Mathematical Programming Computation
https://doi.org/10.1007/s12532-018-0147-4

FULL LENGTH PAPER

QPLIB: a library of quadratic programming instances

Fabio Furini, et al. ***[full author details at the end of the article]***

Received: 2 April 2017 / Accepted: 6 September 2018

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 and/or the constraints are quadratic. QP is a very diverse class of problems, comprising sub-classes ranging from trivial to undecidable. This diversity is reflected in the variety of QP solution methods, ranging from entirely combinatorial approaches to completely continuous algorithms, including many methods 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 different, interested communities is therefore important. We propose a simple taxonomy for QP instances leading 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 · Mixed-Integer Quadratically Constrained Quadratic Programming · Binary quadratic programming

Mathematics Subject Classification 90C06 · 90C25

1 Introduction

Quadratic programming (QP) problems—mathematical optimization problems for which the objective function [145], the constraints [146], or both are polynomial function of the variables of degree two—include a notably diverse set of different

The software that was reviewed as part of this submission was given the DOI (Digital Object Identifier) https://doi.org/10.5281/zenodo.1412045.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12532-018-0147-4) contains supplementary material, which is available to authorized users.

✉ Fabio Furini
fabio.furini@dauphine.fr

Extended author information available on the last page of the article

instances. This is not surprising, given the vast scope of practical applications of such problems, and of solution methods designed to solve them [70]. Depending on the formulation specifics, 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 a class of problems where collaboration between the communities interested in combinatorial and in nonlinear optimization is 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. Therefore, it is perhaps not surprising that, unlike for “simpler” problems classes like Mixed-Integer Linear Programming [84], there has been no single library devoted to all different kinds of instances of QP. While several specialized libraries devoted to particular classes of QP are available, each of them is either focused on a particular application (a specific problem that can be modeled 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 overwhelmingly onerous but sufficiently challenging for the many different interested communities has not been attempted. This work constitutes a first step in this direction.

This paper reports our steps towards collecting what we consider to be a quality QP instance library, filtering a much larger set of currently available (or specifically provided) instances and proposing 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 constructing 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 library instances 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 for almost seventy years. Part of the rationale for this interest is 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, e.g., the simplex method, enumeration methods, cut-generation techniques, reformulation techniques, and many others [27]. 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 developing general techniques for nonconvex NLP, and a very specific class enabling the development of specialized approaches [26,44].

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 [85]. Table 1 provides a 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.

The structure of this paper is as follows. In Sect. 2 we review the basic notion of QP. In particular, Sect. 2.1 sets out the notation, Sect. 2.2 proposes a new QP taxonomy that helps discuss the (very) different QP classes, and Sect. 2.3 very briefly reviews the QP solution methods and the solvers we have employed. Next, Sect. 3 describes the process used to obtain the library and its results. Some conclusions are drawn in Sect. 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 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 [139], a collection of currently 1527 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.

Table 1 Application domains of QP

Problem	Discrete	Contributions
Fundamental problems that can be formulated as MIQP		
Quadratic assignment problem‡	✓	[8,100]
Max-cut	✓	[89,120]
Maximum clique‡	✓	[22]
Computational chemistry & Molecular biology		
Zeolites		[72]
Computational geometry		
Layout design	✓	[7,30,39]
Maximizing polygon dimensions		[9–13]
Packing circles‡	✓	[51,57,76,129]
Nesting polygons		[81,119]
Cutting ellipses		[82]
Finance		
Portfolio optimization	✓	[37,51,54–56,80,98,101,113,122]
Process networks		
Crude oil scheduling	✓	[93–95,106,107]
Data reconciliation	✓	[124]
Multi-commodity flow	✓	[130]
Quadratic network design	✓	[51,57]
Multi-period blending	✓	[87,88]
Natural gas networks	✓	[74,96,97]
Pooling‡	✓	[4,31,36,47,102,103,112,114,125]
Open-pit mine scheduling	✓	[20]
Reverse osmosis	✓	[126]
Supply chain	✓	[111]
Water networks‡	✓	[3,14,24,33,58,64,79,83,118,136]
Robotics		
Traveling salesman problem		
With neighborhoods	✓	[59]
Telecommunications		
Delay-constrained routing	✓	[52,53]
Energy		
Unit-commitment	✓	[51,54,56,131]
Data confidentiality		
Controlled Tabular Adjustment	✓	[32]
Trust-region methods		
Trust-region subproblem		[2,46,65,69,73,121]
PDE-constrained optimization		
Optimal control problem		[115,127,128]

‡Applications with many manuscripts cite reviews and recent works

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

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

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_{hk}^i and Q_{kh}^i with their average $(Q_{hk}^i + Q_{kh}^i)/2$;
- 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 quadratic constraints and the integrality requirements on the variables, this class 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 variables. 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 problem data, although bounds 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.

2.2 Classification

Despite the apparent simplicity of the Sect. 2.1 definition, 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 problem data has specific properties. A taxonomy of QP instances should thus strive to identify a 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 (positive or negative) definiteness of Q^i is a sufficient, but not in general necessary, condition for convexity. As detailed in Sect. 3.3, in our taxonomy we mark the constraints "C" based on the sufficient condition alone, the rationale of this choice being discussed in Sect. 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 ordering in the preceding lists is relevant; 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. The following discussion repeatedly exploits this ordering by assuming that, unless otherwise mentioned, when a method can be applied to a given problem, it can also be applied to all simpler problems where the value of each field is arbitrarily replaced with a value denoting a less-general class.

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.

2.2.2 Examples and reformulations

We now give a general discussion about the different problem classes that our proposed taxonomy defines. For simplicity, this section assumes minimization problems. 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

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

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 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.,

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

is not only $\mathcal{NP}$-hard, but undecidable [78]. 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 [17,18].

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

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

i.e., a *L*Q* (*L*C*) problem. 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 $Q^i = L^i(L^i)^\top$, e.g., by the (incomplete) Cholesky factorization, the term $\frac{1}{2}x^\top Q^i x \equiv \frac{1}{2}\sum_{j\in\mathcal{N}} {z_j^i}^2$ where $z^{i\,\top} = x^\top L^i$. 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, $L^{i\,\top}x - z^i = 0$. 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., [51,54,56,57]), 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 with 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.

