RAIRO-Oper. Res. 53 (2019) 81–109
https://doi.org/10.1051/ro/2018036

**RAIRO Operations Research**
www.rairo-ro.org

# UNDECIDABILITY AND HARDNESS IN MIXED-INTEGER NONLINEAR PROGRAMMING

Leo Liberti[*]

**Abstract.** We survey two aspects of mixed-integer nonlinear programming which have attracted less attention (so far) than solution methods, solvers and applications: namely, whether the class of these problems can be solved algorithmically, and, for the subclasses which can, whether they are hard to solve. We start by reviewing the problem of representing a solution, which is linked to the correct abstract computational model to consider. We then cast some traditional logic results in the light of mixed-integer nonlinear programming, and come to the conclusion that it is not a solvable class: instead, its formal sentences belong to two different theories, one of which is decidable while the other is not. Lastly, we give a tutorial on computational complexity and survey some interesting hardness results in nonconvex quadratic and nonlinear programming.

**Mathematics Subject Classification.** 90C11, 90C26.

Received July 28, 2017. Accepted May 4, 2018.

## Table of Contents

*Keywords and phrases:* Undecidability, hardness, mathematical programming.

CNRS LIX, École Polytechnique, 91128 Palaiseau, France.

[*] Corresponding author: liberti@lix.polytechnique.fr

## 1. Introduction

This survey paper discusses two of the lesser known aspects of mixed-integer nonlinear programming (MINLP): computability and computational complexity. More precisely, we focus on their negative sides: undecidability and hardness.

Mathematical programming (MP) formulations of the MINLP class can model any single-level, single-objective MP problem, as long as it does not involve black-box functions or oracles. An MINLP formulation is formally written as follows:

$$\left.\begin{array}{rrcl} \min\limits_{x\in\mathbb{F}^n} & f(x) & & \\ & g(x) & \in & [g^L, g^U] \\ & x & \in & [x^L, x^U] \\ \forall j \in Z & x_j & \in & \mathbb{Z}. \end{array}\right\} \tag{1.1}$$

The symbols $n, f, g, g^L, g^U, x^L, x^U, Z$ in equation (1.1) encode the input of the problem (also known as the *instance*). The formal sentence equation (1.1) reads informally as follows: given a positive integer $n$, a function $f : \mathbb{F}^n \to \mathbb{F}$ (where $\mathbb{F}$ is a number field containing $\mathbb{Z}$), a vector-valued function $g : \mathbb{F}^n \to \mathbb{F}^m$, vectors $g^L \leq g^U \in \mathbb{F}^m$ and $x^L \leq x^U \in \mathbb{F}^n$, and an index set $Z \subseteq \{1, \ldots, n\}$, find a vector $x^* \in \mathbb{F}^n$ to be assigned to the decision variable vector $x$, such that $x^*$ minimizes the given objective function $f(x)$ and is feasible with respect to (w.r.t.) the given constraints, *i.e.*, $g(x^*) \in [g^L, g^U]$ and $x^* \in [x^L, x^U]$.

The field $\mathbb{F}$ is usually one of the following:

- the field $\mathbb{R}$ of reals (replaced for practical computation by the set of floating point representable reals);
- the field $\mathbb{Q}$ of rationals (for complexity results);
- the field $\mathbb{C}$ of complex numbers (which turns up in few but important applications, *e.g.*, alternating current optimal power flow problems [78]);
- a cartesian product of an appropriate number of copies (matching the number of components of $x$) of any of the above.

The symbols $f, g$ in equation (1.1) do not range over all possible functions: they are assumed to describe functions in "closed form". More formally, they are sentences of a formal language involving variable symbols, numerical constants from $\mathbb{Q}$ and a finite number of arithmetical, algebraic and possibly even transcendental operators, *e.g.*, $3x_1 - 4x_2$, $x_1^2 x_3$ and $\log(\frac{1}{x_1})$. This language is used to represent functions in a way that can be encoded by a computer. Note that a well-formed string of this language corresponds to a unique function (however, the same function might have more than a single string description), and that there are countably many strings in this language, but uncountably many functions: it follows that we cannot represent every function. But since polynomials are representable in this form, and they are dense in all continuous functions over an interval, we can at least approximate all continuous bounded functions as closely as we wish, provided the string length is not bounded.

