An Adaptive Refinement Algorithm for Discretizations of Nonconvex QCQP

Akshay Gupte[1]
School of Mathematics, The University of Edinburgh, UK

Arie M. C. A. Koster
Lehrstuhl II für Mathematik, RWTH Aachen University, Germany

Sascha Kuhnke
Lehrstuhl II für Mathematik, RWTH Aachen University, Germany

Abstract

We present an iterative algorithm to compute feasible solutions in reasonable running time to quadratically constrained quadratic programs (QCQPs), which form a challenging class of nonconvex continuous optimization. This algorithm is based on a mixed-integer linear program (MILP) which is a restriction of the original QCQP obtained by discretizing all quadratic terms. In each iteration, this MILP restriction is solved to get a feasible QCQP solution. Since the quality of this solution heavily depends on the chosen discretization of the MILP, we iteratively adapt the discretization values based on the MILP solution of the previous iteration. To maintain a reasonable problem size in each iteration of the algorithm, the discretization sizes are fixed at predefined values. Although our algorithm did not always yield good feasible solutions on arbitrary QCQP instances, an extensive computational study on almost 1300 test instances of two different problem classes – box-constrained quadratic programs with complementarity constraints and disjoint bilinear programs, demonstrates the effectiveness of our approach. We compare the quality of our solutions against those from heuristics and local optimization algorithms in two state-of-the-art commercial solvers and observe that on one instance class we clearly outperform the other methods whereas on the other class we obtain competitive results.

2012 ACM Subject Classification Theory of computation → Mixed discrete-continuous optimization; Mathematics of computing → Nonconvex optimization

Keywords and phrases Quadratically Constrained Quadratic Programs, Mixed Integer Linear Programming, Heuristics, BoxQP, Disjoint Bilinear

Digital Object Identifier 10.4230/LIPIcs.SEA.2022.24

Supplementary Material *Software (Source Code)*: `https://github.com/skuhnke/qcqp_sourcecode`

Funding The second and third authors' visit was supported by a grant from DAAD, the German Academic Exchange Service.
Akshay Gupte: Initial phase of this research supported by NSF grant DMS-1913294.

Acknowledgements This research was initiated during the second and third authors' visit to Clemson University, USA, in 2019, where the first author was a faculty member.

1 Introduction

A quadratically constrained quadratic program (QCQP) is the optimization problem

$$\begin{aligned} z^* = \max_x \quad & x^\top Q_0\, x + c_0^\top\, x \\ \text{s.t.} \quad & x^\top Q_r\, x + c_r^\top\, x \le b_r, \qquad r \in R \\ & x \in P \end{aligned}$$

[1] Corresponding author

where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the set $P \subseteq \mathbb{R}^n$ is a polytope, the index set R is allowed to be empty so that only the objective is quadratic, and all the data are of conformable dimensions. We assume that the matrices Q_r are symmetric for all $r \in R \cup \{0\}$. When at least one of these matrices has at least one negative eigenvalue (i.e., is not positive semidefinite), then the QCQP instance is nonconvex, which are of interest to us in this paper since they are global optimization problems and solving them to optimality is NP-hard in general. It is common practice in global optimization to assume finite lower and upper bounds on the variables, these bounds may either be pre-specified in the problem or implied by the quadratic constraints or P. The assumption of P being in the nonnegative orthant can be easily achieved by translating the variables. We assume feasible instances.

1.1 Background

QCQPs are a rich and important class of nonconvex optimization with a wide-range of applications. There has been active research for many decades on solution algorithms for them, but a majority of the literature is devoted to convex relaxations and cutting planes that can be used in spatial branch-and-cut algorithms to find a global optimum. The focus of this paper is on generating strong primal (upper) bounds on z^*. A standard choice for doing this is to employ a nonlinear solver to search for a local optima and possibly improve on this local optimum [23]. Although local search heuristics work well sometimes on specific problems, such as box-constrained QPs [4] and also those arising from independent set problem in graphs [22], in general, local solutions can be far away from the global optimum and local solvers have been shown to either fail or produce not-so-good solutions when there are many difficult-to-satisfy nonconvex quadratic constraints. Employing primal heuristics, such as [2, 3, 7, 8, 18, 20], that generate feasible solutions of general mixed-integer nonlinear problems is another approach, but these primarily use integrality of variables and are not always effective for continuous nonconvex problems such as in QCQP. A third way is to solve a convex relaxation of the problem and then round its solution so that it becomes feasible to QCQP; however, a general rounding scheme is not known and difficult to characterise.

Another method for primal bounds is to develop restrictions (i.e., inner approximations) of the feasible region and optimize the objective over it. A common choice for generating restrictions is *variable discretization*, wherein the domain of a subset of variables is constrained to take values in a pre-defined finite set of reals. Although this idea is elementary, to the best of our knowledge, we have not seen it being widely employed for QCQP, except for the aforementioned pooling problem [16, 1, 24, 15, 10], for applications in energy systems [13] and water treatment [17], and for BoxQP [12], which is a subclass of QCQP wherein $R = \emptyset$ (no quadratic constraints) and $P = [0, 1]^n$. Discretising a QCQP yields a mixed-integer QCQP which can be reformulated as a mixed-integer LP (MILP) using different ways of expressing an integer variable as a linear function of binary variables [14]. Convexification studies for such binarization transformations have been carried out for MILPs [9]. The MILP approach to obtaining primal bounds for QCQP is attractive due to many decades of advances in solving MILPs and very sophisticated and powerful commercial solvers available for them. It also complements MILP-based relaxation techniques [5].

