1.0e+2	0.00	10	0.40
	25	10^5	10	0.08	1.2e+2	0.08	10	5.65
	25	10^6	10	0.88	1.3e+2	0.15	10	68.63

Table 1 continued

ε	n	m	EllAS				Gurobi	
			#sol	Time	iter	%ps	#sol	Time
	50	10^3	9	0.01	1.7e+2	0.33	10	0.09
	50	10^4	8	0.02	2.4e+2	0.32	10	1.25
	50	10^5	8	0.22	2.5e+2	0.35	10	46.39
	50	10^6	7	2.63	2.9e+2	0.45	0	–
0.01	25	10^3	10	0.00	7.4e+1	0.14	10	0.03
	25	10^4	10	0.01	8.6e+1	0.23	10	0.39
	25	10^5	8	0.07	1.0e+2	0.12	10	5.60
	25	10^6	10	0.79	1.1e+2	0.10	10	65.06
	50	10^3	10	0.01	1.6e+2	0.19	10	0.09
	50	10^4	9	0.02	2.0e+2	0.17	10	1.33
	50	10^5	9	0.19	2.3e+2	0.10	10	46.91
	50	10^6	8	2.25	2.5e+2	0.35	0	–

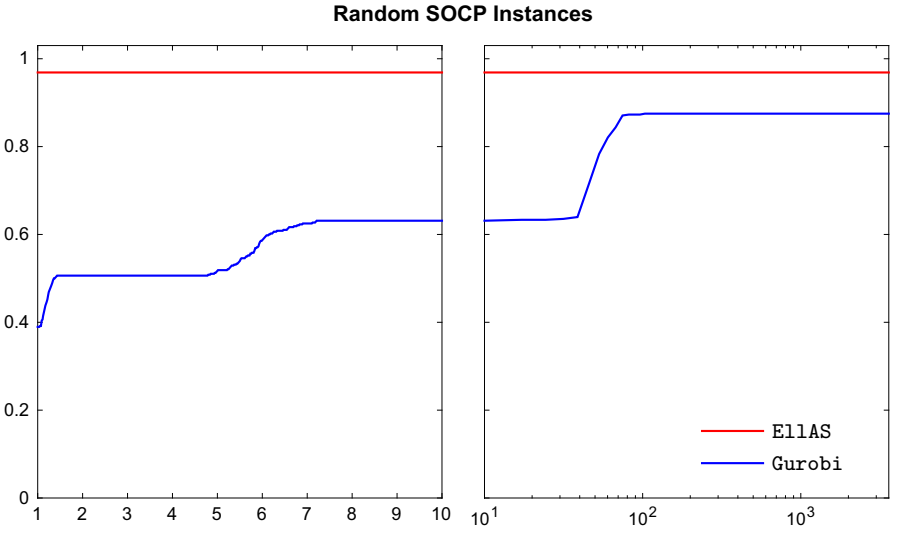

Fig. 1 Performance profile with respect to running times for convex SOCP instances

and Gurobi. Note that the average number of branch-and-bound nodes enumerated by `BB-EllAS` is generally larger than the number of nodes needed by Gurobi, but always by less than a factor of 10 on average. On all instance classes, `BB-EllAS` is able to solve significantly more instances than Gurobi and Gurobi LC within the time limit, and in general has a faster running time. This is confirmed by the performance profiles presented in Fig. 2, where `BB-EllAS` clearly outperforms both Gurobi and the improved Gurobi LC.

Table 2 Comparison on random binary instances, $q = 0.5$

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
0.5	25	10^3	10	0.30	1.7e+3	1.5e+4	1.13	10	0.67	1.7e+3	8.7e+4	0.16	10	3.24	7.1e+2	10	3.85	1.8e+3
	25	10^4	10	1.76	1.9e+3	2.2e+4	1.20	10	4.96	1.9e+3	1.3e+5	0.21	10	31.95	8.1e+2	10	12.12	3.7e+3
	25	10^5	10	51.73	3.7e+3	4.6e+4	1.22	10	124.58	3.8e+3	2.6e+5	0.26	10	472.83	1.1e+3	10	48.27	7.0e+3
	25	10^6	10	716.27	5.9e+3	7.0e+4	1.34	8	1581.84	3.7e+3	2.6e+5	0.45	3	2917.23	5.6e+2	10	282.05	1.1e+4
	50	10^3	10	7.66	1.2e+4	1.3e+5	2.26	10	41.52	1.2e+4	1.4e+6	0.97	10	29.55	5.0e+3	10	33.55	7.3e+3
	50	10^4	10	47.60	1.4e+4	2.0e+5	3.21	10	179.50	1.8e+4	2.3e+6	1.39	10	333.71	5.2e+3	10	76.01	1.2e+4
	50	10^5	10	1469.06	4.6e+4	8.3e+5	2.08	3	520.90	5.5e+3	8.6e+5	3.77	2	2147.47	1.7e+3	10	725.10	7.7e+4
	50	10^6	2	1585.30	5.9e+3	1.1e+5	1.69	0	–	–	–	–	0	–	–	0	–	–
0.2	25	10^3	10	0.20	1.2e+3	1.0e+4	1.01	10	0.46	1.2e+3	6.3e+4	0.06	10	3.23	7.3e+2	10	4.91	1.9e+3
	25	10^4	10	1.45	1.7e+3	1.8e+4	0.98	10	4.13	1.7e+3	1.0e+5	0.08	10	41.75	8.3e+2	10	14.45	3.6e+3
	25	10^5	10	35.78	3.0e+3	3.4e+4	1.05	10	89.95	3.0e+3	1.9e+5	0.10	10	560.75	1.1e+3	10	55.94	7.3e+3
	25	10^6	10	579.44	5.2e+3	5.7e+4	1.15	8	1269.11	3.5e+3	2.2e+5	0.20	2	2905.02	4.2e+2	10	283.50	1.1e+4
	50	10^3	10	6.78	1.2e+4	1.2e+5	1.99	10	31.18	1.2e+4	1.3e+6	0.64	10	38.89	5.4e+3	10	53.50	8.1e+3
	50	10^4	10	51.02	2.0e+4	2.5e+5	2.23	10	171.02	2.1e+4	2.5e+6	0.79	10	452.40	6.1e+3	10	184.89	2.0e+4
	50	10^5	10	1379.43	4.7e+4	7.8e+5	1.74	3	559.16	5.8e+3	8.5e+5	2.70	1	525.32	6.1e+2	8	725.76	5.9e+4
	50	10^6	2	1436.56	4.4e+3	8.2e+4	1.34	0	–	–	–	–	0	–	–	0	–	–
0.1	25	10^3	10	0.20	1.3e+3	9.7e+3	1.09	10	0.47	1.3e+3	6.0e+4	0.03	10	3.10	7.0e+2	10	5.26	1.9e+3
	25	10^4	10	1.26	1.6e+3	1.5e+4	0.72	10	3.52	1.6e+3	8.8e+4	0.05	10	44.89	8.6e+2	10	13.86	3.3e+3
	25	10^5	10	36.55	3.2e+3	3.3e+4	0.89	10	92.62	3.2e+3	1.8e+5	0.05	10	764.64	1.3e+3	10	62.32	6.8e+3
	25	10^6	10	484.66	4.4e+3	4.6e+4	0.95	9	1422.80	3.5e+3	2.1e+5	0.10	2	3223.78	4.1e+2	10	290.73	1.1e+4

