QPLIB: a library of quadratic programming instances

Fabio Furini, Emiliano Traversi, Pietro Belotti, Antonio Frangioni, Ambros Gleixner, Nick Gould, Leo Liberti, Andrea Lodi, Ruth Misener, Hans Mittelmann, et al.

Mathematical Programming Computation manuscript No.
(will be inserted by the editor)

QPLIB: A Library of Quadratic Programming Instances

Fabio Furini · Emiliano Traversi · Pietro Belotti · Antonio Frangioni · Ambros Gleixner · Nick Gould · Leo Liberti · Andrea Lodi · Ruth Misener · Hans Mittelmann · Nikolaos V Sahinidis · Stefan Vigerske · Angelika Wiegele

Received: date / Accepted: date

Fabio Furini
LAMSADE, Université Paris Dauphine, 75775 Paris, France. E-mail: fabio.furini@dauphine.fr

Emiliano Traversi
LIPN, Université de Paris 13, 93430 Villetaneuse, France. E-mail: emiliano.traversi@lipn.fr

Pietro Belotti
Xpress-Optimizer team, FICO, Birmingham UK E-mail: pietrobelotti@fico.com

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 2, 56127 Pisa, Italy, E-mail: frangio@di.unipi.it

Ambros Gleixner
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany. E-mail: gleixner@zib.de

Nick Gould
STFC-Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England. E-mail: nick.gould@stfc.ac.uk

Leo Liberti
CNRS LIX, Ecole Polytechnique, 91128 Palaiseau, France. E-mail: liberti@lix.polytechnique.fr

Andrea Lodi
Ecole Polytechnique de Montral, Canada. E-mail: andrea.lodi@polymtl.ca

Ruth Misener
Department of Computing, Imperial College London, UK. E-mail: r.misener@imperial.ac.uk

Hans Mittelmann
School of Mathematical and Statistical Sciences, Arizona State University, Tempe, AZ 85287-1804, U.S.A. E-mail: mittelmann@asu.edu

Nikolaos V Sahinidis
Departmet of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A. E-mail: sahinidis@cmu.edu

Stefan Vigerske
GAMS Software GmbH, c/o Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany. E-mail: svigerske@gams.com

Angelika Wiegele

Abstract This paper describes a new instance library for Quadratic Programming (QP), i.e., the family of continuous and (mixed)-integer optimization problems where the objective function, the constrains, or both are quadratic. QP is a very "varied" class of problems, comprising sub-classes of problems ranging from trivial to undecidable. Solution methods for QP are very diverse, ranging from entirely combinatorial ones to completely continuous ones, including many for which both aspects are fundamental. Selecting a set of instances of QP that is at the same time not overwhelmingly onerous but sufficiently challenging for the many different interested communities is therefore important. We propose a simple taxonomy for QP instances that leads to a systematic problem selection mechanism. We then briefly survey the field of QP, giving an overview of theory, methods and solvers. Finally, we describe how the library was put together, and detail its final contents.

Keywords Instance Library, Quadratic Programming

Mathematics Subject Classification (2000) 90C06 · 90C25

1. Introduction

Quadratic Programming (QP) problems—mathematical optimization problems for which the objective function [150], the constraints [151], or both are polynomial function of the variables of degree two—include a notably diverse set of different instances. This is not surprising, given the vast scope of practical applications of such problems, and of solution methods designed to solve them [73]. Depending on the specifics of the formulation, solving a QP may require primarily combinatorial techniques, ideas rooted in nonlinear optimization principles, or a mix of the two. In this sense, QP is arguably one of the classes of problems where collaboration between the communities interested in combinatorial and in nonlinear optimization is most needed, and potentially fruitful.

However, this diversity also implies that QP means very different things to different researchers. This is illustrated by the fact that the class of problems that we simply refer to here as "QP" might more accurately be called Mixed-Integer Quadratically-Constrained Quadratic Programming (MIQCQP) in the most general case. It is, therefore, perhaps not surprising that, unlike for "simpler" problems classes [88], there has been, to date, no single library devoted to all different kinds of instances of QP. While several specialised libraries devoted to particular cases of QP are available, each of them is either focussed on a particular application (a specific problem that can be modelled as a QP), or on QPs with specific structural properties that make them suitable for solution by some given class of algorithmic approaches. To the best of our knowledge, collecting a set of QP instances that is at the same time not

Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, 9020 Klagenfurt am Wörthersee, Austria. E-mail: angelika.wiegele@aau.at

overwhelmingly onerous but sufficiently challenging for the many different interested communities has not been attempted. This work constitutes a first step in this direction.

In this paper, we report the steps that have been done to collect what we consider to be a quality library of QP instances, filtering a much larger set of currently available (or specifically provided) instances in order to end up with a manageable set that still contains a meaningful sample of possible QP types. A particularly thorny issue in this process was how to select instances that are "interesting". Our choice has been to take this to mean "challenging for a significant set of solution methods". Our filtering process has then been in part based on the idea that, if a significant fraction of the solvers that can solve a QP instance do so in a "short" time, then the instance is not challenging enough to be included in the library. Conversely, if very few (maybe one) of the solvers can solve it very efficiently by exploiting some specific structure, but most other approaches cannot, then the instance should be deemed "interesting". Putting this approach into practice requires a nontrivial number of technical steps and decisions that are detailed in the paper. We hope that our work can provide useful guidelines for other researchers interested in the constructions of benchmarks for mathematical optimization problems.

A consequence of our focus is that this paper is *not* concerned with the performance of the very diverse available set of QP solvers; we will *not* report any data comparing them. The only reason that solvers are used (and, therefore, described) in this context is to ensure that the instances of the library are nontrivial—at least for a significant fraction of the current solution methods. Providing guidance about which solvers are most suited to some specific class of QPs is entirely outside the scope of our work.

1.1 Motivation

Optimization problems with quadratic constraints and/or objective function (QP) have been the subject of a considerable amount of research over the last almost seventy years. At least some of the rationale for this interest is likely due to the fact that QPs are the "least-nonlinear nonlinear problems". Hence, in particular for the convex case, tools and techniques that have been honed during decades of research for Linear Programming (LP), typically with integrality constraints (MILP), can often be extended to the quadratic case more easily than would be required to tackle general (Mixed-Integer) Nonlinear Programming ((MI)NLP) problems. This has indeed happened over-and-over again with different algorithmic techniques, such as interior-point methods, active-set methods (of which the simplex method is a prototypical example), enumeration methods, cut-generation techniques, reformulation techniques, and many others [29]. Similarly, nonconvex continuous QP is perhaps the "simplest" class of problems that require features such as spatial enumeration techniques for their solution. Hence, QPs are both a natural basis for the development of general techniques for nonconvex NLP, and a very specific class so that specialized approaches can be developed [28, 46].

In addition, QP, with continuous or integer variables, is arguably a considerably more expressive class than (MI)LP. Quadratic expressions are found, either naturally or after appropriate reformulations, in very many optimization problems [89]. Table 1 provides a certainly non-exhaustive collection of applications that lead to formulations with quadratic constraints, quadratic objective function, or both. In general, any continuous function can be approximated with arbitrary accuracy (over a compact set) by a polynomial of arbitrary degree. In turn, every polynomial can be broken down to a system of quadratic expressions. Hence, QP is, in some sense, roughly as expressive as MINLP. This is, in principle, true for MILP as well, but at the cost of much larger and much more complicated formulations. Hence, for many applications QP may represent the "sweet spot" between the effectiveness, but lower expressive power, of MILP and the higher expressive power, but much higher computational cost of MINLP.

Table 1: Application Domains of QP

Problem	Discrete	Contributions
Fundamental problems that can be formulated as MIQP		
Quadratic assignment problem‡	✓	[8, 104]
Max-cut	✓	[93, 125]
Maximum clique‡	✓	[24]
Computational chemistry & Molecular biology		
Zeolites		[75]
Computational geometry		
Layout design	✓	[7, 32, 41]
Maximizing polygon dimensions		[9–13]
Packing circles‡	✓	[53, 59, 79, 134]
Nesting polygons		[85, 124]
Cutting ellipses		[86]
Finance		
Portfolio optimization	✓	[39, 53, 56–58, 84, 102, 106, 118, 127]
Process networks		
Crude oil scheduling	✓	[97–99, 111, 112]
Data reconciliation	✓	[129]
Multi-commodity flow	✓	[135]

‡Applications with many manuscripts cite reviews and recent works

continued

Table 1 (Application Domains of QP) continued

Problem	Discrete	Contributions
Quadratic network design	✓	[53, 59]
Multi-period blending	✓	[91, 92]
Natural gas networks	✓	[77, 100, 101]
Pooling‡	✓	[4, 33, 38, 49, 107, 108, 117, 119, 130]
Open-pit mine scheduling	✓	[22]
Reverse osmosis	✓	[131]
Supply chain	✓	[116]
Water networks‡	✓	[3, 14, 26, 35, 61, 67, 83, 87, 123, 141]
Robotics		
Traveling salesman problem with neighborhoods	✓	[62]
Telecommunications		
Delay-constrained routing	✓	[54, 55]
Energy		
Unit-commitment	✓	[53, 56, 58, 136]
Data confidentiality		
Controlled Tabular Adjustment	✓	[34]
Trust-region methods		
Trust-region subproblem		[2, 48, 68, 72, 76, 126]
PDE-constrained optimization		
Optimal control problem		[120, 132, 133]

‡Applications with many manuscripts cite reviews and recent works.

The structure of this paper is as follows. In §2 we review the basic notion of QP. In particular, §2.1 sets out the notation, §2.2 proposes a new taxonomy of QP that helps us in discussing the (very) different classes of QPs, and §2.3 very briefly reviews the solution methods for QP and the solvers we have employed. Next §3 describes the process used to obtain the library and its results. Some conclusions are drawn in §4, after which Appendix A provides a complete description of all the instances of the library, while Appendix B describes a simple (QPLIB) file format that encodes all of our examples.

While no performance issues of solvers for QP problems are considered in this paper, we refer to the comprehensive benchmark site `http://plato.asu.edu/bench.html`. Of the result on this site, three deal exclusively with QP problems, namely the (1) large SOCP, (2) MISOCP, and the (3) MIQ(C)P benchmarks, while three others contain have partial results for such problems, namely those for (4) parallel barrier solvers on large LP/QP problems, (5) AMPL-NLP and (6) MINLP. Benchmarks (1, 2 & 4) contain only convex instances, while the others include nonconvex ones. Global optima are obtained by several of the solvers in benchmarks (3 & 5), while all solvers in the latest addition (6) compute global optima. Benchmark (6) is based on MINLPLib 2 [144], a collection of currently 1388 instances. In order to give a first representative impression of solver performance, care was taken there to reduce the number of instances and allow all solvers to finish in a reasonable time. More than half of the selected instances are QP or QCP. For details we refer to `http://plato.asu.edu/ftp/minlp.html`.

2. Quadratic Programming in a nutshell

2.1 Notation

In mathematical optimization, a Quadratic Program (QP) is an optimization problem in which either the objective function, or some of the constraints, or both, are quadratic functions. More specifically, the problem has the form

$$
\begin{aligned}
\min \text{or} \max \quad & \tfrac{1}{2}x^\top Q^0 x + b^0 x + q^0 & \\
\text{such that} \quad & c_l^i \le \tfrac{1}{2}x^\top Q^i x + b^i x \le c_u^i & i \in \mathcal{M}, \\
& l_j \le x_j \le u_j & j \in \mathcal{N}, \\
\text{and} \quad & x_j \in \mathbb{Z} & j \in \mathcal{Z},
\end{aligned}
$$

where

- $\mathcal{N} = \{1, \ldots, n\}$ is the set of (indices) of variables, and $\mathcal{M} = \{1, \ldots, m\}$ is the set of (indices) of constraints;
- $x = [x_j]_{j=1}^n$ is a finite vector of real variables;
- Q^i for $i \in \{0\} \cup \mathcal{M}$ are symmetric $n \times n$ real (Hessian) matrices: since one is only interested in the value of quadratic forms of the type $x^\top Q^i x$, symmetry can be assumed without loss of generality by just replacing off diagonal pairs Q^i_{hk} and Q^i_{kh} with their average $(Q^i_{hk} + Q^i_{kh})/2$;
- b^i, c_u^i, c_l^i for $i \in \{0\} \cup \mathcal{M}$, and q^0 are, respectively, real n-vectors and real constants;
- $-\infty \le l_j \le u_j \le \infty$ are the (extended) real lower and upper bounds on each variable x_j for $j \in \mathcal{N}$;
- $\mathcal{M} = \mathcal{Q} \cup \mathcal{L}$ where $Q^i = 0$ for all $i \in \mathcal{L}$ (i.e., these are the linear constraints, as opposed to the truly quadratic ones); and
- the variables in $\mathcal{Z} \subseteq \mathcal{M}$ are restricted to only attain integer values.

Due to the presence of integrality requirements on the variables and of quadratic constraints, this class of problems is often referred to as Mixed-Integer Quadratically Constraint Quadratic Program (MIQCQP). It will be sometimes useful to refer to the (sub)set $\mathcal{B} = \{\, j \in \mathcal{Z} : l_j = 0, u_j = 1 \,\} \subseteq \mathcal{Z}$ of the binary variables, and to $\mathcal{R} = \mathcal{N} \setminus \mathcal{Z}$ as the set of continuous ones. Similarly, it will be sometimes useful to distinguish the (sub)set $\mathcal{X} = \{ j : l_j > -\infty \vee u_j < \infty \}$ of the box-constrained variables from $\mathcal{U} = \mathcal{N} \setminus \mathcal{X}$ of the unconstrained ones (in the sense that finite bounds are not explicitly provided in the data of the problem, although they may be implied by the other constraints).

The relative flexibility offered by quadratic functions, as opposed e.g. to linear ones, allows several reformulation techniques to be applicable to this family of problems in order to emphasize different properties of the various components. Some of these reformulation techniques will be commented later on; here we remark that, for instance, integrality requirements, in particular in the form of binary variables could always be "hidden" by introducing (nonconvex) quadratic constraints utilizing the celebrated relationship $x_j \in \{0, 1\} \iff x_j^2 = x_j$. Therefore, when discussing these problems some effort has to be made to distinguish between features that come from the original model, and those that can be introduced by reformulation techniques in order to extract (and algorithmically exploit) specific properties.

In the rest of this paper, we shall sometimes refer to *exact solutions* of quadratic programs. In view of the fact that their solutions may be irrational, this notation deserves a comment. If the decision version of the problem being referred to is in NP (e.g. LP, MILP, QP [143]), then the assumption is that all rational numbers can be represented exactly by a Turing Machine (TM). If there is no known proof that the problem being solved (or its decision version) is in NP, then there are four main approaches:

1. finding a representable solution x' such that $\|x' - x^*\|_\infty \leq \varepsilon$, where x^* is the true solution, $\varepsilon > 0$ is given, and representable means having a polynomially sized description length (in function of the instance size) [80];
2. using the *Thom encoding* of an algebraic number [15, Prop. 2.28] (limited to problems involving polynomials);
3. using the optimality gap: finding a representable solution x' such that $|f(x') - f(x^*)| \leq \varepsilon$, where f is the objective function, x^* is the true solution, $\varepsilon > 0$ is given (limited to optimization problems);
4. using a computational model according to which every elementary computation on the reals takes $O(1)$ and returns a precise result [23, p. 24].