Variable discretization algorithms do not fix variables values, and in that sense they bear an advantage to some primal heuristics that obtain feasible solutions by fixing a subset of the variables to certain values [3] (typically obtained through a convex relaxation of the feasible set) and then optimizing over the variables that were left unfixed. However, as they are currently designed and implemented, they have the drawback of requiring that the discretised variables and values be pre-specified. Thus, these methods are "static" in the

sense that they solve only once and do not adapt to the solution that has been found, i.e., once a MILP has been solved and feasible solution obtained, there is no guidance on how to solve another MILP to improve on the current solution. An arbitrary discretization can likely miss close-to-optimal values that may be suggested by a convex relaxation. Of course, one could employ immense computing power and parallelise the solve of multiple MILPs, each with a different discretization scheme, and then take the best primal bound out of all these runs. However, such a brute-force approach is undesirable since it relies on a great amount of computing power and time that may not always be available, and can be rendered needless if instead one adopts a dynamic/adaptive approach where the discretization scheme gets updated according to the previous solution. We know of only two studies [13, 17] that take an adaptive discretization approach for nonconvex problems, but they are focused on specific applications. There are also other areas of optimization where adaptive schemes are used, for e.g., in stochastic programming [25].

1.2 Contributions of this Paper

We devise an adaptive variable discretization algorithm for any QCQP and test it extensively on different problem classes. Our iterative algorithm solves in each iteration an MILP restriction of the original QCQP based on discretization. The discretizations of the MILP restrictions are adapted after each iteration based on the previous solution. The numerical experiments show that our adaptive discretization algorithm yields very good feasible solutions that are competitive and frequently superior than those obtained from global solvers (which employ local solvers and also MILP heuristics) and other discretizations from literature, not only in terms of objective value but also in terms of the running time required for obtaining them. We remark that dynamic updating has been recently carried out for MILP *relaxations* [6, 19] of nonconvex problems, and so our algorithm serves as a complement and lays the groundwork for future work on a global optimization algorithm where both dual and primal bounds are computed by adapting MILP problems.

Section 2.1 introduces the procedure to discretize the quadratic terms. Then, we present in Section 2.2 a procedure to decide which variables in the quadratic terms to discretize. Section 2.3 explains the adaption of the discretization which we apply in each iteration of the algorithm. In Section 2.4, we present the whole adaptive discretization algorithm. Our computational experiments are described in section 3. The algorithms used in this study are introduced in Section 3.1 while details about their implementations are given in Section 3.2. Subsequently, we apply the algorithms to two different problem classes. Then, we perform in Section 3.3 calculations on a huge test set of 1230 BoxQPcc instances. Finally, in Section 3.4, we evaluate the performance of the algorithms on 60 DisjBLP instances.

2 Adaptive Discretization Algorithm

2.1 Discretization of Quadratic Terms

To solve nonconvex QCQPs to optimality, a global optimization algorithm is required which handles the quadratic terms $x_i x_j$. Global solvers such as `BARON` sometimes struggle to compute good feasible solutions in reasonable running time for several mid-size instances. Therefore, a common approach to obtain strong feasible solutions for such problems is discretization. In this paper, we reduce the solution space of the QCQP by restricting at least one of the continuous variables in each quadratic term to certain discrete values. This restriction can be equivalently reformulated into an MILP which is likely to be easier to solve than the original QCQP.

We propose a discretization method which is similar to the unary expansion [14]. To introduce this discretization method, we consider a single quadratic term $w_{ij} = x_i\, x_j$ where $x_i \in [\ell_i, u_i]$ and $x_j \in [\ell_j, u_j]$ with $\ell_i < u_i$ and $\ell_j < u_j$. This term can also be strictly quadratic, i.e., $i = j$. Let $u \geq 2$ be a positive integer which defines the size of the discretization. Then, we allow the originally continuous variable x_i to assume only one of the predefined equidistant values $x_{i1} < \cdots < x_{iu}$ with $x_{i1} \geq \ell_i$ and $x_{iu} \leq u_i$. This means we have $x_i \in \{x_{in} \mid n \in N\}$ where $N := \{1, \ldots, u\}$ is the set of indices of the discretized values. In the following, we express w_{ij} using linear expressions instead of a quadratic one. To this end, we introduce additional binary variables $z_{i1}, \ldots, z_{iu}$ which are used to set x_i equal to one of the values x_{in}:

$$x_i = \sum_{n \in N} x_{in}\, z_{in}, \qquad \sum_{n \in N} z_{in} = 1, \qquad z_{in} \in \{0,1\} \quad \forall\, n \in N. \tag{1}$$

We first consider the case where w_{ij} is bilinear, i.e., $i \neq j$. In this case, we add non-negative continuous variables $y_{ij1}, \ldots, y_{iju}$ defined by $y_{ijn} := x_j\, z_{in}$ for all $n \in N$. In the unary expansion, the bilinear terms $x_j\, z_{in}$ are linearized by applying their McCormick envelopes which consist of $4\,u$ additional constraints. By exploiting the SOS-1 property of the binary variables z_{in}, we use an equivalent but smaller linearization with only $2\,u + 1$ additional constraints:

$$x_j = \sum_{n \in N} y_{ijn}, \qquad \ell_j\, z_{in} \leq y_{ijn} \leq u_j\, z_{in} \quad \forall\, n \in N. \tag{2}$$

The above constraints (1)-(2) allow us to rewrite the bilinear term w_{ij} as a linear term where we sum over each discretized value x_{in} multiplied by corresponding additional variable y_{ijn}:

$$w_{ij} = \sum_{n \in N} x_{in}\, y_{ijn}\,. \tag{3}$$

In the second case where w_{ij} is strictly quadratic, i.e., $i = j$, the additional continuous variables y_{ijn} and their corresponding linearization (2) are not necessary. Along with constraints (1), we can rewrite w_{ii} as follows:

