J Glob Optim (2010) 48:595–632
DOI 10.1007/s10898-009-9507-y

# An efficient combined DCA and B&B using DC/SDP relaxation for globally solving binary quadratic programs

**Tao Pham Dinh · Nam Nguyen Canh · Hoai An Le Thi**

Received: 17 April 2007 / Accepted: 12 December 2009 / Published online: 10 January 2010

**Abstract** This paper addresses a new continuous approach based on the DC (Difference of Convex functions) programming and DC algorithms (DCA) to Binary quadratic programs (BQP) which play a key role in combinatorial optimization. DCA is completely different from other avalaible methods and featured by generating a convergent finite sequence of feasible binary solutions (obtained by solving linear programs with the same constraint set) with decreasing objective values. DCA is quite simple and inexpensive to handle large-scale problems. In particular DCA is explicit, requiring only matrix-vector products for Unconstrained Binary quadratic programs (UBQP), and can then exploit sparsity in the large-scale setting. To check globality of solutions computed by DCA, we introduce its combination with a customized Branch-and-Bound scheme using DC/SDP relaxation. The combined algorithm allows checking globality of solutions computed by DCA and restarting it if necessary and consequently accelerates the B&B approach. Numerical results on several series test problems provided in OR-Library (Beasley in J Global Optim, 8:429–433, 1996), show the robustness and efficiency of our algorithm with respect to standard methods. In particular DCA provides $\epsilon$-optimal solutions in almost all cases after only one restarting and the combined DCA-B&B-SDP always provides ($\epsilon -$)optimal solutions.

**Keywords** DC programming · DC algorithms (DCA) · Nonconvex quadratic programming · Exact penalty · DC relaxation · Branch-and-bound (B&B) · Semidefinite programming (SDP) · Binary quadratic programming (BQP)

T. Pham Dinh (✉) · N. Nguyen Canh
Laboratory of Modelling, Optimization & Operations Research (LMI), National Institute for Applied Sciences-Rouen, BP08-Avenue de l'Université, 76801 Saint-Etienne-du-Rouvray, France
e-mail: pham@insa-rouen.fr

N. Nguyen Canh
e-mail: nguyencn@insa-rouen.fr

H. A. Le Thi
Laboratory of Theoretical and Applied Computer Science (LITA), Informatics Department UFR MIM, Metz University, Ile du Saulcy, 57045 Metz Cedex, France
e-mail: lethi@sciences.univ-metz.fr

## 1 Introduction

A general binary quadratic program is of the form

$$\alpha := \min \quad \left\{ f(x) = \frac{1}{2} x^T Q x + q^T x : x \in F = S \cap \{0,1\}^n \right\}, \tag{BQP}$$

where $Q$ is a symmetric real $n \times n$ matrix, $q \in \mathbb{R}^n$ and $S$ is a polyhedral convex set of $\mathbb{R}^n$, i.e.,

$$S := \left\{ x \in \mathrm{R}^n : \left\langle a^i, x \right\rangle = b_i \quad i = 1, 2, \ldots, p \qquad \left\langle a^i, x \right\rangle \le b_i \quad i = p+1, \ldots, m \right\} \quad \text{with } m \ge 0. \tag{1}$$

Unconstrained binary quadratic programs correspond to the particular case where $F = S \cap \{0,1\}^n = \{0,1\}^n$.

$$\alpha := \min \quad \left\{ f(x) = \frac{1}{2} x^T Q x + q^T x : x \in \{0,1\}^n \right\}. \tag{UBQP}$$

Binary quadratic programs (BQP) appear in many areas including economics, machine scheduling, solid-state physics, traffic message management, computer-aided design and location, facility location, Frequency Assignment, Register Allocation, Pattern Matching, Analysis of Biological and Archeological Data, see [9,20]. A lot of methods, heuristic or deterministic have been developed for their solutions, among them: Hammer-Rudeanu [10], Adams-Sherali [2], Pardalos-Rodgers [35] Jha-Pardalos [19], Adams-Dearing [1], Helmberg-Rendl [13], Glover-Kochenberger-Alidaee [8], Le Thi-Pham Dinh [25], Billonnet-Elloumi [5]). In the deterministic approaches, Branch-and-Bound scheme in [25,35] and SDP technique with cutting planes in [13] were developed for solving BQP. The work using SDP technique [5] for unconstrained binary quadratic programs (UBDP) only deals with finding initial solutions for the MIQP solver of CPLEX 8.1 (ILOG 2002) in order to further improve the convergence speed of B&B scheme in MIQP. Billonnet-Elloumi have in fact proposed two approaches for computing the lower bounds (LB) for the optimal value $\alpha$, The first approach, related to the greatest quadratic convex minorization with the matrix of the form $Q - \rho I$ such that $\rho \le \lambda_1(Q)$, (the smallest eigenvalue $\lambda_1(Q)$ of $Q$), is not new: it was used in [21–25,32,34], whereas the second one computes the best LB corresponding to the matrix of the form $Q - \mathrm{Diag}(d)$ by using the solver SDP_S designed by Delaporte-Jouteau-Roupin [6] for solving the dual (DSDP) of the (SDP) problem, which is the equivalent formulation of the above mentioned problem of computing the best LB. Note that the (DSDP) problem is nothing but the well known SDP relaxation of unconstrained binary quadratic programs (UBQP) in Poljak-Rendl-Wolkowicz [39]. SDP_S works with SBmethod, an SDP solver based on the spectral bundle method of Helmberg-Rendl [14]. SDP techniques for computing lower bounds for nonconvex quadratic programs including (BQP) were largely investigated by Le Thi-Pham Dinh-Nguyen Canh [32,34].

The main contribution of our work relies on a new deterministic approach based on DC programming and DCA for globally solving binary quadratic programs (BQP). Equivalent DC programs of (BQP) are first established by using exact penalty techniques in DC programming developed in [29,30]. The resulting DC programs then are tackled by DCA.

DC programming and DCA were introduced by Pham Dinh Tao in 1985, as an extension of his earlier subgradient algorithms for concave programming, and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994. The DCA has been successfully applied to real world nonconvex programs in different fields of Applied Sciences, to which it quite often gave global solutions and proved to be more robust and more efficient than related standard methods, especially in large scale settings. It is worth noting that DCA is one of the rare efficient algorithms for nonsmooth nonconvex programming which allows solving large-scale DC programs (see [22,26,28,32,34,36,37]).

To globalize the local DCA, its combination with a customized Branch-and-Bound scheme (B&B) using, respectively, DC relaxation/SDP relaxation techniques [12,44] is introduced. The combined algorithm allows checking globality of solutions computed by DCA and restarting it if necessary, and, consequently, accelerates the B&B approach.

The paper is organized as follows. After a short introduction presenting the general model of binary quadratic programs, DC programming and DCA are briefly outlined in Sect. 2. The main contents of Sect. 3 concerns DC reformulations of (BQP) and the description of DCA applied to equivalent DC programs. A combination of DCA with a customized Branch-and-Bound and relaxation techniques is developed in Sect. 4. Finally computational experiments are reported in the last section as well as some conclusions.

## 2 DC programming and DCA

To give the reader an easy understanding of the theory of DC programming and DCA, we first briefly outline these tools in the following.

Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all the lower semicontinuous proper (i.e., not identically equal to $+\infty$) convex functions defined on $\mathbb{R}^n$ and taking values in $\mathbb{R} \cup \{+\infty\}$, and let $\|.\|$ be the Euclidean norm on $\mathbb{R}^n$. For $\theta : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the effective domain of $\theta$, denoted dom $\theta$, is defined by

$$\operatorname{dom}\theta := \left\{x \in \mathbb{R}^n : \theta(x) < +\infty\right\}.$$

For a nonempty set $C \subset \mathbb{R}^n$, co $C$ denotes the convex hull of $C$ and $\overline{\text{co}}C$ the closure of co $C$.

A DC program is defined by

$$\alpha = \inf\left\{f(x) := g(x) - h(x) \ : \ x \in \mathbb{R}^n\right\} \quad (P_{dc})$$

with $g, h \in \Gamma_0(\mathbb{R}^n)$. Such a function $f$ is called DC function, and $g - h$, DC decomposition of $f$ while the convex functions $g$ and $h$ are DC components of $f$. It should be noted that a constrained DC program, whose feasible set $C$ is a nonempty closed convex set, can always be transformed into a unconstrained DC program by adding the indicator function $\chi_C$ of $C$ ($\chi_C(x) = 0$ if $x \in C$, $+\infty$ otherwise) to the first DC component $g$. In DC programming [36], the convention

$$(+\infty) - (+\infty) := +\infty \tag{2}$$

has been adopted to avoid the ambiguity on the determination of $(+\infty) - (+\infty)$. Such a case does not present any interest and can be discarded. In fact, we are actually concerned with the following problem

$$\alpha = \inf\{f(x) := g(x) - h(x) \ : \ x \in \operatorname{dom} h\}, \tag{3}$$

which is equivalent to $(P_{dc})$ under the convention (2). It should be noted that a constrained DC program ($\varphi, \psi \in \Gamma_0(\mathbb{R}^n)$ and $C \subset \mathbb{R}^n$ being a nonempty closed convex set)

$$\alpha = \inf\{\varphi(x) - \psi(x) \ : \ x \in C\} \tag{4}$$

is equivalent to the unconstrained DC program by adding the indicator function $\chi_C$ of $C$ ($\chi_C(x) = 0$ if $x \in C$, $+\infty$ otherwise) to the first DC component $\varphi$:

$$\alpha = \inf \left\{ g(x) - h(x) \ : \ x \in \mathbb{R}^n \right\},$$

where $g := \varphi + \chi_C$ and $h := \psi$. The form (3) is preferred when one must emphasize the convex feasible set $C$ , for example in DC relaxation techniques for B&B scheme presented in Sect. 4.1.

Let

$$g^*(y) := \sup \left\{ \langle x, y \rangle - g(x) \ : \ x \in \mathbb{R}^n \right\}$$

be the conjugate function of $g$. By using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$h(x) := \sup \left\{ \langle x, y \rangle - h^*(y) : y \in \mathbb{R}^n \right\},$$

we have

$$\begin{aligned} \alpha &= \inf \left\{ g(x) - \sup \left\{ \langle x, y \rangle - h^*(y) : y \in \mathbb{R}^n \right\} : x \in \mathbb{R}^n \right\} \\ &= \inf \left\{ \alpha(y) : y \in \mathbb{R}^n \right\} \end{aligned}$$

with

$$(P_y) \quad \alpha(y) := \inf \left\{ g(x) - [\langle x, y \rangle - h^*(y)] : x \in \mathbb{R}^n \right\}.$$

It is clear that $(P_y)$ is a convex program and

$$\alpha(y) = h^*(y) - g^*(y) \quad \text{if } y \in \operatorname{dom} h^*, \text{ and } +\infty \text{ otherwise.} \tag{5}$$

Finally we state the dual program of $(P_{dc})$

$$\alpha = \inf \left\{ h^*(y) - g^*(y) : y \in \operatorname{dom} h^* \right\},$$

that is written, in virtue of the convention (2):

$$(D_{dc}) \quad \alpha = \inf \left\{ h^*(y) - g^*(y) : y \in \mathbb{R}^n \right\}.$$

The dual program of $(P_{dc})$ is of the form

$$\alpha = \inf \left\{ h^*(y) - g^*(y) : y \in \operatorname{dom} h^* \right\}$$

that is written, in virtue of the natural convention in DC programming, say $+\infty - (+\infty) = +\infty$:

$$\alpha = \inf \left\{ h^*(y) - g^*(y) : y \in Y \right\}. \quad (D_{dc})$$

where

$$g^*(y) := \sup \left\{ \langle x, y \rangle - g(x) \ : \ x \in \mathbb{R}^n \right\}$$

is the conjugate function of $g$.

Remark that if the optimal value $\alpha$ is finite then $\operatorname{dom} g \subset \operatorname{dom} h$ and $\operatorname{dom} h^* \subset \operatorname{dom} g^*$.

If $g$ or $h$ are polyhedral convex functions then $(P_{dc})$ is called a polyhedral DC program, which plays a main role in nonconvex programming (see [21,22,28,36,37] and references therein), and enjoys interesting properties (from both theoretical and practical viewpoints) concerning the local optimality and the convergence of the DCA. Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in \text{dom}\,\theta$, $\partial\theta(x_0)$ denotes the subdifferential of $\theta$ at $x_0$, i.e.,

$$\partial\theta(x_0) := \left\{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\right\} \tag{6}$$

(see [15,40]). The subdifferential $\partial\theta(x_0)$ is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that $\theta$ is differentiable at $x_0$ if and only if $\partial\theta(x_0)$ is a singleton which is exactly $\{\nabla\theta(x_0)\}$. The domain of $\partial\theta$, denoted dom $\partial\theta$, is defined by: dom $\partial\theta := \{x \in \text{dom}\,\theta : \partial\theta(x) \neq \emptyset\}$. Calling ri $C$ the relative interior of the convex set $C$, there holds ([15,40]):

$$\text{ir}(\text{dom}\,\theta) \subset \text{dom}\,\partial\theta \subset \text{dom}\,\theta \tag{7}$$

DC programming investigates the structure of the vector space $DC(\mathbb{R}^n) := \Gamma_0(\mathbb{R}^n) - \Gamma_0(\mathbb{R}^n)$, DC duality and optimality conditions for DC programs. The complexity of DC programs resides, of course, in the lack of practical optimal globality conditions. We developed instead the following necessary local optimality conditions for DC programs in their primal part, by symmetry their dual part is trivial (see [21,22,28,36,37]):

$$\partial h\left(x^*\right) \cap \partial g\left(x^*\right) \neq \emptyset \tag{8}$$

(such a point $x^*$ is called *critical point* of $g - h$ or for $(P_{dc})$), and

$$\emptyset \neq \partial h\left(x^*\right) \subset \partial g\left(x^*\right). \tag{9}$$

The condition (9) is also sufficient (for local optimality) in many important classes of DC programs. In particular it is sufficient for the next cases quite often encountered in practice:

- In polyhedral DC programs with $h$ being a polyhedral convex function ( [22,28,36,37]). In this case, if $h$ is differentiable at a critical point $x^*$, then $x^*$ is actually a local minimizer for $(P_{dc})$. Since a convex function is differentiable everywhere except for a set of measure zero, one can say that a critical point $x^*$ is almost always a local minimizer for $(P_{dc})$.
- In case the function $f$ is locally convex at $x^*$ (see [28,36] and references therein). Note that, if $h$ is polyhedral convex, then $f = g - h$ is locally convex everywhere $h$ is differentiable.

The transportation of global solutions between $(P_{dc})$ and $(D_{dc})$ is expressed by:

$$\left[\bigcup_{y^* \in \mathcal{D}} \partial g^*(y^*)\right] \subset \mathcal{P}, \left[\bigcup_{x^* \in \mathcal{P}} \partial h(x^*)\right] \subset \mathcal{D} \tag{10}$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of $(P_{dc})$ and $(D_{dc})$, respectively. Under technical conditions, this transportation holds also for local solutions of $(P_{dc})$ and $(D_{dc})$ (see [28,36] and references therein).

Based on local optimality conditions and duality in DC programming, the DCA consists in constructing of two sequences $\{x^k\}$ and $\{y^k\}$ of trial solutions of the primal and dual programs, respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing, and $\{x^k\}$ (resp. $\{y^k\}$) converges to a primal feasible solution $\widetilde{x}$ (resp. a dual feasible solution $\widetilde{y}$) satisfying local optimality conditions and

$$\widetilde{x} \in \partial g^*\left(\widetilde{y}\right), \quad \widetilde{y} \in \partial h\left(\widetilde{x}\right). \tag{11}$$

The sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that $x^{k+1}$ (resp. $y^{k+1}$) is a solution to the convex program $(P_k)$ (resp. $(D_{k+1})$) defined by ($x^0 \in$ dom $\partial h$ being a given initial point and $y^0 \in \partial h(x^0)$ being chosen)

$$\begin{aligned} (P_k) \quad & \inf\{g(x) - [h(x^k) + \langle x - x^k, y^k\rangle] : x \in \mathbb{R}^n\}, \\ (D_{k+1}) \quad & \inf\{h^*(y) - [g^*(y^k) + \langle y - y^k, x^{k+1}\rangle] : y \in \mathbb{R}^n\}. \end{aligned}$$

The DCA has the quite simple interpretation: at the $k$-th iteration, one replaces in the primal DC program $(P_{dc})$ the second component $h$ by its affine minorization $h^{(k)}(x) := h(x^k) + \langle x - x^k, y^k\rangle$ defined by a subgradient $y^k$ of $h$ at $x^k$ to give birth to the primal convex program $(P_k)$, the solution of which is nothing but $\partial g^*(y^k)$. Dually, a solution $x^{k+1}$ of $(P_k)$ is then used to define the dual convex program $(D_{k+1})$ obtained from $(D_{dc})$ by replacing the second DC component $g^*$ with its affine minorization $(g^*)^{(k)}(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ defined by the subgradient $x^{k+1}$ of $g^*$ at $y^k$ : the solution set of $(D_{k+1})$is eaxctly $\partial h(x^{k+1})$. The process is repeated until convergence. DCA performs a double linearization with the help of the subgradients of $h$ and $g^*$ and the DCA then yields the next scheme: (starting from given $x^0 \in \operatorname{dom} \partial h$)

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k), \quad \forall k \geq 0. \tag{12}$$

DCA's distinctive feature relies upon the fact that DCA deals with the convex DC components $g$ and $h$ but not with the DC function $f$ itself. DCA is one of the rare algorithms for nonconvex nonsmooth programming. Moreover, a DC function *f has infinitely many DC decompositions which have crucial implications for the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA. For a given DC program, the choice of *optimal* DC decompositions is still open. Of course, this depends strongly on the very specific structure of the problem being considered. In order to tackle the large-scale setting, one tries in practice to choose $g$ and $h$ such that sequences $\{x^k\}$ and $\{y^k\}$ can be easily calculated, i.e., either they are in an explicit form or their computations are inexpensive.

