Optim Lett (2017) 11:967–981
DOI 10.1007/s11590-015-0988-y

ORIGINAL PAPER

Linear transformation based solution methods for non-convex mixed integer quadratic programs

Eric Newby[1] · **M. M. Ali**[2]

Received: 13 March 2015 / Accepted: 11 December 2015 / Published online: 26 December 2015

Abstract Two preprocessing techniques for mixed integer quadratic programs with non-convex objective functions are presented. The first is a convexification scheme and can be applied to problems were the continuous part of the Hessian is positive semidefinite. The second technique aims to reduce the size of the underestimating problems solved by branch-and-bound algorithms and can be applied to problems were the continuous part of the Hessian is singular. Numerical results are presented showing the effect of the preprocessing techniques.

Keywords Mixed integer programming · Quadratic programming · Linear transformation · Non-convex optimization

1 Introduction

In this paper we present two preprocessing techniques, based on carefully chosen linear transformations, for linearly constrained mixed integer quadratic programs (MIQPs) with nonconvex objective functions. The preprocessing techniques were developed to decrease the solution time of MIQP subproblems arising in the derivative free algorithm developed in [1,2]. The derivative free algorithm uses MIQPs to approximate

✉ M. M. Ali
montaz.ali@wits.ac.za

Eric Newby
Eric.Newby@students.wits.ac.za

1 School of Computer Science and Applied Mathematics, Faculty of Science, University of the Witwatersrand, Wits 2050, South Africa

2 TCSE, Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, South Africa

the objective. A number of these MIQPs need to be solved by the derivative free algorithm and preprocessing techniques which can reduce the solution times of the individual MIQPs result in a large reduction in the solution time of the derivative free algorithm. The development of improved solution techniques for MIQPs is also important in its own right since they have a number of applications; a list of examples is given in Billionnet et al. [3]. A number of papers consider the case with binary rather than general integer variables, for a good overview see Burer, Misener and Floudas [4,5] and the references contained therein. Fewer papers consider the general integer case. A good review of the convex relaxations and valid inequalities that have been developed for this problem can be found in Burer and Saxena [6]. In addition the gap inequalities for the max-cut problem [7] have been generalised to MIQPs with non-convex objective functions [8]. In Buchheim and Wiegele [9] a branch-and-bound method using semidefinite programming relaxations is developed for unconstrained MIQPs. In Billionnet et al. [3] a convex reformulation scheme is developed using semidefinite programming; the reformulation scheme can be applied to problems which become convex if all of the integer variables are fixed. The solution approaches listed above cannot solve MIQPs with general non-convex objective functions. MIQPs with non-convex objective functions can be solved using general nonconvex mixed integer non-linear programming solvers such as BARON [10], Couenne [11], SCIP [12] and LINDO [13,14]. Finally, nonconvex MIQPs can be solved using the mixed integer quadratically constrained quadratic programming solver GloMIQO [5,15]. These approaches all make use of some form of branch-and-bound algorithm.

The first preprocessing technique is a convexification scheme which can be applied to problems which become convex if all of the integer variables are fixed. The convex reformulation scheme for bilinear integer terms developed in Pörn et al. [16] is used to perform the convexification of the transformed problem. To the best of the author's knowledge the approach in Pörn et al. [16] is the only existing approach that can be used in conjunction with the transformation. A different approach to solving the same problem has been presented in [17]. The results in this paper are less positive than those in [17] and are communicated with the aim of preventing duplication of the work by other researchers. The second preprocessing technique presented here aims to reduce solution times when solving MIQPs using branch-and-bound algorithms and can be applied to problems with non-convex objective functions where the continuous part of the Hessian is singular. In both the preprocessing techniques additional theories have been developed within the framework of the basic linear transformation suggested in [17]. A related, transformation based, approach for solving a different class of MIQP has been presented in [18]

The rest of the paper is organised as follows. In Sect. 2 the basic form of the linear transformation used in the preprocessing techniques is developed. In Sect. 3 the convexification scheme is developed. In Sect. 4 the preprocessing technique branch-and-bound algorithms is developed. Computational results are presented in Sect. 5. Concluding remarks are made in Sect. 6.

2 The linear transformation

We consider the following mixed integer quadratic program (MIQP):

$$\begin{aligned} \min_x \quad & f(x) = \frac{1}{2} x^T H x + g^T x \\ \text{s.t.} \quad & Ax \le b,\ Dx = e,\ l \le x \le u, \\ & x = \left[x_c^T,\ x_d^T \right]^T \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_d}, \end{aligned} \tag{1}$$

where $H \in \mathbf{S}_n$ (space of symmetric matrices of order n), $g \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $D \in \mathbb{R}^{p \times n}$ and $e \in \mathbb{R}^p$. The numbers of discrete and continuous variables are denoted by n_d and n_c respectively. In this paper we consider the case when H is indefinite.

In this section we describe the general form of a linear transformation which can be applied to problem (1). This transformation was developed in Newby [1]. In deriving the transformation we make use of the fact that H can be expressed in the following form

$$H = \begin{bmatrix} H_{cc} & H_{cd} \\ H_{cd}^T & H_{dd} \end{bmatrix}, \tag{2}$$

where $H_{cc} \in \mathcal{S}^{n_c}$, $H_{dd} \in \mathcal{S}^{n_d}$ and $H_{cd} \in \mathbb{R}^{n_c \times n_d}$. Now, consider a matrix V with the following form

$$V = \begin{bmatrix} U_{cc} & U_{cd} \\ 0 & U_{dd} \end{bmatrix}, \tag{3}$$

where $U_{cc} \in \mathbb{R}^{n_c \times n_c}$ and $U_{dd} \in \mathbb{R}^{n_d \times n_d}$ are arbitrary invertible matrices and $U_{cd} \in \mathbb{R}^{n_c \times n_d}$ is an arbitrary matrix. Any matrix with this form is invertible [19]. Problem (1) is equivalent to the following problem:

$$\begin{aligned} \min_y \quad & h(Vy) = \frac{1}{2} y^T V^T H V y + g^T V y \\ \text{s.t.} \quad & AVy \le b, \quad DVy = e,\ l \le Vy \le u, \\ & y = \left[y_c^T,\ y_d^T \right]^T, \\ & U_{dd} y_d \in \mathbb{Z}^{n_d}, \quad U_{cc} y_c + U_{cd} y_d \in \mathbb{R}^{n_c}. \end{aligned} \tag{4}$$