If the objective function and the continuous relaxation of the feasible region of equation (1.1) are convex, the formulation is called "convex MINLP" (cMINLP). Notice, however, that because of the integrality constraints, a convex MINLP is *not* in general a convex optimization problem. If $Z = \varnothing$ equation (1.1) is a Nonlinear Programming (NLP) problem, called "convex NLP" (cNLP) if the objective and feasible region are convex.

Natural processes are mostly nonlinear, whereas man-made processes are often linear or at least well-approximated by linear forms. As a result MINLP ends up being the tool of choice in man-made processes at the interface with natural phenomena, such as chemical and bio-chemical engineering [32, 37, 38], electro-mechanical engineering [19, 65], energy management [16, 73, 74], reservoir engineering [58], alternating current optimal power flow [76, 78], and water network design [14]. Typically, the nonlinearities occur as a consequence of the natural processes, and the integer variables (which are often binary) model the turning on and off of certain man-made controllers. MINLP is also used to model some man-made processes, such as portfolio optimization [8, 64], trim loss minimization in the cutting industry [42, 50], and more [26, 53]. Moreover, MINLP occurs in purely mathematical problems, such as the Kissing Number Problem (KNP) [60], where the nonlinearities arising from packing spheres around a spherical surface, with binary variables that control the presence or absence of the spheres.

MINLP is by now a large field of study [53]: large enough to make a "universal MINLP survey" impossible to write in the space of an article, and even a complete MINLP book a significant hurdle. This survey takes the point of view of computability and complexity. Most of the discussed results are "old" (by MINLP standards) and established. But not all may be widely known within the MINLP community. This survey does not aim at relating *every* co-occurrence of MINLP and computability or complexity: the choice is personal, and based on interest, usefulness, and/or elegance. Many proofs are sketched only briefly. Many works in nonlinear discrete optimization do not appear here because a recent survey already exists [44].

For more information about MINLP, nonconvex NLP, applications, and solution algorithms, see the following surveys [6, 18, 27, 39, 69] and books [32, 33, 53, 80, 82]. Some sources also delve in complexity issues, specially as regards efficiently solvable cases [44]. Further information in topics discussed in this survey can be found in more specialized surveys or books [12, 30, 85].

## 2. Representability

The solutions of linear programs (LP) and mixed-integer linear programs (MILP) have rational components as long as the input data is rational. For MINLP, the situation is not so clear, as solution components might involve irrational numbers (both algebraic and transcendental, depending on the involved functions). To the best of our knowledge, there are three main approaches to representability.

### 2.1. The real RAM model

The computational complexity model of Blum *et al.* [10] (often referred to as *real RAM* model) essentially eschews the problem of representation, as it defines the real equivalent of computational complexity classes such as **P**, **NP** and their relationships. In the real RAM model, storing or reading any real number has unit cost. This model of computation is mostly used for theoretical results concerning the complexity of algorithms in numerical analysis, scientific computing, and dynamical systems.

The real RAM computational model focuses on the complexity of the computational process while discounting the difficulties of representing real numbers. It extends the definitions of complexity classes such as **P**, **NP** and the notion of completeness for a complexity class. For example, the problem of solving the class of multivariate polynomial equations of degree 4 is shown to play the same role as SATISFIABILITY (SAT) for the Turing Machine (TM) [83] model, *i.e.*, it is **NP**-complete.

### 2.2. Approximations

The approach taken by the Global Optimization (GO) community, specially those who develop sBB algorithms, consists in finding a (rational, or rather floating point encoded) solution $x^*$ yielding an objective function value $f^* = f(x^*)$ within a given $\varepsilon > 0$ of the true globally optimal function value $\tilde{f}$:

$$|f^* - \tilde{f}| \leq \varepsilon. \tag{2.1}$$

Note that, since the true optimal value $\tilde{f}$ is not known, in practice equation (2.1) is enforced by requiring that $|f^* - \bar{f}| \leq \varepsilon$, where $\bar{f}$ is a guaranteed lower bound to the optimal objective function value, usually computed by solving a convex relaxation ([54], Sect. 3.7) of equation (1.1).

The approach considered in [47] is more accurate, as it insists that the solution $x^*$ is within a given $\varepsilon > 0$ tolerance of the true optimum $\tilde{x}$:

$$\|x^* - \tilde{x}\|_\infty \leq \varepsilon. \tag{2.2}$$

This means that $x^*$ is the same as $\tilde{x}$ in $O(\log \frac{1}{\varepsilon})$ decimal digits. Other than that, all arithmetical operations on reals take $O(1)$ time. Since one only needs to consider $O(\log \frac{1}{\varepsilon})$ decimal digits, this assumption places this representation approach within the TM model of computation.