Approach 3 in the list above is the one most often used in computational papers, including the present one.

2.2 Classification

Despite the apparent simplicity of the definition given in §2.1, Quadratic Programming instances can be of several rather different "types" in practice, depending on fine details of the data. In particular, many algorithmic approaches can only be applied to QP when the data of the problem has specific properties. A taxonomy of QP instances should thus strive to identify the set of properties that an instance should have in order to apply the most relevant computational methods. However, the sheer number of different existing approaches, and the fact that new ones are frequently proposed, makes it hard to provide a taxonomy that is both simple and covers all possible special cases. This is why, in this paper, we propose an approach that aims at finding a good balance between simplicity and coverage of the main families of computational methods.

2.2.1 Taxonomy

Our taxonomy is based on a three-fields code of the form "OVC", where O indicates the type of objective function considered, V records the types of variables, and C designates the types of constraints imposed on the variables. The fields can be given the following values:

- objective function: (L)inear, (D)iagonal convex (if minimization) or concave (if maximization) quadratic, (C)onvex (if minimization) or (C)oncave (if maximization) quadratic, (Q)uadratic (all other cases);
- variables: (C)ontinuous only, (B)inary only, (M)ixed binary and continuous, (I)nteger (including binary) only, (G)eneral (all other cases);
- constraints: (N)one, (B)ox, (L)inear, (D)iagonal convex quadratic, (C)onvex quadratic, nonconvex (Q)uadratic. Note that (D) and (C) are intended to mean that either Q^i is positive semidefinite and $c^i_l = -\infty$, or Q^i is negative semidefinite and $c^i_u = \infty$. Note that (positive or negative) definiteness of Q^i is a sufficient, but not in general necessary, condition for convexity. As detailed in §3.3, in our taxonomy we mark the constraints "C" based on the sufficient condition alone, the rationale of this choice being discussed in §2.2.2. Quadratic constraints with both finite bounds cannot ever be convex (unless $Q^i = 0$, i.e., they are not "truly" quadratic constraints).

The wildcard "*" will be used below to indicate any possible choice, and lists of the form "$\{X, Y, Z\}$" will indicate that the value of the given field can freely attain any of the specified values.

The ordering of the values in the previous lists is not irrelevant; in general, problems become "harder" when going from left to right. More specifically, for the O and C fields the order is that of *strict* containment between problem classes: for instance, linear objective functions are strictly a special case of diagonal convex quadratic ones (by allowing the diagonal elements all to be zero), the latter are a strict subset of general convex quadratic objectives (by allowing the off-diagonal elements all to be zero), and these are strictly subsets of general nonconvex quadratic ones (since these permit any symmetric Hessian including positive semidefinite ones). The only field for which the containment relationship is not a total order is V, for which only the partial orderings

$$C \subset M \subset G, \qquad B \subset M \subset G, \qquad \text{and} \qquad B \subset I \subset G$$

hold. In the following discussion we will repeatedly exploit this by assuming that, unless otherwise mentioned, when a method can be applied to a given problem, it can be applied as well to all simpler problems where the value of each field is arbitrarily replaced with a value denoting a less-general class.

We note that, although the left-to-right progression in the above text marks "harder problems", this does *not* mean that *every* instance of a hard problem is itself "difficult to solve". As is well known, problems as infinite collections of instances; and it is always possible to find subclasses that can be solved efficiently.

2.2.2 *Examples and reformulations*

We now give a general discussion about the different problem classes that our proposed taxonomy defines. For simplicity, we will assume minimization problems for the remaining of this section. Some problem classes are actually "too simple" to make sense in our context. For instance, D*B problems have only diagonal quadratic (hence separable) objective function and bound constraints; as such, they read

$$\min \left\{ \sum_{j \in \mathcal{N}} \left(\tfrac{1}{2} Q^0_j x_j^2 + b^0_j x_j \right) \;:\; l_j \leq x_j \leq u_j \quad j \in \mathcal{N} \;\;,\;\; x_j \in \mathbb{Z} \quad j \in \mathcal{Z} \right\} \;.$$

Hence, their solution only requires the independent minimization of a convex quadratic univariate function in each single variable x_j over a box constraint and possibly integrality requirements, which can be attained trivially in $O(1)$ operations (per variable) by closed-form formulæ, projection and rounding arguments. *A fortiori*, the even simpler cases *L*B*, *D*N* and *L*N* (the latter obviously unbounded unless $b^0 = 0$) will not be discussed here. Similarly, *CCN* are immediately solved by linear algebra techniques, and therefore are of no interest in this context. At the other end of the spectrum, in general QP is a hard problem. Actually, *LIQ*—linear objective function and quadratic constraints in integer variables with no finite bounds, i.e.

$$\min \left\{ b^0 x \;:\; \tfrac{1}{2} x^\top Q^i x + b^i x \leq c^i \quad i \in \mathcal{M} \;\;,\;\; x_j \in \mathbb{Z} \quad j \in \mathcal{N} \right\} \;,$$

is not only $\mathcal{NP}$-hard, but downright undecidable [82]. Hence so are the "harder" $\{C,Q\}IQ$.

It is important to note that the relationships between the different classes can be somehow blurred because reformulation techniques may allow one to move an instance from one class to another. We already mentioned that $x^2 = x \iff x \in \{0,1\}$, and in general **M**—instances with only binary and continuous variables—can be recast as **CQ*; here nonconvex quadratic constraints take the place of binary variables. More generally, this is also true for **G** as long as $\mathcal{U} = \emptyset$, as bounded general integer variables can be represented by binary ones. Hence, the nonconvexity due to binary variables can always be expressed by means of (nonconvex) quadratic constraints. The converse is also true: when only binary variables are present, all quadratic constraints can be converted into convex ones [19, 20].

Another relevant reformulation trick concerns the fact that, as soon as quadratic constraints are allowed, then there is no loss of generality in assuming a linear objective function. Indeed, any Q^{**} (C^*C) problem can always be rewritten as

$$
\begin{aligned}
&\min \ x^0 \\
&-\infty \leq \tfrac{1}{2}x^\top Q^0 x + b^0 x \leq x^0 \\
&c_l^i \leq \tfrac{1}{2}x^\top Q^i x + b^i x \leq c_u^i && i \in \mathcal{M} \\
&l_j \leq x_j \leq u_j && j \in \mathcal{N} \\
&x_j \in \mathbb{Z} && j \in \mathcal{Z}
\end{aligned}
$$

i.e., a L^*Q (L^*C) pronlem. Hence, it is clear that quadratic constraints are, in a well-defined sense, the most general situation (cf. also the result above about hardness of LIQ).

When a Q^i is positive semidefinite (PSD), i.e., the corresponding constraint/objective function is convex, general Hessians are in fact equivalent to diagonal ones. In particular, since every PSD matrix can be factorized as $Q^i = L^i(L^i)^T$, e.g. by the (incomplete) Cholesky factorization, the term $\frac{1}{2}x^T Q^i x \equiv \frac{1}{2}\sum_{j\in\mathcal{N}} {z_j^i}^2$ where ${z^i}^T = x^T L^i$. Hence, one might maintain that D** problems need not be distinguished from C** ones. However in reality, this is only true for "complicated" constraints but not for "simple" ones, because the above reformulation technique introduces additional linear constraints, ${L^i}^T x - z^i = 0$. Indeed, while C^*L (and, a fortiori, $C^*\{C,Q\}$) can always be brought to D^*L ($D^*\{C,Q\}$), using the above technique C^*B becomes D^*L, which may be significantly different from D^*B. In practice, a diagonal convex objective function under linear constraints is found in many applications (e.g., [53, 56, 58, 59]), so that it still makes sense to distinguish the D^*L case where the objective function is "naturally" separable from that where separability is artificially introduced.

Furthermore, as previously remarked, a not (positive or negative) definite Q^i does not necessarily correspond to a nonconvex feasible region. For instance, it is well-known that Second-Order Cone Programs have convex feasible regions; when represented in terms of quadratic constraints, however, they correspond to Q^i in one negative eigenvalue. In our taxonomy we still consider the corresponding instances as $**Q$ ones, with no attempt to detect the different special structures that actually correspond to convex feasible regions. Although this may lead to classify as "potentially nonconvex" some instances that are in fact convex, our choice is justified by the fact that not all QP solvers are capable of detecting and exploiting these structures, which means that the instance can actually be treated as a nonconvex one even if it is not.

2.2.3 QP classes

The proposed taxonomy can then be used to describe the main classes of QP according to the type of algorithms that can be applied for their solution:

- *Linear Programs LCL* and *Mixed-Integer Linear Programs LGL* have been subject of an enormous amount of research and have their well-established instance libraries [88], so they will not be explicitly addressed here.

- *Convex Continuous Quadratic Programs CCC* can be solved in polynomial time by Interior-Point techniques [152]; the simpler *CCL* can also be solved by means of "simplex-like" techniques, usually referred to as active-set methods [42]. Actually, a slightly larger class of problems can be solved with Interior-Point methods: those that can be represented by Second-Order Cone Programs. When written as QPs the corresponding Q^i may not be positive semidefinite, but nonetheless such problems can be efficiently solved. Of course just as for *LCL*, these problems may still require considerable computational effort when the size of the instance grows. In this sense, like in the linear case, there is a significant distinction between solvers that need all the data of QP to work, and those that are "matrix-free", i.e., only require the application of simple operations (typically, matrix-vector products) with the problem data. While when building our instance library we never exploited such characteristics, since they are not amenable to standard modelling tools, but this may be relevant for the solution of very-large-scale *CIC*.
- *Nonconvex Continuous Quadratic Programs QCQ* are generally $\mathcal{NP}$-hard, even if the constraints are very specific (*QCB*) and only a single eigenvalue of Q^0 is negative [78]. They therefore require enumerative techniques, such as spatial Branch&Bound [16, 52], to be solved to optimality. Of course, local approaches are available that are able to efficiently provide saddle points (hopefully, local optima) of the *CQC*, but providing global guarantees about the quality of the obtained solutions is challenging. In our library we have specifically focused on *exact* solution of the instances.
- *Convex Integer Quadratic Programs CGC* are, in general, $\mathcal{NP}$-hard, and therefore require enumerative techniques to be solved. However, convexity of the objective function and constraints implies that efficient techniques (see *CCC*) can be used at least to solve continuous relaxations. The general view is that *CGC* are not, all other things being equal, substantially more difficult than *LGL* to solve, especially if the objective function and/or the constraints have specific properties (e.g., *DGL*, *CGL*). Often integer variables are in fact binary ones, so several *CCC* models are $C\{B,M\}C$ ones. In practice binary variables are considered to lead to somewhat easier problems than general integer ones (cf. the cited result about hardness of unbounded integer quadratic programs), and several algorithmic techniques have been specifically developed for this special case. However, the general approaches for *CBC* are basically the same as for *CGC*, so there is seldom the need to distinguish between the two classes as far as solvability is concerned, although matters can be different regarding actual solution cost. Programs with only binary variables *CBC* can be easier than mixed-binary or integer ones $C\{M,I\}C$ because some techniques are specifically known for the binary-only case, cf. the next point [20]. Absence of continuous variables, even in the presence of integer ones *CIC*, can also lead to specific techniques [19].
- *Nonconvex Binary Quadratic Programs* $QB\{B, N, L\}$ obviously are $\mathcal{NP}$-hard. However, the special nature of binary variables combined with

quadratic forms allows for quite specific techniques to be developed, one of which is the reformulation of the problem as a *LBL*. Also, many well-known combinatorial problems can be naturally reformulated as problems of this class, and therefore a considerable number of results have been obtained by exploiting specific properties of the set of constraints [105, 125].
- *Nonconvex Integer Quadratic Programs QGQ* is the most general, and therefore is the most difficult, class. Due to the lack of convexity even when integrality requirements are removed, solution methods must typically combine several algorithmic ideas, such as enumeration (distinguishing the role of integral variables from that of continuous ones involved into nonconvex terms) and techniques (e.g., outer approximation, semidefinite programming relaxation, ...) that allow the efficient computation of bounds. As in the convex case, *QBQ*, *QMQ*, and *QIQ* can benefit from more specific properties of the variables [27, 40].

This description is deliberately coarse; each of these classes can be subdivided into several others on the grounds of more detailed information about structures present in their constraints/objective function. These can have a significant algorithmic impact, and therefore can be of interest to researchers. Common structures are, e.g., network flows [53–55, 59, 135] or knapsack-type linear constraints [53, 59, 60], and semi-continuous variables [53–59], or the fact that a nonconvex quadratic objective function/constraint can be reformulated as a second-order cone (hence, convex) one [53–55, 58, 59]. It would be very hard to collect a comprehensive list of all types of structures that might be of interest to any individual researcher, since these are as varied as the different possible approaches for specialized sub-classes of QP. For this reason we do not attempt such a more refined classification, and limit ourselves to the coarser one described in this section.

2.3 Solution Methods and Solvers

In this section we provide a quick overview of existing solution methods for QP, restricting ourselves to these implemented by the set of solvers considered in this paper (see §2.3.1). For each approach we briefly describe the formulation they address according to the classification set out in §2.2. We remark that many solvers implement more than one algorithm, which the user can choose at runtime. Moreover, algorithms are typically implemented in different ways within different solvers, so that the same conceptual algorithm can sometimes yield different results or performance measures on the same instances.

Solution methods for QP can be broadly organized in four categories [115]: *incomplete*, *asymptotically complete*, *complete*, and *rigorous*.

- *Incomplete* methods are only able to identify solutions, often locally optimal according to a suitable notion, and may even fail to find one even when one exists; in particular, they are typically unable to determine that an instance has no solution.

– *Asymptotically complete* methods can find a globally optimal solution with probability one in infinite time, but again they cannot prove that a given instance is infeasible.
– *Complete* methods find an approximate globally optimal solution within a prescribed optimality tolerance within finite time, or prove that none such exists (but see §2.3.4 below); they are often referred to as *exact* methods in the computational optimization community.
– *Rigorous* methods find globally optimal solutions within given tolerances even in the presence of rounding errors, except for "near-degenerate cases". Since none of the solvers we are using can be classified as rigorous, we limit ourselves to declaring solvers complete.

We refer the interested reader to [18] and [96] for further details on the solution methods.

2.3.1 Solvers

When compiling QPLIB, we have worked with the QP solvers that come with the GAMS distribution[1]. Table 2 provides a list of these solvers, together with a classification of their algorithm, and references. For more details on the solvers, we refer to the given references, solver manuals, and the survey [30]. In the table, we mark a pair (solver, problem) with "I" if the solver accepts the problem as input but it is an incomplete solver for the problem, with "A" if it is asymptotically complete, with "C" if it is complete, and leave it blank if the solver won't accept the provided problem. When a solver implements several algorithms, we have chosen, for each problem class, the algorithm that potentially provides the "strongest" results ("C" > "A" > "I" > blank).