We mention now the main convergence properties of DCA. Let $C$ (resp. $D$) a convex set containing the sequence $\{x^k\}$ (resp. $\{y^k\}$) and $\rho(g, C)$ (or $\rho(g)$ if $C = \mathbb{R}^n$) the modulus of strong convexity of $g$ on $C$ given by:

$$\rho(g, C) = \sup\left\{\rho \geq 0 : g - (\rho/2)\|\cdot\|^2 \text{ be convex on } C\right\}.$$

**DCA's convergence properties:**
DCA is a descent method without linesearch which enjoys the following properties:

i) The sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and

- $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ iff $y^k \in \partial g(x^k) \cap \partial h(x^k)$, $y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1})$ and $[\rho(g, C) + \rho(h, C)]\|x^{k+1} - x^k\| = 0$. Moreover if $g$ or $h$ are strictly convex on $C$ then $x^k = x^{k+1}$.
  In such a case DCA terminates at the $k$th iteration (finite convergence of DCA)
- $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ iff $x^{k+1} \in \partial g^*(y^k) \cap \partial h^*(y^k)$, $x^{k+1} \in \partial g^*(y^{k+1}) \cap \partial h^*(y^{k+1})$ and $[\rho(g^*, D) + \rho(h^*, D)]\|y^{k+1} - y^k\| = 0$. Moreover if $g^*$ or $h^*$ are strictly convex on $D$, then $y^{k+1} = y^k$.
  In such a case DCA terminates at the $k$th iteration (finite convergence of DCA).

ii) If $\rho(g, C) + \rho(h, C) > 0$ (resp. $\rho(g^*, D) + \rho(h^*, D) > 0$)) then the series $\{\|x^{k+1} - x^k\|^2$ (resp. $\{\|y^{k+1} - y^k\|^2\}$ converges.
iii) If the optimal value $\alpha$ of problem $(P_{dc})$ is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point $\widetilde{x}$ (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).
iv) DCA has a linear convergence for general DC programs.
v) DCA has a finite convergence for polyhedral DC programs.

It is worth noting that for suitable DC decompositions, DCA generates almost standard algorithms in convex and nonconvex programming. For a complete study of DC programming and DCA the reader is referred to [21,22,28,36,37] and references therein.

## 3 DC reformulation and DCA for solving the equivalent DC programs

In order to apply DCA, we need to reformulate (BQP) as a DC program by using exact penalty techniques in DC programming. According to Le Thi et al. [30], the exact penalty is stated as follows

**Theorem 1** *Let $K$ be a nonempty bounded polyhedral convex set, $f$ be a finite concave function on $K$ and p be a finite nonnegative concave function on $K$. Assume that the problem $(P)$ be feasible*

$$(P) \qquad \alpha = \inf\{f(x) : x \in K, \, p(x) \le 0\},$$

*whose solution set is denoted by $\mathcal{P}$ and let $(P_\tau)$, for $\tau \ge 0$, be the penalized problem*

$$(P_\tau) \qquad \alpha(\tau) = \inf\{f(x) + \tau p(x) : x \in K\},$$

*whose solution set is denoted by $\mathcal{P}_\tau$. Then there exists $\tau_0 \ge 0$ such that for all $\tau > \tau_0$, the two problems $(P)$ and $(P_\tau)$ have the same optimal value and the same solution set. The optimal parameter $\tau_0$ is determined as follows:*

(a) *if the vertex set $V(K)$ of $K$ is contained in $\{x \in K : p(x) \le 0\}$, then $\tau_0 = 0$ and $\alpha(0) = \alpha$. Conversely $\alpha(0) = \alpha$ if and only if* $\inf\{f(x) : x \in V(K), p(x) > 0\} \ge \alpha$ *(resp.* $\inf\{f(x) : x \in K, p(x) > 0\} \ge \alpha$*);*

(b) *if $\alpha(0) < \alpha$ then $\tau_0 := \max\left\{\dfrac{\alpha - f(x)}{p(x)} : x \in V(K), p(x) > 0\right\}$, and there hold*

(b1) $\alpha(\tau) = \alpha$ *if and only if* $\tau \ge \tau_0$,
(b2) $\mathcal{P}_\tau \cap \{x \in K : p(x) \le 0\} \neq \emptyset \Leftrightarrow \mathcal{P} \subset \mathcal{P}_\tau \Leftrightarrow \tau \ge \tau_0$,
(b3) $\mathcal{P}_\tau = \mathcal{P}$ *if* $\tau > \tau_0$.

*Remark 2* It has been proved in [29] that Theorem 1 still holds true in case the objective function $f$ is a DC function.

To apply this theorem we first reformulate Problem (BQP) as a concave quadratic program (a particular DC program of minimizing a concave quadratic function over a polyhedral convex set) by using the binary character of $x$. The motivation of choosing such DC decomposition will pointed out in Remark 6

Consider the following quadratic function, with $\rho$ being a given real number:

$$f_\rho(x) = \frac{1}{2}x^T(Q - \rho I_n)x + q^T x + \frac{1}{2}\rho x^T e^{(n)}, \tag{13}$$

where $e^{(n)} \in \mathbb{R}^n$ is the vector of ones and $I_n$ is the identity matrix of order $n$. It is clear that $f_\rho$ and the objective function $f$ in (BQP) agree on $\{0, 1\}^n$ and a fortiori on the feasible set of (BQP).

So if we choose $\rho \ge \lambda_n(Q)$, the largest eigenvalue of matrix $Q$, then the function $f_\rho$ becomes concave.

On the other hand, let the function $p$ be defined by

$$p(x) = p_1(x) := \sum_{i=1}^{n} \min\{x_i, 1 - x_i\}. \tag{14}$$

It is easy to see that $p_1$ is finite concave on $\mathbb{R}^n$, (more exactly $-p_1$ is finite polyhedral convex on $\mathbb{R}^n$) and $p_1(x) \geq 0$ for all $x \in K := S \cap [0, 1]^n\}$. Moreover we have

$$\left\{x \in \mathbb{R}^n : x \in S, x \in \{0, 1\}^n\right\} = \{x \in K : p_1(x) \leq 0\}.$$

Hence the binary quadratic program (BQP) can be rewritten as

$$\alpha = \min\{f_\rho(x) : x \in K, p_1(x) \leq 0\} \tag{15}$$

Consider, from now on, the polyhedral DC program (with $\rho \geq \lambda_n(Q)$ and $\tau > \tau_0$) :

$$\min\{f_{\rho,\tau}(x) := f_\rho(x) + \tau p_1(x) : x \in K\}, \tag{16}$$

that is a nonsmooth nonconvex program.

As a result of Theorem 1, we get

**Proposition 3** *The penalized Problem* (16) *is equivalent to Problem* (15) *in the sense given in Theorem* 1.

*Remark 4* (i) We have $f_\rho(x) - f(x) = \frac{1}{2}\rho x^T[e^{(n)} - x] \geq 0, \forall x \in [0, 1]^n$ if $\rho \geq 0$. More precisely the difference is positive on $[0, 1]^n \setminus \{0, 1\}^n$ if $\rho > 0$. Theoretically, by giving $\rho$ large values, one might force DCA applied to (16) to converge to feasible solutions of (BQP). That is an interesting feature of the DC decomposition ( 13).

(ii) In the case $K = S \cap [0, 1]^n = [0, 1]^n$, i.e., there is no linear constraint, if we choose $\rho > \lambda_n(Q)$, the function $f_\rho$ is strictly concave. As a result, we obtain directly the following equivalent DC program of (BQP)

$$\min\left\{f_\rho(x) = \frac{1}{2}x^T(Q - \rho I_n)x + q^T x + \frac{1}{2}\rho x^T e^{(n)} : x \in [0, 1]^n\right\} \tag{17}$$

(iii) Another usual exact penalty is given by $p_2(x) = \sum_{i=1}^{n} x_i(1 - x_i)$, which leads also to an equivalent polyhedral DC program.

(iv) Other equivalent DC programs can be estabished by the exact penalty techniques in DC programming developed in [29].

The DC decomposition of $f_{\rho,\tau}$ is derived from the concavity of the functions $f_\rho$ and $\tau p_1$. It leads to the DC program with

$$g(x) = \chi_K(x) \quad \text{and} \quad h(x) = -f_\rho(x) - \tau p_1(x), \tag{18}$$

where $\chi_K$ is the indicator function of $K$.

*Remark 5* It is worthwhile noting that a DC function has infinitely many DC decompositions *which have crucial impacts on the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions, ...) of DCA. For a given DC program, the choice of *optimal* DC decompositions is still open. Of course, this depends strongly on the very specific structure of the problem being considered. In order to tackle the large scale setting, one tries in practice to choose $g$ and $h$ such that sequences $\{x^k\}$ and $\{y^k\}$ can be easily calculated, i.e. either they are in explicit form or their computations are inexpensive. The advantages of the above DC decomposition (15 ) will be discussed below in the description of the resulting DCA.

According to Sect. 2, performing DCA for Problem (16) amounts to computing the two sequences $\{x^k\}$ and $\{y^k\}$ defined by

$$y^k \in \partial h(x^k), x^{k+1} \in \partial g^*(y^k)$$

In other words, we have to compute the subdifferentials $\partial h$ and $\partial g^*$. As usually, $\partial h$ is often explicitly computed with the help of known rules in convex analysis. Here we have

$$\partial h = \partial(-f_\rho) + \tau \partial(-p_1) \tag{19}$$

with the explicit computations of $\partial(-f_\rho)$ and $\partial(-p)$ . Indeed,

$$\partial(-f_\rho) = -(Q - \rho I_n)x - q - \frac{1}{2}\rho e^{(n)} \tag{20}$$

and since

$$-p_1(x) = \sum_{i=1}^{n} \max\{-x_i, x_i - 1\} = \sum_{i=1}^{n} \max\left\{\left(-e_i^{(n)}\right)^T x, \left(e_i^{(n)}\right)^T x - 1\right\}$$

where $\{e_1^{(n)}, \dots, e_n^{(n)}\}$ is the canonical basis of $\mathbb{R}^n$. It follows that

$$\partial(-p_1)(x) = \sum_{i=1}^{n} u_i \text{ where } u_i = \begin{cases} -e_i^{(n)} & \text{if } x_i < 0.5 \\ e_i^{(n)} & \text{if } x_i > 0.5 \\ \text{belongs to } [-e_i^{(n)}, e_i^{(n)}] & \text{if } x_i = 0.5 \end{cases} \tag{21}$$

As for computing $\partial g^*(y)$, we need only to solve the following linear program

$$\min\{-\langle x, y\rangle : x \in K\}. \tag{22}$$

*Remark 6* The primal sequence $\{x^k\}$ can then be included in the vertex set $V(K)$ of $K$. This property constitutes one of interesting advantages of the DC decomposition (13), knowing that $V(K)$ contains the feasible set $F$ of (BQP).

We are now in a position to describe the DCA applied to Problem (16)

**Algorithm 1** (DCA applied to (16))

1. Set $k = 1$, let $\varepsilon_1, \varepsilon_2$ be small enough positive number. Choose an initial point $x^k$ (not necessarily in $K$).
2. Compute $y^k \in \partial h(x^k)$ by using (19), (20) and (21)
3. Compute $x^{k+1} \in \partial g^*(y^k) \cap V(K)$ by solving the linear program (22)
4. If either

$$\left\| x^{k+1} - x^k \right\| \le \varepsilon_1$$

or

$$\left\| (f_\rho + \tau p_1)\left(x^{k+1}\right) - (f_\rho + \tau p_1)\left(x^k\right) \right\| \le \varepsilon_2$$

then STOP and $x^{k+1}$ is the computed solution.
Otherwise, set $k \leftarrow k + 1$ and go to Step 2.

Performing DCA amounts to solving a finite number of linear programs (22) having the same constraint set $K$. Moreover, for unconstrained binary quadratic programs, DCA is explicit: $K = [0, 1]^n$ in (22 ), according to (17). Its finite convergence will be emphasized in Theorem 9.

*Remark 7* (Practical choice of the parameter penalty $\tau > \tau_0$ and its use in our combined DCA and BB).

In general it is difficult to compute explicitly any upper bound of $\tau_0$ in Problem (16). In practice we take $\tau$ sufficiently large in order for Problem (16) to be equivalent to Problem (BQP). To check equivalence of these problems, we use the exact penalty results in Theorem 1 : $\alpha(\tau) \leq \alpha$ for every $\tau \geq 0$ and if an optimal solution to Problem (16) with a given $\overline{\tau} \geq 0$ is feasible to Problem ( BQP) then it is also an optimal solution of the latter one and $\overline{\tau} \geq \tau_0$.

## 3.1 Special feature of DCA for Problem(16)

It is crucial for local algorithms applied to the penalty equivalent (16) to provide feasible solutions $x^*$ of the binary quadratic program (BQP), i.e., $p_1(x^*) = 0$. We shall prove that DCA bears this feature, due to the fact that (16) is a polyhedral DC program with the first DC component being polyhedral convex.

**Theorem 8** (i) *There is a positive constant $\tau_1$ such that for all $\tau > \tau_1$ and all $x \in V(K), x' \in \partial g^*(\partial h(x)) \cap V(K)$, there holds*

$$(f_\rho + \tau p_1)\left(x'\right) \leq (f_\rho + \tau p_1)(x), \;\; p_1(x') \leq p_1(x) \tag{23}$$

(ii) *For all $\tau > \tau_1$ and all primal sequence $\{x^k\} \subset V(K)$ generated by DCA applied to the penalty equivalent (16), both sequences $\left\{f_\rho(x^k) + \tau p_1\left(x^k\right)\right\}$ and $\left\{p_1\left(x^k\right)\right\}$ are decreasing.*

*Proof* As mentionned above, $V(K)$ always contains the feasible solution set $F$ of (BQP). Hence, if $V(K)$ is contained in $F$, i.e., $V(K) = F$, then the assertion is trivial with $\tau_1 = 0$. Otherwise let

$$\tau_1 := \max\left\{\frac{f_\rho(x) - f_\rho(x')}{p_1(x') - p_1(x)} : (x, x') \in V(K) \times V(K), p_1(x') > p_1(x)\right\},$$

then $0 \leq \tau_1 < +\infty$ since $V(K)$ is a finite. Consider now $\tau > \tau_1$ and $x, x'$ given as above. Assume for contradiction that such that $p_1(x') > p_1(x)$. Then

$$\tau\left[p_1(x') - p_1(x)\right] > \tau_1[p_1(x') - p_1(x)] \geq f_\rho(x) - f_\rho(x').$$

Hence

$$f_\rho(x') + \tau p_1(x') > f_\rho(x) + \tau p_1(x),$$

it contradicts the fact that DCA is a descent method and $x'$ is a next step of DCA from $x$. The assertion (ii) then is immediate. □

The enhancing features of DCA applied to the penalty equivalent (16) can be summarized in the next Theorem whose proof is immediate from Theorem 8 and the general convergence of DCA applied to polyhedral DC program ([28,36]).

**Theorem 9** (*Convergence properties of Algorithm 1*) *For $\rho \geq \lambda_n(Q)$ and $\tau > \max\{\tau_0, \tau_1\}$, there hold*

(i) *Algorithm 1 generates a finite sequence $\{x^k\}$ contained in $V(K)$ such that both the sequences $\{f_\rho(x^k) + \tau p_1(x^k)\}$ and $\{p_1(x^k)\}$ are decreasing.*
(ii) *If $x^r \in \{0, 1\}^n$, then $x^k \in \{0, 1\}^n$ for all $k \geq r$.*
(iii) *The sequence $\{x^k\}$ converges to $x^* \in V(K)$ after a finite number of iterations. Moreover if $\rho > \lambda_n(Q)$, then $\{x^k\}$ is stationary at $x^*$, i.e., there is some $r$ such that $x^* = x^{r+1} = x^r$.*

### 3.2 Initial point and strategies of launching DCA

Theorem 9 says that, starting with a feasible solution of the binary quadratic program (BQP), DCA provides a better one, although it works on a continuous feasible set. A good feasible point can be found by applying beforehand DCA to the concave programming problem as in [27] :

$$0 = \min \left\{ \sum_{i=1}^{n} \min \{x_i, 1 - x_i\} : x \in K \right\}. \tag{24}$$

Problem (24) is a polyhedral DC program with known optimal value and whose solution set is exactly the feasible solution set $F$ of ( BQP). Fortunately, as for linear complementarity problems [27,38], DCA, with starting point $\overline{x}$ not necessarily feasible but such that $p_1(\overline{x}) \leq 0$, converges, almost always in practice, to a global solution of (24).

Another efficient strategy improves upper bounds by launching DCA from an optimal solution of convex programs formed by DC relaxation or SDP relaxation, which will be developed in the next section.

***When do we restart DCA?***

During the B&B process we can restart DCA to improve the current best upper bound. Usually, an upper bound is obtained when a binary solution is found, we call this upper bound (the value of $f$ at this solution) a $Score$. However, by using the exact penalty technique, $f_{\rho,\tau}(x)$, with $x \in K$, is also an upper bound and is denoted $UB_f$**.** So in our algorithm, we will restart DCA when the following condition is satisfied

$$f_{\rho,\tau}(\bar{x}) < \min\{Score, UB_f\} \left(1 + 10e^{-3}\right). \tag{25}$$

where $\bar{x}$ is an optimal solution of the current relaxed convex program (computing lower bound).

## 4 Global optimization algorithm

In what follows, we will establish an algorithm for globally solving the binary quadratic program (BQP). This is a combination of DCA presented in the previous section and a customized Branch-and-Bound approach. We first discuss two basic operations in any Branch-and-Bound scheme : bound estimation and branching procedure.