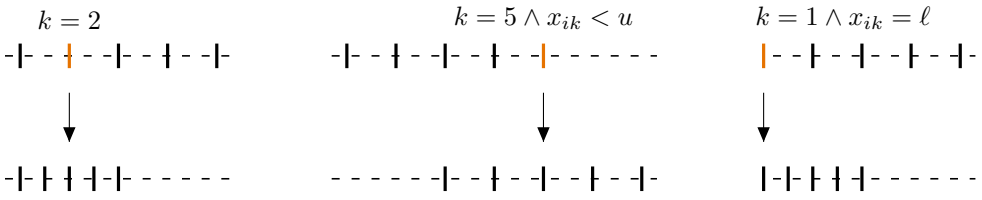
$$w_{ii} = \sum_{n \in N} x_{in}^2\, z_{in}\,. \tag{4}$$

It is known from the results in [14] that the above linearization of the bilinear terms $x_j z_{in}$ is stronger than the standard one using McCormick envelopes.

By discretizing all quadratic terms in the QCQP as described above, the continuous non-convex problem turns into a mixed-integer linear problem which we denote as discretized MILP. This discretized MILP is likely to be easier to solve than the original QCQP since the solution methods for MILPs are very advanced and modern MILP solvers are able to solve very large instances quickly. All feasible solutions to the discretized MILP are also feasible to the original QCQP. However, since we restricted the solution space of the original problem by discretization, an optimal solution to the discretized MILP may not be an optimal solution to the original QCQP.

2.2 Selection of Discretized Variables

To apply the discretization from Section 2.1 to general QCQPs, we have to decide which variables in the quadratic terms we discretize. To this end, we consider a graph $G = (V, E)$ where the set of nodes V is equal to the set of variables of the QCQP and the set of edges E



(a) Internal point. **(b)** End point within feasible region. **(c)** End point on boundary.

Figure 1 Adaption of discretization for $u = 5$.

contains an edge $\{x_i, x_j\}$ if and only if the quadratic term $x_i\, x_j$ exists in the QCQP. In this graph, a vertex cover is equivalent to a feasible discretization for the QCQP where at least one variable in each quadratic term is discretized. It is desirable to discretize as few variables as possible because the problem size of the discretized MILP increases with the number of discretized variables. However, since the minimum vertex cover problem is NP-complete, solving it to optimality might become too time consuming for larger instances. Therefore, we apply the greedy heuristic to obtain a good vertex cover in fast running time.

This heuristic creates a vertex cover by successively adding variables to the set D. After initializing the set $D = \emptyset$, we first add all strictly quadratic variables to D and afterwards remove them along with its adjacent edges from G. Then, we iteratively determine the node with the highest degree in the current graph, add it to D if its degree is positive, and remove it from G afterwards. When the graph is empty, the set D consists of a vertex cover, i.e., D represents a feasible discretization for the QCQP. On our instances, this heuristic performed better than other well-known factor 2 approximation algorithms for the vertex cover problem.

2.3 Adaption of Discretization

The quality of the optimal solution of the discretized MILP depends on the chosen discretized values $x_{i1}, \ldots, x_{iu}$ of each discretized variable $x_i \in D$. Using a big discretization size u might lead to better solutions, but the corresponding discretized MILPs get too large and computationally intractable. On the other hand, if we use a reasonable discretization size, it is unlikely to choose an initial discretization that leads to a very good solution for the original problem. Therefore, we use a computationally tractable discretization size u and iteratively adapt the discretization values $x_{i1}, \ldots, x_{iu}$ while keeping the size u fixed. This leads to an iterative algorithm which allows us to improve the discretization quality in each iteration based on the previous solution of the discretized MILP. By keeping the discretized values corresponding to the solution of the previous iteration, the adapted discretized MILPs yield solutions at least as good as the previous one. Moreover, since the discretization size is fixed, we are able to solve each discretized MILP in reasonable running time.

Let us again consider the discretization of a single quadratic term $w_{ij} = x_i\, x_j$ and let $k \in N$ be the selected index in the solution of the previous discretized MILP, i.e., we have $x_i = x_{ik}$ with $z_{ik} = 1$. To adapt the discretization, we consider three different cases depending on x_{ik} as proposed by [13]. First, we consider the case $1 < k < u$ where the previous solution x_{ik} is an internal point of the discretization. Here, we halve the length of the discretization by moving the new discretized values closer around x_{ik}. This procedure is illustrated in Figure 1a where the second discretized value x_{i2} is selected in the solution of the previous discretized MILP. Second, we consider the case $k \in \{1, u\}$ where the previous solution is an end point of the discretization which lies strictly within the feasible region and not on its boundary, i.e., $x_{ik} \in (\ell_i, u_i)$. Then, we shift the discretization towards x_{ik} without reducing

Algorithm 1 Adaption of discretization.

Input: Discretized values x_{in} for $n \in N$, Previous solution z_{in} for $n \in N$
Output: Adapted discretized values x_{in} for $n \in N$