Table 2 continued

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
	50	10^3	10	6.14	1.2e+4	1.2e+5	1.82	10	28.82	1.3e+4	1.3e+6	0.52	10	46.73	6.1e+3	10	73.92	9.7e+3
	50	10^4	10	38.84	1.5e+4	1.8e+5	2.23	10	128.75	1.6e+4	1.9e+6	0.74	10	609.04	8.1e+3	10	199.28	1.6e+4
	50	10^5	10	1677.85	5.1e+4	7.9e+5	1.40	4	1158.51	1.3e+4	1.7e+6	0.62	2	2066.85	1.9e+3	8	1085.90	5.8e+4
	50	10^6	3	1698.48	6.6e+3	1.1e+5	4.62	0	–	–	–	–	0	–	–	0	–	–
0.05	25	10^3	10	0.14	1.1e+3	7.6e+3	0.59	10	0.37	1.1e+3	4.8e+4	0.03	10	3.19	6.9e+2	10	4.96	1.7e+3
	25	10^4	10	1.21	1.6e+3	1.4e+4	0.73	10	3.24	1.6e+3	8.0e+4	0.04	10	41.66	7.6e+2	10	19.25	4.0e+3
	25	10^5	10	28.22	2.6e+3	2.5e+4	0.72	10	73.74	2.6e+3	1.4e+5	0.04	10	604.16	1.1e+3	10	69.75	6.8e+3
	25	10^6	10	489.52	4.9e+3	4.6e+4	0.79	9	1192.35	3.7e+3	2.0e+5	0.07	2	3390.46	4.0e+2	10	308.27	1.1e+4
	50	10^3	10	5.17	1.3e+4	1.1e+5	1.40	10	21.82	1.3e+4	1.3e+6	0.28	10	58.31	6.2e+3	10	93.96	9.6e+3
	50	10^4	10	37.90	1.6e+4	1.9e+5	1.62	9	133.19	1.8e+4	2.1e+6	0.48	10	688.96	8.9e+3	10	392.07	2.5e+4
	50	10^5	8	1524.78	4.7e+4	6.7e+5	1.20	3	436.85	5.2e+3	7.0e+5	1.74	1	495.35	4.7e+2	6	1071.67	4.7e+4
	50	10^6	3	1656.91	6.4e+3	1.0e+5	4.06	1	2508.22	3.1e+3	4.7e+5	0.10	0	–	–	0	–	–
0.02	25	10^3	10	0.10	1.1e+3	7.1e+3	0.13	10	0.33	1.1e+3	4.4e+4	0.04	10	2.27	4.7e+2	10	4.67	1.4e+3
	25	10^4	10	0.86	1.3e+3	9.9e+3	0.26	10	2.22	1.3e+3	5.8e+4	0.03	10	31.45	6.0e+2	10	15.05	3.4e+3
	25	10^5	10	20.28	2.2e+3	1.8e+4	0.38	10	49.13	2.2e+3	9.9e+4	0.03	10	661.34	9.3e+2	10	59.22	5.9e+3
	25	10^6	10	370.97	3.7e+3	3.2e+4	0.59	10	1062.58	3.7e+3	1.7e+5	0.04	2	3284.41	3.4e+2	10	293.35	1.0e+4
	50	10^3	10	5.26	1.7e+4	1.4e+5	1.00	10	23.47	1.8e+4	1.6e+6	0.15	10	73.19	8.7e+3	10	130.09	1.4e+4
	50	10^4	10	54.86	2.8e+4	2.8e+5	1.21	10	185.21	3.0e+4	3.1e+6	0.30	9	785.83	1.0e+4	10	508.79	2.9e+4
	50	10^5	7	1381.24	6.0e+4	7.3e+5	0.89	3	550.04	7.2e+3	8.6e+5	1.39	1	571.09	4.6e+2	4	749.68	3.3e+4
	50	10^6	2	761.93	3.3e+3	4.6e+4	0.59	1	2030.99	2.5e+3	3.5e+5	0.06	0	–	–	0	–	–

Table 2 continued

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
0.01	25	10^3	10	0.07	8.7e+2	5.0e+3	0.13	10	0.23	8.7e+2	3.1e+4	0.04	10	2.60	5.0e+2	10	3.89	1.4e+3
	25	10^4	10	0.71	1.2e+3	8.1e+3	0.09	10	1.71	1.2e+3	4.7e+4	0.04	10	34.70	6.4e+2	10	11.82	2.6e+3
	25	10^5	10	15.86	1.9e+3	1.4e+4	0.10	10	37.46	1.9e+3	7.5e+4	0.03	10	497.49	7.4e+2	10	46.22	4.5e+3
	25	10^6	9	276.12	2.9e+3	2.2e+4	0.19	10	713.24	2.8e+3	1.1e+5	0.04	2	3270.61	2.9e+2	10	259.69	7.5e+3
	50	10^3	10	5.13	2.0e+4	1.5e+5	0.85	10	25.30	2.0e+4	1.7e+6	0.19	10	102.94	1.1e+4	10	180.27	1.8e+4
	50	10^4	10	62.51	3.5e+4	3.2e+5	0.79	10	206.70	3.8e+4	3.6e+6	0.19	9	1116.72	1.5e+4	10	792.46	4.6e+4
	50	10^5	7	1392.69	5.4e+4	5.9e+5	0.68	3	904.34	1.3e+4	1.4e+6	1.22	1	657.59	5.4e+2	5	1254.26	5.2e+4
	50	10^6	2	799.48	4.2e+3	5.1e+4	0.49	1	2147.40	2.9e+3	3.4e+5	0.08	0	–	–	0	–	–