2.3.2 Incomplete methods

Incomplete methods are usually realized as local search algorithms, asymptotically complete methods are usually realized by meta-heuristic methods such as multi-start or simulated annealing, and complete methods for $\mathcal{NP}$-hard problems such as QP are typically realized as implicit exhaustive exploration algorithms. However, these three categories may exhibit some overlap. For example, any deterministic method for solving *QCQ* locally is incomplete in general, but becomes complete for *CCC*, since any local optimum of a convex QP is also global. Therefore, when we state that a given algorithm is incomplete or (asymptotically) complete we mean that it is so the largest problem class that the solver naturally targets, although it may be complete on specific sub-classes. For example, interior point algorithms naturally target NLPs and are incomplete on NLPs, and therefore on *QCQ*, but become complete for *CCC*. In general, all complete methods for a problem class P must be complete for any problem class $Q \subseteq P$, while a complete method for P might be incomplete for a class $R \supset P$.

[1] `https://www.gams.com`

		CGL	QGL	CGC	QGQ	CCC	QCQ
AlphaECP	[148, 149]	C	I	C	I	C	I
ANTIGONE	[109, 110]	C	C	C	C	C	C
BARON	[138–140]	C	C	C	C	C	C
BONMIN	[25]	C	I	C	I	C	I
CONOPT	[43, 44]					C	I
Couenne	[16]	C	C	C	C	C	C
Cplex	[21, 81]	C	C	C		C	
DICOPT	[47, 90, 146]	C	I	C	I	C	I
Gurobi	[128]	C		C		C	
Ipopt	[147]					C	I
Knitro	[31]	C	I	C	I	C	A
Lindo API	[103]	C	C	C	C	C	C
LGO	[121, 122]					A	A
MINOS	[113, 114]					C	I
MOSEK	[5, 6]	C		C		C	
MsNlp	[95, 142]					C	A
OQNLP	[95, 142]	A	A	A	A	C	A
SBB	[45]	C	I	C	I	C	I
SCIP	[1, 145]	C	C	C	C	C	C
SNOPT	[64, 65]					C	I
Xpress-Optimizer	[50]	C		C		C	

Table 2 Families of QP problems that can be tackled by each solver

The solvers in Table 2 which implement incomplete methods for NLPs (a problem class containing *QCQ*) are CONOPT, Ipopt, MINOS, SNOPT, and Knitro. Note that all these solvers tackle the more general class of NLP, while we use them only for the considerably more restricted class of QP. Aside from solvers provided by GAMS, there are a number of other, specialized, incomplete QP solvers, such as CQP [69], DQP [71] and OOQP [63] for convex problems, and BQPD [51], QPA [74] and QPB [36], QPC [70], SQIC [66] for nonconvex ones.

2.3.3 Asymptotically complete methods

Asymptotically complete methods do not usually require a starting point, and, if given sufficient time (infinite in the worst case) will identify a globally optimal solution with probability one. Most often, these methods are meta-heuristics, involving an element of random choice, which exploit a given (heuristic) local search procedure.

The solvers in Table 2 which implement asymptotically complete methods are OQNLP and Knitro (which apply to *QGQ*) as well as MsNlp and certain sub-solvers of LGO (which apply to *QCQ*).

2.3.4 Complete methods

Complete methods are often referred to as *exact* in a large part of the mathematical optimization community. This term has to be used with care, as it implicitly makes assumptions on the underlying computational model that may not be acceptable in all cases. For example, te decision version of *QCL* is known to be in the complexity class **NP** [143], whereas the same is not known about

LCQ, even with zero objective. On the other hand, there exists a method for deciding feasibility of systems of polynomial equations and inequalities [137], including the solution of LCQ with zero objective function.

To explain this apparent contradiction, we remark that the two statements refer to different computational models: the former is based on the Turing Machine (TM), whereas the latter is based on a computational model that allows operations on real numbers, e.g. the Real RAM (RRAM) machine [23]. Due to the potentially infinite nature of exact real arithmetic computations, exact computations on the RRAM necessarily end up being approximate on the TM. Analogously, a complete method may reasonably be called "exact" on a RRAM; however, the computers we use in practice are more akin to TMs than RRAMs, and therefore calling *exact* a solver that employs floating point computations is, technically speaking, stretching the meaning of the word. However, because the term is well understood in the computational optimization community, in the following we shall loosen the distinction between complete and exact methods, with either properties intended to mean "complete" in the sense of [115].

Nearly all of the complete solvers in Table 2 that address $\mathcal{NP}$-hard problems (i.e. those in $QGQ \smallsetminus CCC$) are based on Branch-and-Bound (BB) [94]. When the BB algorithm is allowed to branch on coordinate directions corresponding to continuous variables, it is called *spatial* BB (sBB) [17, 37]. BB algorithms require exponential time in the worst case, and their exponential behavior unfortunately often shows up in practice. They can also be used heuristically (forsaking their completeness guarantee) in a number of ways, e.g. by terminating them early. The following solvers from Table 2 implement complete BB algorithms for QGQ or some subclasses:

- ANTIGONE, BARON, Couenne, Lindo API, SCIP for QGQ;
- Cplex for QGL and CGC;
- Knitro, BONMIN, SBB, Xpress-Optimizer, Gurobi, and MOSEK for CGC.

We remark that the latter category can be used as incomplete solvers for QGQ. We also note that LGO implements an incomplete BB algorithm for QCQ by using bounds obtained from sampling.

Cutting plane approaches construct and iteratively improve a MILP (LIL) relaxation of the problem [47, 149]. The cutting planes for the MILP are generated by linearization (first-order Taylor approximation) of the nonlinearities. If the latter are convex, the MILP provides a valid lower bound for the problem. Additionally, incomplete methods can be used to provide local solutions. Therefore, these methods are complete on CGC if a complete method is used to solve the MILP. The latter is typically based on BB, which is therefore a crucial technique also for this class of approaches. Solvers in Table 2 that implement complete cutting plane methods for CGC are AlphaECP, BONMIN (in the algorithmic mode B-OA), and DICOPT.

3. Library Construction

In this section we present all the steps we performed in order to build the new instance library. In §3.1, we describe the set of gathered instances, and in §3.2 we present the features used to classify the instances. We describe the selection process used to filter the instances, and graphically present the main features of the selected instances in §3.3, while in §3.4 we provide information on how to access the test collection.

3.1 Instance Collection

In this section we describe the procedure we adopted to gather the instances. In January 2014, we issued an online call for instances using the main international mailing lists of the mathematical optimization and numerical analysis communities, reaching in this way the largest possible set of interested researchers and practitioners. The call remained open for 10 months, during which we received a large number of contributions of different nature. The instances we gathered come both from theoretical studies as well as from real-world applications.

In addition to spontaneous contribution we analysed the other generic libraries of instances available on internet and containing QP instances. Namely, the libraries from which we gathered instances are

- the `BARON` library `http://www.minlp.com/nlp-and-minlp-test-problems`;
- the `CUTEst` library `https://ccpforge.cse.rl.ac.uk/gf/project/cutest`;
- the `GAMS Performance` libraries `http://www.gamsworld.org/performance/performlib.htm`;
- the `MacMINLP` library `https://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP`;
- the `Meszaros` library `http://www.doc.ic.ac.uk/~im/00README.QP`;
- the `MINLP` library `http://www.gamsworld.org/minlp/minlplib.htm`;
- the `POLIP` library `http://polip.zib.de/pipformat.php`.

Other quadratic instances were found in online libraries devoted to specific QP problems as Max-Cut, Quadratic Assignment, Portfolio Optimization, and several others. In addition, we mention that other generic libraries exist, e.g., Conic library CBLIB (`http://cblib.zib.de`) and MIPLIB 2010 (`http://miplib.zib.de/`), to mention just a few.

At the end of this process we had gathered more than eight thousand instances. Three quarters of them contained discrete variables, while the remainder contained only continuous variables. In more detail, we gathered $\approx$ 1800 Quadratic Binary Linear (QBL) instances, $\approx$ 2000 Quadratic Continuous Quadratic (QCQ) instances, and and $\approx$ 2500 Quadratic General Quadratic (QGQ) instances. We also received $\approx$ 1000 Convex General Convex (CGC) instances. We obtained relatively fewer Quadratic Binary Quadratic (QBQ), Convex Continuous Convex (CCC) and Convex Mixed Convex (CMC) instances, ($\approx$ 150, $\approx$ 200 and $\approx$ 200 instances respectively). Finally, we found only 17 Quadratic Mixed Linear (QML) instances. In the call for instances, no specific formats requirements were imposed for the submissions.

To evaluate the instances we decided, for practical reasons, to use GAMS as common platform for all our final selection computations. For this reason, we translated all the instances we received into the GAMS format (`.gms`).

For each instance in this large starting set, we collected important characteristics which allowed us to classify the instances into the QP categories described in §2. As far as the variable types are concerned, we collected the following information:

- the number of binary variables;
- the number of integer variables; and
- the number of continuous variables.

If at least one binary or integer variable is present, then the instance is categorized as *discrete*, otherwise it is categorized as *continuous*. As far as the objective function is concerned, we gathered the following information:

- the percentage of positive and negative eigenvalues of the Hessian Q^0; and
- the density of the Hessian Q^0 (number of nonzero entries divided by the total number of entries).

The number of positive (i.e., larger than 10^{-12}) and negative (i.e., smaller than -10^{-12}) eigenvalues of Q^0 allowed us to identify the objective function type, as in presence of at least one negative (positive) eigenvalue the objective function is nonconvex (nonconcave). Finally, as far as the constraint types are concerned, we collected the following information:

- the number of linear constraints,
- the number of quadratic constraints,
- the number of convex constraints, and
- the number of variable bounds (for non-binary variables).

A constraint is considered quadratic if it contains at least one nonzero in a quadratic term (if present). Among the quadratic constraints, the ones whose Hessians have only non-negative eigenvalues (when $c_u^i < \infty$) and and non-positive eigenvalues (when $c_l^i > -\infty$) are classified as convex constraints; thus, a quadratic constraint with two sided, finite bounds is non-convex. Note that this might occasionally lead us to classify some instances that have conic constraints as non-convex ones, although their feasible region is in fact convex—fortunately, only some solvers are capable of properly exploiting this property. All this information allowed us to analyse the gathered instances and to perform the filters described in the the next paragraph.

3.2 Instance Selection

During the development of the library, a discussion ensued about the expected goals that we wished to achieve. The following four goals where finally identified:

Starting set	$\approx$ 8500 Instances	
	$\Downarrow$	
	$\approx$ 6000 Discr. Inst.	$\approx$ 2500 Cont. Inst.
First Filter	$\Downarrow$	$\Downarrow$
	$\approx$ 3000 Discr. Inst.	$\approx$ 1000 Cont. Inst.
Second Filter	$\Downarrow$	$\Downarrow$
	319 Discr. Inst.	134 Cont. inst.

Table 3 Instance filter steps

1. to represent as far as possible all the different categories of QP problems;
2. to gather "challenging" instances, i.e., ones which can not be easily solved by state-of-the-art solvers;
3. to include, for each of the categories, a set of well-diversified instances; and
4. to obtain a set of instances which is neither too small, so as to preserve statistical relevance, nor too large so as to being computationally manageable.

To achieve such goals, we performed the following two filters, applied in a cascade.

– *First Instances Filter.*
  The first filter was designed to drastically reduce the number of instances by eliminating the "easy" ones. An empirical measure for the hardness of an instance is the CPU time needed by a complete solver (cf. §2.3) to solve it to global optimality. Accordingly, for each of the gathered instance we ran the complete solvers in GAMS, which number depends on the category of the instance under consideration, cf. Table 2. We then filtered according to a first measure of computational difficulty, i.e., we discarded all instances that are solved by at least 30% of the complete solvers within a time limit of 30 seconds.
– *Second Instances Filter.*
  The goal of the second filter was to eliminate "similar" instances. We carefully analysed the instances one by one, and we clustered them according to their features; for each cluster we kept only a few representatives (e.g., very similar size, same donor,...). Finally, in order to only keep computationally challenging instances we ran a a complete solver for QGQ with a time limit of 120 seconds; all the instances which have been solved to proven optimality within this time limit were discarded.

In Table 3 we summarize the two filter steps, which allowed us to identify the final set of 319 discrete instances and 134 continuous instances.

3.3 Analysis of the final set of instances

We now analyse the features of the instances selected to be part of the library. In Table 4, we provide a global overview. The instances have been divided in *continuous* vs *discrete* and *convex* vs *non-convex*, forming in this way, a classification of 4 macro categories. As previously mentioned, an instance

Variables	Convexity	#
continuous	*convex*	32
continuous	*non-convex*	102
discrete	*convex*	31
discrete	*non-convex*	288
Total		453

Table 4 Macro classification of the final set of instances

is classified *discrete* if it contains at least a binary or integer variable, and *continuous* otherwise. On the other hand, an instance is classified as *non-convex* if the objective function is non-convex and/or at least one of the constraints is non-convex, and *convex* otherwise.

The detailed characteristics of the instances are presented in Table 5 for *discrete* instances ($*\{B,M,I,G\}*$) and in Table 6 for *continuous* ones ($*C*$). For each category, the tables report in column "#" the corresponding number of instances. It can be seen that the final set well respects the original distribution of the gathered instances among the different categories. Indeed, the discrete categories (LMQ) or (QBL) are well represented by 118 and 59 instances, respectively. Similarly, the continuous categories (LCQ) and (QCQ) are well represented by 50 and 17 instances, respectively. Moreover, the library actually covers the large majority of all possible categories of instances.

One of the nontrivial choices in our library is that we made no effort to reformulate the instances, and inserted them in the library in the very same format they have been provided to us by the original contributors. Section 2.2.2 is crucial in justifying this choice, as it shows that there are several degrees of freedom to move the instances from one class to another. Tailoring the structure of a problem to the solver is, however, a bias we do not want to add.

We now report some graphs that help in illustrating the main features of the instances. In Figure 1 (left) we plot the number of variables (horizontal axis) versus the number of constrains (vertical axis), both in logarithmic scale. Continuous instances are marked with "+", and discrete ones with "$\times$". The figure shows that the library contains a quite diverse set of instances in terms of number of variables and constraints. The maximal number of constraints is 100000, while the maximal number of variables is almost 40000. Figure 1 (right) plots the number of nonzero elements in the gradient of the objective function and the Jacobian and the number of these nonzeros corresponding to nonlinear variables, that is, it counts the appearances of variables in objectives and constraints and how often such an appearance is in a quadratic term.