Choose $k \in N$ such that $z_{ik} = 1$
if $1 < k < u$ **then** ▷ Internal point
  $\delta \leftarrow (x_{i2} - x_{i1}) / 2$ ▷ Step size of new discretization
  $m \leftarrow \lceil \frac{u}{2} \rceil$ ▷ New selected index
else if $k \in \{1, u\} \wedge x_{ik} \in (\ell, u)$ **then** ▷ End point within feasible region
  $\delta \leftarrow (x_{i2} - x_{i1})$
  $m \leftarrow \lceil \frac{u}{2} \rceil$
else if $k = 1 \wedge x_{ik} = \ell$ **then** ▷ End point on left boundary
  $\delta \leftarrow (x_{i2} - x_{i1}) / 2$
  $m \leftarrow 1$
else if $k = u \wedge x_{ik} = u$ **then** ▷ End point on right boundary
  $\delta \leftarrow (x_{i2} - x_{i1}) / 2$
  $m \leftarrow u$
end if
while $x_{ik} + (1 - m)\,\delta < \ell$ **do** ▷ New discretization is outside of feasible region
  $m \leftarrow m - 1$ ▷ Shift to the right into feasible region
end while
while $x_{ik} + (u - m)\,\delta > u$ **do** ▷ New discretization is outside of feasible region
  $m \leftarrow m + 1$ ▷ Shift to the left into feasible region
end while
$x_i \leftarrow x_{ik}$ ▷ Store previous selected discretized value
for $n \in N$ **do**
  $x_{in} \leftarrow x_i + (n - m)\,\delta$ ▷ Assign new discretized values
end for

its length. See Figure 1b for a visualization of this adaption where the last point of the discretization is selected. Lastly, we consider the case $k \in \{1, u\}$ where the previous solution is an end point of the discretization which lies on the boundary of the feasible region, i.e., $x_{ik} \in \{\ell_i, u_i\}$. We then halve the length of the discretization by moving the discretized values towards x_{ik} while keeping x_{ik} as an end point of the new discretization. This situation is shown in Figure 1c where x_{ik} lies on the left boundary of the feasible region. In all three cases, we ensure that the new discretization is equidistant and that it still contains the previous solution x_{ik}. If parts of the new discretization are located outside the feasible region, we shift the discretization back into the feasible region by keeping the previous solution x_{ik} in the discretization. A detailed description of the whole adaption can be found in Algorithm 1.

To adapt the whole discretized MILP, we perform the above adaption for all discretized variables $x_i \in D$. Since the solution of the previous problem is also feasible to the adapted discretized MILP, it can be used as MILP warm start for the latter. This guarantees that the adapted discretized MILP yields a solution at least as good as the previous one.

2.4 Iterative Algorithm

Now we present the adaptive discretization algorithm for the calculation of feasible solutions to QCQPs. A flowchart of this algorithm is displayed in Figure 2. We start the algorithm by determining the set of discretized variables D according to Section 2.2 and then choosing

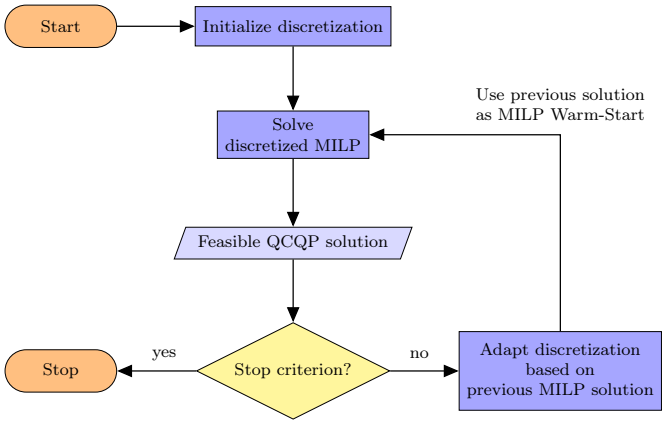


Figure 2 Adaptive discretization algorithm.

Algorithm 2 Adaptive discretization algorithm.

Input: QCQP instance $\mathcal{T}$, Discretization size u
Output: Feasible solution x for $\mathcal{T}$

Determine discretized variables D
for $x_i \in D$ **do** ▷ Initialize discretization
  $\delta \leftarrow (u - \ell) \,/\, (u - 1)$ ▷ Initial step size
  for $n \in N$ **do**
    $x_{in} \leftarrow \ell + (n - 1)\,\delta$ ▷ Assign initial discretized values
  end for
end for
repeat
  Compute solution (x, y, z) to discretized MILP ▷ Use MILP start
  for $x_i \in D$ **do**
    Adapt discretization of x_i with Algorithm 1 ▷ Use previous solution z
  end for
until Stop criterion is fulfilled

an initial discretization $x_{i1}, \ldots, x_{iu}$ for each discretized variable $x_i \in D$. Usually, this initial discretization covers the whole feasible region, i.e., it has the end points $x_{i1} = \ell_i$ and $x_{iu} = u_i$. Next, we solve the corresponding discretized MILP (see Section 2.1) which yields a first feasible solution to the original QCQP. Then, we check if a stop criterion of the algorithm is fulfilled. If no stop criterion is fulfilled, we enter the iteration loop and adapt the discretization of each discretized variable $x_i \in D$ based on the previous MILP solution as stated in Section 2.3. Subsequently, we solve the new discretized MILP where we pass the solution of the previous problem as MILP start. This speeds up the calculations of the current discretized MILP and ensures that the current solution is at least as good as the previous one. Once we calculated the next feasible solution to the original QCQP, we again check for stop criteria. If one of the stop criteria is now fulfilled, we terminate the algorithm instead of entering the next iteration loop and return the QCQP solution calculated in the last iteration. Algorithm 2 describes the above steps in detail.

Our computational experiments in the next section show that this adaptive discretization algorithm computes high quality solutions in reasonable running time for many instances. One drawback of this algorithm is that no feasibility is guaranteed in the first iteration. It

could occur that the discretized MILP with the initial discretization is infeasible even though the original QCQP is feasible. In this case, one could either increase the discretization size u or modify the discretized MILPs to obtain a relaxation instead of a restriction [17]. We do not address in this paper the question of selecting a suitable discretization that is feasible and leave it as a question for separate work in the future.