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 form as they have been provided to us by the original contributors. The rationale of this choice is that reformulation techniques, like the ones discussed here and others, are typically motivated by the fact that they make the instance easier to solve for one specific class of solvers. This being a bias that we do not want to add we have chosen to keep the instances in their "natural" form, this being the one in which the original contributor initially wrote them.

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 [84], so they will not be explicitly addressed here.
- *Convex Continuous Quadratic Programs CCC* can be solved in polynomial time by Interior-Point techniques [147]; the simpler *CCL* can also be solved by means of "simplex-like" techniques, usually referred to as active-set methods [40]. 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. When building our instance library we never exploited such characteristics, since they are not amenable to standard modeling 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 [75]. They therefore require enumerative techniques, such as spatial Branch-and-Bound [15,50], 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 *CGC* 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 [18]. Absence of continuous variables, even in the presence of integer ones (*CIC*), can also lead to specific techniques [17].
- *Nonconvex Binary Quadratic Programs* $QB\{B,N,L\}$ 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 prob-

lem 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 [100,120].

– *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 in nonconvex terms) and techniques that allow the efficient computation of bounds (e.g., outer approximation, semidefinite programming relaxation, ...). As in the convex case, *QBQ*, *QMQ*, and *QIQ* can benefit from more specific properties of the variables [25,38].

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 [51,52,130] or knapsack-type linear constraints [51,57], and semi-continuous variables [52,53,57], or the fact that a nonconvex quadratic objective function/constraint can be reformulated as a second-order cone (hence, convex) one [53,56,57]. 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

This section provides a quick overview of existing solution methods for QP, restricting ourselves to these implemented by the specific solvers considered in this paper (see Sect. 2.3.1). For each approach, we briefly describe the formulation they address according to the Sect. 2.2 classification. 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 [110]: *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 they cannot prove that a given instance is infeasible (see Sect. 2.3.3 below).

- *Complete* methods find an approximate globally optimal solution within a prescribed optimality tolerance within finite time, or prove that none such exists (but see Sect. 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 [16] and [92] for further details on the solution methods.

2.3.1 Solvers

When compiling QPLIB, we have worked with the QP solvers in 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 [28]. 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 $R \supset P$.

The Table 2 solvers 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 QP class. Aside from solvers provided by GAMS, there are a number of other, specialized, incomplete QP solvers, such as CQP [66], DQP

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

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

		CGL	QGL	CGC	QGQ	CCC	QCQ
AlphaECP	[143,144]	C	I	C	I	C	I
ANTIGONE	[104,105]	C	C	C	C	C	C
BARON	[133–135]	C	C	C	C	C	C
BONMIN	[23]	C	I	C	I	C	I
CONOPT	[41,42]					C	I
Couenne	[15]	C	C	C	C	C	C
Cplex	[19,77]	C	C	C		C	
DICOPT	[45,86,141]	C	I	C	I	C	I
Gurobi	[123]	C		C		C	
Ipopt	[142]					C	I
Knitro	[29]	C	I	C	I	C	A
Lindo API	[99]	C	C	C	C	C	C
LGO	[116,117]					A	A
MINOS	[108,109]					C	I
MOSEK	[5,6]	C		C		C	
MsNlp	[91,137]					C	A
OQNLP	[91,137]	A	A	A	A	C	A
SBB	[43]	C	I	C	I	C	I
SCIP	[1,140]	C	C	C	C	C	C
SNOPT	[61,62]					C	I
Xpress-Optimizer	[48]	C		C		C	

[68] and OOQP [60] for convex problems, and BQPD [49], QPA [71] and QPB [34], QPC [67], SQIC [63] for nonconvex ones.

2.3.3 Asymptotically complete methods

An asymptotically complete method reaches a global minimum with certainty or at least with probability one if allowed to run indefinitely long, but has no means to know when a global minimizer has been found (see [110]). 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, the decision version of *QCL* is known to be in the complexity class $\mathcal{NP}$ [138], 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 [132], 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 [21]. 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 [110].

Nearly all of the complete solvers in Table 2 that address $\mathcal{NP}$-hard problems (i.e., those in $QGQ \setminus CCC$) are based on Branch-and-Bound (BB) [90]. When the BB algorithm is allowed to branch on coordinate directions corresponding to continuous variables, it is called *spatial* BB (sBB) [15,35]. 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, and SCIP for *QGQ*;
- CPLEX for *QGL* and *CGC*;
- GUROBI and XPRESS-OPTIMIZER for *QBC*;
- BONMIN, GUROBI, KNITRO, MOSEK, SBB, and XPRESS-OPTIMIZER for *CGC*.

We remark that the solvers BONMIN, KNITRO, and SBB from 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 [45,144]. 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

This section presents all the steps we performed to build the new instance library. In Sect. 3.1, we describe the set of gathered instances, and in Sect. 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 Sect. 3.3, while in Sect. 3.4 we provide information on how to access the test collection.

3.1 Instance collection

This section describes the procedure we adopted to gather the instances. In January 2014, we issued an online call for instances using main international mailing lists of the mathematical optimization and numerical analysis communities, reaching in this way a large set of possibly interested researchers and practitioners. The call remained open for ten 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 these spontaneous contributions, we analyzed existing generic instance libraries available containing QP instances. 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 `Maros-Mészáros` library http://www.doc.ic.ac.uk/~im/00README.QP;
– the `MINLPLib` 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 $\approx$ 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 format requirements were imposed for the submissions.

To evaluate the instances we decided, for practical reasons, we 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 Sect. 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 non-positive eigenvalues (when $c_l^i > -\infty$) are classified as convex constraints; thus, a quadratic constraint with two sided, finite bounds is nonconvex. Note that this might occasionally lead us to classify some instances that have conic constraints as nonconvex 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 next paragraph.

3.2 Instance selection

We chose instances based on the following four goals:

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.

