A FEASIBLE ACTIVE SET METHOD WITH REOPTIMIZATION FOR CONVEX QUADRATIC MIXED-INTEGER PROGRAMMING*

CHRISTOPH BUCHHEIM†, MARIANNA DE SANTIS‡, STEFANO LUCIDI§, FRANCESCO RINALDI¶, AND LONG TRIEU‖

Abstract. We propose a feasible active set method for convex quadratic programming problems with non-negativity constraints. This method is specifically designed to be embedded into a branch-and-bound algorithm for convex quadratic mixed integer programming problems. The branch-and-bound algorithm generalizes the approach for unconstrained convex quadratic integer programming proposed by Buchheim, Caprara and Lodi [8] to the presence of linear constraints. The main feature of the latter approach consists in a sophisticated preprocessing phase, leading to a fast enumeration of the branch-and-bound nodes. Moreover, the feasible active set method takes advantage of this preprocessing phase and is well suited for reoptimization. Experimental results for randomly generated instances show that the new approach significantly outperforms the MIQP solver of `CPLEX 12.6` for instances with a small number of constraints.

Key words. integer programming, quadratic programming, global optimization

AMS subject classifications. 90C10, 90C20, 90C57

1. Introduction.

We consider mixed-integer optimization problems with strictly convex quadratic objective functions and linear constraints:

$$\text{(MIQP)} \qquad \begin{array}{ll} \min & f(x) = x^\top Q x + c^\top x + d \\ \text{s.t.} & Ax \le b \\ & x_i \in \mathbb{Z},\ i = 1, \dots, n_1 \\ & x_i \in \mathbb{R},\ i = n_1 + 1, \dots, n \end{array}$$

where $Q \in \mathbb{R}^{n \times n}$ is a positive definite matrix, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $n_1 \in \{0, \dots, n\}$ is the number of integer variables. Up to now, all solution methods for convex mixed integer quadratic programming are based on either cutting planes [2], branch-and-cut [6], Benders decomposition [19, 20] or branch-and-bound [3, 15, 21]. Numerical experiments by Fletcher and Leyffer [15] showed that branch-and-bound is the most effective approach out of all the common methods, since the continuous relaxations are very easy to solve. Standard commercial solvers that can handle (MIQP) include `CPLEX` [18], `Xpress` [14], `Gurobi` [17] and `MOSEK` [22], while `Bonmin` [7] and `SCIP` [1] are well-known non-commercial software packages being capable of solving (MIQP).

Many optimization problems in real world applications can be formulated as convex mixed-integer optimization problems with few linear constraints, e.g., the clas-

*The second author has been partially supported by the German Research Foundation (DFG) under grant BU 2313/4. A preliminary version of this paper has been published in the Proceedings of ISCO 2014 [9].

†Fakultät für Mathematik, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany christoph.buchheim@tu-dortmund.de

‡Fakultät für Mathematik, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany marianna.de.santis@math.tu-dortmund.de

§Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy stefano.lucidi@dis.uniroma1.it

¶Dipartimento di Matematica, Università di Padova, Via Trieste, 63, 35121 Padova, Italy rinaldi@math.unipd.it

‖Fakultät für Mathematik, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany long.trieu@math.tu-dortmund.de

sical mean-variance optimization problem (MVO) for the selection of portfolios of assets [10]. Thus, the design of effective algorithms for this class of problems plays an important role both in theory and in practice.

Our approach is based on a branch-and-bound scheme that enumerates nodes very quickly. Similar to Buchheim et al. [8], we fix the branching order in advance, thus losing the flexibility of choosing sophisticated branching strategies but gaining the advantage of shifting expensive computations into a preprocessing phase. In each node the dual problem of the continuous relaxation is solved in order to determine a local lower bound. Since all constraints of the continuous relaxation of (MIQP) are affine, strong duality holds if the primal problem is feasible. The dual problem of the continuous relaxation is again a convex Quadratic Programming (QP) problem but contains only non-negativity constraints, for its solution we devise a specific feasible active set method. By considering the dual problem, it suffices to find an approximate solution, as each dual feasible solution yields a valid lower bound. We can thus prune the branch-and-bound node as soon as the current upper bound is exceeded by the value of any feasible iterate produced in a solution algorithm for the dual problem.

Another feature of our algorithm is the use of warmstarts: after each branching, corresponding to the fixing of a variable, we pass the optimal solution from the parent node as starting point in the children nodes. This leads to a significant reduction in the average number of iterations needed to solve the dual problem to optimality.

If the primal problem has a small number of constraints, the dual problem has a small dimension. Using sophisticated incremental computations, the overall running time per node in our approach is linear in the dimension n if the number of constraints is fixed and when we assume that the solution of the dual problem, which is of fixed dimension in this case, can be done in constant time. In this sense, our approach can be seen as an extension of the algorithm devised in [8].

The paper is organized as follows. In Section 2 we present the active set method for convex QP problems with non-negativity constraints. The properties of the proposed active set estimation are discussed and the convergence of the algorithm is analyzed. Section 3 presents an outline of the branch-and-bound algorithm: we discuss the advantages of considering the corresponding dual problem instead of the primal one. Afterwards, we explain the idea of reoptimization, using warmstarts within the branch-and-bound scheme. The end of the section deals with some tricks to speed up the algorithm by using incremental computations and an intelligent preprocessing. In Section 4 we present computational results and compare the performance of the proposed algorithm, applied to randomly generated instances, to the MIQP solver of `CPLEX 12.6`. Section 5 concludes.

2. A Feasible Active Set Algorithm for Quadratic Programming Problems with Non-negativity Constraints. The vast majority of solution methods for (purely continuous) quadratic programs can be categorized into either interior point methods or active set methods (see [23] and references therein for further details). In interior point methods, a sequence of parameterized barrier functions is (approximately) minimized using Newton's method. The main computational effort consists in solving the Newton system to get the search direction. In active set methods, at each iteration, a working set that estimates the set of active constraints at the solution is iteratively updated. Usually, only a single active constraint is added to or deleted from the active set at each iteration. However, when dealing with simple constraints, one can use projected active set methods, which can add to or delete from the current estimated active set more than one constraint at each iteration, and

eventually find the active set in a finite number of steps if certain conditions hold.

An advantage of active set methods is that they are well-suited for warmstarts, where a good estimate of the optimal active set is used to initialize the algorithm. This is particularly useful in applications where a sequence of QP problems is solved, e.g., in a sequential quadratic programming method. Since in our branch-and-bound framework we need to solve a large number of closely related quadratic programs, using active set strategies seems to be a reasonable choice.

In this section, we thus consider QP problems of the form

$$\text{(QP)} \qquad \begin{array}{ll} \min & q(x) = x^\top Q x + c^\top x + d \\ \text{s.t.} & x \geq 0 \\ & x \in \mathbb{R}^m, \end{array}$$

where $Q \in \mathbb{R}^{m \times m}$ is positive semidefinite, $c \in \mathbb{R}^m$ and $d \in \mathbb{R}$. We describe a projected active set method for the solution of such problems that tries to exploit the information calculated in a preprocessing phase at each level of the branch-and-bound tree. Our method is inspired by the work of Bertsekas [4], where a class of active set projected Newton methods is proposed for the solution of problems with non-negativity constraints. The main difference between the two approaches is in the way the active variables are defined and updated. On the one side, the method described in [4] uses a linesearch procedure that both updates active and non-active variables at each iteration. On the other side, our method, at a given iteration, first sets to zero the active variables (guaranteeing a sufficient reduction of the objective function), and then tries to improve the objective function in the space of the non-active variables. This gives us more freedom in the choice of the stepsize along the search direction, since the active variables are not considered in the linesearch procedure.

In the following we denote by $g(x) \in \mathbb{R}^m$ and $H(x) \in \mathbb{R}^{m \times m}$ the gradient vector and the Hessian matrix of the objective function $q(x)$ in Problem (QP), respectively. Explicitly, we have

$$g(x) = 2Qx + c, \quad H(x) = 2Q \; .$$

Given a matrix M, we further denote by $\lambda_{max}(M)$ the maximum eigenvalue of M. Given a vector $v \in \mathbb{R}^m$ and an index set $I \subseteq \{1, \ldots, m\}$, we denote by v_I the sub-vector with components v_i with $i \in I$. Analogously, given the matrix $H \in \mathbb{R}^{m \times m}$ we denote by $H_{I\,I}$ the sub-matrix with components $h_{i,j}$ with $i, j \in I$. Given two vectors $v, y \in \mathbb{R}^m$ we denote by $\max\{v, y\}$ the vector with components $\max\{v_i, y_i\}$, for $i \in \{1, \ldots, m\}$. The open ball with center x and radius $\rho > 0$ is denoted by $\mathcal{B}(x, \rho)$. Finally, we denote the projection of a point $z \in \mathbb{R}^m$ onto $\mathbb{R}^m_+$ by $[z]^\sharp := \max\{0, z\}$.

2.1. Active Set Estimate. The idea behind active set methods in constrained nonlinear programming problems is that of correctly identifying the set of active constraints at the optimal solution $x^\star$.