## 2.3. Representability of algebraic numbers

Better representations, but limited to algebraic numbers, are based on their minimal polynomials. The most common is the *Thom encoding*, a pair $(p_\alpha, \sigma)$, where $p_\alpha(x)$ is the minimal polynomial of $\alpha$ (of degree $d$, say) and $\sigma : \{0, \ldots, d\} \to \{0, -1, 1\}$ encodes the sign of the $k$th derivative of $p_\alpha$ at $\alpha$ ([4], Prop. 2.28). Simpler representations are possible for specific tasks, *e.g.*, $\sigma$ might be replaced by a rational number $a$ which is closer to $\alpha$ than to any other real root of $p_\alpha$ [5].

Identifying minimal polynomials can be achieved using Gröbner bases [17]. There are at least two papers [20, 40] where Gröbner bases are used to diagonalize the polynomial Karush–Kuhn Tucker (KKT) system of a Polynomial Programming (PP) problem (though in those papers the last phase would perform back-substitution with floating point numbers – but those floating point numbers could in principle be used as "closest rationals" in the second representation mentioned in the above paragraph).

Gröbner bases can be found by Buchberger's algorithm [17]. It takes as input a rational multivariate polynomial equation system

$$\forall i \leq m \quad p_i(x) = 0. \tag{2.3}$$

It then proceeds by diagonalizing equation (2.3) to a new polynomial system $\forall i \leq m'$ $(q_i(x) = 0)$ such that the leading terms of $p_i$'s and $q_i$'s (w.r.t. some given fixed monomial order) generate the same ideal.

Like Gaussian elimination, this diagonalization can be used for performing back-substitution as long as the equation $q_{m'}(x)$ only depends on a single variable. Unlike Gaussian elimination, however, $m'$ generally exceeds $m$, and it does not yield a guarantee that $q_i$ will contain strictly fewer variables than $q_{i+1}$: the diagonal structure might contain blocks of polynomials depending on the same set of variables.

Buchberger's algorithm monotonically increases the size of the ideal generated by the leading terms of $G$. The algorithm terminates because, by Hilbert's basis theorem, ascending chains of ideals must eventually become stationary. The termination condition is verified even if the last considered polynomial is not univariate. Although the back-substitution process then generally fails (unless all polynomials are solvable after back-substitution), the diagonal form still holds. Unfortunately, Buchberger's algorithm has doubly exponential computational complexity in general, though it behaves singly exponentially in some cases [3].

An interesting development of Gröbner's bases, involving *chordal networks* [21], yields polynomial systems that are larger than Gröbner bases, but provide the same back-substitution functionality for finding a solution of a polynomial system of equations, and are cheaper to construct.

## 3. Decidability

Can we even solve MINLP? The short answer is no: we cannot solve them in full generality (see Sect. 3.2 below). But the full answer is rather more interesting: MINLP is a class of problems related to two different theories, one of which is decidable, while the other is not.

But first, what does *decidability* mean? This term is almost synonymous with *computability*: a function $f : X \to D$ is computable if there exists a TM that, for any input $x \in X$, provides $f(x)$ as an output. The term "decidable" applies to relations instead of functions: a $k$-ary relation $R$ on $D^k$ is decidable if, given $d_1, \ldots, d_k \in D$, there exists a TM that decides whether the $k$-tuple $d = (d_1, \ldots, d_k)$ is in $R$. Since "being in $R$" can be encoded as a YES/NO type of question, decidability applies to decision problems.

### 3.1. Polynomial feasibility in continuous variables

MINLP contains as a subset all Polynomial Feasibility Problems (PFP) where the variables range over the reals:

$$\forall i \leq m \qquad p_i(x) \quad \mathscr{R} \quad 0, \tag{3.1}$$

where the $p_i$'s are rational polynomials and the relation symbol $\mathscr{R}$ is one of $\{\geq, =\}$. Note that equation (3.1) can in fact be expressed with a single relation $\mathscr{R}$, either $\geq$ or $=$, since every equation can be expressed by a pair of inequalities with opposite sign, and every inequality $p_i(x) \geq 0$ can be rewritten as $p_i(x) + s_i^2 = 0$, where $s_i$ is an additional continuous variable. We consider the question whether there is an algorithm for deciding if an instance of equation (3.1) has a solution or not.