Table 3 Instance filter steps

$$\begin{array}{lcc}
\text{Starting set} & \multicolumn{2}{c}{\approx 8500 \text{ instances}} \\
 & \Swarrow & \Searrow \\
 & \approx 6000 \text{ discr. inst.} & \approx 2500 \text{ cont. inst.} \\
\text{First filter} & \Downarrow & \Downarrow \\
 & \approx 3000 \text{ discr. inst.} & \approx 1000 \text{ cont. inst.} \\
\text{Second filter} & \Downarrow & \Downarrow \\
 & 319 \text{ discr. inst.} & 134 \text{ cont. inst.}
\end{array}$$

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

- *First Instance 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. Sect. 2.3) to solve it to global optimality. Accordingly, for each of the gathered instances we ran the complete solvers in GAMS, whose number depends on the category of the instance under consideration, cf. Table 2. Thanks to these extensive preliminary tests, we discarded all instances that are solved by at least 30% of the complete solvers within a time limit of 30 seconds.
- *Second Instance Filter.*
  The goal of the second filter was to eliminate "similar" instances. We carefully analyzed the instances one by one, eliminating all but a few of those with very similar size and coming from the same donor. The instances discarded by this second filter are instances of the same specific problem, e.g., we gathered many Max Cut Problem instances and we kept in the library only a representative small set of them. The selected representative instances are the larger and computationally harder ones. Finally, in order to only keep computationally challenging instances we ran 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 analyze the features of the instances selected to be part of the library. Table 4 provides a global overview. The instances have been divided in *continuous* vs *discrete* and *convex* vs *nonconvex*, forming in this way, a classification of 4 macro categories. As previously mentioned, an instance is classified *discrete* if it contains at least one binary or integer variable, and *continuous* otherwise. On the other hand, an instance is classified as *nonconvex* if the objective function is nonconvex (if minimization) or nonconcave (if maximization) and/or at least one of the constraints is nonconvex, 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 the corresponding number of instances in column "#". It can be

Table 4 Macro classification of the final set of instances

Variables	Convexity	#
Continuous	*Convex*	32
Continuous	*Nonconvex*	102
Discrete	*Convex*	31
Discrete	*Nonconvex*	288
Total		453

Table 5 Classification of the final set of discrete instances

Obj. fun.	Variables	Constraints	#
Linear	Binary	Quadratic	9
	Mixed	Convex	14
		Quadratic	134
	Integer	Quadratic	2
	General	Quadratic	3
Convex (if min)	Binary	Linear	5
or	Mixed	Linear	12
Concave (if max)		Quadratic	6
Quadratic	Binary	None	23
		Linear	91
		Quadratic	5
	Mixed	Linear	11
		Quadratic	1
	Integer	Linear	2
	General	Quadratic	1
Total			319

seen that the final set well respects the original distribution of the gathered instances among the different categories. Indeed, the discrete categories *LMQ* and *QBL* are well represented by 134 and 91 instances, respectively. Similarly, the continuous categories *LCQ* and *QCQ* are well represented by 52 and 30 instances, respectively. Moreover, the library actually covers the large majority of all possible categories of instances.

We now report some graphs that help in illustrating the main features of the instances. In Fig. 1 (left) we plot the number of variables (horizontal axis) versus the number of constraints (vertical axis), both in logarithmic scale. Continuous instances are marked with "+" and discrete ones with "×". Box constraints are not counted as constraints. The figure shows that the library contains a quite diverse set of instances in terms of number of variables and constraints. The record on the maximal number of variables and constraints (both $\approx 1,000,000$) is set by the instances QPLIB_8547 and QPLIB_9008. 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.

Table 6 Classification of the final set of continuous instances

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

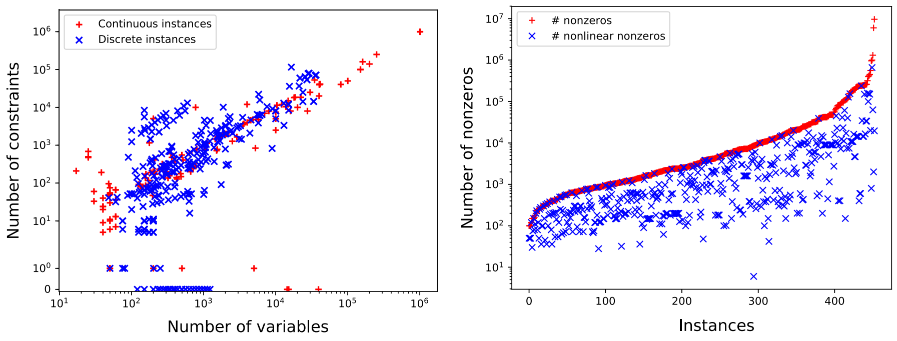

Fig. 1 Distribution of number of variables and constraints of QPLIB instances (left). Number of (nonlinear) nonzeros of QPLIB instances (right)

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 “×”. The pictures clearly show that instances with different percentages 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, Fig. 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 Fig. 3 (left) we learn that while the maximum number of linear constraints exceeds 1,000,000, the maximum number of quadratic constraints tops up at 140,000. 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.