We need to simplify the integral constraint $U_{dd} y_d \in \mathbb{Z}^{n_d}$; we therefore restrict U_{dd} to be some unimodular matrix. A matrix is unimodular if it is integral and has a determinant of ± 1. Now since $|U_{dd}| = \pm 1$ both U_{dd} and U_{dd}^{-1} are integral and it is obvious that $U_{dd} y_d \in \mathbb{Z}^{n_d} \Leftrightarrow y_d \in \mathbb{Z}^{n_d}$. It is also obvious that the following expression holds $U_{cc} y_c + U_{cd} y_d \in \mathbb{R}^{n_c} \Leftrightarrow y_c \in \mathbb{R}^{n_c}$. Problem (4) now takes the following form:

$$\begin{aligned} \min_{x} \quad & h(Vy) = \frac{1}{2} y^T V^T H V y + g^T V y \\ \text{s.t.} \quad & AVy \leq b,\ DVy = e,\ l \leq Vy \leq u, \\ & y = \left[y_c^T,\ y_d^T \right]^T \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_d}. \end{aligned} \tag{5}$$

We now consider the quadratic term, $y^T V^T H V y$, in problem (5). Substituting (2) and (3) into the quadratic term we obtain the following expression

$$\begin{aligned} y^T V^T H V y = {} & y_c^T U_{cc}^T H_{cc} U_{cc} y_c + 2 y_d^T \left(U_{cd}^T H_{cc} U_{cc} + U_{dd}^T H_{cd}^T U_{cc} \right) y_c \\ & + y_d^T \left(U_{cd}^T H_{cc} U_{cd} + U_{cd}^T H_{cd} U_{dd} + U_{dd}^T H_{cd}^T U_{cd} + U_{dd}^T H_{dd} U_{dd} \right) y_d. \end{aligned} \tag{6}$$

In the following sections we shall use the remaining freedom in the elements of V to simplify (6). The choice of elements will depend on the structure of H_{cc}.

3 Approach used when H_{cc} is positive definite

When H_{cc} is positive semidefinite problem (1) can be reformulated as a convex program. This allows us to apply convex mixed integer approaches to the solution of a non-convex problem. Three reformulations are possible. The first reformulation was developed in [3] and is known as Mixed Integer Quadratic Convex Reformulation (MIQCR). MIQCR results in an equivalent convex MIQP and can be applied to any problem where H_{cc} is positive semidefinite. The second reformulation was developed in [17] and combines a linear transformation with the ideas used in [3]. This reformulation is known as Mixed Integer Quadratic Transformation and Convex Reformulation (MIQTCR). MIQTCR also results in an equivalent convex MIQP and can be applied to any problem where H_{cc} positive definite. The third reformulation is the focus of this section of our paper. We call this reformulation Mixed Integer Quadratic Transformation and Bilinear Convexification (MIQTBC). MIQTBC results in an equivalent convex MINLP. MIQTBC can be applied to any problem where H_{cc} is positive definite and another fairly unrestrictive condition on the form of the Hessian holds.

MIQCR and MIQTCR follow similar approaches based on semidefinte programming to achieve the convexification. MIQTBC follows a different approach and uses the linear transformation in Sect. 2 with a convexification scheme developed for bilinear integer programming [16] to convexify problem (1). To use the scheme reported in [16] we require that the only non-convex terms in the objective function of the transformed problem are bilinear terms involving only the integer variables. In the remainder of this section it is shown that a transformation resulting in an objective function with the required form exists and an algorithm is given to find the transformation.

It is shown in [1] that when H_{cc} is positive definite U_{cd} can be chosen such that problem (5) takes the following form form:

$$\begin{aligned} \min_{y} \quad & h(Vy) = \frac{1}{2} \left(y_c^T \Theta_{cc} y_c + y_d^T \Theta_{dd} y_d \right) + g^T V y \\ \text{s.t.} \quad & AVy \leq b,\ DVy = e,\ l \leq Vy \leq u, \end{aligned}$$

$$y^L \leq y \leq y^U,$$
$$y = \left[y_c^T,\ y_d^T\right]^T \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_d}, \tag{7}$$

where $\Theta_{cc} = U_{cc}^T H_{cc} U_{cc}$, $\Theta_{dd} = U_{dd}^T \left(H_{dd} - H_{cd}^T H_{cc}^{-1} H_{cd}\right) U_{dd}$ and y^L and y^U are, respectively, lower and upper bounds on variables. The method used to calculate y^L and y^U is given in [1].

In the following theorem we show that under a fairly unrestrictive condition we can choose V such that the only non-convex terms in the objective function are bilinear integer terms. When we discuss variable reordering in relation to the following theorem we mean interchanging two discrete variables or two continuous variables, we do not allow the interchanging of a continuous variable and a discrete variable.

Theorem 3.1 *If the elements of x can be reordered such that $H_{dd} - H_{cd}^T H_{cc}^{-1} H_{cd}$ has at least two principal leading submatrices which are not negative semidefinite then there exists a unimodular matrix U_{dd} such that the diagonal elements of Θ_{dd} are all positive.*

Proof Let $\Lambda = H_{dd} - H_{cd}^T H_{cc}^{-1} H_{cd}$. Consider a transformation matrix V of the form

$$V = \begin{bmatrix} U_{cc} & U_{cd} \\ 0_{n_d, n_c} & \widetilde{U}_{dd} U_{dd} \end{bmatrix}.$$