### *3.1.1. Quantifier elimination*

Systems like equation (3.1) are decidable by [81]. Most algorithms for deciding equation (3.1) are based on a technique known as *quantifier elimination*, which reformulates a quantified logical sentence with variable terms into one without quantifiers or free variables. Deciding truth or falsity can then be achieved by elementary manipulations leading to either a tautology $1 = 1$ or a contradiction $0 = 1$. Note that Tarski's algorithm can be applied to a larger class of polynomial systems than equation (3.1): the relation symbol $\mathscr{R}$ can actually range over $\{\geq, >, =, \neq\}$.

Tarski's proof extends for 57 pages of a RAND Corporation technical report [81]. To showcase an easier quantifier elimination example, we shall restrict our discussion to dense linear orders ([59], p. 38). This theory includes logical formulae involving $\forall, \exists, \neg, \wedge, \vee$, the variable symbols $x_1, x_2, \ldots$, the terms $\mathsf{True}$, $\mathsf{False}$, the real numbers as constants, and the relation symbols $<, =$. Sentences of this theory are inductively reduced to a standard form $\exists x_i$ s.t. $q(x)$, where $q$ has no quantifiers. Using the properties of linear order relations, each possible atomic formula involving $s \mathscr{R} t$ (where $s, t$ are either variables or real constants, and $\mathscr{R}$ is a relation in $\{<, =\}$) is reduced to either $\mathsf{True}$ or $\mathsf{False}$. The interesting "mixed" cases, where $s$ is a variable and $t$ is a constant (or *vice versa*) are reduced to intervals of the form $r_1 < x_i < r_2$ where $r_1, r_2$ are real constants, and henceforth to $r_1 < r_2$, which evaluates to either $\mathsf{True}$ or $\mathsf{False}$. Essentially, quantifier elimination reduces this theory to deciding whether an interval is empty or not. In the process, it eliminates variable symbols with the corresponding quantifier.

### *3.1.2. Cylindrical decomposition*

Well after Tarski's 1948 quantifier elimination algorithm, it was discovered [23] that the decision procedure of systems such as equation (3.1) is in some sense a nontrivial extension of the dense linear order case. The solution set of equation (3.1) consists of finitely many connected components, which can be recursively built out of cylinders with bases shaped as points or intervals the extremities of which are either points, or $\pm\infty$, or algebraic curves depending on variables involved at previous recursion levels. Although Collins' algorithm is doubly exponential in the number of variables, a singly exponential algorithm was described in [4].

The cylindrical decomposition result of [23] consists of a topological and a geometric part. The topological part states that the feasible regions of equation (3.1) (where $\mathscr{R}$ ranges in $\{>, \geq, =, \neq\}$), also called *semi-algebraic sets*, can be decomposed into a finite number of connected components. The geometric part gives the recursive description mentioned above in terms of cylinders. The topological part was known previous to Collins' contribution, see *e.g.*, [66].

## 3.2. Polynomial feasibility in integer variables

In this section, we shall consider equation (3.1) subject to integrality constraints on the vector $x$ of decision variables (it suffices to consider equations only, since, as mentioned above, an inequality can be reduced to an equation by adding a single squared unrestricted slack variable):

$$\left.\begin{array}{rrcl} \forall i \leq m & p_i(x) & = & 0 \\ \forall j \leq n & x_j & \in & \mathbb{Z}, \end{array}\right\} \tag{3.2}$$

where we assume that the $p_i$'s are polynomials over $\mathbb{Z}$. In other words, given a system equation (3.2) of polynomial equations in integers, can we decide whether it has a solution?

### *3.2.1. Undecidability vs. incompleteness*

We introduce the notion of a *formal system* $S$ informally, as follows. $S$ is a quadruplet $(L, G, A, R)$ where $L$ is a finite language, $G$ is a grammar over $L$ which defines a set of well-formed formulae called *sentences*, $A$ is a set of given sentences called *axioms*, and $R$ is a set of *inference rules* (*e.g.*, *modus ponens*) which define relations over sentences. Given a sentence $p$, a *proof* for $p$ in $S$ is a sequence of iterative applications of inference rules to sentences that are either axioms or for which a proof was already provided.