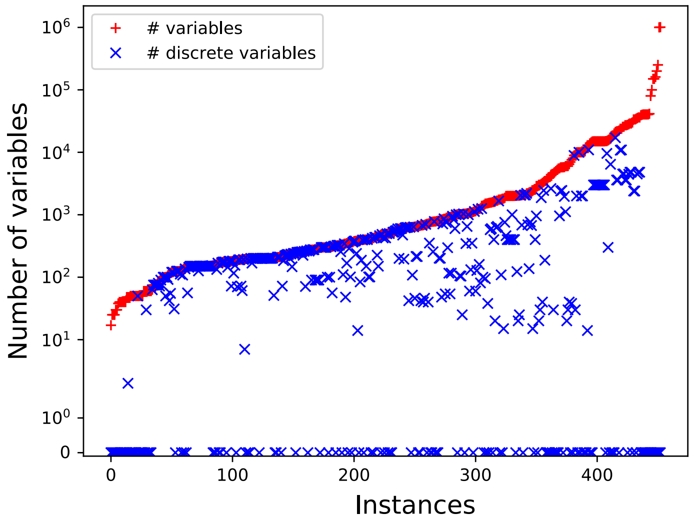

Fig. 2 Number of variables of QPLIB instances

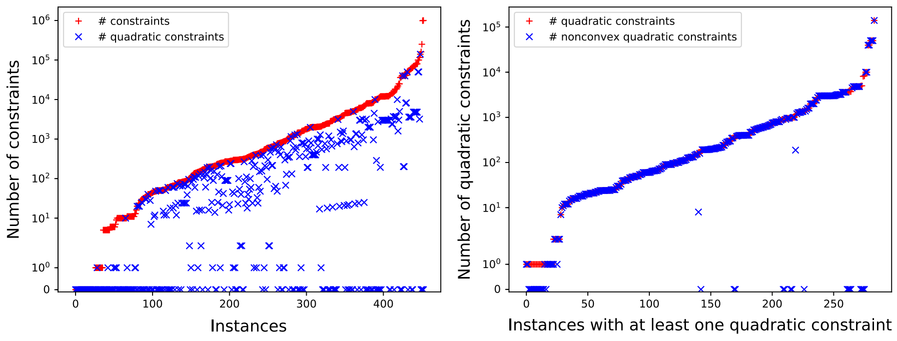

Fig. 3 Number of constraints, quadratic constraints, and nonconvex quadratic constraints of QPLIB instances

Figure 3 (right) shows the instances with at least one quadratic constraint sorted according to the number of quadratic constraints (vertical axis). For 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, Fig. 4 (left) focuses on the instances with a quadratic objective function, ordered by percentage of "problematic" eigenvalues in the Hessian Q^0 (vertical axis), by which we mean eigenvalues below -10^{-12} in case of a minimization problem and eigenvalues above 10^{-12} 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

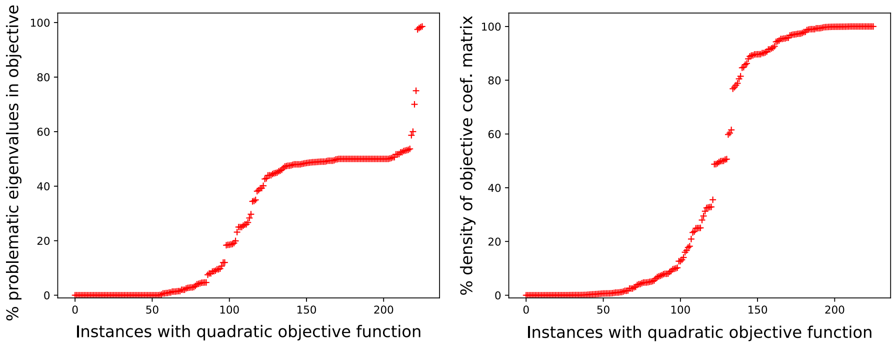

Fig. 4 "Problematic" eigenvalues (left) and density (right) of the Hessian Q^0 for QPLIB instances with a quadratic objective function

50% of "problematic" eigenvalues. However, instances with high or low percentages of "problematic" eigenvalues are present, too.

Similarly, Fig. 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 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 [139]. We provide all instances in GAMS (`.gms`), AMPL (`.mod`), CPLEX (`.lp`) [77], 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, 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 Sect. 3.3), the contributor of the instance, 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, if the instance size allows. 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. Finally, feasible solution points are provided for most instances.

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,

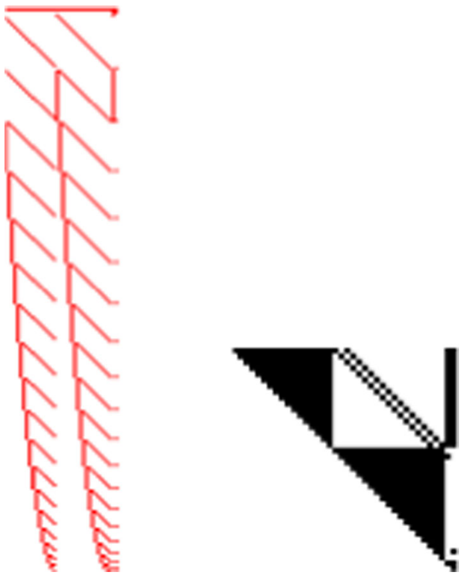

Fig. 5 Example for the sparsity pattern of the Jacobian of the constraint functions (left) and of the upper-right triangle of the Hessian of the Lagrangian function (right) for instance QPLIB_2967. The gradient of the objective function is displayed as the first row of the Jacobian matrix. Non-constant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

problematic 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-separated-values (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 references to the literature.

4 Final remarks

This paper described 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 discussed the main classes of solution methods for QP. We then described 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 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 avoided to perform a computational comparison of the performances of different solution methods or solver implementations. Our goal was instead to provide a broad test bed of instances for researchers and practitioners in the field. This new library will hopefully serve as a point of reference to inspire and test new ideas and algorithms for QP problems.

Finally, we want to emphasize that this QP collection can only be a snapshot of the types of problems that researchers and practitioners have worked on in the past. With the growing interest in this area, we hope that new applications and instances will become available and that the library can be extended dynamically in the future.

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 a license for 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 responsibility for the content of this publication is assumed by the authors.