DEFINITION 2.1. *Let $x^\star \in \mathbb{R}^m$ be an optimal solution for Problem (QP). We define as* active set *at $x^\star$ the following:*

$$\bar{\mathcal{A}}(x^\star) = \big\{ i \in \{1, \ldots, m\} : x^\star_i = 0 \big\}.$$

We further define as non-active set *at $x^\star$ the complementary set of $\bar{\mathcal{A}}(x^\star)$:*

$$\bar{\mathcal{N}}(x^\star) = \{1, \ldots, m\} \setminus \bar{\mathcal{A}}(x^\star) = \big\{ i \in \{1, \ldots, m\} : x^\star_i > 0 \big\}.$$

Our aim is to find a rule that leads us to the identification of the optimal active set for Problem (QP) as quickly as possible. The rule we propose is based on the use of multiplier functions and follows the ideas reported in [13].

DEFINITION 2.2. *Let $x \in \mathbb{R}^m$. We define the following sets as estimates of the non-active and active sets at x:*

$$\mathcal{N}(x) = \{i \in \{1, \dots, m\} : x_i > \varepsilon\, g_i(x)\}$$

and

$$\mathcal{A}(x) = \{1, \dots, m\} \setminus \mathcal{N}(x),$$

where $\varepsilon > 0$ is a positive scalar.

The following result can be proved [13]:

THEOREM 2.3. *Let $x^\star \in \mathbb{R}^m$ be a solution of Problem* (QP)*. Then, there exists a neighborhood of $x^\star$ such that, for each x in this neighborhood, we have*

$$\bar{\mathcal{A}}^+(x^\star) \subseteq \mathcal{A}(x) \subseteq \bar{\mathcal{A}}(x^\star),$$

with $\bar{\mathcal{A}}^+(x^\star) = \bar{\mathcal{A}}(x^\star) \cap \{i \in \{1, \dots, m\} : g_i(x^\star) > 0\}$.

Furthermore, if strict complementarity holds, we can state the following:

COROLLARY 2.4. *Let $x^\star \in \mathbb{R}^m$ be a solution of Problem* (QP) *where strict complementarity holds. Then, there exists a neighborhood of $x^\star$ such that, for each x in this neighborhood, we have*

$$\mathcal{A}(x) = \bar{\mathcal{A}}(x^\star).$$

2.2. Outline of the Algorithm. We now give an outline of our Feasible Active SeT Quadratic Programming Algorithm (`FAST-QPA`) for solving Problem (QP); see Algorithm 1.

At each iteration k, the algorithm first determines the two sets $\mathcal{N}^k := \mathcal{N}(x^k)$ and $\mathcal{A}^k := \mathcal{A}(x^k)$ according to Definition 2.2. Then, the variables belonging to $\mathcal{A}^k$ are set to zero, thus obtaining a new point $\tilde{x}^k$, and a search direction d^k is calculated. More specifically $d^k_{\mathcal{A}^k}$, the components of d^k related to $\mathcal{A}^k$, are set to zero, while a gradient related direction $d^k_{\mathcal{N}^k}$ is calculated in $\mathcal{N}^k$. Here we define that a direction $d_{\mathcal{N}^k}$ is gradient related at $\tilde{x}^k$ (see e.g. [5]) if there exist $\sigma_1, \sigma_2 > 0$ such that

$$d^\top_{\mathcal{N}^k} g(\tilde{x}^k)_{\mathcal{N}^k} \le -\sigma_1 \|g(\tilde{x}^k)_{\mathcal{N}^k}\|^2, \tag{2.1}$$

$$\|d_{\mathcal{N}^k}\| \le \sigma_2 \|g(\tilde{x}^k)_{\mathcal{N}^k}\|. \tag{2.2}$$

In order to obtain the direction $d^k_{\mathcal{N}^k}$, we solve the following unconstrained quadratic problem in the subspace of non-active variables,

$$(\mathrm{QP}_k) \qquad \begin{array}{ll} \min & q^k(y) = y^\top \widetilde{Q} y + \tilde{c}^\top y \\ s.t. & y \in \mathbb{R}^{|\mathcal{N}^k|}, \end{array}$$

where $\widetilde{Q} = Q_{\mathcal{N}^k \mathcal{N}^k}$ and $\tilde{c} = g(\tilde{x}^k)_{\mathcal{N}^k}$. In particular, we apply a conjugate gradient type algorithm (see Algorithm 2). Algorithm 2 is basically a modification of the conjugate gradient method for quadratic problems that embeds a truncated Newton

Algorithm 1 Feasible Active SeT Quadratic Programming Algorithm (FAST-QPA)

0 **Fix** $\delta \in (0,1)$, $\gamma \in (0, \frac{1}{2})$ and ε satisfying (2.3)
1 **Choose** $x^0 \in \mathbb{R}^n$,
2 **For** $k = 0, 1, \ldots$
3 **If** x^k is optimal **then** STOP
4 **Compute** $\mathcal{A}^k := \mathcal{A}(x^k)$ and $\mathcal{N}^k := \mathcal{N}(x^k)$
5 **Set** $\tilde{x}^k_{\mathcal{A}^k} = 0$ and $\tilde{x}^k_{\mathcal{N}^k} = x^k_{\mathcal{N}^k}$
6 **Set** $d^k_{\mathcal{A}^k} = 0$
7 **Compute** a gradient related direction $d^k_{\mathcal{N}^k}$ in $\tilde{x}^k$ **and** the point $\bar{x}^k$
8 **If** $\bar{x}^k$ is optimal **then** STOP
9 **Set** $\alpha^k = \delta^j$ where j is the first non-negative integer for which

$$q([\tilde{x}^k + \delta^j d^k]^\sharp) \leq q(\tilde{x}^k) + \gamma\, \delta^j\, g(\tilde{x}^k)^\top d^k$$

10 **Set**

$$x^{k+1} = [\tilde{x}^k + \alpha^k d^k]^\sharp$$

11 **End For**

like test for the calculation of a gradient related direction (Step 3 of Algorithm 2).

We report two theoretical results, proved in [24], that help us to understand the properties of Algorithm 2:

PROPOSITION 2.5. *Let Problem* (QP_k) *admit an optimal solution and matrix* $\widetilde{Q}$ *be positive semidefinite. Then, the conjugate gradient method terminates with an optimal solution of Problem* (QP_k) *in at most* $|\mathcal{N}^k|$ *iterations.*

PROPOSITION 2.6. *Let matrix* $\widetilde{Q}$ *in Problem* (QP_k) *be positive semidefinite. Then at least one direction* p^l, $1 \leq l \leq |\mathcal{N}^k| - 1$, *computed by the conjugate gradient method satisfies the condition*

$$p^{l\top} \widetilde{Q} p^l = 0.$$

By Propositions 2.5 and 2.6, we are able to prove the following:

PROPOSITION 2.7. *Let matrix* $\widetilde{Q}$ *in Problem* (QP_k) *be positive semidefinite. Algorithm 2 terminates after a finite number of iterations, returning a gradient related direction* $d^k_{\mathcal{N}^k}$. *Furthermore, if Problem* (QP_k) *admits an optimal solution, then* y^l *is an optimal solution of Problem* (QP_k).

Proof. By the results reported in [11, 16] we have that direction $d^k_{\mathcal{N}^k}$ satisfies (2.1) and (2.2) and hence is gradient related at $\tilde{x}^k$. Furthermore, by taking into account Propositions 2.5 and 2.6, the rest of the result follows. □

According to Proposition 2.7, when the minimum of Problem (QP_k) is finite, Algorithm 2 produces a gradient related direction $d^k_{\mathcal{N}^k}$ and an optimal solution y^l of Problem (QP_k) (in at most $|\mathcal{N}^k| - 1$ iterations). The point y^l is then used for building up the candidate optimal point $\bar{x}^k$. In case Problem (QP_k) is unbounded from below, the algorithm still stops after a finite number of iterations giving a gradient related direction $d^k_{\mathcal{N}^k}$, but the point used for generating $\bar{x}^k$ is just a combination of conjugate gradient directions generated along the iterations of the method. As we will see in Section 2.3, calculating the point $\bar{x}^k$ is needed to guarantee, under some specific assumptions, finite convergence of Algorithm 1 (see Proposition 2.15).