Figure 2 describes how discrete and continuous variables are distributed within the instances. The instances are sorted accordingly to the total number of variables. For each instance we report the total number of variables with a "+", and the total number of discrete variables (binary or general integer) with a "$\times$". The pictures clearly show that instances with different percentages

Obj. Fun.	Variables	Constraints	#
Linear	Binary	Quadratic	9
	Mixed	Convex	15
		Quadratic	151
	Integer	Quadratic	2
	General	Quadratic	3
Convex (if min) or Concave (if max)	Binary	Linear	4
	Mixed	Linear	12
		Quadratic	6
Quadratic	Binary	None	23
		Linear	74
		Quadratic	5
	Mixed	Linear	11
		Quadratic	1
	Integer	Linear	2
	General	Quadratic	1
Total			319

Table 5 Classification of the final set of discrete instances

Obj. Fun.	Constraints	#
Linear	Convex	13
	Quadratic	52
Convex (if min) or Concave (if max)	Box	3
	Linear	16
	Quadratic	11
Quadratic	Linear	6
	Convex	3
	Quadratic	30
Total		134

Table 6 Classification of the final set of continuous instances

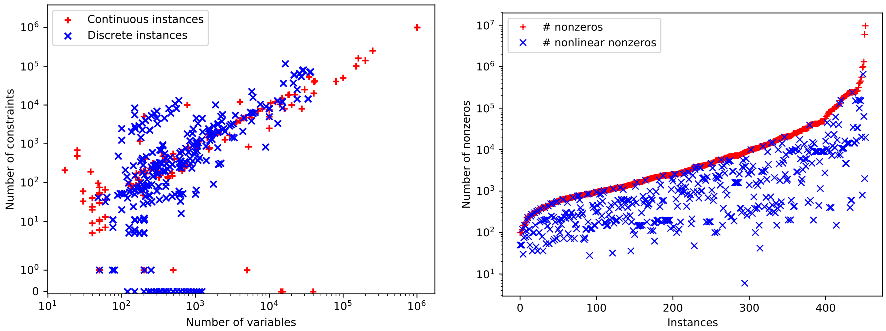

Fig. 1 Distribution of number of variables and constraints of QPLIB instances (left). Number of (nonlinear) nonzeros of QPLIB instances (right).

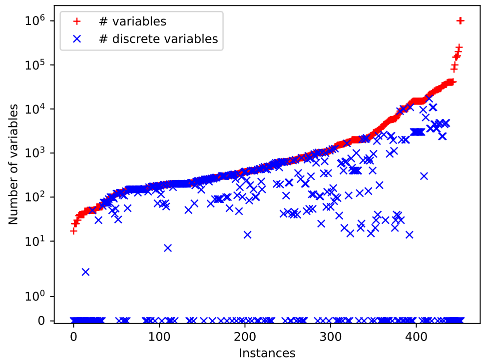

Fig. 2 Number of variables of QPLIB instances.

of integer and continuous variables are present in the library, and that these differences are well distributed across the whole spectrum of variable sizes.

Similarly, Figure 3 (left) describes how the number of linear and quadratic constraints are distributed within the instances. The instances are sorted accordingly to the total number of constraints. For each instance we report the total number of constraints with a "+" and the total number of quadratic constraints with a "×". Also, in this case, different percentages of linear and quadratic constraints are present and well-distributed across the spectrum of constraint sizes, although both medium- and large-size instances show a prevalence of lower percentages of quadratic constraints. In particular, from Figure 3 (left) we learn that while the maximum number of linear constraints exceeds 100000, the maximum number of quadratic constraints tops up at around 10000. This is, however, reasonable, considering how quadratic constraints can, in general, be expected to be much more computationally challenging than linear ones, especially if nonconvex.

Figure 3 (right) shows the instances with at least one quadratic constraint sorted according to the number of quadratic constraints (vertical axis). For

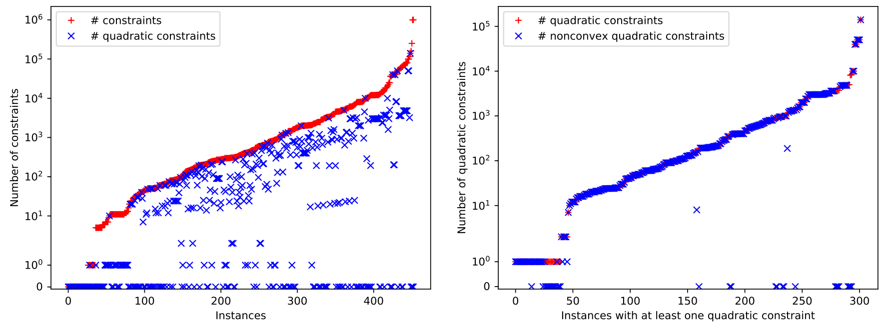

Fig. 3 Number of constraints, quadratic constraints, and nonconvex quadratic constraints of QPLIB instances.

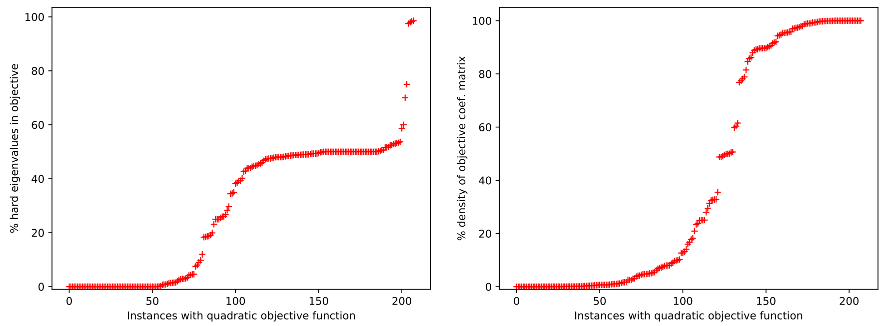

Fig. 4 "Problematic" eigenvalues (left) and density (right) of the Hessian Q^0 for QPLIB instances with a quadratic objective function.

each instance we report the total number of constraints with a "+" and the total number of nonconvex quadratic constraints with a "×". It can be seen that the majority of instances only have nonconvex constraints.

On the theme of nonconvexity, Figure 4 (left) focuses on the instances with a quadratic objective function, ordered by percentage of "problematic" (defined using a tollerance of XXX) eigenvalues in the Hessian Q^0 (vertical axis), by which we mean negative eigenvalues in case of a minimization problem and positive eigenvalues in case of a maximization problem. The instances are mostly clustered around two values. About 25% of the instances have a convex (if minimization) or concave (if maximization) objective function, i.e., they have 0% of "problematic" eigenvalues. Among the others, a vast majority has around 50% of "problematic" eigenvalues. However, instances with high or low percentages of "problematic" eigenvalues are present, too.

please define the tolerance used to identify the "Problematic" eigenvalues and change the label of the plot

Similarly, Figure 4 (right) shows the instances with a quadratic objective function sorted according to the density of the Hessian Q^0 (vertical axis). The majority of the instances have either a very low or a rather high density: indeed, about 30% of the instances have density smaller than 5%, and about 30% of

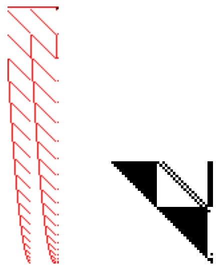

Fig. 5 Example for the sparsity pattern of the Jacobian of the constraint functions (left) and of the upper-right triangle of the Hessian of the Lagrangian function (right) for instance QPLIB_2967. The gradient of the objective function is displayed as the first row of the Jacobian matrix. Non-constant entries are shown in red.

the instances have density larger than 50%. However, also intermediate values are present.

Additional details on the instance features can be found in Appendix A.

3.4 Website

The QPLIB instances are publicly accessible at the website `http://qplib.zib.de`, which was created by extending scripts and tools initially developed for MINLPLib 2 [144]. We provide all instances in GAMS (`.gms`), AMPL (`.mod`), CPLEX (`.lp`) [81], and QPLIB (`.qplib`) formats. The latter is a new format specifically for QP instances. In comparison to more *high level* formats such as `.gms` and `.lp`, the new format offers three main advantages: it is easier to read by a stand-alone parser (provided), it typically produces smaller files, and it permits the inclusion of two-sided inequalities without needless repetition of data. See Appendix B for more details.

Beyond the instances, the website provides a rich set of metadata for each instance: the three letter problem classification (as described in §3.3), basic properties such as the number of variables and constraints of different types, the sense and convexity/concavity of the objective function, and information on the nonzero structure of the problem. In addition, we display a visualization of the sparsity patterns of the Jacobian and the Hessian matrix of the Lagrangian function. In the plots of the Jacobian nonzero pattern, the linear and nonlinear entries are distinguished by color. Figure 5 shows an example for instance QPLIB_2967.

The entire set of instances can be explored in a searchable and sortable table of selected instance features: problem classification, convexity of the continuous relaxation, number of (all, binary, integer) variables, (all, quadratic) constraints, nonzeros, hard eigenvalues in Q^0, and density of Q^0. Finally, a statistics page displays diagrams on the composition of the library according to different criteria: the number of instances according to problem type, variable and constraint types, convexity, problem size, and density. A file containing the relevant metadata for each instance can be downloaded in comma-separatedvalues (`csv`) format, so that researchers can easily compile their own subset of instances according to these statistics.

The complete library can be downloaded as one archive, which contains the website for offline browsing and exploration. In the future, we plan to extend the website by the addition of contributor information and references to the literature.

4. Conclusions

This manuscript describes the first comprehensive library of instances for Quadratic Programming (QP). Since QP comprises different and "varied" categories of problems, we proposed a classification and we briefly discuss the main classes of solution methods for QP.

We then describe the steps of the adopted process used to filter the gathered instances in order to build the new library. Our design goals were to build a library which is computationally challenging and as broad as possible, i.e., it represents the largest possible categories of QP problems, while remaining of manageable size. We have also proposed a stand-alone QP format that is intended for the convenient exchange and use of our QP instances.

We want to stress once again that we intentionally avoid to perform a computational comparison of the performances of the different solvers. Our goal is instead to provide a common test-bed of instances for practitioners and researchers in the field. This new library will hopefully serve as a point of reference to test new ideas and algorithms for QP problems.

5. Acknowledgements

We are grateful to all the donors who provided instances for the library. We gratefully acknowledge the financial support of the Gaspard Monge Program for Optimization and operations research (PGMO) and the logistic support of GAMS for having provided us with the license of their software. Finally, we would like to acknowledge the financial and networking support by the COST Action TD1207. The work of the fifth and twelfth author was supported by the Research Campus MODAL *Mathematical Optimization and Data Analysis Laboratories* funded by the Federal Ministry of Education and Research (BMBF Grant 05M14ZAM). The work of the sixth author was supported by the EPSRC grant EP/M025179/1. All responsibilty for the content of this publication is assumed by the authors.

References

1. Achterberg, T.: SCIP: Solving Constraint Integer Programs. Mathematical Programming Computation **1**(1), 1–41 (2009)

2. Adachi, S., Iwata, S., Nakatsukasa, Y., Takeda, A.: Solving the trust-region subproblem by a generalized eigenvalue problem. SIAM Journal on Optimization **27**(1), 269–291 (2017). doi:10.1137/16M1058200
3. Ahmetović, E., Grossmann, I.E.: Global superstructure optimization for the design of integrated process water networks. AIChE Journal **57**(2), 434–457 (2011)
4. Alfaki, M., Haugland, D.: A multi-commodity flow formulation for the generalized pooling problem. Journal of Global Optimization **56**(3), 917–937 (2013)
5. Andersen, E., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Mathematical Programming **95**(2), 249–277 (2003)
6. Andersen, E.D., Andersen, K.D.: The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In: H. Frenk, K. Roos, T. Terlaky, S. Zhang (eds.) High Performance Optimization, pp. 197–232. Springer US, Boston, MA (2000)
7. Anjos, M.F., Liers, F.: Global approaches for facility layout and VLSI floor-planning. In: M.F. Anjos, J.B. Lasserre (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, *International Series in Operations Research & Management Science*, vol. 166, pp. 849–877. Springer US (2012)
8. Anstreicher, K.M.: Recent advances in the solution of quadratic assignment problems. Mathematical Programming **97**(1-2), 27–42 (2003)
9. Audet, C., Guillou, A., Hansen, P., Messine, F., Perron, S.: The small hexagon and heptagon with maximum sum of distances between vertices. Journal of Global Optimization **49**(3), 467–480 (2011)
10. Audet, C., Hansen, P., Messine, F.: The small octagon with longest perimeter. Journal of Combinatorial Theory, Series A **114**(1), 135 – 150 (2007)
11. Audet, C., Hansen, P., Messine, F.: Simple polygons of maximum perimeter contained in a unit disk. Discrete & Computational Geometry **41**(2), 208–215 (2009)
12. Audet, C., Hansen, P., Messine, F., Xiong, J.: The largest small octagon. Journal of Combinatorial Theory, Series A **98**(1), 46 – 59 (2002)
13. Audet, C., Ninin, J.: Maximal perimeter, diameter and area of equilateral unit-width convex polygons. Journal of Global Optimization **56**(3), 1007–1016 (2013)
14. Bagajewicz, M.: A review of recent design procedures for water networks in refineries and process plants. Computers & Chemical Engineering **24**, 2093 – 2113 (2000)
15. Basu, S., Pollack, R., Roy, M.F.: Algorithms in real algebraic geometry. Springer, New York (2006)
16. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software **24**(4-5), 597–634 (2009)

17. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software **24**(4), 597–634 (2009)
18. Best, M.J.: Quadratic programming with computer programs, 1st edn. Advances in applied mathematics. Chapman and Hall/CRC, Taylor & Francis (2017)
19. Billionnet, A., Elloumi, S., Lambert, A.: An efficient compact quadratic convex reformulation for general integer quadratic programs. Computational Optimization & Applications **54**(1), 141–162 (2013)
20. Billionnet, A., Elloumi, S., Plateau, M.: Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. Discrete Applied Mathematics **157**(6), 1185–1197 (2009)
21. Bixby, E.R., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: MIP: Theory and practice – closing the gap. In: M.J.D. Powell, S. Scholtes (eds.) System Modelling and Optimization: Methods, Theory and Applications. 19th IFIP TC7 Conference on System Modelling and Optimization July 12–16, 1999, Cambridge, UK, pp. 19–49. Springer US, Boston, MA (2000)
22. Bley, A., Gleixner, A.M., Koch, T., Vigerske, S.: Comparing MIQCP solvers to a specialised algorithm for mine production scheduling. In: H.G. Bock, X.P. Hoang, R. Rannacher, J.P. Schlöder (eds.) Modeling, Simulation and Optimization of Complex Processes, pp. 25–39. Springer Berlin Heidelberg (2012)
23. Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers: *NP*-completeness, recursive functions, and universal machines. Bulletin of the American Mathematical Society **21**(1), 1–46 (1989)
24. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: D.Z. Du, P.M. Pardalos (eds.) Handbook of Combinatorial Optimization, pp. 1–74. Springer US (1999)
25. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization **5**(2), 186–204 (2008)
26. Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A., Toth, P.: On the optimal design of water distribution networks: A practical MINLP approach. Optimization and Engineering **13**, 219–246 (2012)
27. Buchheim, C., Wiegele, A.: Semidefinite relaxations for non-convex quadratic mixed-integer programming. Mathematical Programming **141**(1), 435–452 (2013)
28. Burer, S.: Copositive programming. In: F.M. Anjos, B.J. Lasserre (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, pp. 201–218. Springer US, Boston, MA (2012)
29. Burer, S., Saxena, A.: The MILP road to MIQCP. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, *The IMA Volumes in Mathematics and its Applications*, vol. 154, pp. 373–405. Springer (2012)