### 4.1 DC relaxation technique for lower bounding in B&B

Lower bounding procedure—which is based, in general, on the statement of a convex program whose optimal value is a lower bound for the optimal value of the nonconvex program being considered—plays an important role in the construction of B&B scheme in global optimization: the tighter the lower bound is, the more efficient the B&B will be. Let us outline now the DC relaxation technique for computing lower bounds in DC programming.

#### *4.1.1 Convex hull of nonconvex functions*

Consider the nonconvex program

$$\alpha := \inf\{\psi(x) : x \in \mathbb{R}^n\}, \tag{26}$$

where $\psi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is proper (dom $\psi \neq \emptyset$) and has an minorization affine on $\mathbb{R}^n$. The optimal way to convexify (26) passes by the convex hull of $\psi$ defined by [15,40]:

$$\text{co } \psi(x) := \inf \left\{ \sum_i \lambda_i \psi(x_i) : \lambda_i \geq 0, \sum_i \lambda_i = 1, x \in \text{dom } \psi, x = \sum_i \lambda_i x_i \right\}, \quad (27)$$

where the infimum is taken over all representations of $x$ as a convex combination of elements $x_i$, such that only finitely many coefficients $\lambda_i$ are nonzero. The convex function co $\psi$ with

$$\text{dom co } \psi = \text{co dom } \psi \quad (28)$$

is the greatest convex function majorized by $\psi$. It leads to the convex programs with the same optimal value

$$\alpha := \inf\{\text{co } \psi(x) : x \in \mathbb{R}^n\} = \inf\{\overline{\text{co}}\ \psi(x) : x \in \mathbb{R}^n\}. \quad (29)$$

It is well known that [15]

(i) $Arg \min \psi \subset Arg \min \text{co } \psi \subset Arg \min \overline{\text{co}}\ \psi$
(ii) $\text{co } (Arg \min \psi) \subset \overline{\text{co}}\ (Arg \min \psi) \subset Arg \min \overline{\text{co}}\ \psi$
(iii) $\overline{\text{co}}\ \psi = \psi^{**}$.
(iv) If, in addition, $\psi$ is lower-semicontinuous and 1-coercive (the latter means $\lim_{\|x\| \to +\infty} \frac{\psi(x)}{\|x\|} = +\infty$), then $\text{co } \psi = \overline{\text{co}}\ \psi = \psi^{**}$.

*Remark 10* (i) Problem (26 ) can be rewritten as

$$\alpha := \inf\{\psi(x) : x \in \text{dom } \psi\}, \quad (30)$$

while in (29), one can replace $x \in \mathbb{R}^n$ by $x \in \text{co dom } \psi$. As usually in convex analysis, a function $\psi : C \subset \mathbb{R}^n \to \mathbb{R}$ is often identified to its extension $\psi + \chi_C$ to the whole $\mathbb{R}^n$.

(ii) For $C \subset \mathbb{R}^n$ and $\psi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ with $C \subset \text{dom } \psi$, we denote by co $_C\psi$ the convex hull of $\psi$ on $C$, i.e., $\text{co}_C\psi := \text{co } (\psi + \chi_C)$. Likewise $\overline{\text{co}}_C\psi$ stands for $\overline{\text{co}}(\psi + \chi_C)$.

Finding the convex hull of a nonconvex function is in general very difficult, except for those of concave functions over bounded polyhedral convex sets (polytopes). One seeks instead some convex relaxations more tractable to compute lower bounds for the optimal value $\alpha$ as by DC relaxations presented below.

### *4.1.2 Convex hull of concave functions over bounded polydedral convex sets*

Let $K$ be a nonempty bounded polyhedral convex set whose vertex set is $V(K) := \{v^1, \ldots, v^m\}$. Then $K = \text{co } V(K)$. The vertices $v^1, \ldots, v^m$ are said affinely independent if there are no real numbers $\lambda_i, i = 1, \ldots, m$ not all zero such that [15,40]

$$\sum_{i=1}^m \lambda_i = 0 \text{ and } \sum_{i=1}^m \lambda_i v^i = 0. \quad (31)$$

In this case $K$ is called an $(m-1)-$ simplex and every $x \in K$ is uniquely expressible as convex combination of $v^1, \ldots, v^m$. If $\psi$ is a finite concave on $K$ then the expression (27) for $\text{co}_K \psi$ becomes simpler and computable ([16,17,41,42])

**Theorem 11** *If $\psi$ is a finite concave on $K$, there hold*

(i) *$co_K\psi$ is the polyhedral convex function on $K$ defined by*

$$co_K\psi(x) = \min\left\{\sum_{i=1}^{m}\lambda_i\psi(v^i) : \lambda_i \geq 0, \sum_{i=1}^{m}\lambda_i = 1, x = \sum_{i=1}^{m}\lambda_i v^i\right\}. \tag{32}$$

*Moreover $co_K\psi$ and $\psi$ agree on $V(K)$.*

(ii) *If $K$ is an $(m-1)-$simplex, then $co_K\psi$ is the affine function determined by*

$$co_K\psi(x) = \sum_{i=1}^{m}\lambda_i\psi(v^i), \lambda_i \geq 0, \sum_{i=1}^{m}\lambda_i = 1, x = \sum_{i=1}^{m}\lambda_i v^i. \tag{33}$$

### *4.1.3 Convex hull of separable function*

Let $\psi = (\psi_1, \ldots, \psi_m)$ be a separable function on $C = \Pi_{i=1}^m C_i$ with $C_i \subset \text{dom}\,\psi_i \subset \mathbb{R}^{n_i}$, $i = 1, \ldots, m$, i.e.,

$$\psi(x) = \sum_{i=1}^{m}\psi_i(x_i), \quad \forall x = (x_1, \ldots, x_m) \in C, \tag{34}$$

then $\text{co}_C\psi$ can be computed explicitly from the $\text{co}_{C_i}\psi_i$, $i = 1, \ldots, m$.

**Proposition 12** *If for $i = 1, \ldots, m$, $\psi_i$ is minorized on $C_i$ by an affine function, then for every $x = (x_1, \ldots, x_m) \in K$*

$$co_C\psi(x) \geq \sum_{i=1}^{m} co_{C_i}\psi_i(x_i) \geq \sum_{i=1}^{m}\overline{co}_{C_i}\psi_i(x_i) = \sum_{i=1}^{m}(\psi_i + \chi_{C_i})^{**}(x_i) = \overline{co}_C\psi(x). \tag{35}$$

*Proof* By assumption $C \subset$ dom $\psi$ and $\psi$ is minorized by an affine function on $C$,then, as mentioned above, $\overline{\text{co}}_C\psi = (\psi + \chi_C)^{**}$ and $\overline{\text{co}}_{C_i}\psi_i(x_i) = (\psi_i + \chi_{C_i})^{**}(x_i)$ for $i = 1, .., m$ and $x_i \in C_i$. The first inequality is trivial because $\sum_{i=1}^m \text{co}_{C_i}\psi_i(x_i)$ is a convex minorization of $\psi$ over $C$. For the last equalities, it suffices to prove that $\sum_{i=1}^m(\psi_i + \chi_{C_i})^{**}(x_i) = (\psi + \chi_C)^{**}(x)$ for every $x = (x_1, \ldots, x_m) \in C = \Pi_{i=1}^m C_i$. We have for $y = (y_1, \ldots, y_m) \in \Pi_{i=1}^m \mathbb{R}^{n_i}$

$$\begin{aligned}(\psi + \chi_C)^*(y_1, \ldots, y_m) &= \sup\{\langle x, y\rangle - \psi(x) : x \in C\}\\ &= \sup\left\{\sum_{i=1}^{m}[\langle x_i, y_i\rangle - \psi_i(x_i)] : x = (x_1, \ldots, x_m) \in C = \Pi_{i=1}^m C_i\right\}\\ &= \sum_{i=1}^{m}\sup\{[\langle x_i, y_i\rangle - \psi_i(x_i)] : x_i \in C_i\} = \sum_{i=1}^{m}(\psi_i + \chi_{C_i})^*(y_i).\end{aligned}$$

It follows that for $x = (x_1, \ldots, x_m) \in \Pi_{i=1}^m \mathbb{R}^{n_i}$

$$(\psi + \chi_C)^{**}(x) = \sum_{i=1}^{m}(\psi_i + \chi_{C_i})^{**}(x_i).$$

□

### *4.1.4 Convex minorization of DC functions over bounded closed convex sets*

To build convex minorization for DC functions in global optimization, we are concerned with DC programs (4) with the explicit constraint $C$ (a nonempty bounded closed convex set in $\mathbb{R}^n$) and $\varphi, \psi \in \Gamma_0(\mathbb{R}^n)$ such that $C \subset \operatorname{dom} \varphi \subset \operatorname{dom} \psi$

$$\alpha = \inf\{\theta(x) := \varphi(x) - \psi(x) \ : \ x \in C\}. \tag{36}$$

According to the results displayed in Sects. 4.1.2 and 4.1.3, we propose the following computable convex minorizations of $\theta$ on $C$ :

(i) $\varphi +\operatorname{co}_C(-h)$ if $V(C)$ is easy to compute, for example in case $C$ is a bounded polyhedral convex set with known vertex set $V(C)$.
(ii) For the general case, $\operatorname{co}_C(-h)$ will be replaced with

+ $\overline{\operatorname{co}}_L(-h)$ where $L$ is a polytope containing $C$ defined in Sect. 4.1.3, ($L_i := [a_i, b_i]$ quite often in practice), if $h$ is separable,
or
+ $\operatorname{co}_S(-h)$ with $S$ being a simplex containing $C$.

Of course we must highlight suitable DC decompositions in order to use these convex minimization as shown just below.

### *4.1.5 Convex quadratic minimization for the binary quadratic program* (BQP)

Let us return to the binary quadratic program (BQP) and indicate how to apply the preceding results to computing minorizations convex by DC relaxation

$$\alpha := \min \ \left\{ f(x) = \frac{1}{2}x^T Qx + q^T x : x \in F = S \cap \{0, 1\}^n \right\}, \tag{37}$$

Computing a lower bound for the optimal value $\alpha$ of (37 ) passes by finding a practical convex quadratic function on $K := \{x \in \mathbb{R}^n : x \in S, x \in [0, 1]^n\}$ which agrees with the objective function $f$ on $F$. The problem of minimizing such a convex quadratic function over $K$ will provide lower bound for $\alpha$.

According to Sect. 4.1.4, we will work with

$$L := \Pi_{i=1}^m L_i = [0, 1]^n, \tag{38}$$

and build a DC decomposition of $f = g_\mu - h_\mu$ on $L$ such that $h_\mu$ be separable with respect to $L_i, i = 1, \ldots, n$. The following seems to be quite suitable to quadratic functions ([21,22, 24,25,42])

$$g_\mu(x) = \frac{1}{2}x^T(Q - \mu I_n)x + q^T x \quad \text{and}$$
$$h_\mu(x) = -\frac{1}{2}\mu x^T x = \sum_{i=1}^{m} \left(h_\mu\right)_i (x_i) = \sum_{i=1}^{m} \left(-\frac{1}{2}\mu\right) x_i^2, \tag{39}$$

with $\mu \leq \min\{0, \lambda_1(Q)\}$. It gives rise to the convex minorization $\varphi_\mu$ of $f$ on $L$ as the sum of $\frac{1}{2}x^T(Q - \mu I_n)x + q^T x$ and $\overline{\operatorname{co}}_L(-h_\mu)$. From Theorem 11 and Proposition 35, it follows

$$\overline{\operatorname{co}}_{L_i}(-h_\mu)_i(x_i) = \frac{1}{2}\mu x_i, \quad \forall i = 1, \ldots, n$$

and

$$\overline{\text{co}}_L(-h_\mu)(x) = \sum_{i=1}^{m} \overline{\text{co}}_{L_i}(-h_\mu)_i(x_i) = \frac{1}{2}\mu \sum_{i=1}^{m} x_i = \frac{1}{2}\mu x^T e^{(n)},$$

where $e^{(n)} \in \mathbb{R}^n$ being the vector of ones. Finally we have

$$\varphi_\mu(x) := \frac{1}{2}x^T(Q - \mu I_n)x + q^T x + \frac{1}{2}\mu x^T e^{(n)} = f(x) + \frac{1}{2}\mu x^T \left[e^{(n)} - x\right]. \quad (40)$$

This convex quadratic underestimation of $f$ has been used in our DC programming and DCA for solving nonconvex quadratic programs ([21,22,24,25,42]).

As mentioned in Sect. 3, thanks to the binary structure of the variable $x$, the concave quadratic function $p_2(x) = \sum_{i=1}^{n} x_i(1 - x_i) = x^T[e^{(n)} - x]$ can be used as exact penalty function for (BQP). The fact that $p_2$ is equal to zero on$\{0, 1\}^n$ suggests the next convex quadratic minorization $\psi_\mu$ of $f$ on $F$ ([21,22,24,25,42])

$$\psi_\mu(x) := f(x) + \frac{1}{2}\mu x^T \left[e^{(n)} - x\right] = \frac{1}{2}x^T(Q - \mu I_n)x + q^T x + \frac{1}{2}\mu x^T e^{(n)}, \quad (41)$$

where $\mu$ is chosen such that the matrix $Q - \mu I_n$ be positive semidefinite, i.e., $\mu \le \lambda_1(Q)$.

The function $\psi_\mu$ agrees with $f$ on $\{0, 1\}^n$(so a fortiori on $F$). Hence (37) can be equivalently written as

$$\alpha := \min\{\psi_\mu(x) : Ax \le b, x \in \{0, 1\}^n\} \quad (42)$$

and the optimal value of the convex quadratic program

$$\beta_\mu := \min\{\psi_\mu(x) : Ax \le b, x \in [0, 1]^n\} \quad (43)$$

is a lower bound for $\alpha$. It is clear that the greatest lower bound $\beta_\mu$ corresponds to $\mu = \lambda_1(Q)$ :

$$\beta_\mu \le \beta_{\lambda_1(Q)} := \min\left\{\psi_{\lambda_1(Q)}(x) := \frac{1}{2}x^T(Q - \lambda_1(Q)I_n)x + q^T x + \frac{1}{2}\lambda_1(Q)x^T e^{(n)} : Ax \le b, x \in [0, 1]^n\right\}, \text{ for all } \mu \le \lambda_1(Q). \quad (44)$$

The lower bound (43) will be used for lower bounding in the combination DCA-B&B-DC relaxation (Sect. 4.5). As for branching procedure (Sect. 4.3), it will be based on another equivalent DC program of (BQP) in the continuous framework, $(p = p_1, p_2)$ :

$$\nu(\tau) := \min\{\psi_\mu(x) + \tau p(x) : Ax \le b, x \in [0, 1]^n\}. \quad (45)$$

Indeed, since the convex quadratic function $\psi_\mu$ is a DC function then, according to exact penalty techniques in DC programming ([29] ), the nonsmooth DC program (45) is equivalent to (42) in the sense given in Theorem 1: there is $\tau_3 \ge 0$ such that, for every $\tau > \tau_3$,(42) and (45) have the same optimal value and the same optimal solution set. On the other hand, since the concave functions $p_1, p_2$ are separable functions on $[0, 1]^n$, it follows from Sect. 4.1.3 that, for $p = p_1, p_2$

$$\overline{\text{co}}_{[0,1]^n}\, p(x) = 0, \quad \forall x \in [0, 1]^n. \quad (46)$$

Hence, in virtue of the DC relaxation and Sect. 4.1.3, $\psi_\mu$ is also the convex quadratic underestimator, (of the DC objective function of (45)), provided by the procedure (ii) in Sect. 4.1.4, i.e., $\psi_\mu + \overline{\text{co}}_{[0,1]^n}\, p = \psi_\mu$.

*Remark 13* (i) The convex quadratic minorization $\varphi_\mu$ defined by DC relaxation for nonconvex quadratic programs (here applied to (37)) coincides with the convex quadratic underestimation $\psi_\mu$ when $\lambda_1(Q) \leq 0$.

(ii) It is worth noting that, in case the quadratic function $f$ is convex, the function $\varphi_{\lambda_1(Q)}$ is greater than $f$ on $[0, 1]^n$ and then provides a lower bound greater than that one given by the objective function $f$ itself:

$$\beta_{\lambda_1(Q)} \geq \min\{f(x) : Ax \leq b, x \in [0, 1]^n\}. \tag{47}$$

(iii) In [29,32,34] and refrences therein, we investigated the general DC relaxation using the form $Q - \text{Diag}(d)$ with $d \in \mathbb{R}^n$ such that $Q - \text{Diag}(d)$ be positive semidefinite. It leads to solving related SDP problems to construct maximal convex quadratic underestimations for nonconvex quadratic programs that we will outline in the next Sect. 4.2.

## 4.2 SDP relaxation technique for lower bounding in B&B

The second lower bounding relies upon SDP relaxation technique, that seems to be quite relevant to the structure of Problem (BQP).

We first summarize the standard SDP relaxation techniques for the binary quadratic programming ([12,44] and references therein) and recapitulate then related works by Billonnet-Elloumi [5] , and by Le Thi -Pham Dinh -Nguyen Canh [32,34].

### *4.2.1 Standard SDP relaxation technique*

The standard SDP relaxation techniques are known to provide tighter bounds than quadratic convex relaxations displayed above but more effort is required to compute them.

The working space then is the vector space $\mathcal{S}_n(\mathbb{R})$ of real symmetric $n \times n$ matrices, which is equipped with the standard dot-product of $\mathbb{R}^{n\times n}$ :

$$A \bullet B := Tr(AB) = \sum_{i,j=1}^{n} a_{ij} b_{ij} \tag{48}$$

where $A = (a_{ij})$ and $B = (b_{ij})$. Throughout the paper, Diag$(x)$ (resp. diag$(X)$) denotes the diagonal matrix of order $n$(resp. the vector in $\mathbb{R}^n$) whose $i$th entry is $x_i$(resp. $x_{ii}$) for every $x \in \mathbb{R}^n$(resp. $X = (x_{ij}) \in \mathbb{R}^{n\times n}$.