Table 3 Comparison on random binary instances, $q = 0.2$

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
0.5	25	10^3	10	0.11	6.3e+2	4.7e+3	1.28	10	0.25	6.4e+2	3.0e+4	0.23	10	1.96	9.2e+1	10	0.73	5.1e+2
	25	10^4	10	0.41	3.3e+2	4.5e+3	1.21	10	0.77	3.3e+2	1.9e+4	0.26	10	22.66	1.9e+0	10	1.66	9.0e+1
	25	10^5	10	4.69	3.4e+2	5.6e+3	1.22	10	9.98	3.4e+2	2.1e+4	0.28	1	1272.20	1.0e+0	10	172.68	2.6e+0
	25	10^6	10	56.31	3.2e+2	5.3e+3	1.19	10	111.83	3.2e+2	2.1e+4	0.33	0	–	–	0	–	–
	50	10^3	10	6.70	9.3e+3	1.2e+5	1.88	10	32.94	9.3e+3	1.3e+6	0.76	10	16.58	2.3e+3	10	47.87	8.3e+3
	50	10^4	10	65.10	2.2e+4	3.2e+5	1.84	10	188.58	2.2e+4	3.0e+6	0.77	10	217.45	4.2e+3	10	197.08	2.3e+4
	50	10^5	9	1612.64	5.4e+4	8.1e+5	1.85	5	2547.54	3.3e+4	4.6e+6	0.98	2	3403.55	4.6e+3	10	629.99	4.3e+4
	50	10^6	0	–	–	–	–	0	–	–	–	–	0	–	–	0	–	–
0.2	25	10^3	10	0.09	6.4e+2	4.4e+3	0.99	10	0.23	6.4e+2	2.9e+4	0.09	10	1.67	9.6e+1	10	1.11	5.4e+2
	25	10^4	10	0.35	3.2e+2	3.9e+3	0.87	10	0.66	3.2e+2	1.7e+4	0.10	10	33.48	4.9e+0	10	1.73	1.0e+2
	25	10^5	10	4.17	3.3e+2	4.8e+3	0.90	10	8.93	3.3e+2	1.8e+4	0.09	0	–	–	10	172.58	8.8e+0
	25	10^6	10	47.52	3.1e+2	4.5e+3	0.86	10	103.31	3.1e+2	1.8e+4	0.13	0	–	–	0	–	–
	50	10^3	10	5.74	9.7e+3	1.2e+5	1.58	10	25.08	9.8e+3	1.3e+6	0.41	10	25.07	2.9e+3	10	86.02	1.1e+4
	50	10^4	10	76.01	2.8e+4	3.5e+5	1.38	10	192.84	2.8e+4	3.5e+6	0.34	10	278.47	4.2e+3	10	299.67	2.4e+4
	50	10^5	8	1510.80	5.3e+4	7.1e+5	1.50	4	2759.62	3.5e+4	4.4e+6	0.71	0	–	–	10	996.93	4.6e+4
	50	10^6	0	–	–	–	–	0	–	–	–	–	0	–	–	0	–	–
0.1	25	10^3	10	0.09	6.9e+2	4.4e+3	0.85	10	0.21	6.9e+2	2.9e+4	0.05	10	1.59	1.0e+2	10	1.30	6.0e+2
	25	10^4	10	0.34	3.4e+2	3.8e+3	0.73	10	0.64	3.3e+2	1.7e+4	0.07	10	52.67	6.4e+0	10	1.82	1.2e+2
	25	10^5	10	3.79	3.2e+2	4.3e+3	0.66	10	7.76	3.2e+2	1.6e+4	0.07	0	–	–	10	172.54	1.3e+1
	25	10^6	10	43.30	3.0e+2	4.0e+3	0.69	10	87.86	3.0e+2	1.7e+4	0.08	0	–	–	0	–	–

Table 3 continued

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
	50	10^3	10	4.79	1.0e+4	1.2e+5	1.13	10	20.01	1.0e+4	1.3e+6	0.19	10	30.17	3.0e+3	10	130.17	9.1e+3
	50	10^4	10	54.10	2.2e+4	2.7e+5	1.09	10	144.23	2.2e+4	2.8e+6	0.20	10	340.87	4.5e+3	10	519.31	2.7e+4
	50	10^5	7	1777.01	6.3e+4	7.5e+5	1.15	4	2464.15	3.8e+4	4.5e+6	0.48	0	–	–	10	1329.95	4.3e+4
	50	10^6	1	2504.67	7.1e+3	1.1e+5	0.78	0	–	–	–	–	0	–	–	0	–	–
0.05	25	10^3	10	0.08	6.8e+2	4.0e+3	0.67	10	0.19	6.8e+2	2.8e+4	0.03	10	1.73	1.0e+2	10	1.28	5.5e+2
	25	10^4	10	0.30	3.1e+2	3.4e+3	0.50	10	0.55	3.1e+2	1.5e+4	0.05	10	54.26	1.2e+1	10	1.80	1.1e+2
	25	10^5	10	3.29	3.0e+2	3.7e+3	0.43	10	7.11	3.0e+2	1.5e+4	0.04	0	–	–	10	173.85	2.0e+1
	25	10^6	10	38.92	2.9e+2	3.4e+3	0.48	10	82.87	2.9e+2	1.5e+4	0.06	0	–	–	0	–	–
	50	10^3	10	4.24	1.1e+4	1.2e+5	0.86	10	18.04	1.1e+4	1.3e+6	0.09	10	34.89	3.3e+3	10	152.95	1.0e+4
	50	10^4	10	56.38	2.4e+4	2.8e+5	0.87	10	147.77	2.4e+4	2.9e+6	0.12	10	424.22	4.9e+3	10	596.00	2.6e+4
	50	10^5	8	1383.49	5.7e+4	6.5e+5	0.93	4	2001.47	3.2e+4	3.8e+6	0.33	0	–	–	10	1750.51	4.8e+4
	50	10^6	1	2849.56	7.5e+3	1.1e+5	0.61	0	–	–	–	–	0	–	–	0	–	–
0.02	25	10^3	10	0.06	6.2e+2	3.5e+3	0.20	10	0.16	6.3e+2	2.4e+4	0.03	10	1.68	1.1e+2	10	1.02	5.4e+2
	25	10^4	10	0.25	2.8e+2	2.8e+3	0.29	10	0.46	2.8e+2	1.3e+4	0.04	10	97.44	2.4e+1	10	1.85	1.4e+2
	25	10^5	10	2.87	2.8e+2	2.9e+3	0.27	10	5.86	2.8e+2	1.2e+4	0.03	0	–	–	10	173.33	2.2e+1
	25	10^6	10	31.11	2.7e+2	2.7e+3	0.34	10	64.41	2.7e+2	1.2e+4	0.04	0	–	–	0	–	–
	50	10^3	10	3.93	1.3e+4	1.2e+5	0.63	10	17.43	1.2e+4	1.4e+6	0.06	10	46.95	3.5e+3	10	174.94	1.1e+4
	50	10^4	10	49.06	2.4e+4	2.6e+5	0.58	10	133.94	2.4e+4	2.8e+6	0.06	10	489.00	4.8e+3	10	618.33	2.7e+4
	50	10^5	9	1580.76	7.8e+4	7.5e+5	0.61	4	1432.18	2.3e+4	2.7e+6	0.22	0	–	–	9	1536.27	4.4e+4
	50	10^6	1	3271.13	1.6e+4	2.1e+5	2.65	0	–	–	–	–	0	–	–	0	–	–