30. Bussieck, M.R., Vigerske, S.: MINLP solver software. In: J.J.C. et.al. (ed.) Wiley Encyclopedia of Operations Research and Management Science. Wiley & Sons, Inc. (2010)
31. Byrd, R.H., Nocedal, J., Waltz, R.: KNITRO: An integrated package for nonlinear optimization. In: G. di Pillo, M. Roma (eds.) Large-Scale Nonlinear Optimization, *Nonconvex Optimization and Its Applications*, vol. 83, pp. 35–59. Springer (2006)
32. Castillo, I., Westerlund, J., Emet, S., Westerlund, T.: Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. Computers & Chemical Engineering **30**(1), 54 – 69 (2005)
33. Castillo, P.A.C., Mahalec, V., Kelly, J.D.: Inventory pinch algorithm for gasoline blend planning. AIChE Journal **59**(10), 3748–3766 (2013)
34. Castro, J., Frangioni, A., Gentile, C.: Perspective Reformulations of the CTA Problem with L_2 Distances. Operations Research **62**(4), 891–909 (2014)
35. Castro, P.M., Teles, J.P.: Comparison of global optimization algorithms for the design of water-using networks. Computers & Chemical Engineering **52**, 249 – 261 (2013)
36. Conn, A.R., Gould, N.I.M., Orban, D., Toint, Ph.L.: A primal-dual trust-region algorithm for non-convex nonlinear programming. Mathematical Programming **87**(2), 215–249 (2000)
37. Dakin, R.: A tree search algorithm for mixed programming problems. Computer Journal **8**, 250–255 (1965)
38. D'Ambrosio, C., Linderoth, J., Luedtke, J.: Valid inequalities for the pooling problem with binary variables. In: O. Günlük, G.J. Woeginger (eds.) Integer Programming and Combinatoral Optimization, *Lecture Notes in Computer Science*, vol. 6655, pp. 117–129. Springer Berlin Heidelberg (2011)
39. Deng, Z., Bai, Y., Fang, S.C., Tian, Y., Xing, W.: A branch-and-cut approach to portfolio selection with marginal risk control in a linear conic programming framework. J. Sys. Sci. Sys. Eng. **22**(4), 385–400 (2013)
40. Dong, H.: Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. SIAM Journal on Optimization **26**(3), 1962–1985 (2016)
41. Dorneich, M.C., Sahinidis, N.V.: Global optimization algorithms for chip layout and compaction. Engineering Optimization **25**, 131–154 (1995)
42. Dostál, Z.: Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities. Springer Verlag, Heidelberg, Berlin, New York (2009)
43. Drud, A.: CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. Mathematical Programming **31**(2), 153–191 (1985)
44. Drud, A.S.: CONOPT – a large-scale GRG code. INFORMS Journal on Computing **6**(2), 207–216 (1994)
45. Drud, A.S.: SBB. ARKI Consulting and Development A/S (2017). URL `http://www.gams.com/24.8/docs/solvers/sbb`

46. Dür, M.: Copositive programming – a survey. In: M. Diehl, F. Glineur, E. Jarlebring, W. Michiels (eds.) Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization, pp. 3–20. Springer, Berlin, Heidelberg (2010)
47. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming **36**(3), 307–339 (1986)
48. Erway, J.B., Gill, P.E.: A subspace minimization method for the trust-region step. SIAM Journal on Optimization **20**(3), 1439–1461 (2010)
49. Faria, D.C., Bagajewicz, M.J.: A new approach for global optimization of a class of minlp problems with applications to water management and pooling problems. AIChE Jornal **58**(8), 2320–2335 (2012)
50. FICO: Xpress optimization suite (2017). URL `http://www.fico.com/en/products/fico-xpress-optimization-suite`
51. Fletcher, R.: Stable reduced hessian updates for indefinite quadratic programming. Mathematical Programming **87**(2), 251–264 (2000)
52. Floudas, C., Visweswaran, V.: A global optimization algorithm (GOP) for certain classes of nonconvex NLPs – I. Theory. Computers & Chemical Engineering **14**(12), 1397 – 1417 (1990)
53. Frangioni, A., Furini, F., Gentile, C.: Approximated Perspective Relaxations: a Project & Lift Approach. Computational Optimization and Applications **63**(3), 705–735 (2016)
54. Frangioni, A., Galli, L., Scutellà, M.: Delay-Constrained Shortest Paths: Approximation Algorithms and Second-Order Cone Models. Journal of Optimization Theory and Applications **164**(3), 1051–1077 (2015)
55. Frangioni, A., Galli, L., Stea, G.: Delay-constrained routing problems: Accurate scheduling models and admission control. Computers & Operations Research **81**, 67–77 (2017)
56. Frangioni, A., Gentile, C.: Perspective Cuts for a Class of Convex 0–1 Mixed Integer Programs. Mathematical Programming **106**(2), 225–236 (2006)
57. Frangioni, A., Gentile, C.: SDP Diagonalizations and Perspective Cuts for a Class of Nonseparable MIQP. Operations Research Letters **35**(2), 181–185 (2007)
58. Frangioni, A., Gentile, C.: A Computational Comparison of Reformulations of the Perspective Relaxation: SOCP vs. Cutting Planes. Operations Research Letters **37**(3), 206–210 (2009)
59. Frangioni, A., Gentile, C., Grande, E., Pacifici, A.: Projected Perspective Reformulations with Applications in Design Problems. Operations Research **59**(5), 1225–1232 (2011)
60. Frangioni, A., Gorgone, E.: A Library for Continuous Convex Separable Quadratic Knapsack Problems. European Journal of Operational Research **229**(1), 37–40 (2013)
61. Geissler, B., Morsi, A., Schewe, L.: A new algorithm for MINLP applied to gas transport energy cost minimization. In: M. Jünger, G. Reinelt (eds.) Facets of Combinatorial Optimization, pp. 321–353. Springer Berlin

Heidelberg (2013)
62. Gentilini, I., Margot, F., Shimada, K.: The travelling salesman problem with neighbourhood:s MINLP solution. Optimization Methods Software **28**(2), 364–378 (2013)
63. Gertz, E.M., Wright, S.J.: Object-oriented software for quadratic programming. ACM Transactions on Mathematical Software **29**(1), 58–81 (2003)
64. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Journal on Optimization **12**, 979–1006 (2002)
65. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Review **47**, 99–131 (2005)
66. Gill, P.E., Wong, E.: Methods for convex and general quadratic programming. Mathematical Programming Computation **7**(1), 71–112 (2015)
67. Gleixner, A.M., Held, H., Huang, W., Vigerske, S.: Towards globally optimal operation of water supply networks. Numerical Algebra, Control and Optimization **2**(4), 695–711 (2012)
68. Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Solving the trust-region subproblem using the lanczos method. SIAM Journal on Optimization **9**(2), 504–525 (1999)
69. Gould, N.I.M., Orban, D., Robinson, D.P.: Trajectory-following methods for large-scale degenerate convex quadratic programming. Mathematical Programming Computation **5**(2), 113–142 (2013)
70. Gould, N.I.M., Orban, D., Toint, Ph.L.: GALAHAD—a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. ACM Transactions on Mathematical Software **29**(4), 353–372 (2003)
71. Gould, N.I.M., Robinson, D.P.: A dual gradient-projection method for large-scale strictly convex quadratic problems. Computational Optimization and Applications (2016). doi:10.1007/s10589-016-9886-1
72. Gould, N.I.M., Robinson, D.P., Thorne, H.S.: On solving trust-region and other regularised subproblems in optimization. Mathematical Programming Computation **2**(1), 21–57 (2010)
73. Gould, N.I.M., Toint, Ph.L.: A quadratic programming bibliography. Numerical Analysis Group Internal Report 2000-1, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England (2000)
74. Gould, N.I.M., Toint, Ph.L.: An iterative working-set method for large-scale non-convex quadratic programming. Applied Numerical Mathematics **43**(1–2), 109–128 (2002)
75. Gounaris, C.E., First, E.L., Floudas, C.A.: Estimation of diffusion anisotropy in microporous crystalline materials and optimization of crystal orientation in membranes. J. Chem. Phys. **139**(12), 124,703 (2013)
76. Hager, W.W.: Minimizing a quadratic over a sphere. SIAM Journal on Optimization **12**(1), 188–208 (2001)
77. Hasan, M.M.F., Karimi, I.A., Avison, C.M.: Preliminary synthesis of fuel gas networks to conserve energy and preserve the environment. Industrial & Engineering Chemistry Research **50**(12), 7414–7427 (2011)

78. Hemmecke, R., Köppe, M., Lee, J., Weismantel, R.: Nonlinear integer programming. In: M. Jünger, M.T. Liebling, D. Naddef, L.G. Nemhauser, R.W. Pulleyblank, G. Reinelt, G. Rinaldi, A.L. Wolsey (eds.) 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, pp. 561–618. Springer Berlin Heidelberg (2010)
79. Hifi, M., M'Hallah, R.: A literature review on circle and sphere packing problems: Models and methodologies. Advances in Operations Research **2009**, 22 (2009)
80. Hochbaum, D.: Complexity and algorithms for nonlinear optimization problems. 4OR **3**(3), 171–216 (2005)
81. IBM ILOG: CPLEX Optimization Studio, 12.7.0 edn. (2016). URL `http://www.ibm.com/support/knowledgecenter/SSSA5P`
82. Jeroslow, R.: There cannot be any algorithm for integer programming with quadratic constraints. Operations Research **21**(1), 221–224 (1973)
83. Jeżowski, J.: Review of water network design methods with literature annotations. Industrial & Engineering Chemistry Research **49**(10), 4475 – 4516 (2010)
84. Kallrath, J.: Exact computation of global minima of a nonconvex portfolio optimization problem. In: C.A. Floudas, P.M. Pardalos (eds.) Frontiers in Global Optimization, pp. 237–254. Kluwer Academic Publishers (2003)
85. Kallrath, J.: Cutting circles and polygons from area-minimizing rectangles. Journal of Global Optimization **43**, 299 – 328 (2009)
86. Kallrath, J., Rebennack, S.: Cutting ellipses from area-minimizing rectangles. Journal of Global Optimization **59**(2-3), 405–437 (2014)
87. Khor, C.S., Chachuat, B., Shah, N.: Fixed-flowrate total water network synthesis under uncertainty with risk management. Journal of Cleaner Production **77**(0), 79 – 93 (2014)
88. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: Miplib 2010. Mathematical Programming Computation **3**(2), 103–163 (2011)
89. Kochenberger, G., Hao, J.K., Glover, F., Lewis, M., Lü, Z., Wang, H., Wang, Y.: The unconstrained binary quadratic programming problem: a survey. Journal of Combinatorial Optimization **28**(1), 58–81 (2014)
90. Kocis, G.R., Grossmann, I.E.: Computational experience with DICOPT solving MINLP problems in process systems engineering. Computers & Chemical Engineering **13**(3), 307–315 (1989)
91. Kolodziej, S.P., Castro, P.M., Grossmann, I.E.: Global optimization of bilinear programs with a multiparametric disaggregation technique. Journal of Global Optimization **57**(4), 1039–1063 (2013)
92. Kolodziej, S.P., Grossmann, I.E., Furman, K.C., Sawaya, N.W.: A discretization-based approach for the optimization of the multiperiod blend scheduling problem. Computers & Chemical Engineering **53**, 122 – 142 (2013)
93. Krislock, N., Malick, J., Roupin, F.: Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problem. ACM Transactions

on Mathematical Software **43**(4), 32:1–32:23 (2017)
94. Land, A., Doig, A.: An automatic method of solving discrete programming problems. Econometrica **28**(3), 497–520 (1960)
95. Lasdon, L., Plummer, J., Ugray, Z., Bussieck, M.: Improved filters and randomized drivers for multi-start global optimization. McCombs Research Paper Series IROM-06-06, McCombs School of Business (2006)
96. Lee, G., Tam, N., Yen, N.: Quadratic Programming and Affine Variational Inequalities: A Qualitative Study. Nonconvex Optimization and Its Applications. Springer US (2006)
97. Li, J., Li, A., Karimi, I.A., Srinivasan, R.: Improving the robustness and efficiency of crude scheduling algorithms. AIChE Journal **53**(10), 2659–2680 (2007)
98. Li, J., Misener, R., Floudas, C.A.: Continuous-time modeling and global optimization approach for scheduling of crude oil operations. AIChE Journal **58**(1), 205–226 (2012)
99. Li, J., Misener, R., Floudas, C.A.: Scheduling of crude oil operations under demand uncertainty: A robust optimization framework coupled with global optimization. AIChE Journal **58**(8), 2373–2396 (2012)
100. Li, X., Armagan, E., Tomasgard, A., Barton, P.I.: Stochastic pooling problem for natural gas production network design and operation under uncertainty. AIChE Journal **57**(8), 2120–2135 (2011)
101. Li, X., Tomasgard, A., Barton, P.I.: Decomposition strategy for the stochastic pooling problem. Journal of Global Optimization **54**(4), 765–790 (2012)
102. Lin, X., Floudas, C.A., Kallrath, J.: Global solution approach for a nonconvex MINLP problem in product portfolio optimization. Journal of Global Optimization **32**, 417–431 (2005)
103. Lin, Y., Schrage, L.: The global solver in the LINDO API. Optimization Methods & Software **24**(4–5), 657–668 (2009)
104. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. European Journal of Operational Research **176**(2), 657 – 690 (2007)
105. Loiola, E.M., de Abreu, N., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. European Journal of Operational Research **176**(2), 657 – 690 (2007)
106. Maranas, C.D., Androulakis, I.P., Floudas, C.A., Berger, A.J., Mulvey, J.M.: Solving long-term financial planning problems via global optimization. Journal of Economic Dynamics and Control **21**(8-9), 1405 – 1425 (1997)
107. Misener, R., Floudas, C.A.: Advances for the pooling problem: Modeling, global optimization, and computational studies. Applied and Computational Mathematics **8**(1), 3 – 22 (2009)
108. Misener, R., Floudas, C.A.: Global optimization of large-scale pooling problems: Quadratically constrained MINLP models. Industrial & Engineering Chemistry Research **49**(11), 5424 – 5438 (2010)

109. Misener, R., Floudas, C.A.: GloMIQO: Global Mixed-Integer Quadratic Optimizer. Journal of Global Optimization **57**(1), 3–50 (2013)
110. Misener, R., Floudas, C.A.: ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations. Journal of Global Optimization **59**(2-3), 503–526 (2014)
111. Mouret, S., Grossmann, I.E., Pestiaux, P.: A novel priority-slot based continuous-time formulation for crude-oil scheduling problem. Industrial & Engineering Chemistry Research **48**(18), 8515–8528 (2009)
112. Mouret, S., Grossmann, I.E., Pestiaux, P.: A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. Computers & Chemical Engineering **35**(12), 2750–2766 (2011)
113. Murtagh, B.A., Saunders, M.A.: Large-scale linearly constrained optimization. Mathematical Programming **14**, 41–72 (1978)
114. Murtagh, B.A., Saunders, M.A.: A projected lagrangian algorithm and its implementation for sparse nonlinear constraints. Mathematic Programming Study **16**, 84–117 (1982)
115. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. Acta Numerica **13**, 271–369 (2004)
116. Nyberg, A., Grossmann, I.E., Westerlund, T.: The optimal design of a three-echelon supply chain with inventories under uncertainty (2012). Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=157`
117. Papageorgiou, D.J., Toriello, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Fixed-charge transportation with product blending. Transportation Science **46**(2), 281–295 (2012)
118. Parpas, P., Rustem, B.: Global optimization of the scenario generation and portfolio selection problems. In: M. Gavrilova, O. Gervasi, V. Kumar, C. Tan, D. Taniar, A. Laganá, Y. Mun, H. Choo (eds.) Computational Science and Its Applications - ICCSA 2006, *Lecture Notes in Computer Science*, vol. 3982, pp. 908–917. Springer-Verlag (2006)
119. Pham, V., Laird, C., El-Halwagi, M.: Convex hull discretization approach to the global optimization of pooling problems. Industrial & Engineering Chemistry Research **48**, 1973 – 1979 (2009)
120. Pillo, G.D., Grippo, L., Lampariello, F.: A class of structured quasi-newton algorithms for optimal control problems. IFAC Proceedings Volumes **16**(8), 101 – 107 (1983). 4th IFAC Workshop on Applications of Nonlinear Programming to Optimization and Control, San Francisco, CA , USA, 20-21 June 1983
121. Pintér, J.D.: LGO – a program system for continuous and Lipschitz global optimization. In: I.M. Bomze, T. Csendes, R. Horst, P.M. Pardalos (eds.) Developments in Global Optimization, pp. 183–197. Springer US, Boston, MA (1997)
122. Pintér, J.D.: A model development system for global optimization. In: R. De Leone, A. Murli, P.M. Pardalos, G. Toraldo (eds.) High Performance Algorithms and Software in Nonlinear Optimization, pp. 301–314. Springer