3 Computational Study

We present an extensive computational study where we apply the adaptive discretization algorithm to two problem classes of QCQP which we describe in the subsequent sections. In preliminary testing, we did try some arbitrary instances of QCQP from the library QPLib [11]; however, for many of these instances it was not easy to determine a initial discretization that is feasible and hence our algorithm would terminate without computing a primal bound.

The source code for our implementations and the instances we used are both available upon request.

3.1 Algorithms

To evaluate the performance of our adaptive algorithm, we use performance profiles to compare objective values and running times to the commercial global solvers `BARON` and `Gurobi`, which are well-known to be state-of-the-art for solving nonconvex QCQP. We also compared against the popular global solver `SCIP` which employs some MILP-based primal heuristics [2, 3], but it performed much worse than `BARON` and `Gurobi` and our algorithm, and so we do not include `SCIP` in the results reported in this paper. As far as we are aware, `BARON` uses a variety of local solvers to compute feasible solutions to nonlinear problems, whereas `Gurobi` employs some MILP heuristics on a disjunctive formulation for QCQP. Thus, our numerical experiments showcase the advantages of our MILP discretizations not only over local solvers but also over other MILP heuristics.

Our adaptive refinement algorithm is denoted by `AD-`u, using the discretization and adaption from Section 2.1 and Section 2.3, respectively, and where the parameter u represents the size of the discretizations used in the algorithm. We calculate solutions to the original QCQPs with the global solvers `BARON` and `Gurobi`. Beside primal solutions, the global solvers also yield dual bounds which allow us to evaluate the optimality gap of our solutions.

3.2 Implementation

GAMS 31.1.1 is used along with its Python API as the mathematical modeling system. `Gurobi` 9.1.1 is used to solve all discretized MILPs, and it is also used as a global solver for the original QCQP. We also compare against the global solver `BARON` 20.4.14. For all solvers, we use a feasibility tolerance 10^{-6} and an integrality tolerance 10^{-5}. For MILPs solved by `Gurobi`, we set the option `mipstart = 1` and for QCQPs solved by `Gurobi`, we set the option `nonconvex = 2`. All remaining solver options are the standard values. Each calculation is performed on a single core of a Linux machine with an Intel Core i9-9900 CPU with 4.7 GHz clock rate and 32 GB RAM where 14 GB RAM is reserved for this calculation.

For the adaptive discretization algorithms `AD-`u and `ADP-`u, we set a global time limit of 3600 seconds. Furthermore, we specify a time limit of 1200 seconds and a relative optimality gap of 0.01% for each discretized MILP. Besides the global time limit of 3600 seconds, the overall algorithm also stops if the relative improvement of the best solution over the last two iterations is smaller than 0.01%. For the QCQP solvers `BARON` and `Gurobi`, calculations are

stopped if the global time limit of 4 hours or a relative optimality gap of 0.01% is reached. Here, we allow a longer calculation time of 4 hours to obtain good dual bounds as well as to show the difficulties of QCQP solvers to find competitive primal solutions even with this advantage in terms of running time.

3.3 Study I: BoxQPs with Complementarity Constraints

A BoxQP has a quadratic objective and lower and upper bounds on variables as the only constraints. Due to triviality of the constraints, good primal bounds for BoxQP can generally be computed very quickly by local solvers, and so it is of no interest to apply our discretization algorithm to BoxQP directly. Instead, we consider BoxQPs with complementarity constraints

$$\begin{aligned} \text{(BoxQPcc):} \quad \max_{x} \quad & x^\top Q\, x + c^\top x \\ \text{s.t.} \quad & x_i\, x_j = 0 \quad \forall\, (i,j) \in E, \\ & 0 \le x_i \le 1 \quad \forall\, i \in \{1,\dots,n\}, \end{aligned}$$

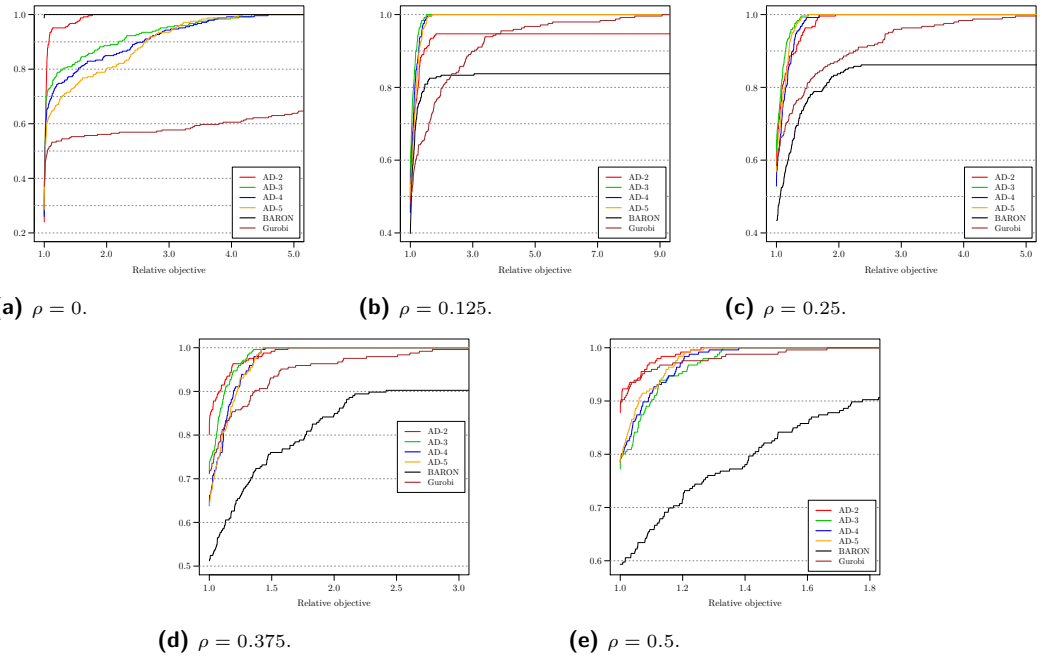
where E is some given subset of the Cartesian product of $\{1,\dots,n\}$ with itself. The complementarity constraints enforce that at least one of the variables x_i and x_j is zero.