Table 3 continued

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
0.01	25	10^3	10	0.05	4.7e+2	2.7e+3	0.11	10	0.12	4.7e+2	1.8e+4	0.03	10	1.76	1.2e+2	10	0.95	5.5e+2
	25	10^4	10	0.19	2.3e+2	2.2e+3	0.23	10	0.35	2.3e+2	1.0e+4	0.03	10	141.54	2.9e+1	10	1.83	1.3e+2
	25	10^5	10	2.10	2.0e+2	2.1e+3	0.29	10	3.97	2.0e+2	9.2e+3	0.05	0	–	–	10	173.90	9.6e+0
	25	10^6	10	22.22	1.9e+2	1.9e+3	0.28	10	44.76	1.9e+2	9.2e+3	0.03	0	–	–	0	–	–
	50	10^3	10	4.25	1.7e+4	1.4e+5	0.43	10	20.47	1.6e+4	1.7e+6	0.05	10	53.57	3.9e+3	10	183.29	1.2e+4
	50	10^4	10	53.43	2.7e+4	2.6e+5	0.49	10	145.39	3.0e+4	3.1e+6	0.05	10	554.16	5.0e+3	10	599.01	2.6e+4
	50	10^5	10	1401.56	6.9e+4	6.4e+5	0.44	5	2009.93	3.7e+4	3.8e+6	0.11	0	–	–	9	1448.73	4.1e+4
	50	10^6	2	2487.15	1.0e+4	1.3e+5	0.37	0	–	–	–	–	0	–	–	0	–	–

Table 4 Comparison on random binary instances, $q = 0.1$

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
0.5	25	10^3	10	0.02	4.8e+1	9.4e+2	0.96	10	0.03	4.8e+1	2.8e+3	0.22	10	0.01	0.2e+0	10	0.01	0.1e+0
	25	10^4	10	0.10	4.5e+1	1.2e+3	0.98	10	0.12	4.5e+1	2.8e+3	0.19	10	0.06	0.0e+0	10	0.07	0.0e+0
	25	10^5	10	1.12	4.6e+1	1.4e+3	0.93	10	1.61	4.6e+1	3.0e+3	0.15	10	2.59	0.0e+0	10	1.94	0.0e+0
	25	10^6	10	14.30	4.6e+1	1.6e+3	0.85	10	18.86	4.6e+1	3.2e+3	0.18	10	21.47	0.0e+0	10	18.77	0.0e+0
	50	10^3	10	2.66	5.9e+3	5.0e+4	1.69	10	10.59	5.9e+3	4.8e+5	0.53	10	5.15	4.5e+2	10	9.46	1.3e+3
	50	10^4	10	5.84	3.1e+3	3.5e+4	1.70	10	18.29	3.1e+3	2.8e+5	0.71	10	54.83	2.1e+2	10	12.11	1.0e+3
	50	10^5	10	58.65	1.2e+3	3.7e+4	1.64	10	113.83	1.2e+3	1.5e+5	0.76	9	621.82	1.0e+0	10	128.44	8.3e+1
	50	10^6	10	729.85	1.3e+3	4.1e+4	1.60	10	1407.93	1.3e+3	1.6e+5	0.74	0	–	–	0	–	–
0.2	25	10^3	10	0.02	4.8e+1	8.4e+2	0.70	10	0.03	4.8e+1	2.6e+3	0.11	10	0.01	0.1e+0	10	0.01	0.1e+0
	25	10^4	10	0.09	4.2e+1	1.0e+3	0.58	10	0.10	4.2e+1	2.4e+3	0.05	10	0.06	0.0e+0	10	0.07	0.0e+0
	25	10^5	10	0.98	4.2e+1	1.2e+3	0.58	10	1.40	4.2e+1	2.6e+3	0.07	10	2.53	0.0e+0	10	1.96	0.0e+0
	25	10^6	10	12.46	4.2e+1	1.4e+3	0.65	10	14.61	4.2e+1	2.6e+3	0.09	10	20.18	0.0e+0	10	19.34	0.0e+0
	50	10^3	10	2.43	6.7e+3	5.1e+4	1.35	10	9.02	6.6e+3	5.1e+5	0.27	10	6.00	4.9e+2	10	12.34	1.3e+3
	50	10^4	10	5.02	2.9e+3	3.0e+4	1.34	10	15.38	3.0e+3	2.7e+5	0.39	10	48.17	2.2e+2	10	14.44	1.1e+3
	50	10^5	10	50.05	1.2e+3	3.1e+4	1.09	10	101.11	1.2e+3	1.4e+5	0.35	6	716.52	1.0e+0	10	128.03	8.9e+1
	50	10^6	10	629.92	1.3e+3	3.5e+4	1.03	10	1195.56	1.3e+3	1.5e+5	0.33	0	–	–	0	–	–
0.1	25	10^3	10	0.01	4.7e+1	7.7e+2	0.60	10	0.02	4.7e+1	2.4e+3	0.06	10	0.01	0.1e+0	10	0.01	0.1e+0
	25	10^4	10	0.07	3.9e+1	9.6e+2	0.43	10	0.09	3.9e+1	2.2e+3	0.06	10	0.07	0.0e+0	10	0.07	0.0e+0
	25	10^5	10	0.86	3.9e+1	1.1e+3	0.51	10	1.12	3.9e+1	2.3e+3	0.04	10	2.55	0.0e+0	10	2.04	0.0e+0
	25	10^6	10	10.48	3.9e+1	1.2e+3	0.40	10	13.70	3.9e+1	2.4e+3	0.03	10	22.21	0.0e+0	10	18.52	0.0e+0