US, Boston, MA (1998)
123. Ponce-Ortega, J.M., El-Halwagi, M.M., Jiménez-Gutiérrez, A.: Global optimization for the synthesis of property-based recycle and reuse networks including environmental constraints. Computers & Chemical Engineering **34**(3), 318 – 330 (2010)
124. Rebennack, S., Kallrath, J., Pardalos, P.M.: Column enumeration based decomposition techniques for a class of non-convex MINLP problems. Journal of Global Optimization **43**(2-3), 277–297 (2009)
125. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Mathematical Programming **121**(2), 307–335 (2008)
126. Rendl, F., Wolkowicz, H.: A semidefinite framework for trust region subproblems with applications to large scale minimization. Mathematical Programming **77**(1), 273–299 (1997)
127. Rios, L.M., Sahinidis, N.V.: Portfolio optimization for wealth-dependent risk preferences. Annals of Operations Research **177**, 63–90 (2010)
128. Rothberg, E.: Solving quadratically-constrained models using Gurobi (2012). URL `http://www.gurobi.com/resources/seminars-and-videos/gurobi-quadratic-constraints-webinar`
129. Ruiz, J.P., Grossmann, I.E.: Exploiting vector space properties to strengthen the relaxation of bilinear programs arising in the global optimization of process network. Optimization Letters **5**, 1–11 (2011)
130. Ruiz, M., Briant, O., Clochard, J.M., Penz, B.: Large-scale standard pooling problems with constrained pools and fixed demands. Journal of Global Optimization **56**(3), 939–956 (2013)
131. Saif, Y., Elkamel, A., Pritzker, M.: Global optimization of reverse osmosis network for wastewater treatment and minimization. Industrial & Engineering Chemistry Research **47**(9), 3060 – 3070 (2008)
132. Schittkowski, K.: Numerical solution of a time-optimal parabolic boundary-value control problem. Journal of Optimization Theory and Applications **27**(2), 271–290 (1979)
133. Stojanovic, S.: Optimal damping control and nonlinear elliptic systems. SIAM Journal on Control and Optimization **29**(3), 594–608 (1991)
134. Szabó, P.G., Markót, C.M., Csendes, T.: Global optimization in geometry – circle packing into the square. In: C. Audet, P. Hansen, G. Savard (eds.) Essays and Surveys in Global Optimization, pp. 233–265. Springer, New York (2005)
135. Tadayon, B., Smith, J.C.: Algorithms for an integer multicommodity network flow problem with node reliability considerations. Journal of Optimization Theory and Applications **161**, 506–532 (2013)
136. Tahanan, M., van Ackooij, W., Frangioni, A., Lacalandra, F.: Large-scale Unit Commitment under uncertainty. 4OR **13**(2), 115–171 (2015)
137. Tarski, A.: A decision method for elementary algebra and geometry. Tech. Rep. R-109, Rand Corporation (1951)
138. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory,

Algorithms, Software, and Applications, *Nonconvex Optimization and Its Applications*, vol. 65. Kluwer Academic Publishers (2002)
139. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical Programming **99**(3), 563–591 (2004)
140. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming **103**(2), 225–249 (2005)
141. Teles, J.P., Castro, P.M., Matos, H.A.: Global optimization of water networks design using multiparametric disaggregation. Computers & Chemical Engineering **40**, 132 – 147 (2012)
142. Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., Martí, R.: Scatter search and local NLP solvers: A multistart framework for global optimization. INFORMS Journal on Computing **19**(3), 328–340 (2007)
143. Vavasis, S.: Quadratic programming is in NP. Information Processing Letters **36**, 73–77 (1990)
144. Vigerske, S.: MINLPLib 2. In: L.G. Casado, I. García, E.M.T. Hendrix (eds.) Proceedings of the XII global optimization workshop MAGO 2014, pp. 137–140 (2014). URL `http://www.gamsworld.org/minlp/minlplib2`
145. Vigerske, S., Gleixner, A.: SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. ZIB Report 16-24, Zuse Institute Berlin (2016). urn:nbn:de:0297-zib-59377
146. Viswanathan, J., Grossmann, I.E.: A combined penalty function and outer-approximation method for MINLP optimization. Computers & Chemical Engineering **14**(7), 769–782 (1990)
147. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)
148. Westerlund, T., Lundquist, K.: Alpha-ECP, version 5.04. an interactive MINLP-solver based on the extended cutting plane method. Tech. Rep. 01-178-A, Process Design Laboratory, Åbo Akademi University, Åbo, Finland (2003)
149. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optimization and Engineering **3**(3), 253–280 (2002)
150. Wikipedia: Quadratic programming (2016). URL `https://en.wikipedia.org/wiki/Quadratic_programming`
151. Wikipedia: Quadratically constrained quadratic program (2016). URL `https://en.wikipedia.org/wiki/Quadratically_constrained_quadratic_program`
152. Wright, S.: Primal-Dual Interior-Point Method. SIAM, Philadelphia (1997)

A. Instance details

Table 7 provides detailed data on all the instances of the final library. Column "name" is the name of the instance with the prefix "QPLIB_" stripped. Column "type" is the classification of the instance according to the taxonomy from §2.2.1. Column "% h.e." provides the fraction of hard eigenvalues of Q^0, the coefficient matrix of the objective function: a positive number implies that the instance is a Q**, "0.0" implies that the instance is a C**, a blank implies that $Q^0 = 0$, i.e., the objective function is linear (hence, the instance is a L**). Column "% d." describes the density of the Q^0 matrix: a blank implies that the corresponding instance has a linear objective function. For both columns ("% n.e." and "% d."), nonzeros values below 0.1 were rounded up to 0.1. The following three columns describe the variables by reporting the number of binary ones ("# b."), general integer ones ("# i."), and continuous ones ("# c."). Finally, the last four columns describe the constraints reporting the number of linear ones ("# l."), nonconvex quadratic ones ("# q."), convex quadratic ones ("# c."), and variable bounds ("# v."). The numbering of the instances reflects the initial order in which we gathered them and the non-consecutiveness of the instance names is due to the filtering.

Table 7: Features of QPLIB instances.

		Q^0		Variables			Constraints			
name	type	% h.e.	% d.	# b.	# i.	# c.	# l.	# q.	# c.	# v.
0018	QCL	48.0	100.0	0	0	50	1	0	0	50
0031	QML	18.3	99.8	30	0	30	32	0	0	30
0032	QML	25.0	99.9	50	0	50	52	0	0	50
0067	QBL	47.5	88.9	80	0	0	1	0	0	0
0343	QCL	48.0	100.0	0	0	50	1	0	0	100
0633	QBL	58.7	98.7	75	0	0	1	0	0	0
0678	LMQ			9600	0	5537	7457	960	0	1474
0681	LMQ			72	0	143	419	48	0	200
0682	LMQ			71	0	190	501	96	0	296
0684	LMQ			101	0	260	815	128	0	408
0685	LMQ			256	0	519	1603	192	0	728
0686	LMQ			692	0	1512	4440	640	0	2200
0687	LMQ			672	0	1651	4875	800	0	2520
0688	LMQ			1964	0	3824	20568	1600	0	6256
0689	LMQ			756	0	1112	9800	288	0	1608
0690	LMQ			6428	0	10048	112400	3200	0	17376
0696	LMQ			187	0	207	390	33	0	260
0698	LMQ			55	0	63	126	15	0	56
0752	QBL	50.0	10.0	250	0	0	1	0	0	0
0911	QCQ	44.0	50.5	0	0	50	0	50	0	100
0975	QCQ	50.0	50.6	0	0	50	0	10	0	100
1055	QCQ	50.0	100.0	0	0	40	0	20	0	80
1143	QCQ	50.0	97.1	0	0	40	4	20	0	80
1157	QCQ	25.0	94.5	0	0	40	8	1	0	80
1353	QCQ	26.0	95.8	0	0	50	5	1	0	100
1423	QCQ	75.0	95.4	0	0	40	4	20	0	80
1437	QCQ	50.0	95.6	0	0	50	10	1	0	100
1451	QCQ	50.0	49.1	0	0	60	6	60	0	120
1493	QCQ	50.0	97.3	0	0	40	4	1	0	80
1507	QCQ	26.7	95.8	0	0	30	3	30	0	60
1535	QCQ	50.0	94.3	0	0	60	6	60	0	120
1619	QCQ	50.0	95.5	0	0	50	5	25	0	100
1661	QCQ	50.0	95.4	0	0	60	12	1	0	120
1675	QCQ	51.7	48.8	0	0	60	12	1	0	120
1703	QCQ	51.7	97.9	0	0	60	6	30	0	120
1745	QCQ	50.0	48.8	0	0	50	5	50	0	100
1773	QCQ	50.0	94.8	0	0	60	6	1	0	120
1886	QCQ	50.0	50.0	0	0	50	0	50	0	100
1913	QCQ	50.0	24.9	0	0	48	0	48	0	96
1922	QCQ	50.0	49.6	0	0	30	0	60	0	60
1931	QCQ	50.0	49.9	0	0	40	0	40	0	80
1940	QCQ	50.0	25.0	0	0	48	0	96	0	96
1967	QCQ	50.0	99.8	0	0	50	0	75	0	100
1976	QBQ	38.2	7.0	152	0	0	136	16	0	0
2017	QBQ	39.3	5.5	252	0	0	231	21	0	0
2022	QBQ	38.5	5.2	275	0	0	253	22	0	0
2029	QBQ	40.1	5.1	299	0	0	276	23	0	0
2036	QBQ	39.2	4.8	324	0	0	300	24	0	0
2047	LBQ			136	0	0	2040	17	0	0
2055	LBQ			153	0	0	2448	18	0	0
2060	LBQ			171	0	0	2907	19	0	0
2067	LBQ			190	0	0	3420	20	0	0
2073	LBQ			210	0	0	3990	21	0	0
2077	LBQ			231	0	0	4620	22	0	0
2085	LBQ			253	0	0	5313	23	0	0
2087	LBQ			276	0	0	6072	24	0	0
2096	LBQ			300	0	0	6900	25	0	0
2165	LMQ			683	0	1376	1366	683	0	683

Table 7: Features of QPLIB instances (continued).

		Q^0		Variables			Constraints			
name	type	% h.e.	% d.	# b.	# i.	# c.	# l.	# q.	# c.	# v.
2166	LMQ			345	0	697	690	345	0	345
2167	LMQ			61	0	131	122	61	0	61
2168	LMQ			214	0	438	428	214	0	214
2169	LMQ			297	0	608	594	297	0	297
2170	LMQ			351	0	736	702	351	0	351
2171	LMQ			150	0	305	300	150	0	150
2173	LMQ			215	0	436	430	215	0	215
2174	LMQ			768	0	1545	1536	768	0	768
2181	LMQ			90	0	190	180	90	0	90
2187	LMQ			90	0	195	180	90	0	90
2192	LMQ			90	0	200	180	90	0	90
2195	LMQ			90	0	205	180	90	0	90
2202	LMQ			90	0	185	180	90	0	90
2203	LMQ			100	0	205	200	100	0	100
2204	LMQ			110	0	225	220	110	0	110
2205	LMQ			958	0	1926	1916	958	0	958
2206	LMQ			194	0	421	388	194	0	194
2315	QBL	44.7	7.5	595	0	0	13090	0	0	0
2353	QML	50.0	23.7	147	0	93	2240	0	0	186
2357	QBL	50.0	7.8	240	0	0	2240	0	0	0
2359	QBL	44.4	4.2	306	0	0	3264	0	0	0
2416	LCQ			0	0	25	153	527	6	48
2430	LCQ			0	0	125	27	65	0	240
2445	LCQ			0	0	143	14	66	0	160
2456	LCD			0	0	5477	4131	0	1369	0
2468	LCD			0	0	14885	11203	0	3721	0
2480	LCQ			0	0	399	199	200	1	400
2482	LCD			0	0	1806	1418	0	361	0
2483	LCQ			0	0	760	40	240	0	1320
2492	QBL	25.5	86.2	196	0	0	28	0	0	0
2505	LCQ			0	0	1039	302	480	0	540
2512	QBL	46.0	77.4	100	0	0	20	0	0	0
2519	LCD			0	0	4806	3802	0	961	0
2540	LCQ			0	0	498	341	210	0	130
2546	CCQ	0.0	0.7	0	0	1015	592	400	0	15
2590	LCQ			0	0	25	93	401	0	48
2626	LCD			0	0	22327	14763	0	3721	0
2635	LCQ			0	0	176	0	188	966	0
2650	LCQ			0	0	1110	228	904	0	1072
2658	LCQ			0	0	184	57	133	0	192
2676	LCD			0	0	1445	1095	0	361	0
2693	LCQ			0	0	791	183	631	0	754
2696	QCQ	1.4	2.5	0	0	3500	1995	1500	0	5
2698	LCQ			0	0	196	36	11	0	280
2702	QML	4.6	1.2	259	0	1	212	0	0	0
2703	LCQ			0	0	799	399	400	1	800
2707	LCQ			0	0	634	151	466	0	640
2708	LMQ			108	0	526	327	30	0	520
2712	QCL	50.0	100.0	0	0	200	1	0	0	400
2714	LCQ			0	0	352	301	298	0	1
2733	QBL	25.9	89.2	324	0	0	36	0	0	0
2738	LCQ			0	0	199	99	100	1	200
2758	LCQ			0	0	303	139	112	0	140
2761	QCL	50.0	100.0	0	0	500	1	0	0	1000
2784	LCD			0	0	4501	3680	0	900	0
2819	LCQ			0	0	334	24	132	0	500
2823	LCQ			0	0	390	103	283	0	396
2834	LCQ			0	0	156	14	72	0	200
2862	LCD			0	0	40501	32640	0	8100	0
2880	QBL	48.8	90.3	625	0	0	50	0	0	0
2881	LCQ			0	0	1512	0	700	20	0
2882	LMQ			56	0	88	257	16	0	32
2894	LCQ			0	0	17	55	154	0	32
2935	LMQ			72	0	108	325	18	0	36
2957	QBL	23.1	60.3	484	0	0	44	0	0	0
2958	LMQ			42	0	70	197	14	0	28
2967	QCC	47.4	5.0	0	0	38	1	0	190	38
2981	CCQ	0.0	0.7	0	0	2015	1192	800	0	15
2987	LCQ			0	0	208	114	90	0	90
2993	LCQ			0	0	266	235	84	0	66
3029	LCD			0	0	5767	3783	0	961	0
3034	LCQ			0	0	780	40	240	0	1320
3049	QCQ	0.8	2.5	0	0	7000	3995	3000	0	5
3060	QML	0.2	6.2	48	0	792	1192	0	0	0
3080	CCQ	0.0	0.7	0	0	4015	2392	1600	0	15
3083	LCQ			0	0	243	107	126	0	120
3088	LCD			0	0	3601	2780	0	900	0
3089	LCQ			0	0	132	12	72	0	228
3105	LCD			0	0	18606	14802	0	3721	0
3120	LCQ			0	0	662	40	204	0	924
3122	QML	2.8	0.1	17136	0	3988	36703	0	0	0
3147	LCQ			0	0	419	32	108	0	550
3170	LCQ			0	0	660	40	160	0	1160
3177	LCQ			0	0	1599	799	800	1	1600
3181	LMQ			84	0	308	180	16	0	222

Table 7: Features of QPLIB instances (continued).