Now the product of two unimodular matrices is unimodular so if both U_{dd} and $\widetilde{U}_{dd}$ are unimodular then so is $\widetilde{U}_{dd} U_{dd}$. Using this form of V we have $\Theta_{dd} = U_{dd}^T \widetilde{U}_{dd}^T \Lambda \widetilde{U}_{dd} U_{dd}$. Now let $\widetilde{\Lambda} = \widetilde{U}_{dd}^T \Lambda \widetilde{U}_{dd}$. We first show that there exists a unimodular matrix $\widetilde{U}_{dd}$ such that $\widetilde{\Lambda}$ has at least one positive diagonal element. Now let $A^{(k)}$ denote a $k \times k$ submatrix of A; the position of $A^{(k)}$ will be clear from the context. We can now write $\widetilde{U}_{dd}$ and Λ as follows

$$\widetilde{U}_{dd} = \begin{bmatrix} \pm 1 & 0_{1, n_d - 1} \\ \widetilde{\mu} & \widetilde{U}_{dd}^{(n_d - 1)} \end{bmatrix},\ \Lambda = \begin{bmatrix} \alpha & \lambda^T \\ \lambda & \Lambda^{(n_d - 1)} \end{bmatrix}, \tag{8}$$

where $\widetilde{U}_{dd}^{(n_d-1)}$ is a lower triangular matrix with ± 1 on its diagonal, $\widetilde{\mu} \in \mathbb{Z}^{n_d - 1}$, $\lambda \in \mathbb{R}^{n_d - 1}$ and $\alpha \in \mathbb{R}$. We can now express $\widetilde{\Lambda}$ as follows

$$\widetilde{\Lambda} = \begin{bmatrix} \widetilde{\mu}^T \Lambda^{(n_d-1)} \widetilde{\mu} \pm 2\lambda^T \widetilde{\mu} + \alpha & \widetilde{\mu}^T \Lambda^{(n_d-1)} \widetilde{U}_{dd}^{(n_d-1)} \pm \lambda^T \widetilde{U}_{dd}^{(n_d-1)} \\ \widetilde{U}_{dd}^{(n_d-1)\,T} \Lambda^{(n_d-1)} \widetilde{\mu} \pm \widetilde{U}_{dd}^{(n_d-1)\,T} \lambda & \widetilde{U}_{dd}^{(n_d-1)\,T} \Lambda^{(n_d-1)} \widetilde{U}_{dd}^{(n_d-1)} \end{bmatrix}.$$

Now since Λ must have at least two leading submatrices which are not negative semidefinite the variables in x can be reordered to make $\Lambda^{(n_d-1)}$ indefinite or positive semidefinite. Suppose that the variables have been ordered such that this is true. If this is the case, $\widetilde{\mu}^T \Lambda^{(n_d-1)} \widetilde{\mu} \pm 2\lambda^T \widetilde{\mu} + \alpha$ is unbounded above, regardless of the values of λ and α. Therefore $\exists\, \widetilde{\mu} \in \mathbb{Z}^{n_d - 1}$ such that $(\widetilde{\Lambda})_{1,1}$ is greater than zero.

Suppose that we choose the elements of $\widetilde{U}_{dd}$ such that $(\widetilde{\Lambda})_{1,1} > 0$. Now Θ_{dd} can be written as $\Theta_{dd} = U_{dd}^T \widetilde{\Lambda} U_{dd}$. We now prove by induction that each leading submatrix of Θ_{dd} can be constructed to have only positive diagonal elements. We

have $\Theta_{dd}^{(1)} = U_{dd}^{(1)\,T}\widetilde{\Lambda}^{(1)}U_{dd}^{(1)} = (\pm 1)^2\widetilde{\Lambda}^{(1)} = \widetilde{\Lambda}^{(1)}$, therefore $\Theta_{dd}^{(1)} > 0$. Now assume the diagonal elements of $\Theta_{dd}^{(k)}$ are positive. We need to show that there exists a unimodular matrix with $U_{dd}^{(k+1)}$ such that $\Theta_{dd}^{(k+1)}$ has positive diagonal elements. Now $\Theta_{dd}^{(k+1)} = U_{dd}^{(k+1)\,T}\widetilde{\Lambda}^{(k+1)}U_{dd}^{(k+1)}$ where

$$U_{dd}^{(k+1)} = \begin{bmatrix} U_{dd}^{(k)} & \mu \\ 0_{1,k} & \pm 1 \end{bmatrix},\ \widetilde{\Lambda}^{(k+1)} = \begin{bmatrix} \widetilde{\Lambda}^{(k)} & \widetilde{\lambda} \\ \widetilde{\lambda}^{T} & a \end{bmatrix},$$

where $U_{dd}^{(k)}$ is an upper triangular matrix with ± 1 on its diagonal, $\mu \in \mathbb{Z}^k$, $\widetilde{\lambda} \in \mathbb{R}^k$ and $a \in \mathbb{R}$. Therefore

$$\Theta_{dd}^{(k+1)} = \begin{bmatrix} U_{dd}^{(k)\,T}\widetilde{\Lambda}^{(k)}U_{dd}^{(k)} & U_{dd}^{(k)\,T}\widetilde{\Lambda}^{(k)}\mu \pm U_{dd}^{(k)\,T}\widetilde{\lambda} \\ \mu^T\widetilde{\Lambda}^{(k)}U_{dd}^{(k)} \pm \widetilde{\lambda}^T U_{dd}^{(k)} & \mu^T\widetilde{\Lambda}^{(k)}\mu \pm 2\widetilde{\lambda}^T\mu + a \end{bmatrix}.$$

We need to show that $\exists\, \mu \in \mathbb{Z}^k$ s.t. $\mu^T\widetilde{\Lambda}^{(k)}\mu \pm 2\widetilde{\lambda}^T\mu + a > 0$. Since this is a quadratic function of μ it is sufficient to show that one of the diagonal elements of $\widetilde{\Lambda}^{(k)}$ is greater than zero. Now $\widetilde{\Lambda}^{(k)}$ has at least one positive element, $(\widetilde{\Lambda}^{(k)})_{1,1}$. Therefore $\exists\, \mu \in \mathbb{Z}^k$ such that $(\Theta_{dd}^{(k+1)})_{k+1,k+1} > 0$ and by the assumption $(\Theta_{dd}^{(k)})_{i,i} > 0\ \forall i = 1, \ldots, k$. Therefore there exists a unimodular matrix U_{dd} such that the diagonal elements of Θ_{dd} are all positive. □

The assumption that the elements of x can be reordered such that $H_{dd} - H_{cd}^T H_{cc}^{-1} H_{cd}$ has at least two principal leading submatrices which are not negative semidefinite will not be satisfied for every H. However numerical experience suggests that the condition will be satisfied for a large number of Hessians, especially as the value of n increases. This is mainly due to the fact that we can rearrange the elements of Λ by relabelling the elements of x. For example if any of the diagonal elements of Λ are greater than zero the elements of x can be relabelled such that the assumption holds.