Table 4 continued

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
	50	10^3	10	2.10	7.0e+3	4.9e+4	1.05	10	7.91	7.0e+3	5.3e+5	0.16	10	7.95	5.3e+2	10	16.79	1.4e+3
	50	10^4	10	5.08	3.3e+3	3.1e+4	1.10	10	15.39	3.3e+3	2.8e+5	0.27	10	48.29	2.2e+2	10	18.20	1.1e+3
	50	10^5	10	43.49	1.2e+3	2.9e+4	0.81	10	93.16	1.2e+3	1.3e+5	0.23	6	1221.66	1.0e+0	10	131.10	7.6e+1
	50	10^6	10	611.92	1.2e+3	3.1e+4	0.75	10	1150.78	1.2e+3	1.4e+5	0.22	0	–	–	0	–	–
0.05	25	10^3	10	0.01	4.5e+1	7.1e+2	0.32	10	0.02	4.5e+1	2.3e+3	0.07	10	0.01	0.1e+0	10	0.01	0.1e+0
	25	10^4	10	0.06	3.7e+1	8.3e+2	0.34	10	0.08	3.7e+1	2.0e+3	0.03	10	0.07	0.0e+0	10	0.07	0.0e+0
	25	10^5	10	0.75	3.7e+1	9.0e+2	0.32	10	0.94	3.7e+1	2.1e+3	0.03	10	2.40	0.0e+0	10	1.95	0.0e+0
	25	10^6	10	9.29	3.6e+1	9.9e+2	0.39	10	10.74	3.6e+1	2.1e+3	0.04	10	20.28	0.0e+0	10	18.53	0.0e+0
	50	10^3	10	1.73	6.3e+3	4.3e+4	0.88	10	6.67	6.3e+3	4.8e+5	0.13	10	11.63	6.0e+2	10	18.79	1.5e+3
	50	10^4	10	4.65	3.0e+3	2.8e+4	0.88	10	13.55	3.0e+3	2.6e+5	0.16	10	41.90	2.5e+2	10	19.63	1.1e+3
	50	10^5	10	44.23	1.1e+3	2.6e+4	0.60	10	83.15	1.1e+3	1.2e+5	0.15	5	726.75	1.0e+0	10	130.09	9.6e+1
	50	10^6	10	517.16	1.2e+3	2.8e+4	0.57	10	1019.99	1.2e+3	1.3e+5	0.14	0	–	–	0	–	–
0.02	25	10^3	10	0.01	3.6e+1	5.8e+2	0.21	10	0.01	3.6e+1	1.8e+3	0.06	10	0.00	0.2e+0	10	0.01	0.2e+0
	25	10^4	10	0.05	3.1e+1	5.9e+2	0.07	10	0.06	3.1e+1	1.6e+3	0.02	10	0.06	0.0e+0	10	0.07	0.0e+0
	25	10^5	10	0.56	3.1e+1	6.3e+2	0.17	10	0.72	3.1e+1	1.6e+3	0.02	10	2.41	0.0e+0	10	1.97	0.0e+0
	25	10^6	10	6.44	3.1e+1	6.6e+2	0.15	10	8.23	3.1e+1	1.6e+3	0.02	10	20.11	0.0e+0	10	18.85	0.0e+0
	50	10^3	10	1.25	5.3e+3	3.4e+4	0.62	10	5.08	5.4e+3	4.1e+5	0.08	10	12.34	5.9e+2	10	17.58	1.3e+3
	50	10^4	10	3.82	2.6e+3	2.4e+4	0.64	10	10.91	2.6e+3	2.2e+5	0.09	10	63.06	3.1e+2	10	20.52	1.1e+3
	50	10^5	10	38.45	1.1e+3	2.3e+4	0.41	10	74.64	1.1e+3	1.1e+5	0.10	4	1532.87	1.1e+1	10	128.18	1.3e+2
	50	10^6	10	476.44	1.1e+3	2.3e+4	0.40	10	881.79	1.1e+3	1.1e+5	0.10	0	–	–	0	–	–

Table 4 continued

ε	n	m	BB-EllAS					BB-EllAS NW					Gurobi			Gurobi LC		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes	#sol	Time	Nodes
0.01	25	10^3	10	0.01	2.6e+1	4.0e+2	0.13	10	0.01	2.6e+1	1.2e+3	0.02	10	0.01	0.1e+0	10	0.01	0.2e+0
	25	10^4	10	0.03	2.5e+1	4.0e+2	0.10	10	0.04	2.5e+1	1.2e+3	0.02	10	0.06	0.0e+0	10	0.07	0.0e+0
	25	10^5	10	0.43	2.6e+1	4.4e+2	0.18	10	0.52	2.6e+1	1.2e+3	0.06	10	2.33	0.0e+0	10	1.93	0.0e+0
	25	10^6	10	4.90	2.5e+1	4.7e+2	0.17	10	5.71	2.5e+1	1.3e+3	0.03	10	20.12	0.0e+0	10	20.57	0.0e+0
	50	10^3	10	0.89	4.2e+3	2.7e+4	0.46	10	3.96	4.3e+3	3.3e+5	0.06	10	15.27	6.9e+2	10	18.18	1.5e+3
	50	10^4	10	3.32	2.4e+3	2.1e+4	0.53	10	9.39	2.4e+3	2.0e+5	0.08	10	77.79	3.6e+2	10	20.88	1.2e+3
	50	10^5	10	31.52	1.1e+3	2.0e+4	0.32	10	62.53	1.1e+3	1.0e+5	0.08	2	2126.38	4.8e+1	10	131.68	1.4e+2
	50	10^6	10	356.82	1.1e+3	2.0e+4	0.30	10	715.08	1.1e+3	1.0e+5	0.08	0	–	–	0	–	–

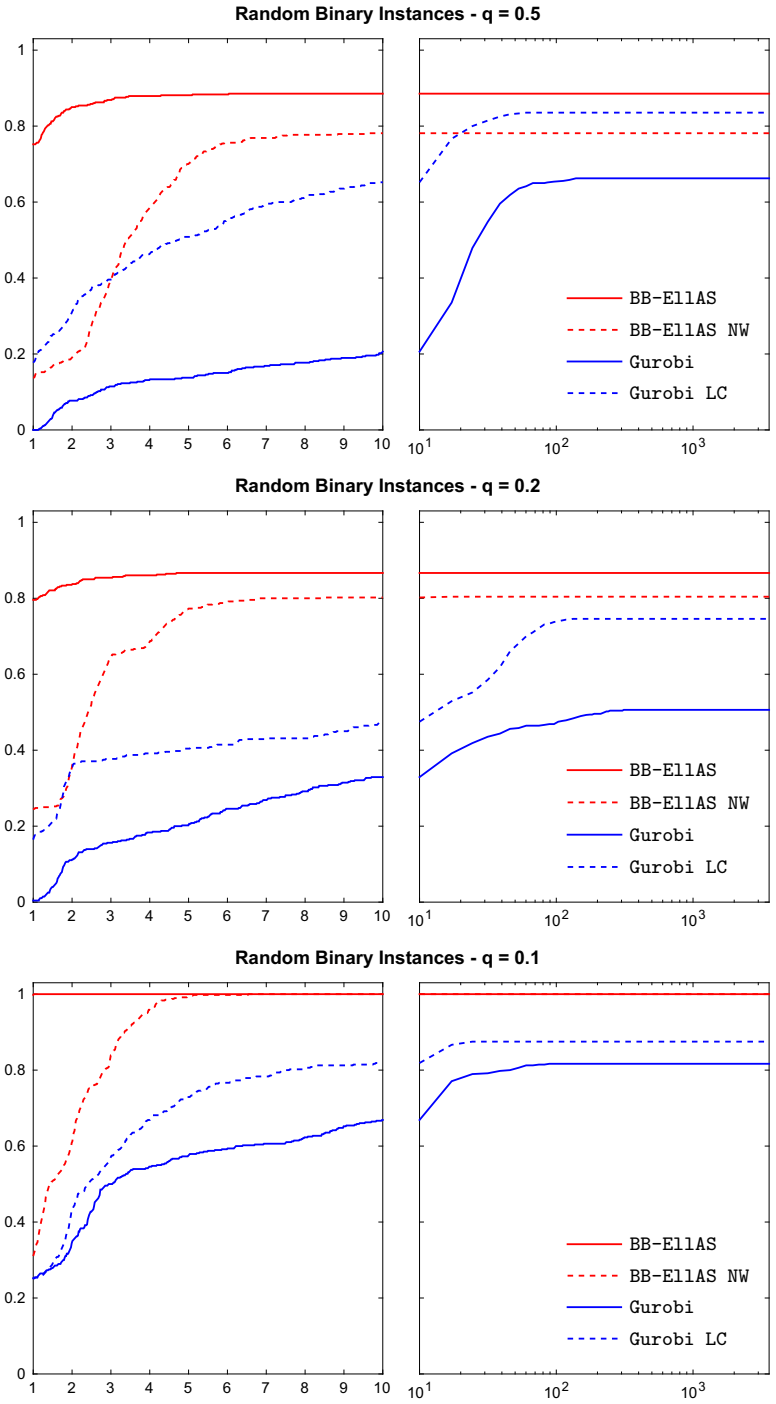

Fig. 2 Performance profiles with respect to running times for random binary instances