For $X = xx^T$, we have $\theta(X) := \frac{1}{2} Q \bullet X + \text{Diag}(q) \bullet X = f(x)$. Hence a semidefinite relaxation can be constructed by replacing $xx^T$ with a semidefinite matrix $X$ whose diagonal elements are in $[0, 1]$. However, this relaxation turns out to be of poor quality. In order to arrive at a better relaxation, we observe that

$$x \in \{0, 1\}^n \Leftrightarrow \begin{cases} X = xx^T \\ \text{diag}(X) = x \end{cases} .$$

Hence

$$X = \text{diag}(X)\,\text{diag}(X)^T \Leftrightarrow X - \text{diag}(X)\,\text{diag}(X)^T = 0.$$

so we will relax the condition $X - \text{diag}(X)\,\text{diag}(X)^T = 0$ by

$$X - \text{diag}(X)\,\text{diag}(X)^T \succeq 0. \tag{49}$$

Finally by Schur Complement [12,44] we have

$$\bar{X} := \begin{bmatrix} 1 & \operatorname{diag}(X)^T \\ \operatorname{diag}(X) & X \end{bmatrix} \succeq 0$$

The crucial step in the design of a semidefinite relaxation is the representation of the linear constraint within $\mathcal{S}_n(\mathbb{R})$. However, the existence of its matrix representation depends on itself ( [12,44] and references therein).

Consider the constraint, $a^T x \leq b$. The first approach is very natural, we model the constraint on the diagonal (because $x_i = x_i^2$): Stabdar

$$\langle \operatorname{Diag}(a), X \rangle \leq b \tag{50}$$

If $|a^T x| \leq b$ we can then square both sides of the original constraint, as a consequence, we obtain

$$\langle aa^T, X \rangle \leq b^2 \tag{51}$$

Another representation appears when we multiply both sides of the constraint by $a^T x$ when $a^T x \geq 0$.

$$0 \leq a^T x(b - a^T x) \Leftrightarrow \left\langle \begin{pmatrix} b \\ -a \end{pmatrix} \begin{pmatrix} 0 \\ a \end{pmatrix}^T, \bar{X} \right\rangle \geq 0 \tag{52}$$

The last approach may be interpreted as a particular case of a more general technique. In 0-1 programming, Lovasz and Schrijver introduced a system of constraints by multiplying the original constraints by $x_i \geq 0$ and $1 - x_i \geq 0$ for each $x_i$. It leads to the system of constraints:

$$\begin{cases} \sum_{j=1}^{n} a_j X_{ij} \leq b X_{ii} \\ \sum_{j=1}^{n} a_j (X_{jj} - X_{ij}) \leq b(1 - X_{ii}) \end{cases} \quad i = 1, 2, \ldots, n \tag{53}$$

The following result shows the relation between (50), ( 51), (52) and (53).

**Lemma 14** ([12]) *Let $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$ and $\mathcal{X}_4$ be the feasible sets defined by* (50), (51), (52), and (53) *jointly with*

$$X - \operatorname{diag}(X) \operatorname{diag}(X)^T \succeq 0$$

*respectively. Then $\mathcal{X}_1 \supseteq \mathcal{X}_2 \supseteq \mathcal{X}_3 \supseteq \mathcal{X}_4$*

*Remark 15* (i) Suppose that $X^*$ is an optimal solution obtained by a SDP relaxation, then $\operatorname{diag}(X^*) \in [0, 1]^n$ according to (49). For the combination DCA-B&B-SDP described in Sect. 4.5, $\operatorname{diag}(X^*)$ will be used as solution of the relaxed convex quadratic program for the branching procedure as well as initial point for DCA.

(ii) It is useful to recall that for the unconstrained binary quadratic program (UBQP), the relaxed (SDP) is uniquely defined and gives quite often tighter bounds than convex quadratic relaxations but more effort is required to compute them.

### *4.2.2 Maximal convex quadratic underestimation in nonconvex quadratic programming*

We developed in [32,34] the following convex quadratic minorization of $f(x) = \frac{1}{2}x^T Qx + q^T x$ on $[a, b] := \Pi_{i=1}^m [a_i, b_i]$, where $a = (a_i)$ and $b = (b_i)$ are in $\mathbb{R}^n$ such that $a < b$, i.e., $a_i < b_i$, for $i = 1, \dots, n$ :

$$\begin{aligned}\Phi_d(x) &:= f(x) + \frac{1}{2}\sum_{i=1}^{n} d_i(x_i - a_i)(b_i - x_i)\\ &= \frac{1}{2}x^T[Q - \mathrm{Diag}(d)]x + q^T x + \frac{1}{2}\sum_{i=1}^{n} d_i[(a_i + b_i)x_i - a_i b_i],\end{aligned} \tag{54}$$

with $d \in \mathbb{R}^n$ such that $Q-\mathrm{Diag}(d)$ be positive semidefinite, to compute lower bounds for the optimal value $\alpha$ of the nonconvex quadratic program

$$\alpha := \min\left\{ f(x) = \frac{1}{2}x^T Qx + q^T x : x \in F = S \cap [a, b]\right\}, \tag{55}$$

where $S \subset \mathbb{R}^n$ is a polyhedral convex set. As pointed out in 4.1.3, the box $[a, b]$ will be used to ease computation of convex quadratic underestimation of $f$ as well as for branching procedure in B&B scheme. We proved that [32,34]

(a) Every convex quadratic underestimation of $f$ on $[a, b]$ which agrees with $f$ on $\Pi_{i=1}^m \{a_i, b_i\}$ is of the form $\Phi_d$ with $d \le 0$.
(b) The maximal $\Phi_{d^*}$ correspond to the solutions $d^*$ of the linear multiobjective program with SDP constraint

$$\max\{d : d \le 0,\ Q - \mathrm{Diag}(d) \succeq 0\}. \tag{56}$$

Recall that a feasible point $d^*$ is a solution of (56) if for every feasible point $d$ such that $d^* \le d$ we have $d^* = d$. Problem (56) is equivalent to the usual mathematical program (with scalar objective function) [31]

$$0 = \min\{\sigma(d) : d \le 0,\ Q - \mathrm{Diag}(d) \succeq 0\}, \tag{57}$$

where the scalar objective function $\sigma$ is defined by

$$\sigma(d) := \max\left\{(\delta - d)^T e^{(n)} : d \le \delta,\ Q - \mathrm{Diag}(\delta) \succeq 0\right\}. \tag{58}$$

It is clear that $\sigma(d) \ge 0$ for feasible to (56) and the solution sets of (57) and (56) are identical. Instead of tackling the difficult nonconvex program (57), we solved the SDP program

$$\max\left\{\sum_{i=1}^{n} d_i : d \le 0,\ Q - \mathrm{Diag}(d) \succeq 0\right\}, \tag{59}$$

whose solution set is contained in that of (57). In the particular case where (56) has the maximum solution $d^*$, i.e., $d^* \ge d$ for all feasible points $d$ of (56), the singleton $\{d^*\}$ is exactly the solution set of both problems (56) and (59).

*Remark 16* (i) Instead of the box constraint $[a, b] = \Pi_{i=1}^m [a_i, b_i]$, if we are concerned with the combinatorial optimization problem

$$\alpha := \min\left\{ f(x) = \frac{1}{2}x^T Qx + q^T x : x \in F = S \cap \Pi_{i=1}^m \{a_i, b_i\}\right\}, \tag{60}$$

then we will use the strictly concave quadratic function $\Psi_u$ which has the same form as $\Phi_u$ in (59):

$$\begin{aligned}\Psi_u(x) &:= f(x) + \frac{1}{2}\sum_{i=1}^{n} u_i(x_i - a_i)(b_i - x_i) \\ &= \frac{1}{2}x^T[Q - \mathrm{Diag}(u)]x + q^T x + \frac{1}{2}\sum_{i=1}^{n} u_i[(a_i + b_i)x_i - a_i b_i], \end{aligned} \qquad (61)$$

except the fact that here the vector $u$ does make the matrix $Q{-}\mathrm{Diag}(d)$ negative definite. As before we consider then the problem

$$\eta_u := \min\{\Psi_u(x) : x \in \Pi_{i=1}^{m}\{a_i, b_i\}\}, \qquad (62)$$

which is equivalent to the concave quadratic program

$$\eta_u := \min\{\Psi_u(x) : x \in \Pi_{i=1}^{m}[a_i, b_i]\}. \qquad (63)$$

For $\Psi_u$ the corresponding $\Phi_d$ in (54) takes the form

$$\begin{aligned}\Phi_{d+u}(x) &:= f(x) + \frac{1}{2}\sum_{i=1}^{n}(d_i + u_i)(x_i - a_i)(b_i - x_i) \\ &= \frac{1}{2}x^T[Q - \mathrm{Diag}(d+u)]x + q^T x \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}(d_i + u_i)[(a_i + b_i)x_i - a_i b_i] \end{aligned} \qquad (64)$$

with $d \le 0$ and $Q{-}\mathrm{Diag}(d+u) \succeq 0$.

(ii) A simpler way for constructing such convex quadratic minorizations of $f$ to compute lower bounds for the optimal value $\alpha$ of (60) is to work directly with the feasible set $F = S \cap \Pi_{i=1}^{m}\{a_i, b_i\}$ and use the function $\Phi_d$ in (54) with only the condition $Q{-}\mathrm{Diag}(d) \succeq 0$. We need not the condition $d \le 0$ in (a) of Sect. (4.2.2). By linear relaxation of the constraint $x \in \Pi_{i=1}^{m}\{a_i, b_i\}$,this convex quadratic underestimation $\Phi_d$ yields the following lower bound for $\alpha$

$$\beta(d) := \min\left\{\Phi_d(x) : x \in S \cap \Pi_{i=1}^{m}[a_i, b_i]\right\}. \qquad (65)$$

Note that (ii) encompasses (i) because (64) is a special case of $\Phi_d$.

### 4.2.3 Maximum lower bound via SDP programming

In addition to the use of the well known DC relaxation involving the smallest eigenvalue of the matrix $Q$, which is outlined in Sect. 4.1.5, the work by Billonnet-Elloumi in [5] is mainly devoted to the computation of the maximum lower bound (given by the quadratic function $\Phi_d$ defined in (ii), 16 for $a_i = 0, b_i = 1, i = 1, \ldots, n$ ) for unconstrained binary quadratic programs (UBQP), a particular case of (BQP) where the feasible set $F$ is simply $\{0, 1\}^n$ :

$$\alpha := \min\left\{f(x) = \frac{1}{2}x^T Qx + q^T x : x \in \{0, 1\}^n\right\}. \qquad \text{(UBQP)}$$

More precisely they considered the following convex quadratic function

$$\Theta_d(x) := f(x) + \frac{1}{2}\sum_{i=1}^{n} d_i x_i(1-x_i) = \frac{1}{2}x^T[Q - \mathrm{Diag}(d)]x + q^T x + \frac{1}{2}\sum_{i=1}^{n} d_i x_i \quad (66)$$

with $Q-\mathrm{Diag}(d) \succeq 0$. It induces the following lower bound for the optimal value $\alpha$

$$\beta(d) := \min\{\Theta_d(x) : x \in [0,1]^n\}. \quad (67)$$

Then finding the greatest lower bound amounts to solve the optimization problem with SDP constraint

$$\beta^* := \beta(d^*) := \max\{\beta_d(x) : d \in \mathbb{R}^n, Q - \mathrm{Diag}(d) \succeq 0\}. \quad (68)$$

The search for an equivalent formulation of SDP of a nonconvex program, if it exists, is one of the most difficult tasks in programming SDP. Fortunately, the problem (68) can be recast as a SDP problem, according to Poljak-Rendl-Wolkowicz [39] and Lemaréchal-Oustry [33]. More precisely $(\beta^* = r^*, d^*)$ is the optimal solution of the SDP problem

$$\begin{aligned} &\beta^* := \min r \\ &\begin{bmatrix} -r & \frac{1}{2}\left(q + \frac{1}{2}d\right)^T \\ \frac{1}{2}\left(q+\frac{1}{2}d\right) & \frac{1}{2}\left[Q - \mathrm{Diag}(d)\right] \end{bmatrix} \succeq 0 \\ &r \in \mathbb{R}, d \in \mathbb{R}^n \end{aligned} \quad \text{(SDP)}$$

whose dual is exactly the SDP problem given by the standard SDP relaxation above displayed in Sect. 4.2.1

$$\begin{aligned} \beta^* := \min\ & \frac{1}{2}Q \bullet X + \mathrm{Diag}(q) \bullet \mathrm{Diag}\ (x) \\ & \mathrm{diag}(X) = x \\ & \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \\ & x \in \mathbb{R}^n, X \in \mathcal{S}_n(\mathbb{R}) \end{aligned} \quad \text{(DSDP)}$$

Their preprocessing phase consits then of computing $(\beta^* = r^*, d^*)$ by applying an existing SDP code to the dual (DSDP) and solving the convex quadratic program (68) to get its optimal solution $x^*$. Finally, as regards the related solution method proposed by Billonnet-Elloumi [5] to unconstrained binary quadratic programs (UBQP), the preprocessing phase is followed by the exact solution phase with the MIQP solver of CPLEX 8.1 (ILOG 2002). In our combined DCA-B&B-SDP for solving (UBQP ), the optimal solution $(X^{**}, x^{**})$ of (DSDP) provides lower bounds and initial points $x^{**}$ for DCA.

## 4.3 Branching procedure using binary subdivision

Thanks to exact penalty techniques, we obtain equivalent DC programs in the continuous framework which make possible the application of the local continuous approach DCA. More precisely, DCA is applied to the first penalty equivalent DC program (24), and the second one (45), related to the exact penalty function $p_1(x) = \sum_{j=1}^{n}(p_1)_j(x_j) = \sum_{j=1}^{n}\min\{x_j, 1-x_j\}$, is used in the branching procedure of our B&B. We turn the binary character (of the variable in the binary quadratic program (BQP)) to advantage in replacing the bisection of a chosen $k$th edge $[0, 1]$ with the binary one: $x_k = 0$ and $x_k = 1$. The procedure is detailed in the following:

Suppose that, at a node in our B&B scheme, we have to solve

$$\min \quad f(x) = \frac{1}{2}x^T Qx + q^T x \tag{SP}$$

subject to $\quad x \in S$

$$x_i = 0 \quad i \in I, \quad x_i = 1 \quad i \in J$$
$$x_i \in \{0, 1\} \quad i \in \{1, 2, \ldots\} \setminus (I \cup J)$$

Let $x^{RP}$ be an optimal solution of a relaxed problem of (SP) (defined by a relaxation technique)

If $x^{RP} \in \{0, 1\}^n$, i.e., $p_1(x^{RP}) = 0$, then $x^{RP}$ is also an optimal solution of (SP). Otherwise there exists $j^*$ such that $x^{RP}_{j^*} \in (0, 1)$. Then we replace problem (SP) with two subproblems by setting $x^{RP}_{j^*} = 0$ and $x^{RP}_{j^*} = 1$, respectively.

The index $j^*$ is chosen such that the gaps between $(p_1)_j(x^{RP}_j) = \min\{x^{RP}_j, 1 - x^{RP}_j\}$ and its convex hull on [0, 1]- which is identical to zero- is maximum with respect to $j = 1, \ldots n$, i.e.,

$$\max_j \{\min\{x^{RP}_j, 1 - x^{RP}_j\}\} = \min\{x^{RP}_{j^*}, 1 - x^{RP}_{j^*}\} \tag{69}$$

Of course, we can also use instead the exact penalty function $p_2$. Note that the criterion (69) was used in branching procedure, related to normal rectangular subdivisions, of B&B for continuous nonconvex programming ([41,42]),

## 4.4 Improving lower bounding by updating the convex quadratic function $\psi_\mu$ defined by (41)

As specified in Sect. 4.1.5 devoted to DC relaxation using the smallest eigenvalue $\lambda_1(Q)$, the first lower bounding procedure in our B&B-DC relaxation relies on the lower bound $\beta_{\lambda_1(Q)}$ given in (44). In the second one, instead of the same matrix $Q$, we will work with the principal submatrices of $Q$, updated at each branching iteration (binary subdivision), in order to increase the lower bounds. For that, we explain below how to proceed.

Our B&B begins by solving the convex quadratic program (44). If the optimal solution $x \in \{0, 1\}^n$, then $x$ is an optimal solution of (BQP) too.

Othewise, choose $k \in \{i = 1, .., n : x_i \neq 0, 1\}$ by (69) and make a binary subdivision by setting $x_k = 0$ and $x_k = 1$. We then have to compute two lower bounds for

$$\min\left\{f(x) := \frac{1}{2}x^T Qx + q^T x : Ax \le b, x \in \{0, 1\}^n, x_k = 0\right\} \quad (\mathrm{P}_0)$$

and

$$\min\left\{f(x) := \frac{1}{2}x^T Qx + q^T x : Ax \le b, x \in \{0, 1\}^n, x_k = 1\right\} \quad (\mathrm{P}_1).$$

From the results displayed in 4.1.5, one can use the following lower bounds $\zeta_0$ and $\zeta_1$ for the optimal values of (P$_0$) and (P$_1$), respectively:

$$\zeta_0 := \min\left\{ f_{\lambda_1(Q)}(x) := \frac{1}{2}x^T(Q - \lambda_1(Q)I_n)x + q^T x \right.$$
$$\left. + \frac{1}{2}\lambda_1(Q)x^T e^{(n)} : Ax \le b, x \in [0,1]^n, x_k = 0 \right\}, \tag{70}$$
$$\zeta_1 := \min\left\{ f_{\lambda_1(Q)}(x) := \frac{1}{2}x^T(Q - \lambda_1(Q)I_n)x + q^T x \right.$$
$$\left. + \frac{1}{2}\lambda_1(Q)x^T e^{(n)} : Ax \le b, x \in [0,1]^n, x_k = 1 \right\}, \tag{71}$$

and the procedure is repeated until convergence.