While we have proven the existence of a transformation matrix U_{dd} with the required properties we still need to provide a concrete method for finding matrices with this form. The required method is described by Algorithm 1. The algorithm finds a matrix with the required form using two basic stages. The first stage ensures that we have a $\widetilde{\Lambda}$ matrix with $(\widetilde{\Lambda})_{1,1} > 0$; steps ii to v are involved in this process. If Λ does not have a positive diagonal element in step ii then a matrix $\widetilde{U}_{dd}$, which will allow us to define $\widetilde{\Lambda}$ such that it has at least one positive diagonal element, is found in steps iii and iv. The second stage of the algorithm finds a matrix U_{dd} which ensures that the diagonal elements of $U_{dd}^T\widetilde{\Lambda}U_{dd}$ are all positive where $\widetilde{\Lambda}$ was found using the first part of the algorithm. Steps vi to ix are involved in the second stage of the algorithm. We restrict U_{dd} to be an upper triangular matrix with 1 or -1 on its diagonal. We set the upper triangular elements one column at a time. To ensure that the diagonal elements of $U_{dd}^T\widetilde{\Lambda}U_{dd}$ are positive we need to ensure that $\mu^T\widetilde{\Lambda}^{(k)}\mu \pm 2\widetilde{\lambda}^T\mu + a > 0$ is satisfied. To do this we first check, in step vi, whether this equation is satisfied by $\mu = 0$. If this is the case we set the upper triangular elements of the column to zero. Otherwise we use steps vii and viii to set the element of the column in the first row to the smallest number which allows this equation to be satisfied if all the other elements of μ are set

to zero. We then set the diagonal element of U_{dd} to 1 or -1 if our choice of μ satisfies $\mu^T \widetilde{\Lambda}^{(k)} \mu + 2\widetilde{\lambda}^T \mu + a > 0$ or $\mu^T \widetilde{\Lambda}^{(k)} \mu - 2\widetilde{\lambda}^T \mu + a > 0$ respectively.

Algorithm 1 The U_{dd} selection algorithm for MIQTBC

i. Set rhs $= 1$, ind $= 2$ and $U_{dd} = 0_{n_d \times n_d}$.
ii. If Λ has at least one positive diagonal element reorder the variables such that $(\Lambda)_{1,1} > 0$, set $\widetilde{U}_{dd} = I_{n_d}$, set $\widetilde{\Lambda} = \Lambda$ and go to step vi. If all of the diagonal elements of Λ are negative go to step iii.
iii. Solve the following equation numerically for $\xi \in \mathbb{R}^{n_d - 1}$; $\xi^T \Lambda^{(n_d-1)} \xi + 2\lambda^T \xi + \alpha =$ rhs. The equation was solved using the MATLAB function `fsolve`.
iv. Let $\widetilde{\mu} = \text{round}(\xi)$. If $\widetilde{\mu}^T \Lambda^{(n_d-1)} \widetilde{\mu} + 2\lambda^T \widetilde{\mu} + \alpha > 0$ then let

$$\widetilde{U}_{dd} = \begin{bmatrix} 1 & 0_{1,n_d-1} \\ \widetilde{\mu} & I_{n_d-1} \end{bmatrix}$$

and go to step v. Otherwise set rhs $=$ rhs $+ 1$ and go to step iii.
v. Let $\widetilde{\Lambda} = \widetilde{U}_{dd}^T \Lambda \widetilde{U}_{dd}$ and reorder the variables such that $(\widetilde{\Lambda})_{1,1} > 0$.
vi. If $(\widetilde{\Lambda})_{\text{ind,ind}} > 0$ set $(U_{dd})_{1,\text{ind}} = 0$, $(U_{dd})_{\text{ind,ind}} = 1$ and go to step ix. Otherwise go to step vii.
vii. Set μ_1 and μ_2 as the solutions to the following problems

$$\begin{aligned} &\min_{\mu_1} \quad \mu_1 \\ &\text{s.t} \quad \left(\widetilde{\Lambda}\right)_{1,1} \mu_1^2 + 2\left(\widetilde{\Lambda}\right)_{\text{ind},1} \mu_1 + \left(\widetilde{\Lambda}\right)_{\text{ind,ind}} > 0, \quad \mu_1 \geq 0, \quad \mu_1 \in \mathbb{Z}. \\ &\min_{\mu_2} \quad \mu_2 \\ &\text{s.t} \quad \left(\widetilde{\Lambda}\right)_{1,1} \mu_2^2 - 2\left(\widetilde{\Lambda}\right)_{\text{ind},1} \mu_2 + \left(\widetilde{\Lambda}\right)_{\text{ind,ind}} > 0, \quad \mu_2 \geq 0, \quad \mu_2 \in \mathbb{Z}. \end{aligned}$$

viii. If $\mu_1 < \mu_2$ set $(U_{dd})_{1,\text{ind}} = \mu_1$ and $(U_{dd})_{\text{ind,ind}} = 1$ otherwise set $(U_{dd})_{1,\text{ind}} = \mu_2$ and $(U_{dd})_{\text{ind,ind}} = -1$.
ix. If ind $= n$ go to step x otherwise set ind $=$ ind $+ 1$ and go to step vi.
x. Return $\widetilde{U}_{dd}$, U_{dd} and the variable reorderings used during the algorithm.