Table 5 Comparison on robust shortest path instances

r	n	m	BB-EllAS					Gurobi		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes
10	180	360	10	27.28	2.0e+3	2.1e+4	5.97	10	34.08	1.3e+3
11	220	440	10	197.61	5.2e+3	5.8e+4	9.65	10	149.82	2.5e+3
12	264	528	10	174.58	1.2e+4	1.5e+5	1.71	10	505.00	5.5e+3
13	312	624	8	1103.26	2.1e+4	3.1e+5	3.45	10	1451.29	1.2e+4
14	364	728	4	1307.92	4.6e+4	7.3e+5	0.82	5	2530.77	1.6e+4
15	420	840	0	–	–	–	–	0	–	–

7.3 Shortest path problem

Given a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, and weights associated with each edge, the Shortest Path problem is the problem of finding a path between two vertices s and t such that the sum of the weights of its constituent edges is minimized. Our approach uses the following flow based formulation of the Robust Shortest Path problem:

$$\begin{array}{ll} \min & c^\top x + \sqrt{x^\top Q x} \\ \text{s.t.} & \sum_{e\in\delta^+(i)} x_e - \sum_{e\in\delta^-(i)} x_e = 0 \qquad \forall i \in V\backslash\{s,t\} \\ & \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e = 1 \\ & \sum_{e\in\delta^+(t)} x_e - \sum_{e\in\delta^-(t)} x_e = -1 \\ & x \in \{0,1\}^E \end{array} \tag{6}$$

In our test set, we produced squared grid graphs with r rows and columns, where all edges point from left to right or from top to bottom. In this way, we produced graphs with $|V| = r^2$ vertices and $|E| = 2r^2 - 2r$ edges. In the IP model (6), we thus have $n := 2r^2 - 2r$ many variables, $|V| = r^2$ many equations, and $m := 2|E| = 4r^2 - 4r$ many inequalities, all being box constraints. Since this number is polynomial in n, we can separate them by enumeration within `EllAS`, whereas we can pass the formulation (6) to Gurobi directly. Concerning the objective function of (6), we determined the expected lengths c_i uniformly at random in $[0.5, 1.5]$ and built the positive definite matrix Q as described above. Altogether, we generated 60 different problem instances for (6): for each $r \in \{10, \ldots, 15\}$ we generated 10 instances.

In Table 5, we report the comparison between `BB-EllAS` and the MISOCP solver of Gurobi. The average number of branch-and-bound nodes enumerated by `BB-EllAS` is almost in the same order of magnitude of that needed by Gurobi. However, `EllAS` is able to process the nodes very quickly, leading to a branch-and-bound scheme that is faster than Gurobi in terms of computational time for the majority of the instances, as confirmed by the performance profiles shown in Fig. 3.

In terms of robustness, Gurobi outperforms `BB-EllAS`, being able to solve three instances more within the time limit. In fact, the running times of `BB-EllAS` exhibit a larger variance than those of Gurobi. This is mostly due to numerical issues: in

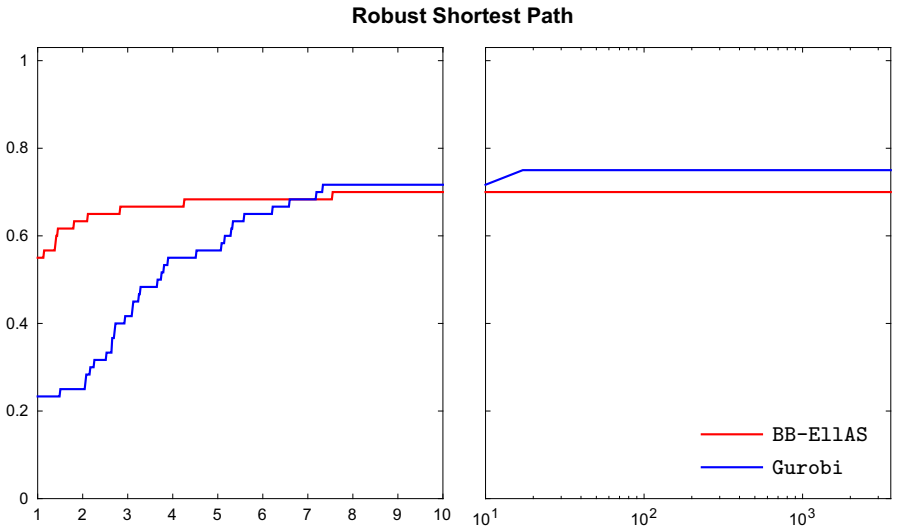

Fig. 3 Performance profile with respect to running times for shortest path instances

instances without numerical difficulties, the incremental update of pseudo-inverses works very well and saves a lot of running time. However, in a few instances, we often have to recompute the pseudo-inverse from scratch to avoid numerical errors. E.g., when considering $r = 10, 11, 12$ in Table 5, it can be observed that average running times for $r = 11$ are larger than for $r = 12$, even if the number of nodes enumerated is smaller. This is due to the larger number of pseudo-inverse recomputations (9.65 % of all iterations for $r = 11$ vs. 1.71 % for $r = 12$). Note that these percentages are significantly smaller for all other types of instances.

7.4 Assigment problem

Given an undirected, bipartite and weighted graph $G = (V, E)$ with bipartition $V = V_1 \cup V_2$, the Assignment problem consists in finding a one-to-one assignment from the nodes in V_1 to the nodes in V_2 such that the sum of the weights of the edges used for the assignment is minimized. In other words, we search for a minimum-weight perfect matching in the bipartite graph G. Our approach uses the following standard formulation of the Assignment problem:

$$\begin{array}{ll} \min & c^\top x + \sqrt{x^\top Q x} \\ \text{s.t.} & \sum_{e \in \delta(i)} x_e = 1 \quad \forall i \in V \\ & x \in \{0, 1\}^E \end{array}$$

In the bipartite case, the above formulation yields a complete description of $\mathrm{conv}(P \cap \mathbb{Z}^n)$, which is not true in the case of general graphs.