References

1. Achterberg, T.: SCIP: solving constraint integer programs. Math. Prog. Comput. **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 J. Optim. **27**(1), 269–291 (2017)
3. Ahmetović, E., Grossmann, I.E.: Global superstructure optimization for the design of integrated process water networks. AIChE J. **57**(2), 434–457 (2011)
4. Alfaki, M., Haugland, D.: A multi-commodity flow formulation for the generalized pooling problem. J. Global Optim. **56**(3), 917–937 (2013)
5. Andersen, E., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Math. Prog. **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: Frenk, H., Roos, K., Terlaky, T., Zhang, S. (eds.) High Performance Optimization, pp. 197–232. Springer, Boston (2000)
7. Anjos, M.F., Liers, F.: Global approaches for facility layout and VLSI floorplanning. In: Anjos, M.F., Lasserre, J.B. (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, International Series in Operations Research and Management Science, vol. 166, pp. 849–877. Springer, Boston (2012)
8. Anstreicher, K.M.: Recent advances in the solution of quadratic assignment problems. Math. Prog. **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. J. Global Optim. **49**(3), 467–480 (2011)
10. Audet, C., Hansen, P., Messine, F.: The small octagon with longest perimeter. J. Comb. Theory Ser. A **114**(1), 135–150 (2007)
11. Audet, C., Hansen, P., Messine, F.: Simple polygons of maximum perimeter contained in a unit disk. Discrete Comput. Geom. **41**(2), 208–215 (2009)
12. Audet, C., Hansen, P., Messine, F., Xiong, J.: The largest small octagon. J. Comb. Theory Ser. A **98**(1), 46–59 (2002)
13. Audet, C., Ninin, J.: Maximal perimeter, diameter and area of equilateral unit-width convex polygons. J. Global Optim. **56**(3), 1007–1016 (2013)
14. Bagajewicz, M.: A review of recent design procedures for water networks in refineries and process plants. Comput. Chem. Eng. **24**(9–10), 2093–2113 (2000)
15. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**(4–5), 597–634 (2009)
16. Best, M.J.: Quadratic Programming with Computer Programs. Advances in Applied Mathematics, vol. 1. Chapman and Hall, London (2017)
17. Billionnet, A., Elloumi, S., Lambert, A.: An efficient compact quadratic convex reformulation for general integer quadratic programs. Comput. Optim. Appl. **54**(1), 141–162 (2013)

18. 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 Appl. Math. **157**(6), 1185–1197 (2009)
19. Bixby, E.R., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: MIP: theory and practice–closing the gap. In: Powell, M.J.D., Scholtes, S. (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, Boston (2000)
20. Bley, A., Gleixner, A.M., Koch, T., Vigerske, S.: Comparing MIQCP solvers to a specialised algorithm for mine production scheduling. In: Bock, H.G., Hoang, X.P., Rannacher, R., Schlöder, J.P. (eds.) Modeling, Simulation and Optimization of Complex Processes, pp. 25–39. Springer, Berlin (2012)
21. Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines. Bull. Am. Math. Soc. **21**(1), 1–46 (1989)
22. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, pp. 1–74. Springer, Boston (1999)
23. 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 Optim. **5**(2), 186–204 (2008)
24. Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A., Toth, P.: On the optimal design of water distribution networks: a practical MINLP approach. Optim. Eng. **13**(2), 219–246 (2012)
25. Buchheim, C., Wiegele, A.: Semidefinite relaxations for non-convex quadratic mixed-integer programming. Math. Program. **141**(1), 435–452 (2013)
26. Burer, S.: Copositive programming. In: Anjos, F.M., Lasserre, B.J. (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, pp. 201–218. Springer, Boston (2012)
27. Burer, S., Saxena, A.: The MILP road to MIQCP. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, The IMA Volumes in Mathematics and its Applications, vol. 154, pp. 373–405. Springer, Boston (2012)
28. Bussieck, M.R., Vigerske, S.: MINLP solver software. In: C, J.J., et al. (eds.) Wiley Encyclopedia of Operations Research and Management Science. Wiley, London (2010)
29. Byrd, R.H., Nocedal, J., Waltz, R.: KNITRO: an integrated package for nonlinear optimization. In: di Pillo, G., Roma, M. (eds.) Large-Scale Nonlinear Optimization, Nonconvex Optimization and Its Applications. Springer, Boston (2006)
30. 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. Comput. Chem. Eng. **30**(1), 54–69 (2005)
31. Castillo, P.A.C., Mahalec, V., Kelly, J.D.: Inventory pinch algorithm for gasoline blend planning. AIChE J. **59**(10), 3748–3766 (2013)
32. Castro, J., Frangioni, A., Gentile, C.: Perspective reformulations of the CTA Problem with L_2 distances. Oper. Res. **62**(4), 891–909 (2014)
33. Castro, P.M., Teles, J.P.: Comparison of global optimization algorithms for the design of water-using networks. Comput. Chem. Eng. **52**, 249–261 (2013)
34. Conn, A.R., Gould, N.I.M., Orban, D., Toint, P.L.: A primal-dual trust-region algorithm for non-convex nonlinear programming. Math. Program. **87**(2), 215–249 (2000)
35. Dakin, R.: A tree search algorithm for mixed programming problems. Comput. J. **8**(3), 250–255 (1965)
36. D'Ambrosio, C., Linderoth, J., Luedtke, J.: Valid inequalities for the pooling problem with binary variables. In: Günlük, O., Woeginger, G.J. (eds.) Integer Programming and Combinatoral Optimization. Lecture Notes in Computer Science, vol. 6655, pp. 117–129. Springer, Berlin (2011)
37. 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. Syst. Sci. Syst. Eng. **22**(4), 385–400 (2013)
38. Dong, H.: Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. SIAM J. Optim. **26**(3), 1962–1985 (2016)
39. Dorneich, M.C., Sahinidis, N.V.: Global optimization algorithms for chip layout and compaction. Eng. Optim. **25**(2), 131–154 (1995)
40. Dostál, Z.: Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities. Springer, Heidelberg (2009)