		Q^0		Variables			Constraints			
name	type	% h.e.	% d.	# b.	# i.	# c.	# l.	# q.	# c.	# v.
3185	LCD			0	0	18001	14560	0	3600	0
3192	LCQ			0	0	479	32	145	0	702
3225	LCQ			0	0	136	14	66	0	160
3240	LCQ			0	0	516	187	220	0	260
3247	LCQ			0	0	361	322	8	148	1
3279	LMQ			56	0	251	148	16	0	222
3297	CCQ	0.0	0.7	0	0	8015	4792	3200	0	15
3307	QBL	19.9	61.5	256	0	0	32	0	0	0
3312	LCD			0	0	41406	33002	0	8281	0
3318	LCQ			0	0	25	93	381	0	48
3326	QCQ	2.9	2.5	0	0	1750	995	750	0	5
3334	LCQ			0	0	715	40	210	0	990
3337	LCQ			0	0	297	0	198	0	396
3338	LCQ			0	0	320	26	110	0	432
3347	QBL	51.8	85.8	676	0	0	52	0	0	0
3358	LCQ			0	0	158	66	106	0	136
3361	QBL	28.3	35.5	1024	0	0	64	0	0	0
3369	LCQ			0	0	485	32	116	0	650
3380	QBL	3.4	0.1	8904	0	0	823	0	0	0
3385	LCQ			0	0	155	77	60	0	80
3387	LCQ			0	0	170	18	65	0	160
3402	QBL	47.2	81.5	144	0	0	24	0	0	0
3413	QBL	45.0	9.0	400	0	0	40	0	0	0
3416	LCQ			0	0	424	32	96	0	400
3496	LGQ			200	56	72	623	64	0	120
3502	LMQ			10920	0	2090	209	3130	0	2090
3505	LMQ			201	0	603	605	2	0	2
3506	QBN	48.4	0.8	496	0	0	0	0	0	0
3508	LMQ			2450	0	891	99	1332	0	891
3510	LMQ			105	0	919	4568	21	0	38
3511	LMQ			2450	0	3292	4950	1283	0	891
3512	LMQ			72	0	119	403	24	0	152
3513	LMQ			123	0	1897	2569	763	0	1880
3514	LMQ			15	0	1800	960	900	0	1800
3515	LMQ			352	0	382	720	48	0	540
3522	LMQ			42	0	588	212	42	0	588
3523	QML	50.0	13.2	155	0	27	1456	0	0	54
3524	LMQ			132	0	949	3165	192	0	288
3525	QGQ	47.5	0.1	0	1662	87	52	39	0	3324
3529	LMQ			38	0	1488	1580	544	0	800
3533	LMQ			240	0	143	176	25	0	8
3547	DML	0.0	16.7	462	0	1536	3137	0	0	6
3549	LMQ			650	0	1033	1326	583	0	408
3554	QML	12.0	100.0	14	0	370	556	0	0	0
3562	LIQ			7	56	0	35	7	0	112
3565	QBN	47.8	1.4	276	0	0	0	0	0	0
3580	LMQ			108	0	24	45	18	0	24
3582	LMQ			184	0	32	60	24	0	32
3584	QBL	43.9	8.0	528	0	0	10912	0	0	0
3587	QBL	50.0	12.7	240	0	0	46	0	0	0
3588	LMQ			600	0	392	49	584	0	392
3592	QML	50.0	0.2	225	0	225	255	0	0	0
3596	LMQ			104	0	921	1054	132	0	428
3600	LMQ			112	0	16	45	12	0	16
3605	LMQ			160	0	1076	4315	192	0	288
3614	QBL	50.0	12.7	210	0	0	44	0	0	0
3620	LMQ			187	0	3285	4071	1344	0	3398
3621	LMQ			109	0	1655	2213	665	0	1624
3622	LMQ			25	0	2000	1040	1000	0	2000
3624	LMQ			40	0	6400	3280	3200	0	6400
3625	LMQ			46	0	598	191	46	0	598
3631	LMQ			750	0	143	210	25	0	8
3642	QBN	48.9	0.4	1035	0	0	0	0	0	0
3643	LGQ			216	72	72	825	68	0	152
3645	LMQ			101	0	302	304	1	1	1
3646	LMQ			20	0	2000	1050	1000	0	2000
3648	LMQ			40	0	680	306	40	0	80
3650	QBN	48.8	0.4	946	0	0	0	0	0	0
3651	LMQ			137	0	2139	2942	861	0	2136
3659	LGQ			0	960	4577	5537	960	0	1474
3661	LMQ			10816	0	12997	11024	3221	0	12906
3662	LMQ			144	0	32	55	24	0	32
3670	LMQ			54	0	864	305	54	0	108
3676	LMQ			30	0	9000	4650	4500	0	9000
3677	LMQ			30	0	6000	3100	3000	0	6000
3678	LMD			200	0	400	402	0	1	0
3680	LMQ			92	0	16	40	12	0	16
3683	LMQ			126	0	24	48	18	0	24
3690	LMQ			20	0	6000	3150	3000	0	6000
3692	LMQ			128	0	1091	751	528	0	592
3693	QBN	48.9	0.3	1128	0	0	0	0	0	0
3694	DML	0.0	0.1	40	0	3200	3280	0	0	3200
3697	LMQ			168	0	32	58	24	0	32
3698	DML	0.0	0.1	30	0	3000	3100	0	0	3000
3699	LMQ			116	0	792	1668	192	0	288

Table 7: Features of QPLIB instances (continued).

		Q^0		Variables			Constraints			
name	type	% h.e.	% d.	# b.	# i.	# c.	# l.	# q.	# c.	# v.
3701	LMQ			60	0	1080	377	60	0	120
3703	QBL	46.7	84.6	225	0	0	30	0	0	0
3705	QBN	48.1	1.0	378	0	0	0	0	0	0
3706	QBN	48.6	0.6	703	0	0	0	0	0	0
3708	DML	0.0	0.1	14	0	12916	12917	0	0	1008
3709	QBL	48.0	91.8	600	0	0	50	0	0	0
3713	LMQ			42	0	630	254	42	0	84
3714	QBL	97.5	32.5	120	0	0	40	0	0	0
3719	LMQ			133	0	28	51	21	0	28
3725	LMQ			81	0	1171	1552	469	0	1112
3726	LMQ			116	0	816	2190	192	0	288
3727	LMQ			20	0	1600	840	800	0	1600
3728	LMQ			72	0	16	35	12	0	16
3729	LMQ			650	0	408	51	608	0	408
3733	LMQ			46	0	644	237	46	0	92
3734	LMQ			38	0	7533	7690	2754	0	4050
3738	QBN	48.3	0.9	435	0	0	0	0	0	0
3745	QBN	48.0	1.2	325	0	0	0	0	0	0
3748	LMQ			75	0	20	37	15	0	20
3750	QBL	98.6	32.9	210	0	0	70	0	0	0
3751	QBL	98.0	32.7	150	0	0	50	0	0	0
3752	QBL	45.5	4.1	462	0	0	6160	0	0	0
3757	QBL	34.4	1.7	552	0	0	8096	0	0	0
3762	QBL	50.0	28.0	90	0	0	480	0	0	0
3772	QBL	50.0	3.8	380	0	0	4560	0	0	0
3775	QBL	98.3	32.8	180	0	0	60	0	0	0
3780	LIQ			12	156	0	60	12	0	312
3785	LMQ			200	0	32	62	24	0	32
3790	QML	9.7	100.0	7	0	188	283	0	0	0
3792	DML	0.0	0.1	20	0	3000	3150	0	0	3000
3794	LMQ			576	0	986	624	602	0	968
3797	LMQ			48	0	296	623	56	0	120
3798	LMQ			54	0	810	251	54	0	810
3803	QBL	42.6	14.1	190	0	0	2280	0	0	0
3809	LMQ			224	0	32	65	24	0	32
3813	LMQ			15	0	2400	1280	1200	0	2400
3814	QMQ	4.2	16.0	2	0	46	13	28	0	80
3815	QBL	50.0	3.1	192	0	0	64	0	0	0
3816	LMQ			70	0	117	363	24	0	148
3822	QBN	48.8	0.5	861	0	0	0	0	0	0
3825	LMQ			60	0	1020	317	60	0	1020
3832	QBN	48.5	0.7	561	0	0	0	0	0	0
3834	QBL	60.0	98.0	50	0	0	1	0	0	0
3838	QBN	48.7	0.5	780	0	0	0	0	0	0
3840	LMQ			2401	0	3334	2499	1374	0	3292
3841	QBL	44.0	10.2	300	0	0	4600	0	0	0
3850	QBN	49.0	0.3	1225	0	0	0	0	0	0
3852	QBN	47.6	1.6	231	0	0	0	0	0	0
3854	LMQ			40	0	640	266	40	0	640
3855	LMQ			400	0	2118	791	1284	0	428
3856	LMQ			168	0	183	50	267	0	174
3857	LMQ			201	0	602	604	1	1	1
3859	LMQ			600	0	968	1225	560	0	392
3860	QBL	44.8	8.7	435	0	0	8120	0	0	0
3861	DML	0.0	0.1	30	0	4500	4650	0	0	4500
3863	LMQ			625	0	1053	675	628	0	1033
3865	QBL	48.0	90.7	525	0	0	50	0	0	0
3870	QML	42.9	23.4	116	0	66	1456	0	0	132
3871	DML	0.0	0.1	25	0	1000	1040	0	0	1000
3872	LMQ			95	0	1413	1874	567	0	1368
3877	QBN	48.6	0.6	630	0	0	0	0	0	0
3879	LMQ			10920	0	12906	21945	3026	0	2090
3883	QBL	50.0	17.8	182	0	0	1456	0	0	0
3913	CBL	0.0	100.0	300	0	0	61	0	0	0
3923	QBL	53.7	8.0	395	0	0	80	0	0	0
3931	QBL	50.3	8.0	316	0	0	80	0	0	0
3980	CBL	0.0	100.0	235	0	0	48	0	0	0
4095	CMQ	0.0	100.0	400	0	1600	1603	400	0	400
4270	CML	0.0	25.1	400	0	1200	1603	0	0	800
4455	LMQ			3000	0	12000	9001	3000	0	3000
4722	LMQ			2000	0	8000	6001	2000	0	2000
4805	LMQ			2000	0	8000	6074	2000	0	4000
5023	LMQ			3000	0	12000	9155	3000	0	6000
5442	LMQ			2000	0	7999	6088	2000	0	3998
5527	DML	0.0	0.1	4492	0	21117	64348	0	0	4738
5543	DML	0.0	0.1	4514	0	21186	64096	0	0	4786
5554	LMQ			4492	0	30878	64769	4800	0	4958
5573	LMQ			4450	0	23692	72976	4800	0	4987
5577	DML	0.0	0.1	1118	0	4896	15690	0	0	1186
5721	QBN	49.0	76.8	300	0	0	0	0	0	0
5725	QBN	50.1	1.7	343	0	0	0	0	0	0
5755	QBN	50.0	1.0	400	0	0	0	0	0	0
5875	QBN	50.0	78.9	200	0	0	0	0	0	0
5881	QBN	49.2	29.5	120	0	0	0	0	0	0
5882	QBN	49.3	78.1	150	0	0	0	0	0	0

Table 7: Features of QPLIB instances (continued).