But, to turn the binary character of the variable in (BQP) to advantage, we introduce the related lower bounds $\beta_0$ and $\beta_1$and prove that

$$\beta_0 \ge \zeta_0, \beta_1 \ge \zeta_1 \tag{72}$$

Set $K := \{1,\ldots,n\}\diagdown\{k\}$. Let $A^K$,(resp.$Q^K$) be the submatrix of $A$ (resp. $Q$) of dimension $m \times |K|$ (resp. $n \times |K|$), made up of the columns $A^k$ (resp. $Q^k$), $k \in K$. Likewise, for $J \subset \{1,\ldots,n\}$, $Q_J^K$ denotes the submatrix of $Q$, of dimension $|J| \times |K|$, which is comprised of the entries $q_{ij}$ with $(i,j) \in J \times K$.

By simple matrix computations, we can show that

$$\frac{1}{2}x^T Qx = \frac{1}{2}x_K^T Q_K^K x_K + x_k Q_k^K x_K + \frac{1}{2}x_k^2 q_{kk} \tag{73}$$

where $x_K$ is the subvector of $x$, comprised of the components $x_i$, $i \in K$. Then, it follows, from Sect. 4.1, that the lower bounds $\beta_0$ and $\beta_1$ can be given by

$$\beta_0 := \min\left\{ \frac{1}{2}x_K^T\left(Q_K^K - \lambda_1\left(Q_K^K\right)(I_n)_K^K\right)x_K + q_K^T x_K \right.$$
$$\left. + \frac{1}{2}\lambda_1\left(Q_K^K\right)x_K^T e_K^{(n)} : A^K x_K \le b, x_K \in [0,1]^K \right\} \tag{74}$$
$$\beta_1 := \min\left\{ \frac{1}{2}x_K^T\left(Q_K^K - \lambda_1\left(Q_K^K\right)I_K^K\right)x_K + \frac{1}{2}\lambda_1\left(Q_K^K\right)x_K^T e_K^{(n)} + Q_k^K x_K + q_K^T x_K \right.$$
$$\left. + \frac{1}{2}q_{kk} + q_K^T x_K : A^K x_K \le b - A^k, x_K \in [0,1]^K \right\}. \tag{75}$$

To prove (72), we have to express the lower bounds $\zeta_0$ and $\zeta_1$ in terms of the variable $x_K$. By using (73), we get

$$\zeta_0 = \min\left\{ \frac{1}{2}x_K^T\left(Q_K^K - \lambda_1(Q)(I_n)_K^K\right)x_K + q_K^T x_K \right.$$
$$\left. + \frac{1}{2}\lambda_1(Q)x_K^T e_K^{(n)} : A^K x_K \le b, x_K \in [0,1]^K \right\} \tag{76}$$

and

$$\zeta_1 := \min\left\{ \frac{1}{2}x_K^T Q_K^K x_K + Q_k^K x_K + \frac{1}{2}q_{kk} + q_k + q_K^T x_K \right.$$
$$\left. + \frac{1}{2}\lambda_1(Q)x_K^T\left(e_K^{(n)} - x_K\right) : A^K x_K \le b - A^k, x_K \in [0,1]^K \right\}. \tag{77}$$

The formulations (74), (75), (76) and (77) allow us to deduce the next result.

**Theorem 17** (i) *The difference between the objective functions in* (74) *(resp.* (75)*) and* (76) *(resp.* (77)*) is*

$$\frac{1}{2}\left[\lambda_1\left(Q_K^K\right)-\lambda_1(Q)\right]x_K^T\left[e_K^{(n)}-x_K\right],$$

*which is nonnegative on* $[0,1]^K$.

(ii) *The new LB,* $\beta_0$ *and* $\beta_1$*, for the optimal values of* $(P_0)$ *and* $(P_1)$*, are greater than the old LB* $\zeta_0$ *and* $\zeta_1$

$$\beta_0 \geq \zeta_0, \beta_1 \geq \zeta_1.$$

*Proof* It suffices to prove that $\lambda_1(Q_K^K)\geq\lambda_1(Q)$ by using the well- known variational properties that eigenvalues of symmetric real matrices enjoy [15], in particular

$$\lambda_1(Q)=\min\left\{\frac{x^TQx}{x^Tx}:x\in\mathbb{R}^n\setminus\{0\}\right\}.$$

Since

$$\min\left\{\frac{x^TQx}{x^Tx}:x\in\mathbb{R}^n\setminus\{0\}\right\}\leq\min\left\{\frac{x^TQx}{x^Tx}:x=(x_K,x_k)\in\mathbb{R}^K\setminus\{0\},x_k=0\right\},$$

and according to (73), the righ hand side of the above inequality is equal to

$$\min\left\{\frac{x_K^TQ_K^Kx_K}{x_K^Tx_K}:x_K\in\mathbb{R}^K\setminus\{0\}\right\},$$

i.e., $\lambda_1\left(Q_K^K\right)$. □

*Remark 18* In the sequel, the B&B scheme using the lower bounds $\zeta's$ is denoted B&B-old, while the B&B-new is related to the lower bounds $\beta's$

## 4.5 The combined DCA and branch-and-bound algorithm

The purpose of combination of DCA with B&B techniques using convex quadratic underestimations (DC relaxation)/SDP relaxation is twofold:

(i) To check globality of solutions computed by DCA, and to find better initial solutions for restarting DCA if need be.
(ii) To improve upper bounds, with the help of DCA, in B&B and thus accelerate its convergence.

We are now in a position to describe the combined algorithm for computing a global optimal solution of problem (BQP).

**Algorithm 2** Let $R_0 := [0,1]^n$.

Solve one of the relaxed convex programs of (BQP) generated by the DC relaxation or SDP relaxation to obtain an optimal solution $x^{R_0}$ and the first lower bound $\beta_0 := \beta(R_0)$.

Apply DCA to the (24) from $x^{R_0}$ to compute an initial point $u^{R_0}$. Apply DCA to the equivalent DC program (16) from the starting point $u^{R_0}$ to obtain $x_\tau^{R_0}$.

```
If $x_\tau^{R_0}$ is feasible to (BQP), then set upper bound $\gamma := \frac{1}{2}(x_\tau^{R_0})^T Q x_\tau^{R_0} + q^T x_\tau^{R_0}$ and set $x^* := x_\tau^{R_0}$.
Otherwise set $\gamma := +\infty$.
**If** $\gamma = \beta_0$ **then** $x^*$ **is an optimal solution of** (BQP).
**else** set $\mathcal{R} \leftarrow \{R_0\}, k \leftarrow 0$.
**While** TRUE **do**
  Select a rectangle $R_k$ such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}$.
  Bisect $R_k$ into two subrectangles $R_{k_0}$ and $R_{k_1}$ via the index $j^*$, chosen as (69), by branching procedure

      $R_{k_i} = \{x \in R_k : x_{j*} = i\} \quad i = 0, 1$

  Set $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma, i = 0, 1\} \setminus R_k$
  Solve the relaxed problem of $(P_{k_i})$ to obtain $\beta(R_{k_i})$ and $x^{R_{k_i}}$:

      $(P_{k_i}) \qquad \min\left\{\frac{1}{2}x^T Q x + q^T x : x \in K, \quad x \in R_{k_i}\right\} \quad (i = 0, 1).$

  **If** $x^{R_{k_i}}$ is feasible to (BQP), and $\gamma_k := \frac{1}{2}(x^{R_{k_i}})^T Q x^{R_{k_i}} + q^T x^{R_{k_i}} < \gamma$ **then**
    Update upper bound $\gamma = \gamma_k$ and solution $x^* = x^{R_{k_i}}$
  **End if**
  **If** the condition of restarting DCA (25) is satisfied **then**
    Apply DCA to (16) from $x^{R_{k_i}}$ to obtain $x_\tau^{R_{k_i}}$.
    **If** $x_\tau^{R_{k_i}}$ is feasible to (BQP), and $\gamma_k := \frac{1}{2}(x_\tau^{R_{k_i}})^T Q x_\tau^{R_{k_i}} + q^T x_\tau^{R_{k_i}} < \gamma$ **then**
      Update upper bound $\gamma = \gamma_k$ and solution $x^* = x_\tau^{R_{k_i}}$
    **End if**
  **End if**
  Update the list of rectangles $\mathcal{R} \leftarrow \mathcal{R} \setminus \{R_j : \beta(R_j) > \gamma\}$
  **If** $(\mathcal{R} = \emptyset)$ or $(\gamma = \min\{\beta(R) : R \in \mathcal{R}\})$ **then**
    STOP, $x^*$ **is an optimal solution**
  **else** $k \leftarrow k + 1$.
**End while**
```

The correctness and the convergence of the algorithm are stated in the following result whose proof is fairly standard from the branching procedure and the bounding one [41,42]. Its finiteness is due to the binary subdivision used in the algorithm.

**Proposition 19** *Algorithm 2 terminates after finitely many iterations and yielding an optimal solution of Problem* (BQP).

## 5 Computational results

This section presents the computational results provided by our algorithms on several sets of test problems. The algorithm has been coded in a C program using CPLEX 7.5 as LP and Convex Quadratic solver [18] and the callable library function SDPA6.0 for solving SDP problem [7]. All computational results have been obtained by implementing our program under Window XP on a 2GH PC portable. We use the efficient library code in C++

(http://www.robertnz.net/VisualC.html) for computing smallest and largest eigenvalues of symmetric matrices under consideration. We specify the following options in our codes:

(i) DCA is applied to the penalty equivalent DC program (16).
(ii) Lower bounding via DC relaxation uses the lower bound $\beta_{\lambda_1(Q)}$ given in (44). The resulting global algorithm is denoted DCA-B&B-old (Remark 18).
(iii) The global algorithm DCA-B&B-new corresponds to updated lower bounds at each iteration (binary subdivision) of B&B, according to 41 (Remark 18).
(iv) Standard SDP relaxation for lower bounds in the global algorithm DCA-B&B-SDP is described in 4.2.1. Here the SDP relaxation (52) is used because it gives best lower bounds within identical execution times.

The test problems are under the form of maximization that we transform into minimization before applying our algorithms. Numerical results are reconverted for the former ones: upper bounds (for maximization) are the opposites of lower bounds (for minimization) and vice versa. The following notations are used:

- #Iter : Number of iterations in the combined DCA-B&B
- LB/Val : Objective value at the solution computed by DCA
- UB : Upper bound obtained by the corresponding relaxation or Opt (optimal value known)
- #DCA: Number of restarting DCA
- $\epsilon$ : The relative error (UB-LB)/(|LB| + 1)

**1)** The first test problem is the unconstrained binary quadratic program (UBQP). The first instances are the Beasley instances. These data sets are due to [4]. The second instances are due to [8]. All data set can be obtained from the OR-Library [3] as well as from [43]. Note that the problems are given as maximization problems.

Finding an exact optimal solution is somehow impossible for large dimension problems. In fact, we always try to find out an $\epsilon$-optimal solution (the value of $\epsilon$ is often equal to 0.05 and sometimes equal to 0.03, depending on the goal of the problem). Algorithms proposed in [4,8] are the heuristics ones. As pointed out in the introduction and Sect. 4.2.3, the work on (UBQP) in Billonnet-Elloumi [5] deals with maximum lower bound via SDP(SDP), which is nothing but the lower bound computed by the standard SDP relaxation (DSDP) in Poljak-Rendl-Wolkowicz [39]. Not surprisingly that, by using this lower bound and $x^*$ or $x^{**}$ (defined in Sect. 4.2.3) in our DCA-B&B-SDP, we obtained similar computational results. We reported three types of combination, the DCA-B&B-old ($\mu$ fixed), the DCA-B&B-new ($\mu$ recalculated at each step of binary subdivision), (see Remark 18), and the DCA-B&B-SDP. The quality of our computed solutions can be seen by the results given in [43]. In all cases, we limit the number of iterations to 1,00,000.

It is also interesting to estimate the quality of solutions given by DCA with respect to the optimal values. *DCA can work with a problem in any dimension and, in practice, gives a very good solution.* It means that the computed solutions are often $\epsilon$-optimal solutions (even exact optimal ones in many cases). We use the sign ** to denote the instances in which exact optimal solutions are found. The sign * denotes the ones in which, with DCA, we obtain an $\epsilon$-optimal solution such that $\epsilon \leq 0.01$ at the first launching time. This is a very important remark. Indeed, when handling large scale real-life problems where computing an exact global solution is somehow impossible, finding such a solution is always a challenge. While, for almost current solvers, the dimensions of these problems have exceeded their limitation.

We presented the computational results for data in [4] in Tables 1 and 2 where in the former the number of variable ($n$) is equal to 50 and in the latter it is equal to 100. The density $d$ of matrix $Q$ is equal to 0.1 in all the instances.

**Table 1** [4] Data, $n = 50$, $d = 0.1$. Numerical results

| Prob | Opt | $\epsilon \leq 0.03$ | | | | $\epsilon \leq 0.05$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | **DCA - B&B old** | | | | | | | |
| bqp50-1 | 2,098 | – | 3133.88 | 2,064 | 1 | 69,777 | 1534.94 | 2,064 | 1 |
| bqp50-2 | 3,702 | 9,628 | 178.94 | **3,702 | 4 | 3,708 | 56.20 | **3,702 | 4 |
| bqp50-3 | 4,626 | 1,048 | 18.59 | **4,626 | 1 | 344 | 4.71 | **4,626 | 1 |
| bqp50-4 | 3,544 | 54,041 | 1332.39 | **3,544 | 3 | 41,372 | 737.53 | 3,400 | 2 |
| bqp50-5 | 4,012 | 45,076 | 1090.86 | **4,012 | 2 | 18,896 | 314.39 | 3,888 | 1 |
| bqp50-6 | 3,693 | 7,376 | 127.27 | **3,693 | 2 | 3,232 | 39.42 | **3,693 | 2 |
| bqp50-7 | 4,520 | 4,819 | 84.03 | *4,510 | 1 | 1,552 | 20.44 | *4,510 | 1 |
| bqp50-8 | 4,216 | 8,144 | 153.50 | 4,112 | 1 | 2,586 | 33.73 | 4,112 | 1 |
| bqp50-9 | 3,780 | 49,523 | 1214.29 | 3,732 | 1 | 20,891 | 318.77 | 3,732 | 1 |
| bqp50-10 | 3,507 | 80,150 | 2298.33 | 3,467 | 1 | 28,799 | 459.97 | 3,467 | 1 |
| | | **DCA - B&B new** | | | | | | | |
| bqp50-1 | 2,098 | 51,752 | 1231.24 | 2,064 | 1 | 27,491 | 428.50 | 2,064 | 1 |
| bqp50-2 | 3,702 | 6,656 | 124.97 | **3,702 | 4 | 2,692 | 41.63 | **3,702 | 4 |
| bqp50-3 | 4,626 | 817 | 15.16 | **4,626 | 1 | 287 | 4.42 | **4,626 | 1 |
| bqp50-4 | 3,544 | 22,271 | 444.09 | **3,544 | 3 | 19,810 | 294.00 | 3,400 | 2 |
| bqp50-5 | 4,012 | 13,881 | 256.97 | **4,012 | 2 | 5,320 | 78.72 | 3,888 | 1 |
| bqp50-6 | 3,693 | 5,908 | 107.23 | **3,693 | 2 | 2,658 | 35.23 | **3,693 | 2 |
| bqp50-7 | 4,520 | 1,313 | 23.83 | *4,510 | 1 | 423 | 6.20 | **4,510 | 1 |
| bqp50-8 | 4,216 | 2,972 | 54.56 | 4,112 | 1 | 992 | 13.95 | 4,112 | 1 |
| bqp50-9 | 3,780 | 14,583 | 263.33 | 3,732 | 1 | 6,344 | 86.00 | 3,732 | 1 |
| bqp50-10 | 3,507 | 44,657 | 1038.28 | 3,467 | 1 | 17,100 | 247.9 | 3,467 | 1 |
| Prob | Opt | $\epsilon \leq 0.03$ | | | | $\epsilon \leq 0.05$ | | | |
| | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | **DCA - B&B - SDP** | | | | | | | |
| bqp50-1 | 2,098 | 224 | 44.13 | 2,040 | 1 | 91 | 19.36 | 2,040 | 1 |
| bqp50-2 | 3,702 | 1 | 0.23 | **3,702 | 1 | 1 | 0.23 | **3,702 | 1 |
| bqp50-3 | 4,626 | 1 | 0.17 | **4,626 | 1 | 1 | 0.17 | **4,626 | 1 |
| bqp50-4 | 3,544 | 23 | 5.58 | **3,544 | 3 | 8 | 2.02 | 3,406 | 1 |
| bqp50-5 | 4,012 | 2 | 0.50 | *3,992 | 1 | 1 | 0.20 | *3,992 | 1 |
| bqp50-6 | 3,693 | 1 | 0.19 | **3,693 | 1 | 1 | 0.19 | **3,693 | 1 |
| bqp50-7 | 4,520 | 4 | 1.03 | 4,470 | 1 | 1 | 0.19 | 4,470 | 1 |
| bqp50-8 | 4,216 | 4 | 1.06 | *4,192 | 1 | 1 | 0.19 | *4,192 | 1 |
| bqp50-9 | 3,780 | 55 | 11.97 | 3,696 | 1 | 10 | 2.48 | 3,696 | 1 |
| bqp50-10 | 3,507 | 164 | 33.06 | 3,427 | 1 | 18 | 4.17 | 3,427 | 1 |

– : Stop after 1,00,000 iterations.