Based on the 246 BoxQPs from [21], we generate a test set of 1230 BoxQPcc instances. Let $\rho \in \{0, 0.125, 0.25, 0.375, 0.5\}$ be a fixed probability and $\mathcal{T}$ be one of the BoxQP instances. Then, we create a new BoxQPcc instance $\mathcal{T}_\rho$ by adding for each quadratic term $x_i\, x_j$ that occurs in the objective function of $\mathcal{T}$ the constraint $x_i\, x_j = 0$ with probability ρ. This means that the instances $\mathcal{T}_0$ are equal to the original 246 BoxQP instances while the remaining instances $\mathcal{T}_\rho$ for $\rho \in \{0.125, 0.25, 0.375, 0.5\}$ have additional quadratic complementarity constraints. We only consider probabilities $\rho \in \{0, 0.125, 0.25, 0.375, 0.5\}$ in this study since higher probabilities yield instances with a very high density of complementarity constraints which force many quadratic terms to zero and thus make the instances very easy for most algorithms.

Now we compare the adaptive discretization algorithm `AD-`u for $u \in \{2, 3, 4, 5\}$ with the global solvers `BARON` and `Gurobi`. Since the gaps for most of the BoxQPcc instances are relatively large, performance profiles using relative optimality gaps are not very informative for these instances. Therefore, we only present performance profiles with relative objective values for the BoxQPcc instances.

Figure 3a shows the performance profile with relative objective values for the BoxQPcc instances with $\rho = 0$, i.e., the original BoxQP instances. We see that `BARON` calculates by far the strongest objectives with the best objective values for 99% of the instances. The second best results are achieved by `AD-2` with objectives at most 5% worse than the best for 88% of the instances and all instances at most 77% worse than the best solutions. The remaining adaptive discretization algorithms perform even weaker and only manage to solve all instances with objective values at most 5 times worse than the best. `Gurobi` achieves clearly the weakest results by terminating for 65% of the instances with objectives more than 5 times worse than the best solutions.

A performance profile with relative objective values for the BoxQPcc instances with $\rho = 0.125$ is presented in Figure 3b. Here, the results look very different as the adaptive discretization algorithms `AD-3`, `AD-4`, and `AD-5` perform clearly best and outclass all remaining algorithms. `AD-2` yields weaker objectives than the other discretization algorithms but is still much stronger than `BARON` and `Gurobi` for up to 95% of the instances. `Gurobi` only calculates better objective values than `AD-2` for percentages between 95% and 100%. While the performance of `BARON` has heavily dropped, `Gurobi` achieves stronger results as for $\rho = 0$.



(a) $\rho = 0$. **(b)** $\rho = 0.125$. **(c)** $\rho = 0.25$.

(d) $\rho = 0.375$. **(e)** $\rho = 0.5$.

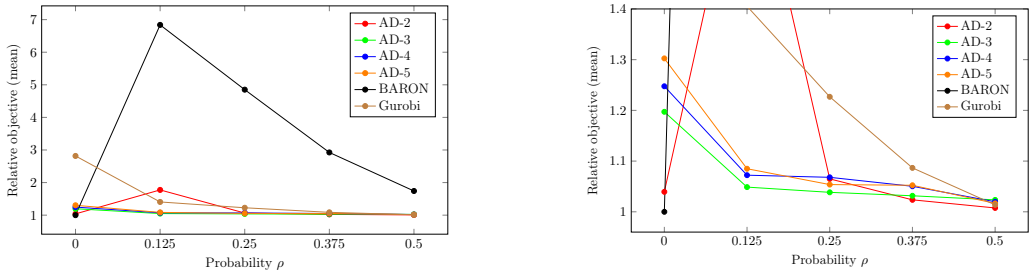
Figure 3 Relative objective values of AD-u and QCQP solvers for BoxQPcc instances.

For $\rho = 0.25$, a corresponding performance profile is depicted in Figure 3c. The adaptive discretization algorithms still yield the best objectives with their results relatively close to each other and AD-3 being the strongest and AD-2 being the weakest. However, AD-2 is now very close behind the remaining discretization algorithms and beats both QCQP solvers by far. While BARON and Gurobi are beaten by magnitudes, Gurobi now has clearly stronger objective values than BARON.

Figure 3d shows the performance profile with relative objective values for the BoxQPcc instances with $\rho = 0.375$. While the adaptive discretization algorithms are still on top, Gurobi is now relatively close behind them and is even stronger than AD-4 and AD-5 for less than 80% of the instances. AD-2 and AD-3 reach the best results where the former is even able to find the best solution for 80% of the instances. The discretization algorithms terminate for all instances with objective values at most 1.63 times worse than the best solution while Gurobi achieves this only with objective values more than 3 times worse than the best. BARON is beaten by magnitudes by all other algorithms.

The performance profile for the BoxQPcc instances with $\rho = 0.5$ is illustrated in Figure 3e. Here, only AD-2 performs stronger than Gurobi for all percentages of instances while the remaining discretization algorithms are weaker than Gurobi for less than 97% of the instances and outperform Gurobi for 97% to 100% of the instances. While Gurobi can only guarantee a relative objective value of at most 66% worse than the best, all discretization algorithms achieve achieve a value of less than 40% while AD-2 even guarantees objective values at most 26% worst than the best ones. Again, BARON is outclassed by all other algorithms.