41. Drud, A.: CONOPT: a GRG code for large sparse dynamic nonlinear optimization problems. Math. Program. **31**(2), 153–191 (1985)
42. Drud, A.S.: CONOPT: a large-scale GRG code. INFORMS J. Comput. **6**(2), 207–216 (1994)
43. Drud, A.S.: SBB. ARKI Consulting and Development A/S (2017). https://www.gams.com/25.0/docs/S_SBB.html. Accessed Sept 2017
44. Dür, M.: Copositive programming: a survey. In: Diehl, M., Glineur, F., Jarlebring, E., Michiels, W. (eds.) Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization, pp. 3–20. Springer, Berlin (2010)
45. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**(3), 307–339 (1986)
46. Erway, J.B., Gill, P.E.: A subspace minimization method for the trust-region step. SIAM J. Optim. **20**(3), 1439–1461 (2010)
47. 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 J. **58**(8), 2320–2335 (2012)
48. FICO: Xpress optimization suite (2017). http://www.fico.com/en/products/fico-xpress-optimization-suite. Accessed Sept 2017
49. Fletcher, R.: Stable reduced Hessian updates for indefinite quadratic programming. Math. Program. **87**(2), 251–264 (2000)
50. Floudas, C., Visweswaran, V.: A global optimization algorithm (GOP) for certain classes of nonconvex NLPs-I. Theory Comput. Chem. Eng. **14**(12), 1397–1417 (1990)
51. Frangioni, A., Furini, F., Gentile, C.: Approximated perspective relaxations: a project and lift approach. Comput. Optim. Appl. **63**(3), 705–735 (2016)
52. Frangioni, A., Galli, L., Scutellà, M.: Delay-constrained shortest paths: approximation algorithms and second-order cone models. J. Optim. Theory Appl. **164**(3), 1051–1077 (2015)
53. Frangioni, A., Galli, L., Stea, G.: Delay-constrained routing problems: accurate scheduling models and admission control. Comput. Oper. Res. **81**, 67–77 (2017)
54. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0–1 mixed integer programs. Math. Program. **106**(2), 225–236 (2006)
55. Frangioni, A., Gentile, C.: SDP diagonalizations and perspective cuts for a class of nonseparable MIQP. Oper. Res. Lett. **35**(2), 181–185 (2007)
56. Frangioni, A., Gentile, C.: A computational comparison of reformulations of the perspective relaxation: SOCP vs cutting planes. Oper. Res. Lett. **37**(3), 206–210 (2009)
57. Frangioni, A., Gentile, C., Grande, E., Pacifici, A.: Projected perspective reformulations with applications in design problems. Oper. Res. **59**(5), 1225–1232 (2011)
58. Geissler, B., Morsi, A., Schewe, L.: A new algorithm for MINLP applied to gas transport energy cost minimization. In: Jünger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 321–353. Springer, Berlin (2013)
59. Gentilini, I., Margot, F., Shimada, K.: The travelling salesman problem with neighbourhoods: MINLP solution. Optim. Methods Softw. **28**(2), 364–378 (2013)
60. Gertz, E.M., Wright, S.J.: Object-oriented software for quadratic programming. ACM Trans. Math. Softw. **29**(1), 58–81 (2003)
61. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**(4), 979–1006 (2002)
62. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005)
63. Gill, P.E., Wong, E.: Methods for convex and general quadratic programming. Math. Program. Comput. **7**(1), 71–112 (2015)
64. Gleixner, A.M., Held, H., Huang, W., Vigerske, S.: Towards globally optimal operation of water supply networks. Numer. Algebra Control Optim. **2**(4), 695–711 (2012)
65. Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Solving the trust-region subproblem using the Lanczos method. SIAM J. Optim. **9**(2), 504–525 (1999)
66. Gould, N.I.M., Orban, D., Robinson, D.P.: Trajectory-following methods for large-scale degenerate convex quadratic programming. Math. Program. Comput. **5**(2), 113–142 (2013)
67. Gould, N.I.M., Orban, D., Toint, P.L.: GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. ACM Trans. Math. Softw. **29**(4), 353–372 (2003)
68. Gould, N.I.M., Robinson, D.P.: A dual gradient-projection method for large-scale strictly convex quadratic problems. Comput. Optim. Appl. **67**(1), 1–38 (2017)

69. Gould, N.I.M., Robinson, D.P., Thorne, H.S.: On solving trust-region and other regularised subproblems in optimization. Math. Program. Comput. **2**(1), 21–57 (2010)
70. Gould, N.I.M., Toint, PhL: A Quadratic Programming Bibliography. Numerical Analysis Group Internal Report 2000-1. Rutherford Appleton Laboratory, Chilton (2000)
71. Gould, N.I.M., Toint, P.L.: An iterative working-set method for large-scale non-convex quadratic programming. Appl. Numer. Math. **43**(1–2), 109–128 (2002)
72. 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)
73. Hager, W.W.: Minimizing a quadratic over a sphere. SIAM J. Optim. **12**(1), 188–208 (2001)
74. Hasan, M.M.F., Karimi, I.A., Avison, C.M.: Preliminary synthesis of fuel gas networks to conserve energy and preserve the environment. Ind. Eng. Chem. Res. **50**(12), 7414–7427 (2011)
75. Hemmecke, R., Köppe, M., Lee, J., Weismantel, R.: Nonlinear integer programming. In: Jünger, M., Liebling, M.T., Naddef, D., Nemhauser, L.G., Pulleyblank, R.W., Reinelt, G., Rinaldi, G., Wolsey, A.L. (eds.) 50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art, pp. 561–618. Springer, Berlin (2010)
76. Hifi, M., M'Hallah, R.: A literature review on circle and sphere packing problems: models and methodologies. Adv. Oper. Res. **2009**, 22 (2009)
77. IBM ILOG: CPLEX Optimization Studio, 12.7.0 edn. (2016). http://www.ibm.com/support/knowledgecenter/SSSA5P
78. Jeroslow, R.: There cannot be any algorithm for integer programming with quadratic constraints. Oper. Res. **21**(1), 221–224 (1973)
79. Jeżowski, J.: Review of water network design methods with literature annotations. Ind. Eng. Chem. Res. **49**(10), 4475–4516 (2010)
80. Kallrath, J.: Exact computation of global minima of a nonconvex portfolio optimization problem. In: Floudas, C.A., Pardalos, P.M. (eds.) Frontiers in Global Optimization, pp. 237–254. Kluwer Academic Publishers, Alphen aan den Rijn (2003)
81. Kallrath, J.: Cutting circles and polygons from area-minimizing rectangles. J. Global Optim. **43**(2–3), 299–328 (2009)
82. Kallrath, J., Rebennack, S.: Cutting ellipses from area-minimizing rectangles. J. Global Optim. **59**(2–3), 405–437 (2014)
83. Khor, C.S., Chachuat, B., Shah, N.: Fixed-flowrate total water network synthesis under uncertainty with risk management. J. Clean. Prod. **77**, 79–93 (2014)
84. 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. Math. Program. Comput. **3**(2), 103–163 (2011)
85. Kochenberger, G., Hao, J.K., Glover, F., Lewis, M., Lü, Z., Wang, H., Wang, Y.: The unconstrained binary quadratic programming problem: a survey. J. Comb. Optim. **28**(1), 58–81 (2014)
86. Kocis, G.R., Grossmann, I.E.: Computational experience with DICOPT solving MINLP problems in process systems engineering. Comput. Chem. Eng. **13**(3), 307–315 (1989)
87. Kolodziej, S.P., Castro, P.M., Grossmann, I.E.: Global optimization of bilinear programs with a multiparametric disaggregation technique. J. Global Optim. **57**(4), 1039–1063 (2013)
88. 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. Comput. Chem. Eng. **53**, 122–142 (2013)
89. Krislock, N., Malick, J., Roupin, F.: BiqCrunch: a semidefinite branch-and-bound method for solving binary quadratic problem. ACM Trans. Math. Softw. **43**(4), 32:1–32:23 (2017)
90. Land, A., Doig, A.: An automatic method of solving discrete programming problems. Econometrica **28**(3), 497–520 (1960)
91. 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, Austin (2006)
92. Lee, G., Tam, N., Yen, N.: Quadratic Programming and Affine Variational Inequalities: A Qualitative Study. Nonconvex Optimization and Its Applications. Springer, Boston (2006)
93. Li, J., Li, A., Karimi, I.A., Srinivasan, R.: Improving the robustness and efficiency of crude scheduling algorithms. AIChE J. **53**(10), 2659–2680 (2007)