**Table 2** [4] Data, $n = 100$, $d = 0.1$. Numerical results

| Prob | Opt | $\epsilon \le 0.03$ | | | | $\epsilon \le 0.05$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | **DCA - B&B old** | | | | | | | |
| bqp100-1 | 7,970 | – | 5139.69 | 7,664 | 1 | – | 5139.69 | 7,664 | 1 |
| bqp100-2 | 11,036 | – | 5202.88 | 10,878 | 1 | – | 5202.88 | 10,878 | 1 |
| bqp100-3 | 12,723 | – | 5363.94 | *12,661 | 1 | – | 5363.94 | *12,661 | 1 |
| bqp100-4 | 10,368 | – | 5300.30 | *10,320 | 1 | – | 5300.30 | *10,320 | 1 |
| bqp100-5 | 9,083 | – | 5161.06 | 8,813 | 1 | – | 5161.06 | 8,813 | 1 |
| bqp100-6 | 10,210 | – | 5243.00 | 9,976 | 1 | – | 5243.00 | 9,976 | 1 |
| bqp100-7 | 10,125 | – | 5094.41 | 9,914 | 1 | – | 5094.41 | 9,914 | 1 |
| bqp100-8 | 11,435 | – | 5288.42 | *11,363 | 1 | – | 5288.42 | *11,363 | 1 |
| bqp100-9 | 11,455 | – | 5640.03 | *11,383 | 1 | 51,884 | 2381.03 | *11,383 | 1 |
| bqp100-10 | 12,565 | – | 5265.88 | 12,403 | 1 | – | 5265.88 | 12,403 | 1 |
| | | **DCA - B&B new** | | | | | | | |
| bqp100-1 | 7,970 | – | 5816.73 | 7,664 | 1 | – | 5816.73 | 7,664 | 1 |
| bqp100-2 | 11,036 | – | 5847.94 | 10,878 | 1 | – | 5847.94 | 10,878 | 1 |
| bqp100-3 | 12,723 | – | 5872.25 | *12,661 | 1 | – | 5872.25 | *12,661 | 1 |
| bqp100-4 | 10,368 | – | 5923.97 | *10,320 | 1 | – | 5923.97 | *10,320 | 1 |
| bqp100-5 | 9,083 | – | 5890.36 | 8,813 | 1 | – | 5890.36 | 8,813 | 1 |
| bqp100-6 | 10,210 | – | 5942.47 | 9,976 | 1 | – | 5942.47 | 9,976 | 1 |
| bqp100-7 | 10,125 | – | 5829.58 | 9,914 | 1 | – | 5829.58 | 9,914 | 1 |
| bqp100-8 | 11,435 | – | 5932.19 | *11,363 | 1 | – | 5932.19 | *11,363 | 1 |
| bqp100-9 | 11,455 | – | 6357.38 | *11,383 | 1 | 18,724 | 919.53 | *11,383 | 1 |
| bqp100-10 | 12,565 | – | 6310.83 | 12,403 | 2 | 53,602 | 2809.58 | 12,403 | 1 |
| | | **DCA - B&B - SDP** | | | | | | | |
| bqp100-1 | 7,970 | 2,556 | 3706.17 | 7,788 | 1 | 299 | 434.70 | 7,788 | 1 |
| bqp100-2 | 11,036 | 445 | 708.22 | 10,952 | 1 | 23 | 34.97 | 10,952 | 1 |
| bqp100-3 | 12,723 | 13 | 23.30 | *12,705 | 1 | 1 | 1.03 | *12,705 | 1 |
| bqp100-4 | 10,368 | 38 | 65.59 | *10,320 | 1 | 3 | 4.63 | *10,320 | 1 |
| bqp100-5 | 9,083 | 831 | 1284.94 | 8871 | 1 | 82 | 126.69 | 8,871 | 1 |
| bqp100-6 | 10,210 | 1,047 | 1592.11 | *10,127 | 1 | 79 | 120.05 | *10,127 | 1 |
| bqp100-7 | 10,125 | 9,673 | 13254.81 | 9,884 | 1 | 534 | 755.20 | 9,884 | 1 |
| bqp100-8 | 11,435 | 88 | 150.05 | *11,363 | 1 | 4 | 5.98 | *11,363 | 1 |
| bqp100-9 | 11,455 | 42 | 73.42 | 11,249 | 1 | 3 | 4.38 | 11,249 | 1 |
| bqp100-10 | 12,565 | 83 | 141.53 | 12,433 | 1 | 4 | 6.00 | 12,433 | 1 |

– : Stop after 1,00,000 iterations.

Numerical experiences for data in [8] and [5] are shown in Tables 3 and 4, respectively. For both series test problems we first once again tried to get an $\epsilon$-solution with $\epsilon \le 0.05$ and after that we decreased $\epsilon$ to 0.03.

**2)** The second test problem is the Maximum Clique Problem (MCP). One of the most important problems in combinatorial optimization. It can be modeled as a unconstrained binary quadratic program [17]. Computational results with three combinations on

**Table 3** DCA in B&B method with different relaxation techniques, data [8]

| Prob | $n$ | $d$ | Opt | $\epsilon \leq 0.03$ | | | | $\epsilon \leq 0.05$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | | | **DCA - B&B - old** | | | | | | | |
| gka1a | 50 | 0.1 | 3,414 | – | 3202.88 | 3,353 | 1 | – | 3202.88 | 3,353 | 1 |
| gka2a | 60 | 0.1 | 6,063 | 1,634 | 35.38 | $^{**}$6,063 | 1 | 395 | 8.69 | $^{**}$6,063 | 1 |
| gka3a | 70 | 0.1 | 6,037 | – | 3930.50 | $^{*}$5,859 | 1 | – | 3930.50 | $^{*}$5,859 | 1 |
| gka4a | 80 | 0.1 | 8,598 | – | 4428.31 | $^{*}$8,554 | 1 | 20,851 | 691.33 | $^{*}$8,554 | 1 |
| gka5a | 50 | 0.2 | 5,737 | 28,703 | 706.89 | 5,584 | 1 | 10,568 | 231.80 | 5,584 | 1 |
| gka6a | 30 | 0.4 | 3,980 | 55 | 0.92 | $^{*}$3,973 | 1 | 24 | 0.45 | $^{*}$3,973 | 1 |
| gka7a | 30 | 0.5 | 4,541 | 113 | 1.88 | $^{**}$4,541 | 1 | 60 | 1.03 | $^{**}$4,541 | 1 |
| gka8a | 100 | 0.0625 | 11,109 | – | 4996.34 | $^{*}$11,051 | 1 | – | 4946.34 | $^{*}$11,051 | 1 |
| gka1b | 20 | 1.0 | 133 | 185 | 2.66 | $^{**}$133 | 4 | 185 | 2.66 | $^{**}$133 | 4 |
| gka2b | 30 | 1.0 | 121 | 1,167 | 21.52 | $^{**}$121 | 2 | 1,167 | 21.52 | $^{**}$121 | 2 |
| gka3b | 40 | 1.0 | 118 | 4,997 | 129.38 | $^{**}$118 | 2 | 4,997 | 129.38 | $^{**}$118 | 1 |
| gka4b | 50 | 1.0 | 129 | 12,374 | 422.77 | $^{**}$129 | 2 | 12,374 | 414.87 | $^{**}$129 | 2 |
| gka5b | 60 | 1.0 | 150 | 414.87 | 1902.11 | $^{**}$150 | 2 | 4,1487 | 1902.11 | $^{**}$150 | 2 |
| gka1c | 40 | 0.8 | 5,058 | 15 | 0.42 | $^{**}$5,058 | 1 | 4 | 0.16 | $^{**}$5,058 | 1 |
| gka2c | 50 | 0.6 | 6,213 | 261 | 7.73 | $^{*}$6,151 | 1 | 60 | 1.91 | $^{*}$6,151 | 1 |
| gka3c | 60 | 0.4 | 6,665 | 564 | 17.73 | $^{*}$6,648 | 1 | 116 | 3.83 | $^{*}$6,648 | 1 |
| | | | | **DCA - B&B - new** | | | | | | | |
| gka1a | 50 | 0.1 | 3,414 | – | 3227.34 | 3,353 | 1 | 58,318 | 1534.06 | 3,353 | 1 |
| gka2a | 60 | 0.1 | 6,063 | 1,215 | 28.09 | $^{**}$6,063 | 1 | 316 | 7.73 | $^{**}$6,063 | 1 |
| gka3a | 70 | 0.1 | 6,037 | – | 4202.94 | $^{*}$5,859 | 1 | – | 4202.94 | $^{*}$5,859 | 1 |
| gka4a | 80 | 0.1 | 8,598 | 3,130 | 1142.09 | $^{*}$8,554 | 1 | 6,035 | 207.72 | $^{*}$8,554 | 1 |
| gka5a | 50 | 0.2 | 5,737 | 15,555 | 335.08 | 5,584 | 1 | 5,846 | 119.53 | 5,584 | 1 |
| gka6a | 30 | 0.4 | 3,980 | 31 | 0.59 | $^{*}$3,973 | 1 | 17 | 0.36 | $^{*}$3,973 | 1 |
| gka7a | 30 | 0.5 | 4,541 | 54 | 0.92 | $^{**}$4,541 | 1 | 31 | 0.56 | $^{**}$4,541 | 1 |
| gka8a | 100 | 0.0625 | 11,109 | – | 5958.44 | $^{*}$11,051 | 1 | – | 5, 958.44 | $^{*}$11,051 | 1 |
| gka1b | 20 | 1.0 | 133 | 84 | 1.08 | $^{**}$133 | 2 | 83 | 1.08 | $^{**}$133 | 2 |
| gka2b | 30 | 1.0 | 121 | 488 | 7.45 | $^{**}$121 | 4 | 488 | 7.45 | $^{**}$121 | 4 |
| gka3b | 40 | 1.0 | 118 | 1,349 | 25.94 | $^{**}$118 | 2 | 1,349 | 25.94 | $^{**}$118 | 2 |
| gka4b | 50 | 1.0 | 129 | 3,781 | 91.39 | $^{**}$129 | 3 | 3,781 | 89.77 | $^{**}$129 | 3 |
| gka5b | 60 | 1.0 | 150 | 8,836 | 279.92 | $^{**}$150 | 2 | 8,836 | 275.78 | $^{**}$150 | 2 |
| gka1c | 40 | 0.8 | 5,058 | 12 | 0.34 | $^{**}$5,058 | 1 | 4 | 0.16 | $^{**}$5,058 | 1 |
| gka2c | 50 | 0.6 | 6,213 | 171 | 4.89 | $^{*}$6,151 | 1 | 38 | 1.27 | $^{*}$6,151 | 1 |
| gka3c | 60 | 0.4 | 6,665 | 380 | 12.05 | $^{*}$6,648 | 1 | 86 | 3.02 | $^{*}$6,648 | 1 |
| | | | | **DCA - B&B - SDP** | | | | | | | |
| gka1a | 50 | 0.1 | 3,414 | 17 | 4.25 | $^{*}$3,380 | 1 | 3 | 0.83 | $^{*}$3,380 | 1 |
| gka2a | 60 | 0.1 | 6,063 | 1 | 0.36 | $^{**}$6,063 | 1 | 1 | 0.36 | $^{**}$6,063 | 1 |
| gka3a | 70 | 0.1 | 6,037 | 216 | 116.27 | $^{*}$5,992 | 1 | 26 | 16.06 | $^{*}$5,992 | 1 |
| gka4a | 80 | 0.1 | 8,598 | 5 | 4.61 | $^{*}$8,543 | 1 | 1 | 0.70 | $^{*}$8,543 | 1 |
| gka5a | 50 | 0.2 | 5,737 | 52 | 11.80 | $^{*}$5,730 | 2 | 8 | 2.13 | $^{*}$5,730 | 1 |

**Table 3** Continued

| Prob | $n$ | $d$ | Opt | $\epsilon \leq 0.03$ | | | | $\epsilon \leq 0.05$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | | | **DCA - B&B - SDP** | | | | | | | |
| gka6a | 30 | 0.4 | 3,980 | 4 | 0.36 | *3,973 | 1 | 2 | 0.23 | *3,973 | 1 |
| gka7a | 30 | 0.5 | 4,541 | 2 | 0.22 | **4,541 | 1 | 1 | 0.11 | **4,541 | 1 |
| gka8a | 100 | 0.0625 | 11,109 | 1 | 1.16 | *11,051 | 1 | 1 | 1.16 | *11,051 | 1 |
| gka1b | 20 | 1.0 | 133 | 37 | 1.80 | **133 | 3 | 37 | 1.80 | **133 | 3 |
| gka2b | 30 | 1.0 | 121 | 231 | 14.11 | **121 | 4 | 231 | 14.14 | **121 | 4 |
| gka3b | 40 | 1.0 | 118 | 514 | 48.66 | **118 | 3 | 514 | 48.66 | **118 | 3 |
| gka4b | 50 | 1.0 | 129 | 1,113 | 175.50 | **129 | 4 | 1,113 | 175.50 | **129 | 4 |
| gka5b | 60 | 1.0 | 150 | 2,035 | 547.88 | **150 | 3 | 2,035 | 547.88 | **150 | 3 |
| gka1c | 40 | 0.8 | 5,058 | 1 | 0.14 | **5,058 | 1 | 1 | 0.14 | **5,058 | 1 |
| gka2c | 50 | 0.6 | 6,213 | 2 | 0.56 | *6,151 | 1 | 1 | 0.27 | *6,151 | 1 |
| gka3c | 60 | 0.4 | 6,665 | 1 | 0.36 | *6,648 | 1 | 1 | 0.36 | *6,648 | 1 |

– : Stop after 1,00,000 iterations.

**Table 4** DCA in B&B method with different relaxation techniques, data [8]

| Prob | $n$ | $d$ | Opt | $\epsilon \leq 0.03$ | | | | $\epsilon \leq 0.05$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | | | **DCA - B&B - old** | | | | | | | |
| be100.1 | 100 | 1 | 19,412 | – | 11954.31 | *19,299 | 1 | 86,583 | 11696.88 | *19,299 | 1 |
| be100.2 | 100 | 1 | 17,290 | – | 12020.16 | 17,076 | 1 | 31,492 | 4945.48 | 17,076 | 1 |
| be100.3 | 100 | 1 | 17,565 | – | 12147.87 | 17,348 | 1 | 54,495 | 9912.92 | 17,348 | 1 |
| be100.4 | 100 | 1 | 19,125 | – | 11632.45 | *19,036 | 1 | 24,586 | 8394.08 | *19,036 | 1 |
| be100.5 | 100 | 1 | 15,868 | – | 13298.67 | 15,538 | 1 | – | 13298.67 | 15,538 | 1 |
| be100.6 | 100 | 1 | 17,368 | – | 12949.38 | 16,958 | 1 | – | 12949.38 | 16,958 | 1 |
| be100.7 | 100 | 1 | 18,629 | – | 12309.89 | 18,308 | 1 | – | 12309.89 | 18,308 | 1 |
| be100.8 | 100 | 1 | 18,649 | – | 12296.95 | 18,081 | 1 | – | 12296.95 | 18,081 | 1 |
| be100.9 | 100 | 1 | 13,294 | – | 12462.45 | 12,651 | 1 | – | 12462.45 | 12,651 | 1 |
| be100.10 | 100 | 1 | 15,352 | – | 12317.91 | 14,690 | 1 | – | 12317.91 | 14,690 | 1 |
| | | | | **DCA - B&B - new** | | | | | | | |
| be100.1 | 100 | 1 | 19,412 | – | 11874.24 | *19,299 | 1 | 48,671 | 9128.31 | *19,299 | 1 |
| be100.2 | 100 | 1 | 17,290 | – | 11560.56 | 17,076 | 1 | 14,574 | 2663.78 | 17,076 | 1 |
| be100.3 | 100 | 1 | 17,565 | 88,412 | 11495.56 | 17,348 | 1 | 11,717 | 2183.02 | 17,348 | 1 |
| be100.4 | 100 | 1 | 19,125 | – | 11980.28 | *19,036 | 1 | 11,487 | 2178.56 | *19,036 | 1 |
| be100.5 | 100 | 1 | 15,868 | – | 11940.14 | 15,538 | 1 | – | 11940.14 | 15,538 | 1 |
| be100.6 | 100 | 1 | 17,368 | – | 11830.09 | 16,958 | 1 | – | 11830.09 | 16,958 | 1 |
| be100.7 | 100 | 1 | 18,629 | – | 11306.61 | 18,038 | 1 | – | 11306.61 | 18,038 | 1 |
| be100.8 | 100 | 1 | 18,649 | – | 11247.66 | 18,081 | 1 | – | 11247.66 | 18,081 | 1 |
| be100.9 | 100 | 1 | 13,294 | – | 11341.42 | 12,651 | 1 | – | 11341.42 | 12,651 | 1 |
| be100.10 | 100 | 1 | 15,352 | – | 11921.03 | 14,690 | 1 | – | 11921.03 | 14,690 | 1 |

**Table 4** Continued

| Prob | $n$ | $d$ | Opt | $\epsilon \leq 0.03$ | | | | $\epsilon \leq 0.05$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #Iter | t(s) | Val | #DCA | #Iter | t(s) | Val | #DCA |
| | | | | **DCA - B&B - SDP** | | | | | | | |
| be100.1 | 100 | 1 | 19,412 | 48 | 122.22 | *19,338 | 1 | 2 | 4.64 | *19,338 | 1 |
| be100.2 | 100 | 1 | 17,290 | 226 | 538.59 | *17,161 | 1 | 15 | 39.67 | *17,161 | 1 |
| be100.3 | 100 | 1 | 17,565 | 424 | 980.34 | 17,312 | 1 | 26 | 68.13 | 17,312 | 1 |
| be100.4 | 100 | 1 | 19,125 | 96 | 238.16 | *19,007 | 1 | 4 | 10.48 | *19,007 | 1 |
| be100.5 | 100 | 1 | 15,868 | 1,639 | 3559.95 | 15,674 | 1 | 162 | 392.08 | 15,674 | 1 |
| be100.6 | 100 | 1 | 17,368 | 460 | 1060.22 | 17,144 | 1 | 35 | 90.24 | 17,144 | 1 |
| be100.7 | 100 | 1 | 18,629 | 1,754 | 3711.80 | 18,348 | 1 | 135 | 322.55 | 18,348 | 1 |
| be100.8 | 100 | 1 | 18,649 | 49,233 | 71798.14 | **18,649 | 2 | 8,589 | 16844.78 | 17,983 | 1 |
| be100.9 | 100 | 1 | 13,294 | 13,085 | 21208.31 | 13,095 | 2 | 7,285 | 14389.28 | 12,719 | 1 |
| be100.10 | 100 | 1 | 15,352 | 15,196 | 26897.54 | **15,352 | 2 | 4,687 | 9896.78 | 14,759 | 1 |

– : Stop after 1,00,000 iterations.

various sizes ($n \leq 200$) of problems are provided in Table 5. We found exact optimal solutions for this type of problem.