Algorithm 2 Calculation of the direction $d^k_{\mathcal{N}^k}$ and of the point $\bar{x}^k$

1 **Set** $y^0 = 0$, $\tilde{g}^0 = 2\widetilde{Q}\,\tilde{x}^k_{\mathcal{N}^k} + \tilde{c}$, $p^0 = -\tilde{g}^0$, $\eta > 0$, $l = 0$ and $check$ = true;
2 **While** ${p^l}^\top \widetilde{Q} p^l > 0$
3 **If** ${p^l}^\top \widetilde{Q} p^l \leq \eta \|p^l\|^2$ **and** $check$ = true **then**

$$\hat{y} = \begin{cases} -\tilde{g}^0, & \text{if } l = 0 \\ y^l, & \text{if } l > 0 \end{cases}, \quad check = \text{false};$$

4 **End If**
5 **Compute** stepsize along the search direction p^l;

$$\alpha^l = \frac{(\tilde{g}^l)^T p^l}{2{p^l}^\top \widetilde{Q} p^l}$$

6 **Update** point $y^{l+1} = y^l + \alpha^l\, p^l$;
7 **Update** gradient of the quadratic function $\tilde{g}^{l+1} = \tilde{g}^l + 2\alpha^l\, \widetilde{Q}\, p^l$;
8 **Compute** coefficient

$$\beta^{l+1} = \frac{\|\tilde{g}^{l+1}\|^2}{\|\tilde{g}^l\|^2};$$

9 **Determine** new conjugate direction $p^{l+1} = -\tilde{g}^{l+1} + \beta^{l+1} p^l$;
10 **Set** $l = l + 1$;
11 **End While**
12 **If** $check$ = true **then**

$$\hat{y} = \begin{cases} -\tilde{g}^0, & \text{if } l = 0 \\ y^l, & \text{if } l > 0 \end{cases}$$

13 **End If**
14 **Return** $d^k_{\mathcal{N}^k} = \hat{y}$ and $\bar{x}^k$, where

$$\bar{x}^k_{\mathcal{A}^k} = 0, \quad \bar{x}^k_{\mathcal{N}^k} = \begin{cases} -\tilde{g}^0, & \text{if } l = 0 \\ y^l, & \text{if } l > 0 \end{cases}.$$

Note that, even if the matrix $\widetilde{Q}$ is ill-conditioned, Algorithm 2 still generates a gradient related direction. In practice, when dealing with ill-conditioned problems, suitable preconditioning techniques can be applied to speed up Algorithm 2.

Once the direction d^k and the point $\bar{x}^k$ are computed, Algorithm 1 checks optimality of point $\bar{x}^k$. If $\bar{x}^k$ is not optimal, the algorithm generates a new point x^{k+1} by means of a projected Armijo linesearch along d^k.

We finally notice that at each iteration of Algorithm 1, two different optimality checks are performed: the first one, at Step 3, to test optimality of the current iterate x^k; the second one, at Step 8, to test optimality of the candidate solution $\bar{x}^k$.

2.3. Convergence Analysis. The convergence analysis of `FAST-QPA` is based on two key results, namely Proposition 2.8 and Proposition 2.9 stated below. These

results show that the algorithm obtains a significant reduction of the objective function both when fixing to zero the variables in the active set estimate and when we perform the projected Armijo linesearch.

Proposition 2.8 completes the properties of the active set identification strategy defined in Section 2.1. More specifically, it shows that, for a suitably chosen value of the parameter ε appearing in Definition 2.2, a decrease of the objective function is achieved by simply fixing to zero those variables whose indices belong to the estimated active set.

ASSUMPTION 1. *The parameter ε appearing in Definition 2.2 satisfies*

$$0 < \varepsilon < \frac{1}{2\lambda_{max}(Q)} \, . \tag{2.3}$$

PROPOSITION 2.8. *Let Assumption 1 hold. Given the point z and the sets $\mathcal{A}(z)$ and $\mathcal{N}(z)$, let y be the point such that*

$$y_{\mathcal{A}(z)} = 0, \quad y_{\mathcal{N}(z)} = z_{\mathcal{N}(z)}.$$

Then,

$$q(y) - q(z) \leq -\frac{1}{2\varepsilon}\|y - z\|^2.$$

Proof. Define $\mathcal{A} = \mathcal{A}(z)$ and $\mathcal{N} = \mathcal{N}(z)$. By taking into account the definition of these two sets and the points y and z, we have

$$q(y) = q(z) + g_{\mathcal{A}}(z)^\top (y - z)_{\mathcal{A}} + \frac{1}{2}(y - z)_{\mathcal{A}}^\top H_{\mathcal{A}\mathcal{A}}(y - z)_{\mathcal{A}} \, .$$

Since $H = 2Q$, the following inequality holds

$$q(y) \leq q(z) + g_{\mathcal{A}}(z)^\top (y - z)_{\mathcal{A}} + \lambda_{max}(Q) \, \|(y - z)_{\mathcal{A}}\|^2.$$

Using (2.3) we obtain

$$q(y) \leq q(z) + g_{\mathcal{A}}(z)^\top (y - z)_{\mathcal{A}} + \frac{1}{2\varepsilon}\|(y - z)_{\mathcal{A}}\|^2$$

and hence

$$q(y) \leq q(z) + \left(g_{\mathcal{A}}(z) + \frac{1}{\varepsilon}(y - z)_{\mathcal{A}} \right)^\top (y - z)_{\mathcal{A}} - \frac{1}{2\varepsilon}\|(y - z)_{\mathcal{A}}\|^2.$$

It thus remains to show

$$\left(g_{\mathcal{A}}(z) + \frac{1}{\varepsilon}(y - z)_{\mathcal{A}} \right)^\top (y - z)_{\mathcal{A}} \leq 0 \, ,$$

which follows from the fact that, for all $i \in \mathcal{A}$,

$$\left(g_i(z) + \frac{1}{\varepsilon}(y_i - z_i) \right) (y_i - z_i) \leq 0.$$

Indeed, for all $i \in \mathcal{A}$ we have $z_i \geq 0$ and $y_i = 0$, hence $z_i - y_i = z_i \leq \varepsilon\, g_i(z)$, so that

$$g_i(z) + \frac{1}{\varepsilon}\,(y_i - z_i) \geq 0\ .$$

□

The following result shows that the projected Armijo linesearch performed at Step 9 of Algorithm 1 terminates in a finite number of steps, and that the new point obtained guarantees a decrease of the objective function of (QP). Its proof is similar to the proof of Proposition 2 in [4].

PROPOSITION 2.9. *Let $\gamma \in (0, \frac{1}{2})$. Then, for every $\bar{x} \in \mathbb{R}^n_+$ which is not optimal for Problem* (QP)*, there exist $\rho > 0$ and $\bar{\alpha} > 0$ such that*

$$q([\tilde{x} + \alpha d]^\sharp) - q(\tilde{x}) \leq \gamma\, \alpha\, d^\top_{\mathcal{N}(x)} g(\tilde{x})_{\mathcal{N}(x)} \tag{2.4}$$

for all $x, \tilde{x} \in \mathbb{R}^n_+$ with $x, \tilde{x} \in \mathcal{B}(\bar{x}, \rho)$ and for all $\alpha \in (0, \bar{\alpha}]$, where $d \in \mathbb{R}^n$ is the direction used at $\tilde{x}$, and such that

(i) $d_{\mathcal{A}(x)} = 0$,

(ii) $d_{\mathcal{N}(x)}$ satisfies (2.1) *and* (2.2).

Using Proposition 2.8 and Proposition 2.9, we can show that the sequence $\{q(x^k)\}$ is monotonically decreasing.

PROPOSITION 2.10. *Let Assumption 1 hold. Let $\{x^k\}$ be the sequence produced by* `FAST-QPA`*. Then, the sequence $\{q(x^k)\}$ is such that*

$$q(x^{k+1}) \leq q(x^k) - \frac{1}{2\varepsilon}\|\tilde{x}^k - x^k\|^2 - \sigma_1 \|g(\tilde{x}^k)_{\mathcal{N}^k}\|^2.$$

Proof. Let $\tilde{x}^k$ be the point produced at Step 5 of Algorithm 1. By setting $y = \tilde{x}^k$ and $z = x^k$ in Proposition 2.8, we have

$$q(\tilde{x}^k) \leq q(x^k) - \frac{1}{2\varepsilon}\|\tilde{x}^k - x^k\|^2\ .$$

Furthermore, by the fact that we use an Armijo linesearch at Step 9 of Algorithm 1 and by Proposition 2.9, we have that the chosen point x^{k+1} satisfies inequality (2.4), that is

$$q(x^{k+1}) - q(\tilde{x}^k) \leq \gamma\, \alpha^k\, d^\top_{\mathcal{N}^k} g(\tilde{x}^k)_{\mathcal{N}^k}\ .$$

By taking into account (2.1), we thus have

$$q(x^{k+1}) - q(\tilde{x}^k) \leq \gamma\, \alpha^k\, d^\top_{\mathcal{N}^k} g(\tilde{x}^k)_{\mathcal{N}^k} \leq -\sigma_1 \|g(\tilde{x}^k)_{\mathcal{N}^k}\|^2\ .$$

In summary, we obtain

$$\begin{aligned} q(x^{k+1}) &\leq q(\tilde{x}^k) - \sigma_1 \|g(\tilde{x}^k)_{\mathcal{N}^k}\|^2 \leq \\ &\leq q(x^k) - \frac{1}{2\varepsilon}\|\tilde{x}^k - x^k\|^2 - \sigma_1 \|g(\tilde{x}^k)_{\mathcal{N}^k}\|^2\ . \end{aligned}$$

□

Proposition 2.10 will allow us to prove two important results: if the minimum of Problem (QP) is finite, then the iterates generated by Algorithm 1 will meet a stopping condition of the type

$$\|\max\{0, -g(x^k)\}\| \leq \epsilon_{tol},$$

with a fixed tolerance $\epsilon_{tol} > 0$ in a finite number of iterations; see Proposition 2.11. Otherwise, the sequence $\{q(x^k)\}$ is unbounded from below; see Proposition 2.12.