94. Li, J., Misener, R., Floudas, C.A.: Continuous-time modeling and global optimization approach for scheduling of crude oil operations. AIChE J. **58**(1), 205–226 (2012)
95. Li, J., Misener, R., Floudas, C.A.: Scheduling of crude oil operations under demand uncertainty: a robust optimization framework coupled with global optimization. AIChE J. **58**(8), 2373–2396 (2012)
96. Li, X., Armagan, E., Tomasgard, A., Barton, P.I.: Stochastic pooling problem for natural gas production network design and operation under uncertainty. AIChE J. **57**(8), 2120–2135 (2011)
97. Li, X., Tomasgard, A., Barton, P.I.: Decomposition strategy for the stochastic pooling problem. J. Global Optim. **54**(4), 765–790 (2012)
98. Lin, X., Floudas, C.A., Kallrath, J.: Global solution approach for a nonconvex MINLP problem in product portfolio optimization. J. Global Optim. **32**(3), 417–431 (2005)
99. Lin, Y., Schrage, L.: The global solver in the LINDO API. Optim. Methods Softw. **24**(4–5), 657–668 (2009)
100. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. Eur. J. Oper. Res. **176**(2), 657–690 (2007)
101. Maranas, C.D., Androulakis, I.P., Floudas, C.A., Berger, A.J., Mulvey, J.M.: Solving long-term financial planning problems via global optimization. J. Econ. Dyn. Control **21**(8–9), 1405–1425 (1997)
102. Misener, R., Floudas, C.A.: Advances for the pooling problem: modeling, global optimization, and computational studies. Appl. Comput. Math. **8**(1), 3–22 (2009)
103. Misener, R., Floudas, C.A.: Global optimization of large-scale pooling problems: quadratically constrained MINLP models. Ind. Eng. Chem. Res. **49**(11), 5424–5438 (2010)
104. Misener, R., Floudas, C.A.: GloMIQO: global mixed-integer quadratic optimizer. J. Global Optim. **57**(1), 3–50 (2013)
105. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. J. Global Optim. **59**(2–3), 503–526 (2014)
106. Mouret, S., Grossmann, I.E., Pestiaux, P.: A novel priority-slot based continuous-time formulation for crude-oil scheduling problem. Ind. Eng. Chem. Res. **48**(18), 8515–8528 (2009)
107. Mouret, S., Grossmann, I.E., Pestiaux, P.: A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. Comput. Chem. Eng. **35**(12), 2750–2766 (2011)
108. Murtagh, B.A., Saunders, M.A.: Large-scale linearly constrained optimization. Math. Program. **14**(1), 41–72 (1978)
109. Murtagh, B.A., Saunders, M.A.: A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. In: Buckley, A.G., Goffin, J.L. (eds.) Algorithms for Constrained Minimization of Smooth Nonlinear Functions, Mathematic Programming Studies, vol. 16, pp. 84–117. Springer, Berlin (1982)
110. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. Acta Numer. **13**, 271–369 (2004)
111. Nyberg, A., Grossmann, I.E., Westerlund, T.: The optimal design of a three-echelon supply chain with inventories under uncertainty (2012). http://www.minlp.org/library/problem/index.php?i=157. Accessed Sept 2017
112. Papageorgiou, D.J., Toriello, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Fixed-charge transportation with product blending. Transp. Sci. **46**(2), 281–295 (2012)
113. Parpas, P., Rustem, B.: Global optimization of the scenario generation and portfolio selection problems. In: Gavrilova, M., Gervasi, O., Kumar, V., Tan, C., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) Computational Science and Its Applications-ICCSA 2006. Lecture Notes in Computer Science, vol. 3982, pp. 908–917. Springer, Berlin (2006)
114. Pham, V., Laird, C., El-Halwagi, M.: Convex hull discretization approach to the global optimization of pooling problems. Ind. Eng. Chem. Res. **48**(4), 1973–1979 (2009)
115. Pillo, G.D., Grippo, L., Lampariello, F.: A class of structured quasi-newton algorithms for optimal control problems. IFAC Proc. Vol. **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
116. Pintér, J.D.: LGO: a program system for continuous and Lipschitz global optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds.) Developments in Global Optimization, pp. 183–197. Springer, Boston (1997)
117. Pintér, J.D.: A model development system for global optimization. In: De Leone, R., Murli, A., Pardalos, P.M., Toraldo, G. (eds.) High Performance Algorithms and Software in Nonlinear Optimization, pp. 301–314. Springer, Boston (1998)