We now consider problem (7). Set U_{cc} to be the matrix whose columns are the normalised eigenvectors of H_{cc} and use Algorithm 1 to choose U_{dd} such that Theorem 3.1 is satisfied. We now have the following MIQP which is equivalent to problem (1):

$$\begin{aligned} \min_{y} \quad & h(Vy) = \frac{1}{2}\left(y_c^T \Theta_{cc} y_c + y_d^T \Theta_{dd} y_d\right) + g^T V y \\ \text{s.t.} \quad & AVy \leq b,\ DVy = b,\ l \leq Vy \leq u,\ y^L < y < y^U, \\ & y = \left[y_c^T,\ y_d^T\right]^T \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_d}, \end{aligned} \tag{9}$$

where $\Theta_{cc} = U_{cc}^T H_{cc} U_{cc}$ is diagonal and all the diagonal elements of Θ_{dd} are positive. We note that since we have used Algorithm 1 to choose U_{dd} the only non-convex terms in objective function of problem (9) are integer bilinear terms and that the y variables may have been reordered by the algorithm. In Pörn et al. [16] a convexification scheme for bilinear integer programs is developed. This scheme allows each bilinear integer term in an optimization problem to be replaced by an equivalent convex term by adding additional variables and constraints to the problem. Each of the bilinear terms in the

objective function of problem (9) was replaced by the substitutions developed in [16]. The resulting problem is a convex MINLP which is equivalent to problem (1) [16]. MIQTBC can be applied to any problem where the n_cth principal leading submatrix is positive definite and the elements of x can be reordered such that $H_{dd} - H_{cd}^T H_{cc}^{-1} H_{cd}$ has at least two principal leading submatrices which are not negative semidefinite.

4 Approach used when H_{cc} is singular

When H_{cc} is singular the aim of the preprocessing technique is to reduce the number of bilinear terms in the objective function that need to be underestimated. This aim is chosen since the available solution approaches for problems of this type, such as SCIP, BARON, LINDO and Couenne, make use of branch-and-bound algorithms. When constructing the lower bounding problems in the branch-and-bound tree each bilinear term in the objective function is underestimated using the convex envelopes in [20]. Each bilinear term which needs to be underestimated adds one additional variable and two constraints to the lower bounding problem [20]. Reducing the number of bilinear terms will decrease the size of the lower bounding problems which could improve the efficiency of the branch-and-bound algorithm. Towards this end we choose the elements of V such that the Hessian Θ of the transformed problem, which is given in (6), can be written in the following form

$$\Theta = \Theta^{(1)} + \Theta^{(2)} = \begin{bmatrix} \Theta_{cc}^{(1)} & 0 \\ 0 & \Theta_{dd}^{(1)} \end{bmatrix} + \begin{bmatrix} \Theta_{cc}^{(2)} & \Theta_{cd}^{(2)} \\ \Theta_{cd}^{(2)T} & \Theta_{dd}^{(2)} \end{bmatrix}, \tag{10}$$

where $\Theta_{cc}^{(1)}$, $\Theta_{cc}^{(2)}$ and $\Theta_{dd}^{(2)}$ are diagonal and $\Theta^{(2)}$ is positive definite. Since $\Theta^{(2)}$ is positive definite none of the terms in $\Theta^{(2)}$ need to be underestimated when obtaining the lower bounds. We now show that there exists a matrix V which gives Θ the form specified by (10). As before we require U_{cc} to be a matrix which diagonalises H_{cc}. We now prove the existence of the required transformation.

Theorem 4.1 *$\exists$ U_{cc} such that U_{cc} diagonalises H_{cc} and Θ can be written in the following form*

$$\Theta = \begin{bmatrix} \Theta_{cc}^{(1)} & 0 \\ 0 & \Theta_{dd}^{(1)} \end{bmatrix} + \Theta^{(2)}, \tag{11}$$

where $\Theta^{(2)}$ is positive definite.

Proof We define $A = U_{cc}^T H_{cc} U_{cc}$, $B = U_{cc}^T (H_{cc} U_{cd} + H_{cd} U_{dd})$ and finally define $C = U_{cd}^T H_{cc} U_{cd} + U_{cd}^T H_{cd} U_{dd} + U_{dd}^T H_{cd}^T U_{cd} + U_{dd}^T H_{dd} U_{dd}$. The Hessian in (6) now takes the following form

$$\Theta = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} = \begin{bmatrix} A^f & 0 \\ 0 & C^f \end{bmatrix} + \begin{bmatrix} A^d & B \\ B^T & C^d \end{bmatrix}, \tag{12}$$

where A^d and C^d are defined as diagonal matrices with positive values on the diagonal and A^f and C^f are defined such that $A = A^d + A^f$ and $C = C^d + C^f$. Denote the

second matrix in (12) as $\Theta^{(2)}$. Now the elements of U_{cd} and U_{dd} can be taken to be fixed, the methods used to fix these matrices will be discussed after the proof of the theorem. Using the definition of B and the fact that U_{cd} and U_{dd} are fixed we see that the elements of B can be written in the following form

$$(B)_{j,k} = \sum_{m=1}^{n_c} \rho_m^{(k)} (U_{cc})_{m,j} , \tag{13}$$

where $\rho_m^{(k)} \in \mathbb{R}$. Now we know that U_{cc} must be a matrix which diagonalises H_{cc} but this does not specify U_{cc} uniquely. If F is some matrix which diagonalises H_{cc} then the matrix U_{cc} obtained by multiplying each column of F by some real number will also diagonlise H_{cc} [21]. Therefore the magnitude of the elements of U_{cc} can be made arbitrarily small. It is then clear from (13) that the elements of U_{cc} can be chosen to make the magnitude of the elements of B arbitrarily small.