The above five performance profiles are summarized in Figure 4a. This figure shows for each considered probability ρ the geometric mean of the relative objective values of the corresponding 246 instances calculated by each algorithm. We see that the adaptive discretization algorithms AD-3, AD-4, and AD-5 yield the most consistent results. They achieve top results for all probabilities except $\rho = 0$ where they are beaten by AD-2 and BARON. AD-2 is also very strong for most probabilities but fails for $\rho = 0.125$. While BARON



(a) All profiles.

(b) Zoomed summary.

Figure 4 Summary of relative objective values of AD-u and QCQP for BoxQPcc instances.

only achieves competitive results for $\rho = 0$, Gurobi gets stronger with increasing probabilities with competitive objective values for $\rho = 0.5$. To evaluate the performance of the individual adaptive discretization algorithms, we show the same figure zoomed to smaller relative objective values in Figure 4b. From this figure follows that AD-3 overall performs most consistent among the adaptive discretization algorithms. AD-4 and AD-5 yield slightly weaker results than AD-3 while AD-2 performs very unstable. On the one hand, AD-2 beats the remaining discretization algorithms for $\rho \in \{0, 0.375, 0.5\}$ and, on the other hand, it performs poorly for $\rho = 0.125$.

3.4 Study II: Disjointly Constrained Bilinear Programs

Another subclass of QCQP that we consider are disjoint bilinear programs

$$\begin{aligned} \text{(DisjBLP)}: \quad \max_{x,y} \quad & x^\top Q\, y + c^\top x + d^\top y \\ \text{s.t.} \quad & A\, x \le a, \qquad B\, y \le b \end{aligned}$$

with variables $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ and inputs $Q \in \mathbb{R}^{n\times m}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}^m$, $A \in \mathbb{R}^{r\times n}$, $B \in \mathbb{R}^{s\times m}$, $a \in \mathbb{R}^r$ and $b \in \mathbb{R}^s$. These are called *disjoint* because they are bipartite due to the variables being partitioned into two sets x and y, and the feasible set is the Cartesian product of a polyhedron in x-space and a polyhedron in y-space.

We generate 60 DisjBLP test instances following the procedure in [26]. For all generated instances, we use the parameters $\delta = 2.5$, $\rho = 1.5$, and randomized them with Householder matrices defined by unit vectors with random numerators and a denominator of 1000. The size of the instances depends on the parameters κ_1 and κ_2 where the total number of variables and constraints is equal to $\kappa_1 + 3\,\kappa_2$ and $\kappa_1 + 5\,\kappa_2$. Here, we generate six classes of DisjBLP instances where each class contains 10 instances with size $\kappa_1 \in \{50, 100, 150, 200, 300, 400\}$ and $\kappa_2 = 2\,\kappa_1$. For each instance, we used the random seed $\kappa_1 + \kappa_2 + \delta + \rho + k$ where $k \in \{0, \ldots, 9\}$ is the number of the instance. After the instance generation, we applied LP based bound tightening on all variables. Moreover, we discovered that the kernel problem 2 described by [26] does not have the optimal solution stated by them in Property 6. Therefore, we added the constraint $y \ge 1$ to kernel problem 2 to fix this issue.

The adaptive algorithm AD-u for $u \in \{5, 6\}$ is compared with the QCQP solvers BARON and Gurobi on the 60 generated instances. Here, we only use greater discretization sizes since the smaller sizes $u \in \{2, 3, 4\}$ struggle to find competitive feasible solutions for many instances due to the randomized bounds on the variables. Figure 5 presents a performance profile with the relative optimality gaps of the above algorithms. This figure shows that

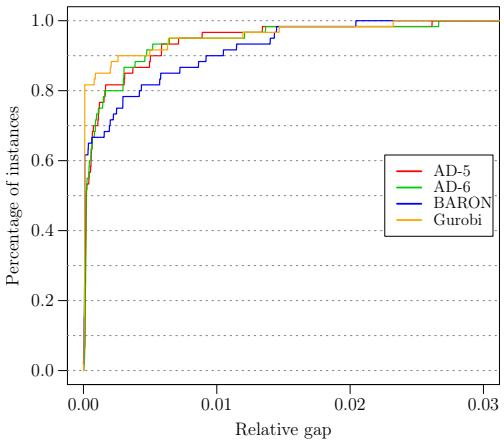


Figure 5 Relative gaps of `AD-5`, `AD-6`, and the QCQP solvers for DisjBLP instances.

all considered algorithms perform relatively similar on the DisjBLP instances as the scale of the relative gaps only reaches to 3%. `Gurobi` yields slightly better objective values than the rest while `BARON` yields slightly worse. The adaptive discretization algorithms `AD-5` and `AD-6` show very similar results and are even able to compete with `Gurobi` for above 90% of the instances. On the other hand, `BARON` is able to terminate for all instances with the best gap of at most 2% while the other algorithms are very close behind. Altogether, these four algorithms perform very similar on the DisjBLP instances where `Gurobi` calculates marginally better objective values than the rest.

4 Conclusion

We presented an iterative algorithm that adaptively refines a MILP restriction of a QCQP. This restriction arises from discretizing all quadratic terms, and our adaptive step modifies the discretization of the MILP after each iteration based on the previous MILP solution. During this adaption, we only change the discretization values while the discretization sizes remain the same. Since arbitrary instances of QCQP are not always amenable to a MILP discretization approach due to the difficulty of finding a good feasible discretization to begin with, for our computational testing, we chose two problem classes of QCQP. On a large test set of 1230 instances of box-constrained quadratic programs with complementarities, we showed that our adaptive discretization algorithm calculates much better objective values than the heuristics employed in global solvers `BARON` and `Gurobi`. For 60 test instances of disjointly constrained BLPs, the solutions obtained by all methods were of similar value with a slight advantage for `Gurobi`.