PROPOSITION 2.11. *Let Assumption 1 hold. Let $\{x^k\}$ be the sequence produced by* `FAST-QPA`*. If the minimum of Problem* (QP) *is finite, then*

$$\lim_{k\to\infty} \| \max\{0, -g(x^k)\}\| = 0.$$

Proof. By Proposition 2.10, the sequence $\{q(x^k)\}$ is monotonically decreasing. Since it is bounded by the minimum of (QP), we have that $\{q(x^k)\}$ converges. In particular, this implies that both $\|\tilde{x}^k - x^k\|^2$ and $\|g(\tilde{x}^k)_{\mathcal{N}^k}\|^2$ go to zero when $k \to \infty$. By noting that

$$0 \le \|g(\tilde{x}^k) - g(x^k)\| = \|Q(\tilde{x}^k - x^k)\| \le \|Q\| \cdot \|\tilde{x}^k - x^k\| \to 0$$

we obtain $\|g(x^k)_{\mathcal{N}^k}\|^2 \to 0$ as well, and thus $\| \max\{0, -g(x^k)\}_{\mathcal{N}^k}\|^2 \to 0$. On the other hand, if $i \in \mathcal{A}^k$, we have $g_i(x^k) \ge 0$, so that $\| \max\{0, -g(x^k)\}_{\mathcal{A}^k}\|^2 = 0$. This shows the result. □

PROPOSITION 2.12. *Let Assumption 1 hold. Let $\{x^k\}$ be the sequence produced by* `FAST-QPA`*. If Problem* (QP) *is unbounded, then*

$$\lim_{k\to\infty} q(x^k) = -\infty.$$

Proof. By Proposition 2.10, the sequence $\{q(x^k)\}$ is monotically decreasing. Assume by contradiction that it is bounded and hence $\lim_{k\to\infty} q(x^k) = \bar{q}$ for some $\bar{q} \in \mathbb{R}$. Then, as in the proof of Proposition 2.11, $\|g(x^k)_{\mathcal{N}^k}\|^2 \to 0$. Since Problem (QP) is unbounded, there must exist some $d \ge 0$ such that $Qd = 0$ and $c^\top d < 0$. In particular, we obtain $g(x)^\top d = c^\top d = \eta < 0$ for all $x \in \mathbb{R}^n$.

Therefore, since $g_i(x^k) \ge 0$ for all $i \in \mathcal{A}^k$, we have

$$\begin{aligned} 0 > \eta = g(x^k)^\top d &= \sum_{i\in\mathcal{A}^k} g_i(x^k)d_i + \sum_{i\in\mathcal{N}^k} g_i(x^k)d_i \\ &\ge \sum_{i\in\mathcal{N}^k} g_i(x^k)d_i = -\left|g(x^k)_{\mathcal{N}^k}^\top d_{\mathcal{N}^k}\right| \\ &\ge -\|g(x^k)_{\mathcal{N}^k}\| \cdot \|d_{\mathcal{N}^k}\| \ge -\|g(x^k)_{\mathcal{N}^k}\| \cdot \|d\|\,, \end{aligned}$$

and we get a contradiction since $\|g(x^k)_{\mathcal{N}^k}\|^2$ goes to zero. □

Finally, we are able to prove the main result concerning the global convergence of `FAST-QPA`.

THEOREM 2.13. *Assume that the minimum of Problem* (QP) *is finite and that Assumption 1 hold. Let $\{x^k\}$ be the sequence produced by Algorithm* `FAST-QPA`*. Then either an integer $\bar{k} \ge 0$ exists such that $x^{\bar{k}}$ or $\bar{x}^{\bar{k}}$ is an optimal solution for Problem* (QP)*, or the sequence $\{x^k\}$ is infinite and every limit point $x^\star$ of the sequence is an optimal point for Problem* (QP)*.*

Proof. Let $x^\star$ be any limit point of the sequence $\{x^k\}$ and let $\{x^k\}_K$ be the subsequence with

$$\lim_{k\to\infty,\, k\in K} x^k = x^\star.$$

Because of the continuity of $g(x)$, the function $\max\{0, -g(x)\}$ is continuous in x, and therefore Proposition 2.11 gives $\max\{0, -g(x^\star)\} = 0$. But this means that $x^\star$ satisfies

the standard first-order optimality condition for (QP). Since (QP) is convex, $x^\star$ must be an optimal point. □

COROLLARY 2.14. *If* (2.3) *holds, then the sequence* $q(x^k)$ *converges to the optimal value of* (QP).

As a final result, we prove that, under a specific assumption, `FAST-QPA` finds an optimal solution in a finite number of iterations.

THEOREM 2.15. *Let Assumption 1 hold. Assume that there exists an accumulation point* $x^\star$ *of the sequence* $\{x^k\}$ *generated by* ***FAST-QPA*** *such that*

$$Q_{\hat{\mathcal{N}}\,\hat{\mathcal{N}}} \succ 0\ ,$$

where $\hat{\mathcal{N}} = \bar{\mathcal{N}}(x^\star) \cup \{i \in \{1,\dots,n\} : x_i^\star = 0,\ g_i(x^\star) = 0\}$. *Then* ***FAST-QPA*** *produces a minimizer of Problem* (QP) *in a finite number of steps.*

Proof. Let $\{x^k\}$ be the sequence generated by `FAST-QPA` and let $\{x^k\}_K$ be the subsequence with

$$\lim_{k\to\infty,\, k\in K} x^k = x^\star.$$

By Theorem 2.13, the limit $x^\star$ of $\{x^k\}_K$ is a minimizer of Problem (QP). Furthermore, by Theorem 2.3, we know that the active set estimation yields $\bar{\mathcal{A}}^+(x^\star) \subseteq \mathcal{A}^k \subseteq \bar{\mathcal{A}}(x^\star)$ for sufficiently large k. Consequently

$$\bar{\mathcal{N}}(x^\star) \subseteq \mathcal{N}^k\ . \tag{2.5}$$

Moreover, taking into account the definition of $\hat{\mathcal{N}}$, we get $\mathcal{N}^k \subseteq \hat{\mathcal{N}}$ and thus

$$Q_{\mathcal{N}^k\,\mathcal{N}^k} \succ 0. \tag{2.6}$$

Hence by (2.5), (2.6), and the fact that $g_i(x^\star) = 0$ for all $i \in \mathcal{N}^k$ we have that solving (QP) is equivalent to solving the following problem

$$\begin{array}{ll} \min & x^\top Q x + c^\top x + d \\ \text{s.t.} & x_{\mathcal{A}^k} = 0, \\ & x \in \mathbb{R}^m. \end{array} \tag{2.7}$$

Since $\bar{x}^k_{\mathcal{N}^k}$ is the only optimal solution for problem (QP_k) and $\bar{x}^k_{\mathcal{A}^k} = 0$, we conclude that $\bar{x}^k$ is the optimal solution of (2.7) and of (QP). □

By Theorem 2.15, we just need that the submatrix $Q_{\hat{\mathcal{N}}\,\hat{\mathcal{N}}}$ is positive definite in order to ensure finite termination of `FAST-QPA`.

Furthermore, by taking into account the scheme of our algorithm, and by carefully analyzing the proof given above, we notice two important facts:

- the algorithm, at iteration k, moves towards the optimal point $x^\star$ by only using an approximate solution of the unconstrained problem (QP_k), i.e. the solution needed in the calculation of the gradient related direction;
- once the point x^k gets close enough to $x^\star$, thanks to the properties of our estimate, we can guarantee that the point $\bar{x}^k$ is the optimum of the original problem (QP).

This explains why, in the algorithmic scheme, we calculate $\bar{x}^k$ and just use it in the optimality check at Step 8.

3. Embedding `FAST-QPA` into a Branch-and-Bound Scheme. As shown in [9], the approach of embedding taylored active set methods into a branch-and-bound scheme is very promising for solving convex quadratic integer programming problems of type (MIQP). In this section we shortly summarize the key ingredients of our branch-and-bound algorithm `FAST-QPA-BB` that makes use of `FAST-QPA`, presented in Section 2, for the computation of lower bounds. The branch-and-bound algorithm we consider is based on the work in [8], where the unconstrained case is addressed. As our branch-and-bound scheme aims at a fast enumeration of the nodes, we focus on bounds that can be computed quickly. A straightforward choice for determining lower bounds is to solve the continuous relaxation of (MIQP). Instead of considering the primal formulation of the continuous relaxation of (MIQP), we deal with the dual one, which again is a convex QP, but with only non-negativity constraints, so that it can be solved by `FAST-QPA`. The solution can be used as a lower bound for f over all feasible integer points and is as strong as the lower bound obtained by solving the primal problem, since strong duality holds.

Our branching strategy and its advantages are discussed in Section 3.1. In Section 3.2, we have a closer look at the relation between the primal and the dual problem, while in Section 3.3 we shortly discuss the advantage of reoptimization. Using a predetermined branching order, some of the expensive calculations can be moved into a preprocessing phase, as described in Section 3.4.