A matrix $F \in \mathbb{R}^{(n,n)}$ is said to be strictly diagonally dominant if $\left|(F)_{i,i}\right| > \sum_{j=1,\, j\neq i}^{n} \left|(F)_{i,j}\right|$, $\forall i$ [19]. A strictly diagonally dominant matrix which is Hermitian and has positive diagonal elements is positive definite [19]. Now from the definitions of A^d and C^d we know that the $\Theta^{(2)}$ has positive diagonal elements and it is obvious that $\Theta^{(2)}$ is Hermitian. We need to show that we can choose the elements of U_{cc} such that $\Theta^{(2)}$ is strictly diagonally dominant. We have shown above that the magnitude of the elements of B can be made arbitrarily small and we know that A^d and C^d are diagonal matrices. Therefore the magnitude of the off diagonal elements of $\Theta^{(2)}$ can be made arbitrarily small. It is then obvious that we can choose U_{cc} such that $\Theta^{(2)}$ is strictly diagonally dominant. Therefore $\exists\ U_{cc}$ such that U_{cc} diagonalises H_{cc} and Θ can be written in the required form. □

We now discuss the methods used to fix the values of U_{cd} and U_{dd}. U_{dd} is set to I_{n_d} using a method described in [1]. In fixing U_{cd} we note that although there will always exist some U_{cc} such that Theorem 4.1 is satisfied the elements of U_{cc} might be very small. This tends to increase the feasible ranges of the variables, $(y^U - y^L)$, in problem (5), see [1]. We attempted to use our freedom in the choice of U_{cd} to minimise this effect. A number of methods were developed to try and achieve this, the most promising of which sets as many of the elements of $H_{cc}U_{cd} + H_{cd}U_{dd}$ to zero as possible. This was done because we need to use U_{cc} to make the elements of B small enough to make $\Theta^{(2)}$ positive definite. It was reasoned that since $B = U_{cc}^T (H_{cc}U_{cd} + H_{cd}U_{dd})$, if $H_{cc}U_{cd} + H_{cd}U_{dd}$ had a large number of zero elements then the elements of U_{cc} could be made larger while still satisfying the requirements in (11). The efficiency of this choice of U_{cd} was tested. However, it was found through numerical experiment that the most efficient value for U_{cd} was the zero matrix. There may be more efficient choice of U_{cd} than that found in this paper and the choice of this matrix warrants further investigation. Algorithm 2 was used to find U_{cc} satisfying (11). In Algorithm 2 μ, ν and ω are parameters set by the user which determine the size of the elements of U_{cc}, A^d and C^d respectively. The effectiveness of the transformation is dependent on the choice of μ, ν and ω. The values used in this work are given in Sect. 5.

Algorithm 2 The U_{cc} selection algorithm for singular H_{cc}

i. Set μ, ν and ω. Set $r = 0$. Let $\tilde{U}_{cc}$ be the matrix with the normalised eigenvectors of H_{cc} as its columns. Let $U_{cc} = \mu\tilde{U}_{cc}$.
ii. Let $\chi = \nu \max(\text{abs}(U_{cc}^T H_{cc} U_{cc}))$ and $\zeta = \omega \max(\text{abs}(C))$ where C is defined prior to (12). Here $\max(F)$ denotes the largest element of the matrix F and $\text{abs}(F)$ denotes the matrix obtained by taking the absolute values of the elements of F.
iii. Set $\tilde{\Theta}^{(2)}$ to the following matrix

$$\tilde{\Theta}^{(2)} = \begin{bmatrix} \chi I_{n_c} & B \\ B^T & \zeta I_{n_d} \end{bmatrix},$$

where B is defined prior to (12). If $\tilde{\Theta}^{(2)} \succ 0$ go to step v, otherwise go to step iv.
iv. Set $r = r + 1$ and $u^{(r)} = 0.9u^{(r)}$ where $u^{(r)}$ is the rth column of U_{cc}. If $r = n_c$ set $r = 0$. Go to step ii.
v. Let $\Theta^{(2)} = \tilde{\Theta}^{(2)}$. Return $\Theta^{(2)}$ and U_{cc} and stop.

Using the specified values of U_{cd}, U_{dd} and U_{cc} the following transformed problem is obtained:

$$\begin{aligned} \min_y \quad & h(Vy) = \frac{1}{2}\left(y_c^T \Theta_{cc}^{(1)} y_c + y_d^T \Theta_{dd}^{(1)} y_d\right) + \frac{1}{2} y^T \Theta^{(2)} y + g^T V y \\ \text{s.t.} \quad & AVy \le b,\ DVy = e,\ l \le Vy \le u,\ y^L \le y \le y^U, \\ & y = \left[y_c^T,\ y_d^T\right]^T \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_d}, \end{aligned} \tag{14}$$

where $\Theta_{cc}^{(1)}$ is diagonal and $\Theta^{(2)}$ is positive definite. We have transformed problem (1) into an equivalent problem which has at most $(n_d^2 - n_d)$ bilinear terms which need to be underestimated. The objective function of problem (14) is not convex, the problem must be solved using a method capable of handling this non-convexity.

5 Computational results

The effectiveness of the transformations developed in Sects. 3 and 4 was tested on randomly generated MIQPs. The method used to generate the random MIQPs is given in Newby [1]. When comparing approaches the superior approach is taken to be the one with the shortest computation time. Unless noted otherwise, all tests were performed on a PC with an Intel Core i5 CPU at 3.2 GHz with 4 GB of RAM running 64-bit Windows 7. Whenever two different algorithms were used to solve the same problem the second algorithm was begun as soon as the first algorithm was terminated. This was done to ensure a similar computational load. All solutions were checked for feasibility. If an algorithm ran for more than 10,000 s on a problem it was stopped and declared unsuccessful for that problem. Two types of constraints were considered in the test problems;

1. Sparse linear inequality constraints.
2. Dense linear inequality constraints.

The constraints were generated using the method described in Newby and Ali [17].

5.1 Results obtained with H_{cc} invertible

We now examine the results obtained when solving the problems generated by MIQCR, MIQTCR and MIQTBC using the MINLP solver Couenne 0.3.2 on the NEOS server [22,23]. The reason that the NEOS server was used for these problems rather than making use of the Couenne binaries is the following. The Couenne binaries require .nl files as input, these files are generated by AMPL. However, the authors only had access to the student version of AMPL which only accepts problems with fewer than 300 variables and constraints. MIQCR and MIQTCR generate transformed problems with a large number of constraints, the number of constraints is generally >300 for $n \geq 6$. We solved problems with constraints of types 1 and 2. We solved randomly generated MIQPs with $n_c = n_d, n$ was varied between 4 and 14 for type 1 constraints and between 4 and 10 for type 2 constraints. The time taken to solve a problem with a certain n was taken as the average time $\bar{t}$ taken to solve 10 randomly generated problems with that n. The average time and the standard deviation s for constraints of types 1 and 2 are given in Tables 1 and 2 respectively. It is clear from Tables 1 and 2 that MIQTBC is the superior convexification scheme for constraints of type 1 when $n < 10$ and for constraints of type 2 for all n examined. The superior performance of MIQTBC is due to the fact that the reformulated problems produced by MIQTBC contain less variables than those produced by MIQCR and MIQTCR. This effect is larger for type 2 constraints as the structure of the constraints gives the variables larger ranges which increases the variables used by MIQCR and MIQTCR. However, the additional variable used in MIQCR and MIQTCR result in reformulated problems which have tight continuous relaxations. For large n the small relaxation gap produced by MIQCR and MIQTCR becomes more