118. 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. Comput. Chem. Eng. **34**(3), 318–330 (2010)
119. Rebennack, S., Kallrath, J., Pardalos, P.M.: Column enumeration based decomposition techniques for a class of non-convex MINLP problems. J. Global Optim. **43**(2–3), 277–297 (2009)
120. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**(2), 307–335 (2008)
121. Rendl, F., Wolkowicz, H.: A semidefinite framework for trust region subproblems with applications to large scale minimization. Math. Program. **77**(1), 273–299 (1997)
122. Rios, L.M., Sahinidis, N.V.: Portfolio optimization for wealth-dependent risk preferences. Ann. Oper. Res. **177**(1), 63–90 (2010)
123. Rothberg, E.: Solving quadratically-constrained models using Gurobi (2012). http://www.gurobi.com/resources/seminars-and-videos/gurobi-quadratic-constraints-webinar. Accessed Sept 2017
124. 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. Optim. Lett. **5**(1), 1–11 (2011)
125. Ruiz, M., Briant, O., Clochard, J.M., Penz, B.: Large-scale standard pooling problems with constrained pools and fixed demands. J. Global Optim. **56**(3), 939–956 (2013)
126. Saif, Y., Elkamel, A., Pritzker, M.: Global optimization of reverse osmosis network for wastewater treatment and minimization. Ind. Eng. Chem. Res. **47**(9), 3060–3070 (2008)
127. Schittkowski, K.: Numerical solution of a time-optimal parabolic boundary-value control problem. J. Optim. Theory Appl. **27**(2), 271–290 (1979)
128. Stojanovic, S.: Optimal damping control and nonlinear elliptic systems. SIAM J. Control Optim. **29**(3), 594–608 (1991)
129. Szabó, P.G., Markót, C.M., Csendes, T.: Global optimization in geometry: circle packing into the square. In: Audet, C., Hansen, P., Savard, G. (eds.) Essays and Surveys in Global Optimization, pp. 233–265. Springer, New York (2005)
130. Tadayon, B., Smith, J.C.: Algorithms for an integer multicommodity network flow problem with node reliability considerations. J. Optim. Theory Appl. **161**(2), 506–532 (2013)
131. Tahanan, M., van Ackooij, W., Frangioni, A., Lacalandra, F.: Large-scale unit commitment under uncertainty. 4OR **13**(2), 115–171 (2015)
132. Tarski, A.: A decision method for elementary algebra and geometry. Technical Reports R-109, Rand Corporation (1951)
133. 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, Alphen aan den Rijn (2002)
134. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Program. **99**(3), 563–591 (2004)
135. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005)
136. Teles, J.P., Castro, P.M., Matos, H.A.: Global optimization of water networks design using multiparametric disaggregation. Comput. Chem. Eng. **40**, 132–147 (2012)
137. 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 J. Comput. **19**(3), 328–340 (2007)
138. Vavasis, S.: Quadratic programming is in NP. Inf. Process. Lett. **36**, 73–77 (1990)
139. 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). http://www.gamsworld.org/minlp/minlplib2
140. Vigerske, S., Gleixner, A.: SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Optim. Methods Softw. **33**(3), 563–593 (2018)
141. Viswanathan, J., Grossmann, I.E.: A combined penalty function and outer-approximation method for MINLP optimization. Comput. Chem. Eng. **14**(7), 769–782 (1990)
142. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
143. Westerlund, T., Lundquist, K.: Alpha-ECP, version 5.04. an interactive MINLP-solver based on the extended cutting plane method. Technical Reports 01-178-A. Process Design Laboratory, Åbo Akademi University, Åbo, Finland (2003)

144. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optim. Eng. **3**(3), 253–280 (2002)
145. Wikipedia: Quadratic programming (2016). https://en.wikipedia.org/wiki/Quadratic_programming. Accessed Sept 2017
146. Wikipedia: Quadratically constrained quadratic program (2016). https://en.wikipedia.org/wiki/Quadratically_constrained_quadratic_program. Accessed Sept 2017
147. Wright, S.: Primal-Dual Interior-Point Method. SIAM, Philadelphia (1997)

Affiliations

Fabio Furini[1] · Emiliano Traversi[2] · Pietro Belotti[3] · Antonio Frangioni[4] · Ambros Gleixner[5] · Nick Gould[6] · Leo Liberti[7] · Andrea Lodi[8] · Ruth Misener[9] · Hans Mittelmann[10] · Nikolaos V. Sahinidis[11] · Stefan Vigerske[12] · Angelika Wiegele[13]

Emiliano Traversi
emiliano.traversi@lipn.fr

Pietro Belotti
pietrobelotti@fico.com

Antonio Frangioni
frangio@di.unipi.it

Ambros Gleixner
gleixner@zib.de

Nick Gould
nick.gould@stfc.ac.uk

Leo Liberti
liberti@lix.polytechnique.fr

Andrea Lodi
andrea.lodi@polymtl.ca

Ruth Misener
r.misener@imperial.ac.uk

Hans Mittelmann
mittelmann@asu.edu

Nikolaos V. Sahinidis
sahinidis@cmu.edu

Stefan Vigerske
svigerske@gams.com

Angelika Wiegele
angelika.wiegele@aau.at

[1] LAMSADE, Université Paris Dauphine, 75775 Paris, France

[2] LIPN, Université de Paris 13, 93430 Villetaneuse, France

[3] Xpress-Optimizer Team, FICO, Birmingham, UK

[4] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 2, 56127 Pisa, Italy

[5] Department of Mathematical Optimization, Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany

[6] STFC-Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England

[7] CNRS LIX, École Polytechnique, 91128 Palaiseau, France

[8] CERC, École Polytechnique de Montreal, Montreal, Canada

[9] Department of Computing, Imperial College London, London, UK

[10] School of Mathematical and Statistical Sciences, Arizona State University, Tempe, AZ 85287-1804, USA

[11] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[12] GAMS Software GmbH, c/o Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany

[13] Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, 9020 Klagenfurt am Wörthersee, Austria