3.1. Branching. At every node in our branch-and-bound scheme, we branch by fixing a single *primal* variable in increasing distance to its value in the solution of the continuous relaxation $x^\star$. For example, if the closest integer value to $x_i^\star$ is $\lfloor x_i^\star \rfloor$, we fix x_i to integer values $\lfloor x_i^\star \rfloor, \lceil x_i^\star \rceil, \lfloor x_i^\star \rfloor - 1, \lceil x_i^\star \rceil + 1$, and so on. After each branching step, the resulting subproblem is a quadratic programming problem of type (MIQP) again, with a dimension decreased by one. We are not imposing bound constraints on the integer variables (i.e. $x_i \leq \lfloor x_i^\star \rfloor$ and $x_i \geq \lfloor x_i^\star \rfloor$) since they are taken into account as fixings (i.e. $x_i = \lfloor x_i^\star \rfloor$) in the construction of the reduced subproblem by adapting properly the matrices Q_ℓ and A_ℓ, the linear term $\bar{c}$, the constant term $\bar{d}$ and the right hand side $\bar{b}$ (see Section 3.4). The branching order of these variables at every level ℓ is set to $x_1, \ldots, x_{n-\ell}$, assuming that ℓ variables are already fixed. Hence, at every level we have a predetermined branching order. Let $x_i^\star$ be the value of the next branching variable in the continuous minimizer. Then, by the strict convexity of f, all consecutive lower bounds obtained by fixing x_i to integer values in increasing distance to $x_i^\star$, on each side of $x_i^\star$, are increasing. Thus, we can cut off the current node of the tree and all its outer siblings as soon as we fix a variable to some value for which the resulting lower bound exceeds the current best known upper bound. Since f is strictly convex we get a finite algorithm even without bounds on the variables.

Once all integer variables have been fixed, we compute the optimal solution of the QP problem in the reduced continuous subspace. If the computed point is feasible, it yields a valid upper bound for the original problem. As our enumeration is very fast and we use a depth-first approach, we do not need any initial feasible point nor do we apply primal heuristics.

3.2. Dual Approach. In the following, we derive the dual problem of the continuous relaxation of (MIQP) and point out some advantages when using the dual approach in the branch-and-bound framework. The dual can be computed by first forming the Lagrangian of the relaxation

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

and then, for fixed λ, minimizing $\mathscr{L}$ with respect to the primal variables x. As Q is assumed to be positive definite, the unique minimizer can be computed from the first order optimality condition

$$\nabla_x \mathscr{L}(x,\lambda) = 2Qx + c + A^\top \lambda = 0 \iff x = -\frac{1}{2}Q^{-1}(c + A^\top \lambda). \tag{3.1}$$

Having x as a function of λ, we can insert it into the Lagrangian $\mathscr{L}$ yielding the following dual function

$$\mathscr{L}(\lambda) = \lambda^\top \big(-\frac{1}{4} A Q^{-1} A^\top \big) \lambda - \big(b^\top + \frac{1}{2} c^\top Q^{-1} A^\top \big) \lambda - \frac{1}{4} c^\top Q^{-1} c + d.$$

Defining $\widetilde{Q} := \frac{1}{4} A Q^{-1} A^\top$, $\widetilde{c} := \frac{1}{2} A Q^{-1} c + b$ and $\tilde{d} := \frac{1}{4} c^\top Q^{-1} c - d$, we can thus write the dual of the continuous relaxation of (MIQP) as

$$\begin{aligned} -\min\ & \lambda^\top \widetilde{Q} \lambda + \widetilde{c}^\top \lambda + \tilde{d} \\ \text{s.t. } & \lambda \geq 0 \\ & \lambda \in \mathbb{R}^m. \end{aligned} \tag{3.2}$$

Note that (3.2) is again a convex QP, since $\widetilde{Q}$ is positive semidefinite.

The first crucial difference in considering the dual problem is that its dimension changed from n to m, which is beneficial if $m \ll n$. The second one is that $\lambda = 0$ is always feasible for (3.2). Finally, note that having the optimal solution $\lambda^\star \in \mathbb{R}^m$ of (3.2), it is easy to reconstruct the corresponding optimal primal solution $x^\star \in \mathbb{R}^n$ using the first order optimality condition (3.1).

Within a branch-and-bound framework, a special feature of the dual approach is the *early pruning*: we can stop the iteration process and prune the node as soon as the current iterate λ_k is feasible and its objective function value exceeds the current upper bound, since each dual feasible solution yields a valid bound. Note however that, in case we cannot prune, an optimal solution of the dual problem is required, since it is needed for the computation of the corresponding primal solution $x^\star$ which in turn is needed to decide the enumeration order in the branch-and-bound scheme.

During the tree search it may occur that a node relaxation is infeasible due to the current fixings. In this case infeasibility of the primal problem implies the unboundness of the dual problem. Therefore, during the solution process of the dual problem, an iterate will be reached such that its objective function value exceeds the current upper bound and the node can be pruned. This is why in our implementation of `FAST-QPA` we set the following stopping criterion: the algorithm stops either if the norm of the projected gradient is less than a given optimality tolerance, or if an iterate is computed such that its objective function value exceeds the current upper bound. More precisely, we declare optimality when the point $\lambda \in \mathbb{R}^m_+$ satisfies the following condition:

$$\| \max\{0, -g(\lambda)\} \| \leq 10^{-5}.$$

By Propositions 2.11 and 2.12, we then have a guarantee that the algorithm stops after a finite number of iterations.

3.3. Reoptimization. At every node of the branch-and-bound tree, we use our algorithm `FAST-QPA` described in Section 2 for solving Problem (3.2). A crucial advantage of using an active set method is the possibility of working with warmstarts,

i.e., of passing information on the optimal active set from a parent node to its children. In the dual approach the dimension of all subproblems is m, independently of the depth ℓ in the branch-and-bound tree. When fixing a variable, only the objective function changes, given by $\widetilde{Q}$, $\widetilde{c}$ and $\widetilde{d}$. So as the starting guess in a child node, we choose $\mathcal{A}^0 := \mathcal{A}(\lambda^\star)$, i.e., we use the estimated active set for the optimal solution $\lambda^\star$ of the parent node, according to Step 3 of Algorithm 1. We also pass the solution $\lambda^\star$ to the child nodes to initialize the variables in the line search procedure in Step 4 of Algorithm 1, that is we set $\tilde{x}^0_{\mathcal{N}^0} = \lambda^\star_{\mathcal{N}^0}$. Our experimental results presented in Section 4 show that this warmstarting approach reduces the average number of iterations of `FAST-QPA` significantly.

3.4. Incremental Computations and Preprocessing. A remarkable speed-up can be achieved by exploiting the fact that the subproblems enumerated in the branch-and-bound tree are closely related to each other. Let $\ell \in \{0, \ldots, n_1 - 1\}$ be the current depth in the branch-and-bound tree and recall that after fixing the first ℓ variables, the problem reduces to the minimization of

$$\bar{f} \colon \mathbb{Z}^{n_1-\ell} \times \mathbb{R}^{n-n_1} \to \mathbb{R},\ x \mapsto x^\top Q_\ell x + \bar{c}^\top x + \bar{d}$$

over the feasible region $\bar{\mathcal{F}} = \{x \in \mathbb{Z}^{n_1-\ell} \times \mathbb{R}^{n-n_1} \mid A_\ell x \leq \bar{b}\}$, where $Q_\ell \succ 0$ is obtained by deleting the corresponding ℓ rows and columns of Q and $\bar{c}$ and $\bar{d}$ are adapted properly by

$$\bar{c}_{j-\ell} := c_j + 2\sum_{i=1}^{\ell} q_{ij} r_i, \text{ for } j = \ell + 1, \ldots, n$$

and

$$\bar{d} := d + \sum_{i=1}^{\ell} c_i r_i + \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} q_{ij} r_i r_j,$$

where $r = (r_1, \ldots, r_\ell) \in \mathbb{Z}^\ell$ is the current fixing at depth ℓ. Similarly, A_ℓ is obtained by deleting the corresponding ℓ columns of A and the reduced right hand side $\bar{b}$ is updated according to the current fixing.

Since we use a predetermined branching order, the reduced matrices Q_ℓ, Q_ℓ^{-1} and A_ℓ only depend on the depth ℓ, but not on the specific fixings. Along with the reduced matrix Q_ℓ, the quadratic part of the reduced dual objective function $\widetilde{Q}_\ell$ can then be computed in the preprocessing phase, because they only depend on Q_ℓ and A_ℓ. The predetermined branching order also allows the computation of the maximum eigenvalues $\lambda_{max}(\widetilde{Q}_\ell)$ in the preprocessing phase, needed for ensuring proper convergence of our active set method as described in Section 2; compare (2.3).

Concerning the linear part $\widetilde{c}$ and the constant part $\widetilde{d}$ of the dual reduced problem, both can be computed incrementally in linear time per node: let $r = (r_1, \ldots, r_\ell) \in \mathbb{Z}^l$ be the current fixing at depth ℓ. By definition of $\widetilde{c}$, we have

$$\widetilde{c}(r) = \frac{1}{2} A_\ell Q_\ell^{-1} c(r) + b(r)\ ,$$

where the suffix (r) always denotes the corresponding data after fixing the first ℓ variables to r.