Table 1 The time taken to solve problems with constraints of type 1 with H_{cc} positive definite using Couenne

n	MIQCR		MIQTCR		MIQTBC	
	$\bar{t}$	s	$\bar{t}$	s	$\bar{t}$	s
4	4.5	2.9	4.7	2.0	1.5	0.8
6	45.1	20.7	21.0	8.4	5.9	3.5
8	48.3	16.8	51.0	15.8	22.6	8.4
10	129.0	35.0	199.5	50.1	148.7	36.0
12	299.2	66.7	443.6	110.4	457.5	86.5
14	975.8	200.8	1524.9	367.9	1993.9	339.2

Table 2 The time taken to solve problems with constraints of type 2 with H_{cc} positive definite using Couenne

n	MIQCR		MIQTCR		MIQTBC	
	$\bar{t}$	s	$\bar{t}$	s	$\bar{t}$	s
4	2.8	1.6	1.8	0.9	0.8	0.3
6	12.9	6.7	10.8	4.6	11.5	5.9
8	100.7	27.7	251.4	50.2	67.9	17.6
10	5642.4	887.8	3777.7	517.0	1968.2	380.2

important than the additional variables which are added to the problem. This is why MIQCR and MIQTCR become more efficient than MIQTBC when $n > 10$. However, while MIQTBC generates a convex MINLP, MIQCR and MIQTCR generate convex MIQPs. If the problems generated by MIQCR and MIQTCR are solved using a convex MIQP solver, such as CPLEX, MIQCR and MIQTCR outperform MIQTBC [17].

5.2 Results obtained with H_{cc} singular

We now consider the results obtained when H_{cc} is singular. The values of μ, ν and ω in Algorithm 2 were set by numerical experiment. In order to simplify the selection of these parameters it was decided that each of the three parameters would be given the same value and this value would be denoted by σ. For problems with $n < 10$ we used $\sigma = 2$ and for $n \geq 10$ we used $\sigma = 2.5$.

The results in this section are presented in the form of performance profiles [24]. Performance profiles are constructed as follows. Let $t_{p,s}$ be the time taken by algorithm s to solve problem p. The performance ratio $\rho_{p,s}$ is then given by $\rho_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in \mathcal{S}\}$ where $\mathcal{S}$ is the set of algorithms. Denote the fraction of performance ratios that are less than a factor $\tau \geq 1$, $P(\rho_{p,s} \leq \tau : s \in \mathcal{S})$. The performance profile is a plot, for each algorithm, of $P(\rho_{p,s} \leq \tau : s \in \mathcal{S})$, the performance factor, vs τ, the time factor.

Making use of the special structure of the objective function of problem (14) requires a change in the underestimating procedure used by the branch-and-bound algorithm. The algorithm needs to be told to split the Hessian into the two terms $\Theta^{(1)}$ and $\Theta^{(2)}$ and that none of the bilinear terms in the $\Theta^{(2)}$ need to be underestimated. Since this requires a change in the algorithm rather than the simple preprocessing applied in Sect. 5.1 a basic branch-and-bound algorithm with the required underestimating rules was written in Matlab 7.11. The algorithm, named Algorithm BB, is based on the algorithms in Sahinidis [10] and Zamora and Grossman [25]. A high level description of the key features of the algorithm is given below. As it is not a novel contribution, the full description of Algorithm BB is not given here, the interested reader is referred to [1]. Convex underestimators were constructed using the convex envelopes in [20]. The branching node chosen at each iteration is the node with the smallest lower bound. The branching variable was chosen by finding the non-convex term with the greatest difference between the non-convex term and its convex underestimator at the solution of the convex underestimation of the problem.

We recall that the transformed problem has at most $(n_d^2 - n_d)$ bilinear terms that need to be underestimated. Clearly as n_c increases the reduction in the number of bilinear terms increases so we expect the transformation to become more effective as n_c increases. We therefore consider two different sets of test problems; the first containing problems with $n_c > n_d$ and the second with $n_c < n_d$. Random MIQP test problems were generated for both types of constraints and a range of values of n and n_c. The parameters used to generate the test problems are detailed in [1]. The number of problems in the test sets for $n_c > n_d$ and $n_c < n_d$ are 140 and 112 respectively. The results for $n_c > n_d$ and $n_c < n_d$ are presented in the form of performance profiles in Figs. 1 and 2 respectively. In the figures Original shows the results when solving prob-

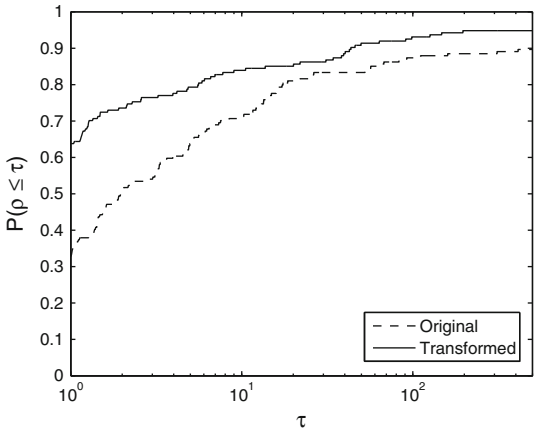

Fig. 1 Performance profile obtained when solving problems with $n_c > n_d$ and H_{cc} singular using Algorithm BB

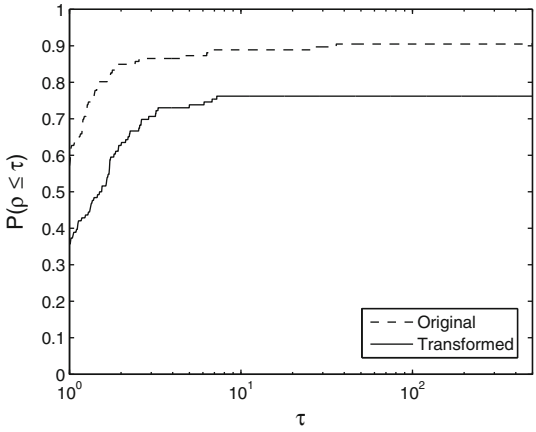

Fig. 2 Performance profile obtained when solving problems with $n_c < n_d$ and H_{cc} singular using Algorithm BB

lem (1) and Transformed shows the results when solving problem (14). The results include the time taken to perform the preprocessing. The time taken to perform the preprocessing is negligible compared to the time taken to solve problems (1) and (14).