We consider complete bipartite graphs, so that the number of variables is $n = \frac{1}{4}|V|^2$. The number of equations is $|V|$ while the number of inequalities is equal to n, since only

Table 6 Comparison on robust assignment instances

$\|V\|$	n	m	BB-EllAS					Gurobi		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes
20	100	100	10	4.10	1.1e+4	8.7e+4	0.10	10	56.13	4.1e+3
22	121	121	10	33.87	4.6e+4	3.8e+5	0.38	10	237.45	1.3e+4
24	144	144	9	156.24	1.9e+5	1.7e+6	0.23	10	742.23	3.0e+4
26	169	169	10	941.32	5.8e+5	5.4e+6	0.50	7	2328.94	6.7e+4
28	196	196	2	2723.25	1.5e+6	1.5e+7	0.28	0	–	–
30	225	225	0	–	–	–	–	0	–	–

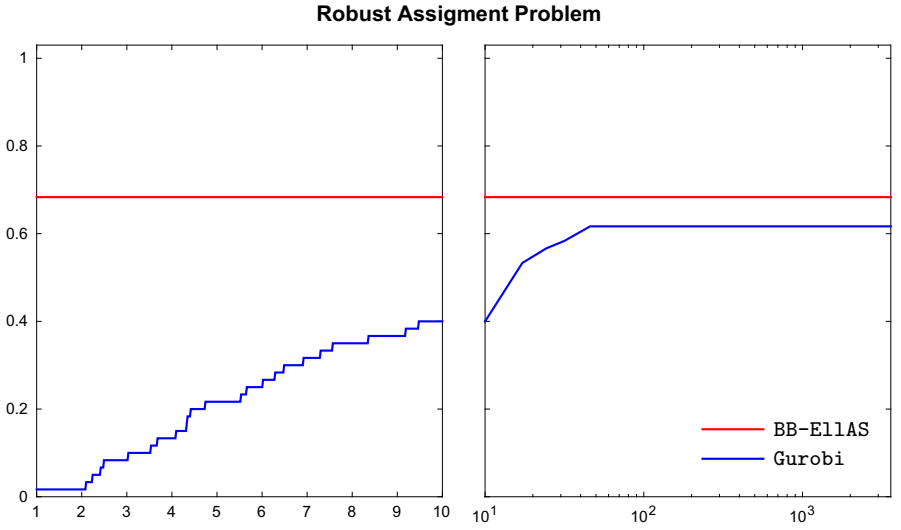

Fig. 4 Performance profile with respect to running times for assignment instances

box constraints of the type $x \geq 0$ need to be added; the remaining box constraints are then implied by the equations. In our instances, we use randomly generated expected weights $c_i \in [0.5, 1.5]$ again, while the non-linear part of the objective function is generated as before. Altogether, we generated 60 different problem instances: for each $|V| \in \{20, 22, \ldots, 30\}$ we generated 10 different instances. Results are presented in Table 6 and Fig. 4. `BB-EllAS` is able to solve four instances more than Gurobi within the time limit and outperforms Gurobi in terms of computational time.

7.5 Spanning tree problem

Given an undirected weighted graph $G = (V, E)$, a minimum spanning tree is a subset of edges that connects all vertices, without any cycles and with the minimum total edge weight. Our approach uses the following formulation of the Robust Spanning Tree problem:

Table 7 Comparison on robust minimum spanning tree instances (complete graphs)

$\|V\|$	n	m	BB-EllAS					Gurobi		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes
12	66	4160	10	53.51	9.5e+4	5.9e+5	2.83	10	110.61	1.1e+4
13	78	8268	10	231.11	1.9e+5	1.3e+6	5.92	10	546.84	2.9e+4
14	91	16,473	10	312.34	5.9e+5	4.5e+6	0.21	7	1802.43	7.2e+4
15	105	32,871	5	2388.39	2.8e+6	2.3e+7	0.64	2	3271.12	1.0e+5
16	120	65,654	1	1490.94	1.4e+6	1.3e+7	0.50	0	–	–

Table 8 Comparison on robust minimum spanning tree instances (grid graphs)

r	n	m	BB-EllAS					Gurobi		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes
6	60	6.9e+10	10	2.96	1.9e+3	2.0e+4	1.37	10	389.07	2.3e+4
7	84	5.6e+14	10	340.57	6.7e+4	8.3e+5	4.74	2	701.55	1.9e+4
8	112	1.8e+19	3	2112.00	2.2e+5	3.1e+6	2.32	0	–	–

$$\begin{array}{ll} \min & c^\top x + \sqrt{x^\top Q x} \\ \text{s.t.} & \sum_{e \in E} x_e = |V| - 1 \\ & \sum_{e \subseteq X} x_e \leq |X| - 1 \forall \emptyset \neq X \subseteq V \\ & x \in \{0,1\}^E \end{array} \tag{7}$$

In the above model, the number of inequalities, taking into account also the non-negativity constraints, is $m = 2^{|V|} - 2 + |E|$. Since this number is exponential in the input size, we also have to use a separation algorithm for Gurobi. For both `BB-EllAS` and Gurobi, we essentially use the same simple implementation based on the Ford–Fulkerson algorithm.

For our experiments, we considered both complete graphs and grid graphs, the latter being produced as for the Shortest Path Problem. In both cases, expected edge weights are randomly generated in [0.5, 1.5] again, while we built the positive definite matrix Q as above. Altogether, we generated 80 different problem instances: for each $|V| \in \{12, \ldots, 16\}$ we generated 10 different complete instances, while for each $r \in \{6, 7, 8\}$ we generated 10 different grid instances. As shown in Tables 7 and 8, `BB-EllAS` clearly outperforms the MISOCP solver of Gurobi on all the instances considered. For the performance profile, see Fig. 5.

7.6 Traveling salesman problem

Given an undirected, complete and weighted graph $G = (V, E)$, the Traveling Salesman problem consists in finding a path starting and ending at a given vertex $v \in V$ such that all the vertices in the graph are visited exactly once and the sum of the weights of its constituent edges is minimized. Our approach uses the following formulation of the Traveling Salesman problem:

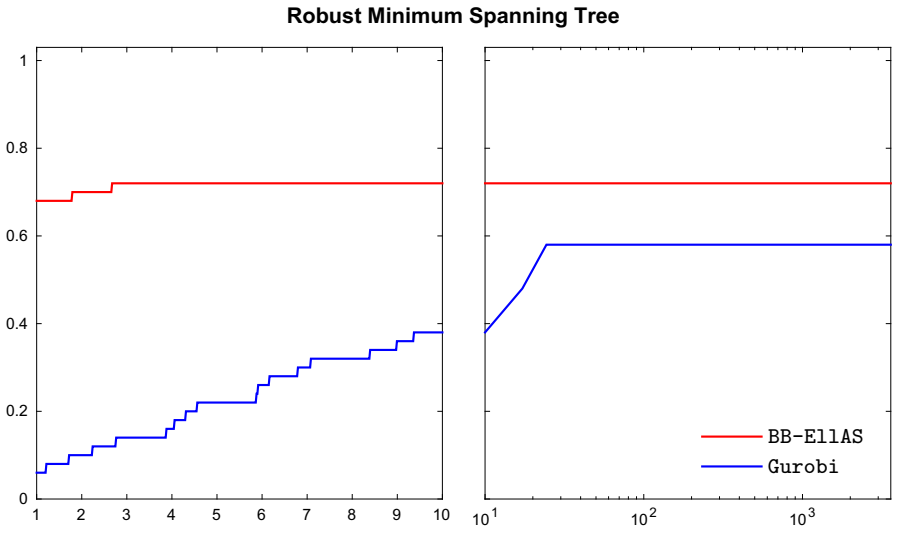

Fig. 5 Performance profile with respect to running times for spanning tree instances