THEOREM 3.1. *After a polynomial time preprocessing, the vector $\widetilde{c}(r)$ can be constructed incrementally in $O(n - \ell + m)$ time per node.*

Proof. Defining $y(r) := -\frac{1}{2}Q_\ell^{-1}c(r)$, we have

$$\frac{1}{2}A_\ell Q_\ell^{-1}c(r) = -A_\ell \cdot y(r).$$

Note that $y(r)$ is the unconstrained continuous minimizer of $f(r)$. In [8], it was shown that $y(r)$ can be computed incrementally by

$$y(r) := [y(r') + \alpha z^{\ell-1}]_{1,\dots,n-\ell} \in \mathbb{R}^{n-\ell}$$

for some vector $z^{\ell-1} \in \mathbb{R}^{n-\ell+1}$ and $\alpha := r_\ell - y(r')_\ell \in \mathbb{R}$, where $r' = (r_1, \dots, r_{\ell-1})$ is the fixing at the parent node. This is due to the observation that the continuous minima according to all possible fixings of the next variable lie on a line, for which $z^{\ell-1}$ is the direction. It can be proved that the vectors $z^{\ell-1}$ only depend on the depth ℓ and can be computed in the preprocessing [8]. Updating y thus takes $O(n-\ell)$ time. We now have

$$\begin{aligned}\widetilde{c}(r) &= -A_\ell[y(r') + \alpha z^{\ell-1}]_{1,\dots,n-\ell} + b(r)\\ &= -A_\ell[y(r')]_{1,\dots,n-\ell} - \alpha A_\ell[z^{\ell-1}]_{1,\dots,n-\ell} + b(r)\\ &= -(A_{\ell-1}y(r') - y(r')_{n-\ell+1} \cdot A_{\bullet,n-\ell+1}) - \alpha A_\ell[z^{\ell-1}]_{1,\dots,n-\ell} + b(r).\end{aligned}$$

In the last equation, we used the fact that the first part of the computation can be taken over from the parent node by subtracting column $n-\ell+1$ of A, scaled by the last component of $y(r')$, from $A_{\ell-1}y(r')$, which takes $O(m)$ time. The second part $A_\ell[z^{\ell-1}]_{1,\dots,n-\ell}$ can again be computed in the preprocessing. The result then follows from the fact that also $b(r)$ can easily be computed incrementally from $b(r')$ in $O(m)$ time. □

THEOREM 3.2. *After a polynomial time preprocessing, the scalar $\tilde{d}(r)$ can be constructed incrementally in $O(n-\ell)$ time per node.*

Proof. Recalling that

$$\tilde{d}(r) = \frac{1}{4}c(r)^\top Q_\ell^{-1}c(r) - d(r)\ ,$$

this follows from the fact that $y(r) = -\frac{1}{2}Q_\ell^{-1}c(r)$ and $c(r)$ can be computed in $O(n-\ell)$ time per node [8]. □

COROLLARY 3.3. *After a polynomial time preprocessing, the dual problem* (3.2) *can be constructed in $O(n-\ell+m)$ time per node.*

Besides the effort for solving the QP with the active set method, computing the optimal solution of the primal problem from the dual solution is the most time consuming task in each node. The following observation is used to speed up its computation.

THEOREM 3.4. *After a polynomial time preprocessing, the optimal primal solution $x^\star(r)$ can be computed from the optimal dual solution $\lambda^\star(r)$ in $O(m\cdot(n-\ell))$ time per node.*

Proof. From (3.1) we derive

$$\begin{aligned}x^\star(r) &= -\frac{1}{2}Q_\ell^{-1}\left(\sum_{i=1}^m \lambda^\star(r)_i a_i + c(r)\right)\\ &= y(r) + \sum_{i=1}^m \lambda^\star(r)_i\left(-\frac{1}{2}Q_\ell^{-1}a_i\right)\ .\end{aligned}$$

The first part can again be computed incrementally in $O(n-\ell)$ time per node. For the second part, we observe that $-\frac{1}{2}Q_\ell^{-1}a_i$ can be computed in the preprocessing phase for all $i = 1, \ldots, m$. □

The above results show that the total running time per node is linear in $n - \ell$ when the number m of constraints is considered a constant and when we make the reasonable assumption that Problem (QP) can be solved in constant time in fixed dimension.

3.5. Postprocessing. Using `FAST-QPA`, we can solve the dual problem of the continuous relaxation in every node of the branch-and-bound tree, in case it admits a solution, up to high precision. If the dual problem is bounded, strong duality guarantees the existence of a primal feasible solution. However, in practice, computing the primal solution by formula (3.1), can affect its feasibility. This numerical problem is negligible in the pure integer case (where we end up fixing all the variables), while it becomes crucial in the mixed integer case. Indeed, when dealing with mixed integer problems, the primal solution of the relaxation in the optimal branch-and-bound node (i.e. the node that gives the optimal value) is actually used to build up the solution of the original problem.

Hence, in case a high precision is desired for the primal solution, we propose the following approach: the branch-and-bound node related to the optimal value gives an optimal fixing of the integer variables. Therefore, we consider the convex QP that arises from Problem (MIQP) under these optimal fixings. Then, we call a generic solver to deal with this QP problem (the values of the primal continuous variables obtained by formula (3.1) can be used as a starting point). Since the hardness of Problem (MIQP) is due to the integrality constraints on the variables, the running time for this postprocessing step is negligible.

In the experiments reported below, we apply this approach using `CPLEX 12.6` as solver for the QP problem (we choose 10^{-6} as feasibility tolerance). The time required for the postprocessing step is included in all stated running times. As noticed above, this postprocessing phase is not needed in the pure integer case.

4. Experimental Results. In order to investigate the potential of our algorithm `FAST-QPA-BB`, we implemented it in `C++/Fortran 90` and compared it to the MIQP solver of `CPLEX 12.6`. We also tested the branch-and-bound solver `B-BB` of `Bonmin 1.74`. However, we did not include the running times for the latter into the tables since its performance was not competitive at all, not even for mixed-integer instances with a few number of integer variables. All experiments were carried out on Intel Xeon processors running at 2.60 GHz. We used an absolute optimality tolerance and a relative feasibility tolerance of 10^{-6} for all algorithms.

In order to obtain a feasible solution to Problem (MIQP) and thus an initial upper bound – or to determine infeasibility of (MIQP) – we replace the objective function in (MIQP) by the zero function and use the `CPLEX 12.6` ILP solver. In principle, the algorithm also works when setting the initial upper bound to a very large value. Then it is either replaced as soon as a feasible solution is found in some branch-and-bound node, or it will remain unchanged until the algorithm terminates, in which case Problem (MIQP) must be infeasible.

Altogether, we randomly generated 1600 different problem instances for (MIQP), considering percentages of integer variables $p := \frac{n_1}{n} \in \{0.25, 0.50, 0.75, 1.0\}$. For $p = 0.25$ (0.5 / 0.75 / 1.0), we chose $n \in \{50, 100, 150, 200, 250\}$ ($\{50, 75, 100, 125, 150\}$ / $\{50, 60, 70, 80, 90\}$ / $\{50, 55, 60, 65, 70\}$), respectively. The number of constraints m was chosen in $\{1, 10, 25, 50\}$. For each combination of p, n and m, we generated 10

instances. For every group of instances with a given percentage of integer variables p, the parameter n was chosen up to a number such that at least one of the tested algorithms was not able to solve all of the 10 instances to optimality for $m = 1$ within our time limit of 3 cpu hours.

For generating the positive definite matrix Q, 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 $[-1, 1]$, then we set $Q = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$. The entries of c were chosen uniformly at random from $[-1, 1]$, moreover we set $d = 0$. For the entries of A and b, we considered two different choices:

(a) the entries of b and A were chosen uniformly at random from $[-1, 1]$,
(b) the entries of A were chosen uniformly at random from $[0, 1]$ and we set $b_i = \frac{1}{2}\sum_{j=1}^{n} a_{ij}$, $i = 1, \ldots, m$.

The constraints of type (b) are commonly used to create hard instances for the knapsack problem. At `www.mathematik.tu-dortmund.de/lsv/instances/MIQP.tar.gz` all instances are publicly available.

The performance of the considered algorithms for instances of type (a) can be found in Tables 1–4. We do not inlcude the tables for instances of type (b), since there are no significant differences in the results, except that they are in general easier to solve for our algorithm as well as for `CPLEX`. All running times are measured in cpu seconds. The tables include the following data for the comparison between `FAST-QPA-BB` and `CPLEX 12.6`: numbers of instances solved within the time limit, average preprocessing time, average running times, average number of branch-and-bound nodes, average number of iterations of `FAST-QPA` in the root node and average number of iterations of `FAST-QPA` per node in the rest of the enumeration tree. All averages are taken over the set of instances solved within the time limit.

From our experimental results, we can conclude that when $p \in \{0.25; 0.50\}$ `FAST-QPA-BB` clearly outperforms `CPLEX 12.6` for m up to 25 and is at least competitive to `CPLEX 12.6` if $p = 0.75$. For the mixed-integer case we can see that the average running times of `FAST-QPA-BB` compared to `CPLEX 12.6` are the better, the bigger the percentage of continuous variables is, even with a larger number of constraints. For the pure integer case we still outperform `CPLEX 12.6` for m up to 10. For all instances, the preprocessing time is negligible.