**3)** The last test problem is the 0-1 Quadratic Knapsack problem (QK01). In this case, there is only one linear constraint, see [12].

We tested 40 instances provided by OR-Library [3] with dimension 100. The different densities of matrix $Q$ are 25, 50, 75 and 100%. We will denote the various instances as 100.D.O where D is the density expressed as percentage and O is the order**.** The penalty parameter $\tau$ in our numerical simulations takes the value $10^5$. The notation "1st DCA" denotes the values given by DCA at the first iteration where a binary solution is found. Our algorithms stop either an $\epsilon$-optimal solution, ($\epsilon \leq 0.05$), is found or after 1,00,000 iterations.

We reported the computational experiments in Tables 6 and 7 with the same notations as in the unconstrained binary quadratic program.

From the results we can observe the efficiency of DCA and the combined DCA-B&B ("old" and "new"). In all cases, DCA gives a very good binary solution, most of them are $\epsilon$-optimal solutions of (BQP) (with $\epsilon \leq 1\%$ or even exact optimal one) since the number

**Table 5** Numerical results of maximum clique problem

| Prob | Nodes | Edges | Clique size | #Iter | UB | LB | #DCA | Time(s) |
|---|---|---|---|---|---|---|---|---|
| | | | | **B&B - DCA - old** | | | | |
| johnson8-2-4 | 28 | 210 | 4 | 4 | 4 | 4 | 2 | 0.06 |
| MANN_a9 | 45 | 918 | 16 | – | 17 | 16 | 2 | 2380.30 |
| hamming6-2 | 64 | 1,824 | 32 | – | 34 | 32 | 1 | 3023.56 |
| hamming6-4 | 64 | 704 | 4 | 1,912 | 4 | 4 | 1 | 70.45 |
| johnson8-4-4 | 70 | 1,855 | 14 | 37,807 | 14 | 14 | 4 | 1099.88 |
| c-fat200-1 | 200 | 1,534 | 12 | 38,332 | 12 | 12 | 1 | 11248.30 |

**Table 5** Continued

| Prob | Nodes | Edges | Clique size | #Iter | UB | LB | #DCA | Time(s) |
|---|---|---|---|---|---|---|---|---|
| | | | | **DCA - B&B new** | | | | |
| johnson8-2-4 | 28 | 210 | 4 | 4 | 4 | 4 | 2 | 0.05 |
| MANN_a9 | 45 | 918 | 16 | 38,623 | 16 | 16 | 2 | 504.42 |
| hamming6-2 | 64 | 1,824 | 32 | 1,9174 | 32 | 32 | 1 | 240.88 |
| hamming6-4 | 64 | 704 | 4 | 148 | 4 | 4 | 1 | 1.81 |
| johnson8-4-4 | 70 | 1,855 | 14 | 4,565 | 14 | 14 | 1 | 49.69 |
| c-fat200-1 | 200 | 1,534 | 12 | 1,589 | 12 | 12 | 1 | 35.09 |
| | | | | **B&B - DCA - SDP** | | | | |
| johnson8-2-4 | 28 | 210 | 4 | 4 | 4 | 4 | 2 | 0.18 |
| MANN_a9 | 45 | 918 | 16 | 1,588 | 16 | 16 | 3 | 143.77 |
| hamming6-2 | 64 | 1,824 | 32 | 8,372 | 32 | 32 | 1 | 1076.64 |
| hamming6-4 | 64 | 704 | 4 | 59 | 4 | 4 | 2 | 7.84 |
| johnson8-4-4 | 70 | 1,855 | 14 | 1,430 | 14 | 14 | 2 | 149.61 |
| c-fat200-1 | 200 | 1,534 | 12 | 205 | 12 | 12 | 1 | 644.09 |

– : Stop after 1,00,000 iterations.

**Table 6** Numerical results of 0-1 Quadratic knapsack problem—part 1

| Prob | #Iter | UB | LB | $\varepsilon$ | #DCA | 1st DCA | Time (s) |
|---|---|---|---|---|---|---|---|
| | **DCA - B&B old** | | | | | | |
| 100.25.1 | – | 21,521 | 12,560 | 0.42 | 1 | 12,560 | 12147.48 |
| 100.25.2 | 79 | 58,148 | 55,412 | 0.05 | 156 | 55,412 | 36.22 |
| 100.25.3 | – | 5,565 | 872 | 0.84 | 1 | 872 | 12565.06 |
| 100.25.4 | – | 52,423 | 48,535 | 0.07 | 1 | 48,535 | 12155.25 |
| 100.25.5 | 5 | 62,727 | 59,639 | 0.05 | 1 | 63,062 | 0.63 |
| 100.25.6 | – | 38,610 | 35,360 | 0.08 | 1 | 35,360 | 12061.77 |
| 100.25.7 | – | 16,714 | 10,256 | 0.39 | 1 | 10,256 | 12247.25 |
| 100.25.8 | – | 22,615 | 15,852 | 0.30 | 1 | 15,852 | 11969.79 |
| 100.25.9 | – | 37,835 | 34,916 | 0.08 | 1 | 34,916 | 12071.30 |
| 100.25.10 | – | 27,945 | 23,145 | 0.17 | 1 | 23,145 | 12062.84 |
| | **DCA - B&B new** | | | | | | |
| 100.25.1 | – | 20,553 | 12,560 | 0.39 | 1 | 12,560 | 8740.16 |
| 100.25.2 | 71 | 58,056 | 55,412 | 0.05 | 140 | 55,412 | 31.67 |
| 100.25.3 | – | 4,993 | 872 | 0.83 | 1 | 872 | 8899.31 |
| 100.25.4 | – | 51,903 | 48,535 | 0.07 | 1 | 48,535 | 8825.56 |
| 100.25.5 | 5 | 62,717 | 59,639 | 0.05 | 1 | 63,062 | 0.59 |
| 100.25.6 | – | 37,747 | 35,360 | 0.06 | 1 | 35,360 | 8587.64 |
| 100.25.7 | – | 15,882 | 10,256 | 0.35 | 1 | 10,256 | 8781.89 |
| 100.25.8 | – | 21,709 | 15,852 | 0.27 | 1 | 15,852 | 8638.11 |
| 100.25.9 | – | 36,967 | 34,916 | 0.06 | 1 | 34,916 | 8514.67 |
| 100.25.10 | – | 26,772 | 23,145 | 0.14 | 1 | 23,145 | 8525.84 |

**Table 6** Continued

| Prob | #Iter | UB | LB | $\varepsilon$ | #DCA | 1st DCA | Time (s) |
|---|---|---|---|---|---|---|---|
| | **DCA - B&B - SDP** | | | | | | |
| 100.25.1 | 967 | 18,559 | 18,558 | 0.00 | 2 | 17,028 | 3364.77 |
| 100.25.2 | 3 | 56,576 | 56,320 | 0.01 | 3 | 56,320 | 9.08 |
| 100.25.3 | 256 | 3,752 | 3,752 | 0.00 | 2 | 3,020 | 1025.20 |
| 100.25.4 | 1 | 50,526 | 48,088 | 0.05 | 1 | 48,088 | 2.88 |
| 100.25.5 | 8 | 61,579 | 61,494 | 0.00 | 14 | 61,494 | 41.66 |
| 100.25.6 | 4 | 36,484 | 35,744 | 0.00 | 6 | 35,744 | 18.03 |
| 100.25.7 | 1 | 14,848 | 14,282 | 0.04 | 1 | 14,282 | 2.45 |
| 100.25.8 | 122 | 20,452 | 20,452 | 0.00 | 2 | 19,069 | 481.80 |
| 100.25.9 | 1 | 35,600 | 34,924 | 0.02 | 1 | 34,924 | 2.61 |
| 100.25.10 | 1 | 25,216 | 24,748 | 0.02 | 1 | 24,748 | 3.05 |
| | **DCA - B&B old** | | | | | | |
| 100.50.1 | – | 87,189 | 80,843 | 0.07 | 1 | 80,843 | 14329.17 |
| 100.50.2 | 20,845 | 1,07,931 | 1,02,535 | 0.05 | 1 | 1,02,535 | 2709.97 |
| 100.50.3 | – | 39,189 | 25,636 | 0.35 | 1 | 25,636 | 14270.22 |
| 100.50.4 | 721 | 1,08,979 | 1,03,532 | 0.05 | 1 | 1,03,532 | 92.88 |
| 100.50.5 | – | 61,322 | 53,165 | 0.13 | 1 | 53,165 | 14436.91 |
| 100.50.6 | – | 21,834 | 5,679 | 0.74 | 1 | 5,679 | 15348.38 |
| 100.50.7 | – | 58,127 | 44,763 | 0.23 | 1 | 44,763 | 13887.09 |
| 100.50.8 | – | 59,025 | 49,130 | 0.17 | 1 | 49,130 | 14303.06 |
| 100.50.9 | – | 74,434 | 67,930 | 0.09 | 1 | 67,930 | 14246.74 |
| 100.50.10 | – | 92,900 | 87,748 | 0.06 | 1 | 87,748 | 14118.11 |
| | **DCA - B&B new** | | | | | | |
| 100.50.1 | – | 85,567 | 80,843 | 0.06 | 1 | 80,843 | 11074.86 |
| 100.50.2 | 3,661 | 1,07,931 | 1,02,535 | 0.05 | 1 | 1,02,535 | 391.77 |
| 100.50.3 | – | 37,003 | 25,636 | 0.31 | 1 | 25,636 | 10842.88 |
| 100.50.4 | 278 | 1,08,980 | 1,03,532 | 0.05 | 1 | 1,03,532 | 32.05 |
| 100.50.5 | – | 59,189 | 53,165 | 0.10 | 1 | 53,165 | 11035.95 |
| 100.50.6 | – | 19,939 | 5,679 | 0.72 | 1 | 5,679 | 11722.67 |
| 100.50.7 | – | 55,871 | 44,763 | 0.22 | 1 | 44,763 | 11092.20 |
| 100.50.8 | – | 56,910 | 49,130 | 0.14 | 1 | 49,130 | 10951.94 |
| 100.50.9 | – | 72,259 | 67,930 | 0.06 | 1 | 67,930 | 10927.86 |
| 100.50.10 | 19,911 | 92,366 | 87,748 | 0.05 | 1 | 87,748 | 2025.42 |
| | **DCA- B&B - SDP** | | | | | | |
| 100.50.1 | 1 | 83,976 | 82,737 | 0.02 | 1 | 82,737 | 2.97 |
| 100.50.2 | 1 | 1,05,088 | 1,04,389 | 0.01 | 1 | 1,04,389 | 3.31 |
| 100.50.3 | 1 | 34,268 | 33,095 | 0.04 | 1 | 33,095 | 2.84 |
| 100.50.4 | 1 | 1,06,097 | 1,05,876 | 0.00 | 1 | 1,05,876 | 3.19 |
| 100.50.5 | 2 | 56,701 | 56,297 | 0.01 | 3 | 56,297 | 8.72 |
| 100.50.6 | 1 | 16,270 | 16,021 | 0.02 | 1 | 16,021 | 3.30 |
| 100.50.7 | 1 | 52,979 | 52,646 | 0.01 | 1 | 52,646 | 2.91 |
| 100.50.8 | 1 | 54,506 | 53,790 | 0.01 | 1 | 53,790 | 2.93 |

**Table 6** Continued

| Prob | #Iter | UB | LB | $\varepsilon$ | #DCA | 1st DCA | Time (s) |
|---|---|---|---|---|---|---|---|
| 100.50.9 | 1 | 69,035 | 68,974 | 0.00 | 1 | 68,974 | 2.86 |
| 100.50.10 | 2 | 89,011 | 88,045 | 0.01 | 2 | 88,045 | 8.44 |

– : Stop after 1,00,000 iterations.

**Table 7** Numerical results of 0-1 Quadratic knapsack problem—part 2

| Prob | #Iter | UB | LB | $\varepsilon$ | #DCA | 1st DCA | Time (s) |
|---|---|---|---|---|---|---|---|
| | **DCA - B&B old** | | | | | | |
| 100.75.1 | 1 | 1,90,435 | 1,88,190 | 0.01 | 1 | 1,88,190 | 0.25 |
| 100.75.2 | – | 1,05,802 | 80,815 | 0.24 | 1 | 80,815 | 16103.58 |
| 100.75.3 | – | 70,794 | 51,086 | 0.28 | 1 | 51,086 | 16405.92 |
| 100.75.4 | – | 8,19,570 | 47,771 | 0.42 | 1 | 47,771 | 15823.16 |
| 100.75.5 | – | 3,60,954 | 14,150 | 0.61 | 1 | 14,150 | 16691.56 |
| 100.75.6 | 1,004 | 1,51,813 | 1,44,227 | 0.05 | 1 | 1,44,227 | 147.94 |
| 100.75.7 | – | 1,20,412 | 1,09,814 | 0.09 | 1 | 1,09,814 | 16046.69 |
| 100.75.8 | – | 28,604 | 5,613 | 0.80 | 1 | 5,613 | 16697.94 |
| 100.75.9 | – | 1,13,465 | 1,00,040 | 0.12 | 1 | 1,00,040 | 16095.06 |
| 100.75.10 | 16,178 | 1,48,956 | 1,41,509 | 0.05 | 1 | 1,41,509 | 2392.69 |
| | **DCA - B&B new** | | | | | | |
| 100.75.1 | 1 | 1,90,435 | 1,88,190 | 0.01 | 1 | 1,88,190 | 0.25 |
| 100.75.2 | – | 1,02,318 | 80,815 | 0.21 | 1 | 80,815 | 13273.38 |
| 100.75.3 | – | 67,119 | 51,086 | 0.24 | 1 | 51,086 | 13332.73 |
| 100.75.4 | – | 78,513 | 47,771 | 0.39 | 1 | 47,771 | 12628.19 |
| 100.75.5 | – | 3,33,264 | 14,150 | 0.58 | 1 | 14,150 | 13475.13 |
| 100.75.6 | 268 | 1,51,797 | 1,44,227 | 0.05 | 1 | 1,44,227 | 37.94 |
| 100.75.7 | – | 1,17,181 | 1,09,814 | 0.06 | 1 | 1,09,814 | 13317.66 |
| 100.75.8 | – | 25,875 | 5,613 | 0.78 | 1 | 5,613 | 13457.06 |
| 100.75.9 | – | 1,10,114 | 1,00,040 | 0.09 | 1 | 1,00,040 | 13365.81 |
| 100.75.10 | 2,497 | 1,48,956 | 1,41,509 | 0.05 | 1 | 1,41,509 | 314.84 |
| | **DCA - B&B - SDP** | | | | | | |
| 100.75.1 | 2 | 1,89,137 | 1,89,137 | 0.00 | 1 | 1,89,137 | 3.14 |
| 100.75.2 | 1 | 95,515 | 93,658 | 0.02 | 1 | 93,658 | 2.89 |
| 100.75.3 | 1 | 62,285 | 62,046 | 0.00 | 1 | 62,046 | 3.41 |
| 100.75.4 | 1 | 72,777 | 71,226 | 0.02 | 1 | 71,226 | 3.16 |
| 100.75.5 | 1 | 27,903 | 27,233 | 0.03 | 1 | 27,233 | 3.14 |
| 100.75.6 | 1 | 1,45,448 | 1,44,227 | 0.01 | 1 | 1,44,227 | 2.83 |
| 100.75.7 | 1 | 1,11,357 | 1,09,625 | 0.01 | 1 | 1,09,625 | 3.17 |
| 100.75.8 | 34 | 19,573 | 19,525 | 0.00 | 2 | 18,345 | 154.42 |
| 100.75.9 | 1 | 1,04,901 | 1,02,663 | 0.02 | 1 | 1,02,663 | 3.09 |
| 100.75.10 | 2 | 1,43,956 | 1,42,617 | 0.01 | 3 | 1,42,617 | 8.61 |

**Table 7** Continued

| Prob | #Iter | UB | LB | ε | #DCA | 1st DCA | Time (s) |
|---|---|---|---|---|---|---|---|
| | **DCA - B&B old** | | | | | | |
| 100.100.1 | – | 89,718 | 68,522 | 0.24 | 1 | 68,522 | 18018.20 |
| 100.100.2 | 86,245 | 1,98,210 | 1,88,300 | 0.05 | 1 | 18,830 | 15128.28 |
| 100.100.3 | 1 | 2,35,006 | 2,23,990 | 0.05 | 1 | 2,23,990 | 0.27 |
| 100.100.4 | – | 81,957 | 47,771 | 0.42 | 1 | 47,771 | 16237.17 |
| 100.100.5 | 1 | 2,39,653 | 2,28,311 | 0.05 | 1 | 2,28,311 | 0.27 |
| 100.100.6 | – | 85,705 | 63,502 | 0.35 | 1 | 63,502 | 17481.24 |
| 100.100.7 | – | 86,306 | 56,152 | 0.35 | 1 | 56,152 | 17952.64 |
| 100.100.8 | – | 72,879 | 46,249 | 0.58 | 1 | 46,249 | 17450.33 |
| 100.100.9 | 18 | 2,39,550 | 2,31,191 | 0.04 | 34 | 2,31,191 | 26.05 |
| 100.100.10 | 4,372 | 2,01,660 | 1,91,578 | 0.05 | 68 | 1,91,578 | 733.67 |
| | **DCA - B&B new** | | | | | | |
| 100.100.1 | – | 86,149 | 68,522 | 0.21 | 1 | 68,522 | 14431.92 |
| 100.100.2 | 6,956 | 1,98,210 | 1,88,300 | 0.05 | 1 | 18,830 | 1050.69 |
| 100.100.3 | 1 | 2,35,006 | 2,23,990 | 0.05 | 1 | 2,23,990 | 0.27 |
| 100.100.4 | – | 78,513 | 47,771 | 0.39 | 1 | 47,771 | 13328.52 |
| 100.100.5 | 1 | 2,39,653 | 2,28,311 | 0.05 | 1 | 2,28,311 | 0.27 |
| 100.100.6 | – | 80,870 | 63,502 | 0.22 | 1 | 63,502 | 14539.05 |
| 100.100.7 | – | 81,632 | 56,152 | 0.31 | 1 | 56,152 | 15605.75 |
| 100.100.8 | – | 68,355 | 46,249 | 0.32 | 1 | 46,249 | 14697.13 |
| 100.100.9 | 27 | 2,38,707 | 2,31,191 | 0.03 | 52 | 2,31,191 | 38.28 |
| 100.100.10 | 844 | 2,01,660 | 1,91,578 | 0.05 | 147 | 1,91,578 | 191.52 |
| | **DCA - B&B - SDP** | | | | | | |
| 100.100.1 | 1 | 82,625 | 81,760 | 0.01 | 1 | 81,760 | 3.29 |
| 100.100.2 | 1 | 1,90,901 | 1,88,993 | 0.01 | 1 | 1,88,993 | 3.28 |
| 100.100.3 | 1 | 2,25,560 | 2,20,471 | 0.02 | 1 | 2,20,471 | 3.39 |
| 100.100.4 | 1 | 72,777 | 71,226 | 0.02 | 1 | 71,226 | 5.08 |
| 100.100.5 | 7 | 2,30,235 | 2,30,076 | 0.00 | 13 | 2,30,076 | 40.75 |
| 100.100.6 | 1 | 74,715 | 74,074 | 0.01 | 1 | 74,074 | 3.14 |
| 100.100.7 | 1 | 75,215 | 74,341 | 0.01 | 1 | 74,341 | 3.30 |
| 100.100.8 | 1 | 62,874 | 62,251 | 0.01 | 1 | 62,251 | 2.98 |
| 100.100.9 | 25 | 2,34,112 | 2,32,200 | 0.01 | 47 | 2,32,200 | 162.73 |
| 100.100.10 | 12 | 1,94,953 | 1,92,300 | 0.01 | 22 | 1,92,300 | 62.30 |

– : Stop after 1,00,000 iterations.