Considering Figs. 1 and 2 we see that, as expected, the efficiency of the transformation increases as n_c increases. When $n_c > n_d$ it is clearly more efficient to solve problem (14) and when $n_c < n_d$ is it more efficient to solve problem (1). Results presented here are positive. The value of σ used was obtained by numerical testing. During the testing it was found that some values of σ result in poor performance. No general method for setting σ has been developed.

6 Conclusions

We have presented two preprocessing techniques that can be applied when solving mixed integer quadratic programs with non-convex objective functions. The first technique is a convexification scheme and can be applied to problems where the continuous

part of the Hessian is positive definite. The convex problem produced by the scheme is a MINLP. There are two convexification schemes in the literature which produce convex MIQPs. Due to the superiority of existing convex MIQP solvers over convex MINLP solvers the technique developed in this paper is outperformed by the existing techniques.

The second preprocessing technique can be applied to problems where the continuous part of the Hessian is singular. The technique reduces the number of bilinear terms in the objective function that need to be underestimated when solving the transformed problems using a branch-and-bound algorithm. This reduction is useful because each bilinear term requires additional variables and constraints to be added to the lower bounding problem used in the branch-and-bound algorithm. Promising numerical results are presented for the second preprocessing technique. However, the results are sensitive to a parameter in the algorithm used to generate the transformation matrix and no method has been determined to set the value of this parameter beyond numerical experiment. Methods to set the value of this parameter are an area of future research.

Acknowledgements The second author would also like to thank Professor Tapio Westerlund of Åbo Akademi, Turku, Finland, for introducing the research area to him. Financial support was provided by the National Research Foundation of South Africa under Grant CPR2010030300009918.

References

1. Newby, E.: General solution methods for mixed integer quadratic programming and derivative free mixed integer non-linear programming problems. Ph.D. dissertation, University of the Witwatersrand (2013)
2. Newby, E., Ali, M.M.: A trust-region-based derivative free algorithm for mixed integer programming. Comput. Optim. Appl. **60**, 199–229 (2015)
3. Billionnet, A., Elloumi, S., Lambert, A.: Extending the QCR method to general mixed-integer programs. Math. Program. **131**, 381–401 (2012)
4. Burer, S.: On the copositive representation of binary and continuous nonconvex quadratic programs. Math. Program. **120**, 479–495 (2009)
5. Misener, R., Floudas, C.: GloMIQO: Global mixed-integer quadratic optimizer. J. Glob. Optim. (2012). doi:10.1007/s10898-012-9874-7
6. Burer, S., Saxena, A.: The MILP road to MIQCP. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, pp. 373–405. Springer, New York (2012)
7. Laurent, M., Poljak, S.: Gap inequalities for the cut polytope. Eur. J. Combin. **17**, 233–254 (1996)
8. Galli, L., Kaparis, K., Letchford, A.N.: Gap inequalities for non-convex mixed-integer quadratic programs. Oper. Res. Lett. **39**, 297–300 (2011)
9. Buchheim, C., Wiegele, A.: Using semidefinite programming for solving non-convex mixed-integer quadratic problems. In: Proceedings of the European Workshop on Mixed Integer Nonlinear Programming (2010)
10. Sahinidis, N.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**, 201–205 (1996)
11. Belotti, P., Lee, J., Liberti, L., Margot, F., Wachter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**, 597–634 (2009)
12. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**, 1–41 (2009)
13. Gau, C.Y., Schrage, L.E.: Implementation and testing of a branch-and-bound based method for deterministic global optimization: operations research applications. In: Floudas, C.A., Pardalos, P.M. (eds.) Frontiers in Global Optimization, pp. 145–164. Kluwer Academic Publishers, Berlin (2003)
14. Lin, Y., Schrage, L.E.: The global solver in the LINDO API. Optim. Methods Software. **24**, 657–668 (2009)

15. Misener, R., Floudas, C.: GloMIQO: global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. Math. Program. B **136**, 155–182 (2012)
16. Pörn, R., Harjunkoski, I., Westerlund, T.: Convexification of different classes of non-convex MINLP problems. Comput. Chem. Eng. **23**, 439–448 (1999)
17. Newby, E., Ali, M.M.: A note on convex reformulation schemes for mixed integer quadratic programs. J. Optim. Theory Appl. **160**, 457–469 (2013)
18. Newby, E., Ali, M.M.: Transformation-based preprocessing for mixed-integer quadratic programs. J. Optim. Theory Appl. (2015). doi:10.1007/s10957-015-0806-9
19. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press, Cambridge (1985)
20. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems. Math. Program. **10**, 147–175 (1976)
21. Anton, H., Rorres, C.: Elementary Linear Algebra. Wiley, New Jersey (2005)
22. Czyzyk, J., Mesnier, M.P., More, J.J.: The NEOS server. IEEE Comput. Sci. Eng. **5**, 68–75 (1998)
23. Gropp, W., Moré, J.J.: Optimization environments and the NEOS server. In: Buhmann, M.D., Iserles, A. (eds.) Approximation Theory and Optimization, pp. 167–182. Cambridge University Press, Cambridge (1997)
24. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
25. Zamora, J.M., Grossman, I.E.: A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. J. Glob. Optim. **14**, 217–249 (1999)