This experimental study shows that `FAST-QPA-BB` is able to solve 644 instances of type (a) to optimality, while `CPLEX 12.6` can only solve 606 instances. Note that the average number of branch-and-bound nodes in our dual approach is approximately 30 times greater than that needed by `CPLEX 12.6`. Nevertheless the overall running times of our approach are much faster for moderate sizes of m, emphasizing both the quick enumeration process within the branch-and-bound tree and the benefit of using reoptimization. Note that the performance of our approach highly depends on m. As the number of constraints grows, the computational effort for both solving the dual problem and recomputing the primal solution (see Theorem 3.4), is growing as well.

In Table 5 we compare the performance of `FAST-QPA-BB` with `FAST-QPA-BB-NP`, a version in which the early pruning is not implemented (see Section 3.2). We show the results for the pure integer instances of type (a) with $p = 1.0$. The benefits from the early pruning are evident: the average number of iterations of `FAST-QPA` is decreased leading to faster running times so that 9 more instances can be solved.

Our experimental results also underline the strong performance of `FAST-QPA`. The number of iterations of `FAST-QPA` needed in the root node of our branch-and-bound algorithm is very small on average: for $m = 50$ it is always below 60 and often much

inst		FAST-QPA-BB						CPLEX 12.6		
n	m	#	ptime	time	nodes	it root	it	#	time	nodes
50	1	10	0.00	6.83	9.24e+6	1.50	1.16	10	48.79	5.22e+5
50	10	10	0.00	83.15	3.16e+7	3.80	1.54	10	157.87	1.38e+6
50	25	9	0.01	2337.87	1.62e+8	8.56	2.09	10	1993.92	1.13e+7
50	50	0	-	-	-	-	-	0	-	-
55	1	10	0.00	28.47	3.62e+7	1.60	1.25	10	149.52	1.31e+6
55	10	10	0.00	427.66	1.46e+8	3.40	1.48	10	1030.84	7.30e+6
55	25	4	0.01	3843.30	3.37e+8	7.50	1.86	5	3126.64	1.64e+7
55	50	0	-	-	-	-	-	0	-	-
60	1	10	0.00	133.32	1.60e+8	1.30	1.11	10	692.71	5.67e+6
60	10	8	0.01	1894.81	6.86e+8	2.62	1.51	8	3397.34	2.35e+7
60	25	2	0.02	8963.19	7.55e+8	5.50	2.00	3	7357.56	3.72e+7
60	50	0	-	-	-	-	-	0	-	-
65	1	10	0.01	349.11	4.13e+8	1.40	1.15	10	1352.04	1.05e+7
65	10	8	0.01	4010.42	1.50e+9	2.75	1.48	4	4270.80	2.73e+7
65	25	0	-	-	-	-	-	1	7818.69	3.64e+7
65	50	0	-	-	-	-	-	0	-	-
70	1	10	0.01	1113.47	1.30e+9	1.60	1.27	9	4609.44	3.24e+7
70	10	4	0.01	6915.67	2.38e+9	2.50	1.51	1	9510.90	5.33e+7
70	25	0	-	-	-	-	-	0	-	-
70	50	0	-	-	-	-	-	0	-	-

TABLE 1
Results for instances of type (a) with $p = 1.0$.

inst		FAST-QPA-BB						CPLEX 12.6		
n	m	#	ptime	time	nodes	it root	it	#	time	nodes
60	1	10	0.00	1.81	2.01e+6	1.30	1.11	10	12.12	1.12e+5
60	10	10	0.01	5.65	1.82e+6	2.80	1.47	10	17.63	1.34e+5
60	25	10	0.02	32.60	2.36e+6	9.10	1.84	10	26.49	1.51e+5
60	50	10	0.08	641.91	6.64e+6	55.80	2.79	10	129.51	4.80e+5
70	1	10	0.01	12.15	1.30e+7	1.60	1.20	10	84.02	6.67e+5
70	10	10	0.01	27.11	8.19e+6	3.40	1.47	10	77.70	5.07e+5
70	25	10	0.03	159.81	1.23e+7	9.20	1.76	10	183.43	8.85e+5
70	50	10	0.08	2222.15	2.79e+7	18.60	2.41	10	593.70	1.89e+6
80	1	10	0.02	65.98	6.51e+7	1.40	1.12	10	446.60	2.91e+6
80	10	10	0.03	151.77	4.37e+7	3.80	1.45	10	386.33	2.08e+6
80	25	10	0.04	963.38	7.06e+7	10.30	1.77	10	791.62	3.20e+6
80	50	7	0.09	3339.39	5.38e+7	15.29	2.16	9	1903.79	5.27e+6
90	1	10	0.03	417.90	3.79e+8	1.30	1.11	10	2332.52	1.25e+7
90	10	10	0.04	1538.99	4.36e+8	3.00	1.42	10	2745.36	1.26e+7
90	25	10	0.06	4468.73	3.55e+8	5.90	1.69	9	3094.58	1.08e+7
90	50	1	0.09	5361.11	1.02e+8	10.00	2.01	4	5199.56	1.27e+7
100	1	6	0.05	1897.13	1.69e+9	1.33	1.10	5	5918.25	2.61e+7
100	10	6	0.06	5893.96	1.52e+9	4.83	1.41	0	-	-
100	25	1	0.06	4897.75	3.95e+8	5.00	1.64	0	-	-
100	50	1	0.11	3622.67	8.05e+7	15.00	1.99	1	2664.13	5.47e+6

TABLE 2
Results for instances of type (a) with $p = 0.75$.

inst		FAST-QPA-BB						CPLEX 12.6		
n	m	#	ptime	time	nodes	it root	it	#	time	nodes
50	1	10	0.00	0.02	9.83e+3	1.50	1.16	10	0.18	1.26e+3
50	10	10	0.00	0.03	5.46e+3	3.80	1.51	10	0.16	7.47e+2
50	25	10	0.01	0.11	4.68e+3	8.20	1.95	10	0.23	8.70e+2
50	50	10	0.06	1.14	8.30e+3	21.40	3.00	10	0.61	1.63e+3
75	1	10	0.01	0.25	1.76e+5	1.60	1.21	10	2.20	1.30e+4
75	10	10	0.02	0.68	1.65e+5	3.60	1.46	10	2.50	1.17e+4
75	25	10	0.02	2.23	1.68e+5	9.80	1.75	10	3.80	1.47e+4
75	50	10	0.09	12.08	1.70e+5	17.00	2.31	10	6.56	1.74e+4
100	1	10	0.05	13.84	1.08e+7	1.50	1.15	10	76.62	3.20e+5
100	10	10	0.05	14.13	3.46e+6	4.80	1.41	10	63.18	2.35e+5
100	25	10	0.07	56.86	4.35e+6	8.50	1.66	10	87.39	2.55e+5
100	50	10	0.13	322.71	5.39e+6	51.80	2.03	10	216.89	4.38e+5
125	1	10	0.11	193.38	1.33e+8	1.40	1.13	10	2449.15	6.70e+6
125	10	10	0.13	516.50	1.15e+8	3.40	1.39	10	2265.19	5.70e+6
125	25	10	0.14	1342.59	1.03e+8	8.80	1.56	10	2111.49	4.22e+6
125	50	10	0.21	4524.80	9.28e+7	52.60	1.86	10	2843.59	4.22e+6
150	1	9	0.22	4011.63	2.34e+9	1.78	1.25	2	5026.81	9.86e+6
150	10	6	0.24	6857.64	1.29e+9	3.50	1.40	0	-	-
150	25	3	0.24	9440.33	6.65e+8	7.33	1.54	0	-	-
150	50	0	-	-	-	-	-	0	-	-

TABLE 3
Results for instances of type (a) with $p = 0.5$.

inst		FAST-QPA-BB						CPLEX 12.6		
n	m	#	ptime	time	nodes	it root	it	#	time	nodes
50	1	10	0.01	0.01	1.31e+2	1.50	1.18	10	0.01	3.44e+1
50	10	10	0.00	0.01	1.96e+2	3.80	1.58	10	0.02	4.46e+1
50	25	10	0.01	0.03	1.40e+2	8.20	2.34	10	0.03	3.85e+1
50	50	10	0.06	0.10	1.14e+2	21.40	4.39	10	0.07	3.32e+1
100	1	10	0.06	0.09	1.05e+4	1.50	1.16	10	0.50	1.06e+3
100	10	10	0.06	0.11	3.88e+3	4.80	1.43	10	0.45	6.32e+2
100	25	10	0.06	0.18	4.91e+3	8.50	1.73	10	0.55	7.36e+2
100	50	10	0.12	0.75	8.24e+3	51.80	2.16	10	1.30	1.53e+3
150	1	10	0.25	0.62	1.68e+5	1.70	1.23	10	7.49	1.31e+4
150	10	10	0.22	0.97	1.24e+5	3.00	1.41	10	7.37	1.09e+4
150	25	10	0.24	2.85	1.56e+5	6.40	1.58	10	12.81	1.65e+4
150	50	10	0.31	4.74	7.77e+4	11.20	1.82	10	9.50	8.69e+3
200	1	10	0.58	14.79	6.38e+6	1.50	1.16	10	283.67	2.73e+5
200	10	10	0.52	21.32	3.00e+6	3.20	1.38	10	244.09	2.19e+5
200	25	10	0.57	74.93	4.03e+6	5.90	1.52	10	344.88	2.56e+5
200	50	10	0.68	400.99	7.44e+6	40.60	1.73	10	818.13	4.83e+5
250	1	10	1.12	329.54	1.21e+8	1.50	1.15	7	4413.64	2.54e+6
250	10	10	1.10	617.59	7.35e+7	3.80	1.37	9	5291.97	2.67e+6
250	25	10	1.17	2115.57	1.01e+8	6.90	1.48	5	4984.95	2.08e+6
250	50	9	1.28	3709.07	6.27e+7	49.11	1.68	4	5444.02	1.90e+6