"#DCA" of restarting DCA is surprisingly small and equal to 1 in most cases. The efficiency of DCA can be clearly seen when we decrease the $\epsilon$ in the stopping criterion. We can see that our global algorithm could not stop since the lower bound (resp. upper bound for maximization problem) could not sufficiently approach the upper bound (resp. lower bound for maximization problem) given by DCA.

Due to SDP relaxation, the confirmation of the optimality of the solution given by DCA is quite faster than other relaxations. As for the combination DCA-B&B ("old" and "new") without SDP relaxation, their lower bounds increase too slowly to reach the relative error

5% before 1,00,000 iterations. So for many test problems (where the number of iterations is denoted by "–" in the tables) they failed to provide $\epsilon$-optimal solutions. It is worthwhile remarking that within these two combinations (without using SDP relaxation) DCA always gives $\epsilon$-optimal solutions after only one restarting, but neither DCA-B&B-old nor DCA-B&B-new are able to confirm this fact in the cases "–". We must have recourse to the combination DCA-B&B-SDP for it. By recalculating $\mu' s$ values during binary subdivisions, the combination DCA-B&B -new improved the efficiency of the algorithm and arrived at some solutions while the "old" failed. The increase of $\mu' s$ is important: in many cases, the maximum difference between $\mu' s$ is near 1,000.

**Comparison with existing algorithms**
As we have clearly specified in the introduction, our main objective is to prove that our local approach based on DC programming and DCA with good initial points is inexpensive and efficient, and can thus be applicable to large-scale (BQP). Indeed these problems (BQP) are NP-hard and no existing algorithms can be able to globally solve them in the large-scale setting. Its combination with B&B aims at

(i) Computing good initial points for DCA, and restarting the algorithm if a feasible (binary or not) solution of the penalized DC program (17) for (UBQP) or (16) for (BQP)), generated by the lower bounding process in B&B, is found with smaller objective value than that computed by DCA.
(ii) Checking globality of solutions computed by DCA. It is only possible with small sized (BQP). That is why the dimension of ( BQP) is limited to $n = 100$ for all test problems, except for the Maximum Clique Problem where $n$ is up to 200, to ensure the end of the global algorithms.

It is interesting to report here the comparative results (on the *unconstrained binary quadratic programs* (UBQP)), which were provided by different global algorithms based on B&B techniques (as Pardalos-Rodgers [35], Hansen-Jaumard-Meyer [11], Helmberg-Rendl [13] and Billonnet-Elloumi [5]) and summarized in the very recent work of Billonnet-Elloumi [5]. As indicated above, the global algorithms in [5] is the powerful commercial MIQP solver of CPLEX 8.1 (ILOG 2002) to which the authors incorporated the preprocessing phase. In DC relaxation (resp. SDP relaxation) this phase amounts to compute the smallest eigenvalue $\lambda_1(Q)$ of the matrix $Q$ and the solution of an associated convex quadratic program (resp. a SDP problem) for providing initial solutions to MIQP. The corresponding global algorithms will be denoted by MIQP* and MIQP**, respectively. The computational experiments in [5] concern a large set of instances of (UBQP), randomly generated according to [35] and benchmark instances introduced by Glover-Kochenberger-Alidaee and available in OR-Library by Beasley [3]. Our numerical experiences are about these test problems for unconstrained binary quadratic programs (UBQP) and instances of 0-1 Quadratic Knapsack problem picked from [3] for (constrained) binary quadratic programs. According to [5], due to tighter bounds provided by SDP relaxation, MIQP** is better than MIQP* and the aforementioned global algorithms. These comparisons, based on the quality (gap) of computed solutions, #Iter and CPU time, were not always possible. The algorithms are different in their conception and construction: For instance, the B&B of MIQP has very large #Iter but each iteration is less costly, while the #Iter of the algorithm in [13] is much smaller, but each iteration is more expansive. Note that the performance of an algorithm also depends on the programming skill and the power of the machines on which the experiments are carried out. As noticed in [5], for a max-cut problem with the 80 nodes G.5 instances of Table 6, Helmberg and Rendl's method [13] needs about 1 h and 46 min (average on 2 instances) of

CPU time on a HP9000/715 workstation while MIQP** needs about 8 min, on the above Pentium IV PC. But the B&B algorithm in [13] performs 154 nodes when MIQP** performs 8,76,623 nodes. *Obviously, the smaller its #Iter is, the better the algorithm will be for high dimensions.*

As indicated in DC/SDP relaxation, and under the precautions mentioned above, our computational experiments on these test problems (UBQP ) show that DCA-B&B-SDP is very competitive and provides the numerical results comparable to MIQP** in [5] while our #Iter, like Helmberg-Rendl [13], are much smaller. Note that MIQP** is too expensise to handle large-scale problems: with $n = 100$,it failed to find optimal binary solutions after 3 h of CPU time for many cases. On the other hand, it is worth remarking the nice computational behaviour of DCA-B&B-SDP for solving $0 - 1$ Quadratic Knapsack problem (constrained binary quadratic programs) with $n = 100$, in Tables 6 & 7. It found $\epsilon$-optimal *binary* solutions with $\epsilon = 0$ for 4/40(4 instances over 40 test problems), $0.01 \leq \epsilon \leq 0.02$ for 32/40 , $\epsilon = 0.03, 0.04, 0.05$ for the last four problems; and CPU time is less than 3.41 s for 26/40.

But the remarquable fact, through these numerical experiments, remains the possibility for DCA to reach $\epsilon$-optimal solution with only one restarting from initial points computed by DC/SDP relaxation (solving only one linear program/ SDP problem).

## Conclusion

We have investigated a DC programming approach via the local DCA and a global algorithm combining DCA and B&B with DC/ SDP relaxation techniques for solving binary quadratic programs (BQP). The binary character of the variable in this problem and exact penalty in DC programming allow reformulating (BQP) into an appropriate equivalent polyhedral DC program. That leads to a very simple DCA, consisting of simply solving a finite number of linear programs with the same constraint set. In particular DCA is explicit for unconstrained binary quadratic programs (UBQP). DCA-B&B-new is better than DCA-B&B-old: each iteration of the former is a little more costly than that one of the latter, but #Iter of the latter is much smaller. Even they are not able to locate global solutions in all instances, these combined algorithms can provide DCA with initial points in order for it to reach $\epsilon$-optimal *binary* solutions with #DCA = 1. On the other hand, SDP relaxation technique seems to be quite relevant to the structure of (BQP) for the construction of relaxed convex quadratic programs with tight lower bounds. These approaches have been implemented and tested on three classes of binary quadratic programs: (UBQP), (QK01) and (MCP).

Preliminary numerical simulations show that the combination DCA-B&B-SDP is very efficient. Its success is due to DCA which is featured by the two facts:

(i) Although being a continuous approach, DCA generates a finite sequence of binary solutions with decreasing objective values. DCA gives $\epsilon$ -optimal solutions in almost all cases with only one restarting for the first two classes and exact solutions to (MCP).
(ii) SDP relaxed convex programs provide quite tight lower bounds for the optimal value of (BQP) to force DCA to tend towards $\epsilon$ -optimal solutions after only one restarting of DCA in almost all cases.

They confirm the practical observations concerning DCA: DCA is inexpensive and can be applied to large-scale DC programs to which it gives quite often optimal solutions, while starting from a good initial point. Finally, the combined DCA-B&B-DC/SDP will be able to handle large-scale (BQP) if its number of iterations remains in certain reasonable limits. Otherwise, thanks to its inexpensiveness, DCA still works well to find good local binary solutions, but we do not have any more means of checking their globality. Note that our approaches are easily extended to mixed 0-1 quadratic programming. We hope that the DCA will be useful for people having to solve large-scale real-world (BQP). Their computational results will make it possible to strongly appreciate the robustness and effectiveness of the DCA.

**Acknowledgments** The authors would like to thank the two referees for their useful comments and suggestions which have considerably improved the presentation of the revised paper.

## References

1. Adams, W.P., Dearing, P.M.: On the equivalence between roof duality and Lagrangian duality for unconstrained 0–1 quadratic programming problems. Discrete Appl. Math. **48**(1), 1–20 (1994)
2. Adams, W.P., Sherali, H.D.: A tight linearization and an algorithm for 0-1 quadratic programming problems. Manag. Sci. **32**(10), 1274–1290 (1986)
3. Beasley, J.E.: Obtaining test problems via internet. J. Global Optim. **8**, 429–433 (1996). http://people.brunel.ac.uk/~mastjjb/jeb/info.html
4. Beasley, J.E.: Heuristic algorithms for unconstrained binary quadratic programming problem. Technical report, The management school, Imperial college, London SW7 2AZ, England (1998)
5. Billionnet, A., Elloumi, S.: Using a mixed integer quadratic programming solver for unconstrained quadratic 0–1 problem. Math. Program. **109**(1, Ser.A), 55–68 (2007)
6. Delaporte, G., Jouteau, S., Roupin, F.: SDP S :A tool to formulate and solve semidefinite relaxations for bivalent quadratic problems (2002). http://semidef.free.fr/
7. Fujisawa, K., Kojima, M., Nakata, K., Yamashita, M.: SDPA—Users manual version 6.00. Research reports on Mathematical and Computing Sciences, Tokyo Institue of Technology (2002)
8. Glover, F., Kochenberger, G.A., Alidaee, B.: Adaptive memory tabu search for binary quadratic programs. Manag. Sci. **44**, 336–345 (1998)
9. Hammer, P.L.: Plant location—A pseudo-Boolean approach. Isr. J. Technol. **6**, 330–332 (1968)
10. Hammer, P.L., Rudeanu, S.: Boolean Methods in Operations Research. Springer, New York (1968)
11. Hansen, P., Jaumard, B., Meyer, C.: A simple enumerative method algorithm for unconstrained 0-1 quadratic programming. Technical ReportG-2000-59, Les Cahiers du GERAD (2000)
12. Helmberg, C.: Semidefinite programming for combinatorial optimization, ZIB-Report 00-34, Oct (2000)
13. Helmberg, C., Rendl, F.: Solving quadratic 0-1 problem by semidefinite programs and cutting planes. Math. Program. **82**(3), 291–315 (1998)
14. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**(3), 673–696 (2000)
15. Hiriart Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer, Berlin, Heidelberg (1993)
16. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3rd edn. Springer, Berlin (2000)
17. Horst, R., Pardalos, P.M., Van Thoai, N.: Introduction to Global Optimization, 2nd edn. Kluwer, Dordrecht (2000)
18. ILOG. CPLEX 7.5. Ilog cplex 7.5 reference manual. ILOG CPLEX Division, Gentile, France. http://www.ilog.com/products/cplex
19. Jha, S., Pardalos, P.M.: Graph separation techniques for quadratic 0–1 programming. Comput. Math. Appl. **21**(6/7), 107–113 (1991)
20. Laughunn, D.J.: Quadratic binary programming. Oper. Res. **14**, 454–461 (1970)
21. Le Thi, H.A.: Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications, Habilitation à Diriger des recherches, Université de Rouen, Juin (1997)
22. Le Thi, H.A., Pham Dinh, T.: Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. J. Global Optim. **11**, 253–285 (1997)
23. Le Thi, H.A., Pham Dinh, T.: DC optimization algorithms for solving the trust region subproblem. SIAM J. Optim. **8**, 476–505 (1998)
24. Le Thi, H.A., Pham Dinh, T.: A combined DC optimization —ellipsoidal branch-and-bound algorithm for solving nonconvex quadratic programming problems. J. Comb. Optim. **2**(1), 9–29 (1998)
25. Le Thi, H.A., Pham Dinh, T.: A continuous approach for large-scale constrained quadratic zero-one programming. (In Honor of Professor ELSTER, Founder of the Journal Optimization) Optimization **45**(3), 1–28 (2001)

26. Le Thi, H.A., Pham Dinh, T.: Large scale molecular optimization from distance matrices by a DC optimization approach. SIAM J. Optim. **14**(1), 77–117 (2003)
27. Le Thi, H.A., Pham Dinh, T.: DC programming approaches and DCA for globally solving linear complementarity problems, Research Report, National Institute for Applied Sciences, Rouen (2004)
28. Le Thi, H.A., Pham Dinh, T.: The DC programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
29. Le Thi, H.A., Pham Dinh, T., Huynb Van, N.: Exact penalty techniques in DC programming, Research Report, National Institute for Applied Sciences, Rouen (2005)
30. Le Thi, H.A., Pham Dinh, T., Le Dung, M.: Exact penalty in DC programming. Vietnam J. Math. **2**, 169–178 (1999)
31. Le Thi, H.A., Pham Dinh, T., Le Dung, M.: Simplicially constrained D.C. optimization over the efficient and weakly efficient sets. J. Optim. Theory Appl. **117**, 503–521 (2003)
32. Le Thi, H.A., Pham Dinh, T., Nguyen Canh, N.: Local and global approaches based on DC programming, branch-and-bound and SDP techniques for nonconvex quadratic programming, Research Report , National Institute for Applied Sciences, Rouen (2005)
33. Lemaréchal, C., Oustry, F. : SDP relaxations in combinatorial optimization from a Lagrangian point of view. In: Hadjisavvas, N., Pardalos, P.M. (eds.) Advances in Convex Analysis and Global Optimization, pp. 119–134. Kluwer, Dordrecht (2001)
34. Nam, N.C.: Approches Locales et Globales basées sur la Programmation DC & DCA et les Techniques B&B avec Relaxation SDP pour certaines classes des programmes non convexes. Simulations Numériques et Codes à l'Usage Industriel, PhD thesis, National Institute for Applied Sciences-Rouen, France (2007)
35. Pardalos, P.M., Rodgers, G.P.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 131–144 (1990)
36. Pham Dinh, T., Le Thi, H.A.: Convex analysis approach to DC programming: theory, algorithms and applications. Acta Math. Vietnamica, Dedicated to Professor Hoang Tuy on the Occasion of His 70th Birthday **70**(1), 289–357 (1997)
37. Pham Dinh, T., Le Thi, H.A.: DC optimization algorithms for solving the trust region subproblem. SIAM J. Optim. **8**, 476–505 (1998)
38. Pham Dinh, T., Le Thi, H.A., Akoa, F.: Combining DCA and interior point techniques for large-scale nonconvex quadratic programming, optimization, methods and softwares. **23**(4), 609–629 (2008)
39. Poljak, S., Rendl, F., Wolkowicz, H.: A recipe for semidefinite relaxation for (01)-quadratic programming. J. Global Optim. **7**, 51–73 (1995)
40. Rockafeller, R.T.: Convex Analysis. Princeton University, Princeton (1970)
41. Thai Quynh, P., Le Thi, H.A., Pham Dinh, T.: Decomposition branch and bound method for globally solving linearly constrained indefinite quadratic minimization problems. Oper. Res. Lett. **17**, 215–222 (1996)
42. Thai Quynh, P., Le Thi, H.A., Pham Dinh, T.: On the global solution of linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method. RAIRO, Rech. Opér. **30**(1), 31–49 (1996)
43. Wiegele, A.: Big Mac library. http://biqmac.uni-klu.ac.at/biqmaclib.html (2007)
44. Wolkowicz, H., Saigal, R., Vandenberghe, L.: Handbook of Semidefinite Programming—Theory, Algorithms, and Application. Kluwer, Dordrecht (2000)