		Q^0		Variables			Constraints			
name	type	% h.e.	% d.	# b.	# i.	# c.	# l.	# q.	# c.	# v.
5909	QBN	50.0	9.6	250	0	0	0	0	0	0
5922	QBN	49.8	9.8	500	0	0	0	0	0	0
5924	DML	0.0	0.7	300	0	15220	36060	0	0	150
5925	LMQ			100	0	1300	271	100	0	100
5926	LMQ			2400	0	31200	11923	2400	0	2400
5927	LMQ			2400	0	31200	11963	2400	0	2400
5935	QBL	49.0	99.0	100	0	0	1237	0	0	0
5944	QBL	49.0	99.0	100	0	0	2475	0	0	0
5962	QBL	49.3	99.3	150	0	0	2793	0	0	0
5971	QBL	49.3	99.3	150	0	0	5587	0	0	0
5980	QBL	49.3	99.3	150	0	0	8381	0	0	0
6287	LCQ			0	0	171	36	81	0	150
6310	LCQ			0	0	208	22	390	0	324
6311	LCQ			0	0	212	43	128	0	186
6324	QBL	50.6	31.3	640	0	0	16	0	0	0
6487	QBL	35.0	20.9	618	0	0	309	0	0	0
6597	QBL	45.7	97.3	600	0	0	60	0	0	0
6647	QBL	70.0	7.2	627	0	0	33	0	0	0
6757	QBL	18.5	4.7	2046	0	0	297	0	0	0
6764	QBL	19.1	4.7	2071	0	0	297	0	0	0
6799	QBL	18.7	4.7	2075	0	0	297	0	0	0
6941	QBL	18.7	4.5	2203	0	0	315	0	0	0
7127	QBL	50.6	6.8	1000	0	0	50	0	0	0
7139	QBL	53.3	89.2	180	0	0	100	0	0	0
7144	QBL	53.2	89.6	220	0	0	121	0	0	0
7149	QBL	53.0	89.6	264	0	0	144	0	0	0
7154	QBL	52.9	89.7	312	0	0	169	0	0	0
7159	QBL	52.5	89.7	364	0	0	196	0	0	0
7164	QBL	52.4	89.7	420	0	0	225	0	0	0
7579	LMD			100	0	200	202	0	1	0
8009	LMQ			101	0	303	305	2	0	2
8153	LMQ			31	0	93	95	2	0	2
8381	LMQ			51	0	153	155	2	0	2
8495	DCL	0.0	0.1	0	0	27543	8000	0	0	22743
8500	DCL	0.0	0.1	0	0	250997	250498	0	0	126002
8505	QCL	49.9	0.1	0	0	20050	10001	0	0	40100
8515	CCL	0.0	0.1	0	0	16002	8002	0	0	16002
8547	DCL	0.0	0.1	0	0	1003001	1001000	0	0	4002
8553	QCQ	0.0	0.1	0	0	79998	796	39601	0	158404
8559	CCL	0.0	0.1	0	0	10000	5000	0	0	20000
8567	CCL	0.0	0.1	0	0	10000	7500	0	0	20000
8585	DCQ	0.0	0.1	0	0	99999	0	49999	0	2
8595	DCQ	0.0	0.1	0	0	2500	0	1275	0	0
8602	DCL	0.0	0.1	0	0	34552	52983	0	0	69104
8605	DCQ	0.0	0.1	0	0	5000	0	1	0	1
8616	DCL	0.0	0.1	0	0	13870	10404	0	0	409
8683	DCQ	0.0	0.1	0	0	200008	0	140000	0	14
8685	DCQ	0.0	0.1	0	0	772	0	10000	0	0
8758	QCQ	4.3	50.0	0	0	2070	0	1981	0	0
8777	QCL	34.6	0.1	0	0	10000	2500	0	0	20000
8784	QCC	49.5	1.0	0	0	200	98	0	4950	204
8785	DCL	0.0	0.1	0	0	10399	11362	0	0	20798
8790	CCB	0.0	0.1	0	0	39204	0	0	0	39204
8792	CCB	0.0	0.1	0	0	15129	0	0	0	30258
8803	DCQ	0.0	0.1	0	0	150002	50000	50000	0	50003
8810	DCQ	0.0	0.1	0	0	150002	50000	50000	0	4
8815	QCD	0.1	25.0	0	0	30010	20004	0	5001	0
8845	CCL	0.0	59.8	0	0	1546	777	0	0	441
8906	CCL	0.0	3.0	0	0	5223	838	0	0	1941
8938	DCL	0.0	0.1	0	0	4001	11999	0	0	0
8991	CCB	0.0	0.1	0	0	14400	0	0	0	28800
9002	DCL	0.0	0.1	0	0	2890	1649	0	0	3617
9004	QCQ	25.0	0.1	0	0	40000	10001	10001	0	20000
9008	DCL	0.0	0.1	0	0	1009306	989604	0	0	39208
9030	QIL	0.1	0.1	0	10000	0	5000	0	0	20000
9048	QIL	29.7	18.2	0	202	0	1	0	0	404
10001	LMC			426	0	59	295	0	1	118
10002	LMC			426	0	59	295	0	1	118
10003	LMC			999	0	59	866	0	1	118
10004	LMC			150	0	250	100	0	1	500
10005	LMC			1000	0	1000	793	0	1	2000
10006	LMC			1875	0	1250	1489	0	1	2500
10007	LMC			2625	0	1750	2086	0	1	3500
10008	LMC			713	0	132	415	0	1	264
10009	LMC			473	0	132	245	0	1	264
10010	LMC			262	0	7	146	0	1	14
10011	LMC			1258	0	132	872	0	1	264
10012	LMC			835	0	132	537	0	1	264
10013	LMQ			3600	0	18106	55968	3600	0	3600
10014	LMQ			3600	0	18113	55834	3600	0	3600
10015	LMQ			3600	0	23527	50083	3600	0	3600
10016	LMQ			3600	0	23524	50427	3600	0	3600
10017	LMQ			4800	0	24149	74451	4800	0	4800
10018	LMQ			4800	0	24145	75293	4800	0	4800
10019	LMQ			4800	0	31370	66484	4800	0	4800

Table 7: Features of QPLIB instances (continued).

		Q^0		Variables			Constraints			
name	type	% h.e.	% d.	# b.	# i.	# c.	# l.	# q.	# c.	# v.
10020	LMQ			4800	0	31372	66912	4800	0	4800
10021	LMQ			3000	0	12000	9155	3000	0	6000
10022	LMQ			3000	0	12000	9155	3000	0	6000
10023	LMQ			3000	0	12000	9155	3000	0	6000
10024	LMQ			3000	0	12000	9089	3000	0	6000
10025	CMQ	0.0	100.0	400	0	1600	1603	400	0	400
10026	CMQ	0.0	100.0	400	0	1600	1603	400	0	400
10027	CMQ	0.0	100.0	400	0	1600	1603	400	0	400
10028	CMQ	0.0	100.0	400	0	1600	1603	400	0	400
10029	CMQ	0.0	100.0	400	0	1600	1603	400	0	400
10030	LMQ			3000	0	12000	9001	3000	0	3000
10031	LMQ			3000	0	12000	9001	3000	0	3000
10032	LMQ			3000	0	12000	9001	3000	0	3000
10033	LMQ			3000	0	12000	9001	3000	0	3000
10034	DCL	0.0	0.2	0	0	40400	40200	0	0	802
10035	LCQ			0	0	40401	40000	200	1	1200
10036	LCQ			0	0	40401	40000	200	1	1200
10037	LCQ			0	0	40401	200	40000	1	400
10038	DCL	0.0	0.1	0	0	160800	160400	0	0	1602
10039	LCQ			0	0	12097	11713	193	0	384
10040	LMQ			125	0	1	6	1	0	0
10041	LMQ			125	0	1	6	1	0	0
10042	QBL	0.8	99.9	125	0	0	5	0	0	0
10043	LMQ			150	0	1	10	1	0	0
10044	QBL	8.0	97.0	150	0	0	6	0	0	0
10045	LMQ			150	0	1	10	1	0	0
10046	QBL	0.7	92.1	150	0	0	6	0	0	0
10047	LMQ			150	0	1	10	1	0	0
10048	QBL	1.3	99.9	150	0	0	5	0	0	0
10049	LMQ			150	0	1	10	1	0	0
10050	CBL	0.0	100.0	150	0	0	5	0	0	0
10051	LMQ			150	0	1	10	1	0	0
10052	QBL	1.3	99.9	150	0	0	6	0	0	0
10053	LMQ			150	0	1	10	1	0	0
10054	QBL	4.6	90.1	175	0	0	11	0	0	0
10055	QBL	2.9	91.5	175	0	0	5	0	0	0
10056	CBL	0.0	98.8	175	0	0	5	0	0	0
10057	LMQ			200	0	1	11	1	0	0
10058	QBL	7.5	88.0	200	0	0	11	0	0	0
10059	LMQ			200	0	1	10	1	0	0
10060	LMQ			200	0	1	10	1	0	0
10061	QBL	9.0	97.6	200	0	0	5	0	0	0
10062	LMQ			200	0	1	10	1	0	0
10063	QBL	3.0	99.5	200	0	0	5	0	0	0
10064	LMQ			200	0	1	11	1	0	0
10065	QBL	1.0	99.0	200	0	0	11	0	0	0
10066	QBL	1.5	100.0	200	0	0	11	0	0	0
10067	QBL	2.5	99.7	200	0	0	5	0	0	0
10068	QBL	2.0	99.9	200	0	0	11	0	0	0
10069	LMC			200	0	1	10	0	1	0
10070	QBL	1.5	99.9	200	0	0	11	0	0	0
10071	LMQ			200	0	1	11	1	0	0
10072	LMQ			75	0	1	10	1	0	0
10073	LMQ			75	0	1	6	1	0	0
10074	LMQ			75	0	1	10	1	0	0

B. The file format

The QPLIB format is defined in Table 8, with the notation of §2.

The data is in free format (blanks separate values), but must occur in the order given here. Any blank lines, or lines starting with any of the characters !, % or # are ignored. Each term in the first column of Table 8 denotes a required value. Any strings beyond those required on a given line will be regarded as comments and ignored. Real values may either by in decimal or exponential formats; for the latter, the exponent may be preceded by either the character `D` or `E`, e.g. `12.56D+2` or `12.56E+2`. Variable indices, j, must lie in the range $1 \leq j \leq n$, while constraint indices, i, must satisfy $1 \leq i \leq m$, that is they are both **one-based**. The case for character strings is irrelevant.

Table 8: The QPLIB file format: refer to the notes after the table for more details.

data	description	note
name	problem name (character string)	
type	problem type (character string)	[1]
sense	one of the words minimize or maximize (character string)	
n	number of variables (integer)	
m	number of constraints (integer)	[2]
n^{Q^0}	number of nonzeros (integer) in lower triangle of Q^0	[3]
$h\ \ k\ \ Q^0_{hk}$	row and column indices (integers) and value (real) for each nonzero entry of Q^0, if $n^{Q^0} > 0$, one triple on each line	[3]
b^0_d	default value (real) for entries in b^0	
n^{b^0}	number of non-default entries (integer) in b^0	
$j\ \ b^0_j$	index (integer) and value (real) for each non-default term in b^0, if $n^{b^0} > 0$, one pair per line	
q^0	constant part of the objective function	
$\sum_{i\in\mathcal{M}} n^{Q^i}$	number of nonzeros (integer) in lower triangles of Q^i, summed over all $i \in \mathcal{M}$	[2,4]
$i\ \ h\ \ k\ \ Q^i_{hk}$	i, row and column indices (integers) and value (real) for each entry of Q^i for every $i \in \mathcal{M}$, if $n^{Q^i} > 0$, one quadruple on each line	
$\sum_{i\in\mathcal{M}} n^{b^i}$	number of nonzeros (integer) in b^i, summed over all $i \in \mathcal{M}$	[2]
$i\ \ j\ \ b^i_j$	i and index (integers) and value (real) for each nonzero entry of b^i for every $i \in \mathcal{M}$, if $n^{b^i} > 0$, one triple on each line	[2]
c_∞	value (real) for infinity for constraint or variable bounds—any bound greater than or equal to this in, absolute value, is infinite	
$c_{l,d}$	default value (real) for entries in c_l	[2]
$n^{c_{l,d}}$	number of non-default entries (integer) in c_l	[2]
$i\ \ c^i_l$	index (integer) and value (real) for each non-default term in $c_{l,d}$, if $n^{c_{l,d}} > 0$, one pair per line	[2]
$c_{u,d}$	default value (real) for entries in c_u	[2]
$n^{c_{u,d}}$	number of non-default entries (integer) in c_u	[2]
$i\ \ c^i_u$	index (integer) and value (real) for each non-default term in $c_{u,d}$, if $n^{c_{u,d}} > 0$, one pair per line	[2]
l_d	default value (real) for entries in l	[6]
n^{l_d}	number of non-default entries (integer) in l	[6]
$i\ \ l_i$	index (integer) and value (real) for each non-default term in l, if $n^{l_d} > 0$, one pair per line	[6]
u_d	default value (real) for entries in u	[6]
n^{u_d}	number of non-default entries (integer) in u	[6]
$i\ \ u_i$	index (integer) and value (real) for each non-default term in u, if $n^{u_d} > 0$, one pair per line	[6]
v_d	default variable type (integer, 0 for continuous variables, 1 for integer variables, 2 for binary variables)	[5]
n^v	number of non-default variables (integer)	[5]
$i\ \ v_i$	index and type (integers) for each non-default variable type, if $n^v > 0$, one pair per line	[5]
x^0_d	default value (real) for the components of the starting point x^0 for the variables x	
n^{x^0}	number of non-default starting entries (integer) in x	
$i\ \ x^0_i$	index (integer) and value (real) for each non-default starting value in x^0, if $n^{x^0} > 0$, one pair per line	

Table 8: The QPLIB file format (continued)

data	description	note
y_d^0	default value (real) for the components of the starting point y^0 for the Lagrange multipliers y for the general constraints	[2]
n^{y^0}	number of non-default starting entries (integer) in y	[2]
i y_i^0	index (integer) and value (real) for each non-default starting value in y^0, if $n^{y^0} > 0$, one pair per line	[2]
z_d^0	default value (real) for the components of the starting point z^0 for the dual variables z for the simple bound constraints	
n^{z^0}	number of non-default starting entries (integer) in z	
i z_i^0	index (integer) and value (real) for each non-default starting value in z^0, if $n^{z^0} > 0$, one pair per line	
n_d^x	number of non-default names (integer) of variables—default for variable i is the character string representing the numerical value i	
j `var_name`$_j$	index (integer) and name (character string) for each non-default variable name, if $n_d^x > 0$, one pair per line	
n_d^c	number of non-default names (integer) of general constraints—default for constraint i is the character string representing the numerical value i	
i `cons_name`$_i$	index (integer) and name (character string) for each non-default constraint name, if $n_d^c > 0$, one pair per line	

`[1]` The problem type is represented by a three character string as given in §2.2.1
`[2]` For problems of type `**N` or `**B`, these lines/sections are omitted.
`[3]` For problems of type `L**`, this section is omitted.
`[4]` For problems of type `**N`, `**B` or `**L`, this section is omitted.
`[5]` For problems of type `*C*`, `*B*` or `*I*`, this section is omitted. For problems of type `*I*`, binary variables should be specified as integer variables with lower and upper bounds 0 and 1.
`[6]` For problems of type `*B*`, this section is omitted.

Binary variables defined either implicitly via the type `*B*` or explicitly in the variable type section will be assumed to have lower and upper bounds 0 and 1, and this will override any explicit bounds l_d, u_d, l_i, and u_i set in the lower and upper bound sections. To fix a binary variable to 0 or 1, its variable type should be changed to continuous or general integer and the corresponding lower and upper bounds set accordingly in the lower and upper bound sections.

As a simple example, consider the mixed-integer QP

$$\begin{array}{l} \min_{x\in\mathbb{R}^3} x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 - 0.2x_1 - 0.4x_2 - 0.2x_3 \\ \text{subject to } 1 \le x_1 + x_2,\ 1 \le x_1 + x_3,\ 0 \le x_1 \le 1,\ 0 \le x_2 \le 2, \text{ and binary } x_3, \end{array}$$

for which the Hessian of the objective function is

$$Q^0 = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

This may then be represent in QPLIB format as follows:

```
! ---------------
! example problem
! ---------------
MIPBAND   # problem name
QML       # problem is a mixed-integer quadratic program
Minimize  # minimize the objective function
```

```
3         # variables
2         # general linear constraints
5         # nonzeros in lower triangle of Q^0
1 1 2.0   5 lines row & column index & value of nonzero in lower triangle Q^0
2 1 -1.0  |
2 2 2.0   |
3 2 -1.0  |
3 3 2.0   |
-0.2      default value for entries in b_0
1         # non default entries in b_0
2 -0.4    1 line of index & value of non-default values in b_0
0.0       value of q^0
4         # nonzeros in vectors b^i (i=1,...,m)
1 1 1.0   4 lines constraint, index & value of nonzero in b^i (i=1,...,m)
1 2 1.0   |
2 1 1.0   |
2 3 1.0   |
1.0E+20   infinity
1.0       default value for entries in c_l
0         # non default entries in c_l
1.0E+20   default value for entries in c_u
0         # non default entries in c_u
0.0       default value for entries in l
0         # non default entries in l
1.0       default value for entries in u
1         # non default entries in u
2 2.0     1 line of non-default indices and values in u
0         default variable type is continuous
1         # non default variable types
3 2       variable 3 is binary
1.0       default value for initial values for x
0         # non default entries in x
0.0       default value for initial values for y
0         # non default entries in y
0.0       default value for initial values for z
0         # non default entries in z
0         # non default names for variables
0         # non default names for constraints
```