TABLE 4
Results for instances of type (a) with $p = 0.25$.

inst		FAST-QPA-BB				FAST-QPA-BB-NP			
n	m	#	time	nodes	it	#	time	nodes	it
50	1	10	6.83	9.24e+6	1.16	10	8.79	9.24e+6	1.48
50	10	10	83.15	3.16e+7	1.54	10	145.95	3.16e+7	2.74
50	25	9	2337.87	1.62e+8	2.09	7	4587.82	1.35e+8	4.25
50	50	0	-	-	-	0	-		-
55	1	10	28.47	3.62e+7	1.25	10	34.86	3.62e+7	1.74
55	10	10	427.66	1.46e+8	1.48	10	743.12	1.46e+8	2.53
55	25	4	3843.30	3.37e+8	1.86	3	4216.79	1.56e+8	3.63
55	50	0	-	-	-	0	-	-	-
60	1	10	133.32	1.60e+8	1.11	10	154.74	1.60e+8	1.34
60	10	8	1894.81	6.86e+8	1.51	8	3259.91	6.86e+8	2.62
60	25	2	8963.19	7.55e+8	2.00	0	-		-
60	50	0	-	-	-	0	-	-	-
65	1	10	349.11	4.13e+8	1.15	10	410.14	4.13e+8	1.47
65	10	8	4010.42	1.50e+9	1.48	7	5238.44	1.23e+9	2.52
65	25	0	-	-	-	0	-	-	-
65	50	0	-	-	-	0	-	-	-
70	1	10	1113.47	1.30e+9	1.27	10	1318.95	1.30e+9	1.81
70	10	4	6915.67	2.38e+9	1.51	1	8714.63	1.31e+9	2.95
70	25	0	-	-	-	0	-	-	-
70	50	0	-	-	-	0	-	-	-

TABLE 5
Results for instances of type (a) with $p = 1.0$ turning early pruning on/off.

smaller. Using warmstarts, the average number of iterations drops to 1–6.

Besides the tables of average running times, we visualized our results by performance profiles in Figure 1, as proposed in [12]. They confirm the result that FAST-QPA-BB outperforms CPLEX 12.6 significantly.

5. Conclusions. We presented a new branch-and-bound algorithm for convex quadratic mixed integer minimization problems based on the use of an active set method for computing lower bounds. Using a dual instead of a primal algorithm considerably improves the running times, as it may allow an early pruning of the node. Moreover, the dual problem only contains non-negativity constraints, making the problem accessible to our tailored active set method FAST-QPA. Our sophisticated rule to estimate the active set leads to a small number of iterations of FAST-QPA in the root node that however grows as the number of constraints increases. This shows that for a large number of constraints the QPs addressed by FAST-QPA are nontrivial and their solution time has a big impact on the total running time, since we enumerate a large number of nodes in the branch-and-bound tree. Nevertheless, reoptimization helps to reduce the number of iterations of FAST-QPA per node substantially, leading to an algorithm that outperforms CPLEX 12.6 on nearly all problem instances considered.

REFERENCES

[1] T. ACHTERBERG, *SCIP: solving constraint integer programs*, Mathematical Programming Computation, 1 (2009), pp. 1–41.

[2] S. C. AGRAWAL, *On mixed integer quadratic programs*, Naval Research Logistics Quarterly, 21 (1974), pp. 289–297.

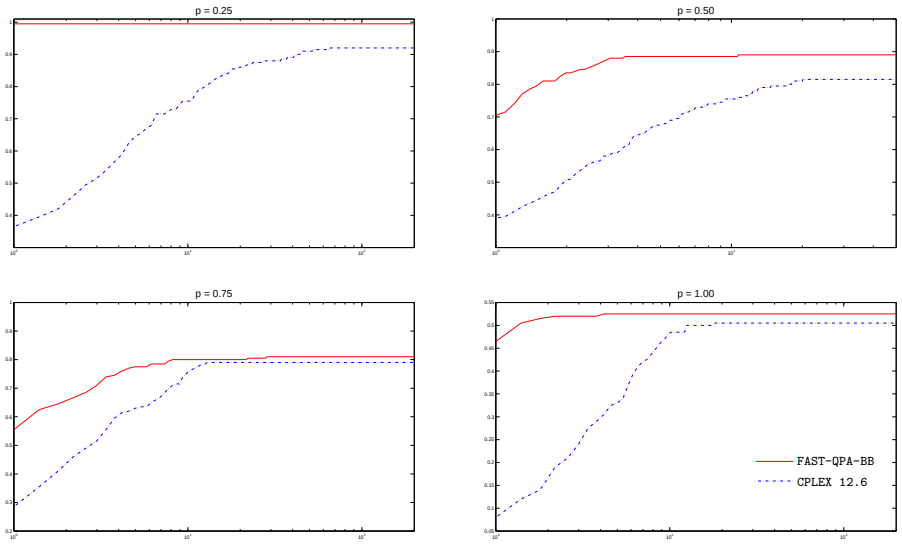

FIG. 1. *Performance profiles for all instances of type (a)*

[3] F. A. AL-KHAYYAL AND C. LARSEN, *Global optimization of a quadratic function subject to a bounded mixed integer constraint set*, Annals of Operations Research, 25 (1990), pp. 169 – 180.
[4] D. P. BERTSEKAS, *Projected Newton methods for optimization problems with simple constraints*, SIAM Journal on Control and Optimization, 20(2) (1982), pp. 221–246.
[5] D. P. BERTSEKAS, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999.
[6] D. BIENSTOCK, *Computational study of a family of mixed-integer quadratic programming problems*, Mathematical Programming, 74 (1996), pp. 121–140.
[7] P. BONAMI, L.T. BIEGLER, A.R. CONN, G. CORNUÉJOLS, I.E. GROSSMANN, C.D. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, 5 (2008), pp. 186 – 204.
[8] C. BUCHHEIM, A. CAPRARA, AND A. LODI, *An effective branch-and-bound algorithm for convex quadratic integer programming*, Mathematical Programming, 135 (2012), pp. 369–395.
[9] C. BUCHHEIM AND L. TRIEU, *Active set methods with reoptimization for convex quadratic integer programming*, in Combinatorial Optimization, vol. 8596 of Lecture Notes in Computer Science, 2014, pp. 125–136.
[10] G. CORNUÉJOLS AND R. TÜTÜNCÜ, *Optimization methods in finance*, Mathematics, finance, and risk, Cambridge University Press, Cambridge, U.K., New York, 2006.
[11] R. S. DEMBO, S. C. EISENSTAT, AND T. STEINHAUG, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.
[12] E.D. DOLAN AND J.J. MORÉ, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.
[13] F. FACCHINEI AND S. LUCIDI, *Quadratically and superlinearly convergent algorithms for the solution of inequality constrained minimization problems*, Journal of Optimization Theory and Applications, 85 (1995), pp. 265–289.
[14] *FICO Xpress Optimization Suite*, 2015. www.fico.com.
[15] R. FLETCHER AND S. LEYFFER, *Numerical experience with lower bounds for MIQP branch-and-bound*, SIAM Journal on Optimization, 8 (1998), pp. 604–616.
[16] L. GRIPPO, F. LAMPARIELLO, AND S. LUCIDI, *A truncated newton method with nonmonotone line search for unconstrained optimization*, Journal of Optimization Theory and Applications, 60 (1989), pp. 401–419.
[17] *GUROBI Optimizer*, 2015. www.gurobi.com.
[18] *IBM ILOG CPLEX Optimizer*, 2015. www.ibm.com/software/commerce/optimization/cplex-optimizer.
[19] R. LAZIMY, *Mixed-integer quadratic programming*, Mathematical Programming, 22 (1982),

pp. 332–349.
[20] ———, *Improved algorithm for mixed-integer quadratic programs and a computational study*, Mathematical Programming, 32 (1985), pp. 100–113.
[21] S. Leyffer, *Deterministic Methods for Mixed Integer Nonlinear Programming*, PhD thesis, University of Dundee, Scotland, UK, 1993.
[22] *MOSEK Optimization Software*, 2015. mosek.com/products/mosek.
[23] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer, 2000.
[24] B. N. Pshenichny and Y. M. Danilin, *Numerical Methods in Extremal Problems*, Mir Publishers, 1978.