Table 9 Comparison on robust traveling salesman instances

$\|V\|$	n	m	BB-EllAS					Gurobi		
			#sol	Time	Nodes	iter	%ps	#sol	Time	Nodes
14	91	16,564	10	29.34	9.3e+4	6.1e+5	0.08	10	213.86	1.5e+4
15	105	32,976	10	266.85	4.4e+5	3.1e+6	0.71	10	1027.32	5.5e+4
16	120	65,774	10	932.06	1.7e+6	1.3e+7	0.12	4	1897.40	8.0e+4
17	136	131,342	3	2148.13	2.9e+6	2.3e+7	0.16	0	–	–
18	153	262,448	0	–	–	–	–	0	–	–

$$\begin{array}{lll} \min & c^\top x + \sqrt{x^\top Q x} & \\ \text{s.t.} & \sum_{e\in\delta(i)} x_e = 2 & \forall i \in V \\ & \sum_{e\in\delta(X)} x_e \geq 2 & \forall \emptyset \neq X \subsetneq V \\ & x \in \{0,1\}^E & \end{array} \tag{8}$$

Again, we consider complete graphs. The number of inequalities including the bounds $x \in [0,1]^E$ is $m := 2^{|V|} - 2 + 2|E|$ and hence again exponential. For both `BB-EllAS` and Gurobi, we basically use the same separation algorithm as for the Spanning Tree problem; see Sect. 7.5. Instances are identical to those generated for the Spanning Tree problem, but we can consider slightly larger graphs, namely graphs with $|V| \in \{14, \ldots, 18\}$. See Table 9 and Fig. 6 for the results.

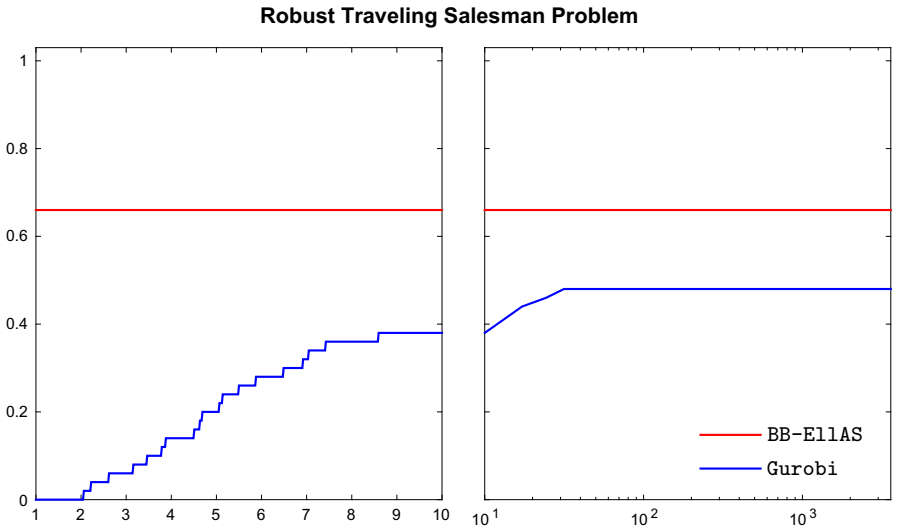

Fig. 6 Performance profile with respect to running times for traveling salesman instances

8 Conclusions

We presented a new branch-and-bound algorithm for robust combinatorial optimization problems under ellipsoidal uncertainty. We assume that the set of feasible solutions is given by a separation algorithm that decides whether a given point belongs to the convex hull of the feasible set or not, and, in the negative case, provides a valid but violated inequality. The branch-and-bound algorithm is based on the use of an active set method for the computation of dual bounds. Dealing with the Lagrangian dual of the continuous relaxation has the advantage of allowing an early pruning of the node. The closed form solution of the active set subproblems, the smart update of pseudo-inverse matrices, as well as the possibility of using warmstarts, leads to an algorithm that clearly outperforms the mixed-integer SOCP solver of Gurobi on most of the problem instances considered.

Acknowledgements The first author acknowledges support within the project "Mixed-Integer Non Linear Optimisation: Algorithms and Applications", which has received funding from the Europeans Union's EU Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie Actions Grant Agreement No 764759. The second author acknowledges support within the project "Nonlinear Approaches for the Solution of Hard Optimization Problems with Integer Variables"(No RP11715C7D8537BA) which has received funding from Sapienza, University of Rome. Moreover, the authors are grateful to Philipp Speckenmeyer for developing and implementing the special treatment of equations in our code (and for the careful reading of the manuscript).

References

1. Atamtürk, A., Gómez, A.: Simplex QP-based methods for minimizing a conic quadratic objective over polyhedra. Math. Prog. Comp. (2018). https://doi.org/10.1007/s12532-018-0152-7

2. Ben-Tal, A., Nemirovski, A.: Robust convex optimization. Math. Oper. Res. **23**(4), 769–805 (1998)
3. Ben-Tal, A., Nemirovski, A.: Robust solutions of uncertain linear programs. Oper. Res. Lett. **25**, 1–13 (1999)
4. Ben-Tal, A., Nemirovski, A.: Lectures on Modern Convex Optimization. SIAM, Philadelphia (2001)
5. Bertsimas, D., Popescu, I.: Optimal inequalities in probability theory: a convex optimization approach. SIAM J. Optim. **15**, 780–804 (2005)
6. Buchheim, C., Kurtz, J.: Robust combinatorial optimization under convex and discrete cost uncertainty. Technical report, Optimization Online (2017)
7. Buchheim, C., De Santis, M., Lucidi, S., Rinaldi, F., Trieu, L.: A feasible active set method with reoptimization for convex quadratic mixed-integer programming. SIAM J. Optim. **26**(3), 1695–1714 (2016)
8. Buchheim, C., De Santis, M., Rinaldi, F., Trieu, L.: A Frank-Wolfe based branch-and-bound algorithm for mean-risk optimization. J. Glob. Optim. **70**(3), 625–644 (2018)
9. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
10. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2016)
11. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Springer, Dordrecht (1996)
12. Meyer Jr., C.D.: Generalized inversion of modified matrices. SIAM J. Appl. Math. **24**(3), 315–323 (1973)
13. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia (2000)
14. Mittelmann, H.D.: Latest benchmark results—informs annual conference. http://plato.asu.edu/talks/informs2018.pdf. Accessed 4–7 November 2018
15. MOSEK ApS: The MOSEK optimization toolbox for MATLAB manual. Version 8.0.0.81 (2017)
16. Nesterov, Y., Nemirovski, A.: Interior-Point Polynomial Algorithms in Convex Programming. SIAM, Philadelphia (1993)
17. Nikolova, E.: Approximation algorithms for offline risk-averse combinatorial optimization. Technical report (2010)
18. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer-Verlag, New York (2006)
19. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11–12**, 625–653 (1999). Version 1.05 available from http://fewcal.kub.nl/sturm