References

1 Mohammed Alfaki and Dag Haugland. Comparison of discrete and continuous models for the pooling problem. In Alberto Caprara and Spyros Kontogiannis, editors, *11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 112–121, 2011. `doi:10.4230/OASIcs.ATMOS.2011.112`.

2 Timo Berthold. RENS. *Mathematical Programming Computation*, 6(1):33–54, 2014. `doi:10.1007/s12532-013-0060-9`.

3 Timo Berthold and Ambros M Gleixner. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. *Mathematical Programming*, 144(1-2):315–346, 2014. doi:10.1007/s10107-013-0635-2.

4 Endre Boros, Peter L Hammer, and Gabriel Tavares. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *Journal of Heuristics*, 13(2):99–132, 2007.

5 Samuel Burer and Anureet Saxena. The MILP road to MIQCP. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *IMA Volumes in Mathematics and its Applications*, pages 373–405. Springer, 2012.

6 Robert Burlacu, Björn Geißler, and Lars Schewe. Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes. *Optimization Methods and Software*, 35(1):37–64, 2020.

7 C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. Experiments with a feasibility pump approach for nonconvex MINLPs. In P. Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 350–360. Springer, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-13193-6_30.

8 Claudia D'Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. A storm of feasibility pumps for nonconvex MINLP. *Mathematical Programming*, 136(2):375–402, 2012. doi:10.1007/s10107-012-0608-x.

9 Sanjeeb Dash, Oktay Günlük, and Robert Hildebrand. Binary extended formulations of polyhedral mixed-integer sets. *Mathematical Programming*, 170(1):207–236, 2018. doi:10.1007/s10107-018-1294-0.

10 Santanu S Dey and Akshay Gupte. Analysis of MILP techniques for the pooling problem. *Operations Research*, 63(2):412–427, 2015. doi:10.1287/opre.2015.1357.

11 Fabio Furini, Emiliano Traversi, Pietro Belotti, Antonio Frangioni, Ambros Gleixner, Nick Gould, Leo Liberti, Andrea Lodi, Ruth Misener, Hans Mittelmann, et al. QPLIB: a library of quadratic programming instances. *Mathematical Programming Computation*, 11(2):237–265, 2019.

12 Laura Galli and Adam N Letchford. A binarisation heuristic for non-convex quadratic programming with box constraints. *Operations Research Letters*, 46(5):529–533, 2018.

13 S. Goderbauer, B. Bahl, P. Voll, M.E. Luebbecke, A. Bardow, and A.M.C.A. Koster. An adaptive discretization MINLP algorithm for optimal synthesis of decentralized energy supply systems. *Computers & Chemical Engineering*, 95:38–48, 2016. doi:10.1016/j.compchemeng.2016.09.008.

14 Akshay Gupte, Shabbir Ahmed, Myun S. Cheon, and Santanu S Dey. Solving mixed integer bilinear problems using MILP formulations. *SIAM Journal on Optimization*, 23(2):721–744, 2013. doi:10.1137/110836183.

15 Akshay Gupte, Shabbir Ahmed, Santanu S. Dey, and Myun Seok Cheon. Relaxations and discretizations for the pooling problem. *Journal of Global Optimization*, 67(3):631–669, 2017. doi:10.1007/s10898-016-0434-4.

16 Scott P Kolodziej, Ignacio E Grossmann, Kevin C Furman, and Nicolas W Sawaya. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. *Computers & Chemical Engineering*, 53:122–142, 2013.

17 Arie M. C. A. Koster and Sascha Kuhnke. An adaptive discretization algorithm for the design of water usage and treatment networks. *Optimization and Engineering*, 20(2):497–542, June 2019. doi:10.1007/s11081-018-9413-6.

18 Leo Liberti, Nenad Mladenović, and Giacomo Nannicini. A recipe for finding good solutions to MINLPs. *Mathematical Programming Computation*, 3(4):349–390, 2011. doi:10.1007/s12532-011-0031-y.

19 Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. *Journal of Global Optimization*, 74(4):639–675, 2019.

20 Giacomo Nannicini and Pietro Belotti. Rounding-based heuristics for nonconvex MINLPs. *Mathematical Programming Computation*, 4(1):1–31, 2012. doi:10.1007/s12532-011-0032-x.

21 Carlos J Nohra, Arvind U Raghunathan, and Nikolaos Sahinidis. Spectral relaxations and branching strategies for global optimization of mixed-integer quadratic programs. *SIAM Journal on Optimization*, 31(1):142–171, 2021.

22 Foad Mahdavi Pajouh, Balabhaskar Balasundaram, and Oleg A Prokopyev. On characterization of maximal independent sets via quadratic optimization. *Journal of Heuristics*, 19(4):629–644, 2013. doi:10.1007/s10732-011-9171-5.

23 Jaehyun Park and Stephen Boyd. General heuristics for nonconvex quadratically constrained quadratic programming. arXiv Preprint, May 2017. arXiv:1703.07870.

24 V. Pham, C. Laird, and M. El-Halwagi. Convex hull discretization approach to the global optimization of pooling problems. *Industrial and Engineering Chemistry Research*, 48(4):1973–1979, 2009.

25 Wim van Ackooij, Welington de Oliveira, and Yongjia Song. Adaptive partition-based level decomposition methods for solving two-stage stochastic programs with fixed recourse. *INFORMS Journal on Computing*, 30(1):57–70, 2018. doi:10.1287/ijoc.2017.0765.

26 Luis N Vicente, Paul H Calamai, and Joaquim J Júdice. Generation of disjointly constrained bilinear programming test problems. *Computational Optimization and Applications*, 1(3):299–306, 1992.