Undecidability is sometimes related to Gödel's incompleteness theorem [34]. Undecidability and incompleteness are, however, two different notions. A consistent formal system $S$ is *incomplete* when it has at least a sentence $p$ such that neither $p$ nor $\neg p$ are provable within $S$. Gödel's (first) incompleteness theorem states that any $S$ capable of encoding arithmetic in $\mathbb{N}$ is either inconsistent or incomplete. This leaves open the question of decidability, *i.e.*, of whether there exists an algorithm that, given any $p$ in $S$, decides whether $p$ is provable within $S$ or not.

There is a common misconception that if a formal system $S$ is decidable, then it cannot be incomplete. The (faulty) argument goes as follows. If a formal system $S$ is decidable, then for any given sentence $p$ it is possible to decide whether $p$ is provable in $S$ or not. So, in case $p$ is provable, then $\neg p$ cannot be, and *vice versa*. Hence, either $p$ or $\neg p$ must be provable within $S$, which implies that $S$ cannot be incomplete.

The error arises in thinking that the "*vice versa*" above exhausts all possibilities. For two sentences $p$ and $\neg p$ in $S$, there are four assignments of the property "provability within $S$" to the two sentences: (i) both are provable, (ii) the first is provable and the second is not, (iii) the first is not and the second is, and (iv) neither is provable. The first assignment leads to the inconsistency of $S$, it should be discarded since we assumed that $S$ is consistent. The second and third assignments are part of the above (faulty) argument. The error stems from forgetting the fourth assignment, *i.e.*, that both $p$ and $\neg p$ may fail to be provable within $S$. This condition is described by saying that $p$ is *independent* of $S$. If $p$ is independent of $S$, a decision algorithm for provability in $S$ would answer NO when given either $p$ or $\neg p$ as input. So, a formal system can be decidable but incomplete.

Let us see an example, the formal systems of algebraically closed fields of prime characteristic are both decidable and incomplete. Up to isomorphism, there is an algebraically closed field of prime characteristic for each distinct prime integer $q$. Given $q$, let us consider the formal system $\mathsf{ACF}_q$. Its axioms are: the field axioms, an axiom schema stating that every polynomial splits, and an axiom stating that the characteristic of the field is a finite prime $q$, namely $C_q \equiv \sum_{i\le q} 1 = 0$ (the language, grammar, and rules of inference are the ordinary ones from elementary mathematics). Because there exist algebraically closed fields of characteristic zero (*e.g.*, $\mathbb{C}$), it is clear that the finite characteristic axioms are independent of the rest of the axioms (otherwise, if there existed proofs of the finite characteristic axioms from the other field axioms, $\mathbb{C}$ could not have characteristic zero). And yet, the finite characteristic axioms are easy to recognize: a TM could scan a sentence and syntactically decide whether it is a finite characteristic axiom or not, for example using a regular expression. Therefore, a decision algorithm for $\mathsf{ACF}_q$ can be devised as follows. Given a sentence $p$:

(i) if $p$ is a finite characteristic axiom for $q$ return True (meaning: the sentence is an axiom of the FS so it is its own zero-line proof within the FS);
(ii) if $p$ is a finite characteristic axiom for $q' \neq q$, or its negation, return False (meaning: there is no proof for $p$ within the FS);
(iii) if $p$ is neither, use a quantifier elimination algorithm for finite fields (*e.g.*, [35]) to determine whether the $p$ is True or False.

So, these formal systems are decidable but incomplete. Note that the decision algorithm operates within the FS, but the reason why it works has to be explained in the meta-language, outside of the FS.

As it turns out, if $S$ encodes arithmetic in $\mathbb{N}$ and is consistent, it is not only incomplete (by Gödel's first incompleteness theorem) but also undecidable. This was settled by in [83] by a diagonalization argument involving an undecidable decision problem called the *Halting problem* – is there an algorithm for deciding whether a given TM terminates or not?

### *3.2.2. Hilbert's 10th problem*

We now come back to our MINLP feasibility problem in equation (3.2). Each equation in the system is known as a *Diophantine equation* (DE). Let $S$ be a formal system encoding arithmetic in $\mathbb{N}$. Systems of DEs are certainly well-formed formulae within $S$. Although the set of *all* well-formed formulae in $S$ is undecidable, we still do not know whether the limited subset described by the form of equation (3.2) is "general enough" to be undecidable. The common property of all feasibility systems is that the integer variables $x_1, \ldots, x_n$ are implicitly quantified by *existential* quantifiers "$\exists$", but no *universal* quantifier "$\forall$". Moreover, equation (3.2) only consists of sentences involving polynomials, whereas exponentiation is also part of arithmetic in integers. We can now frame the question as follows: is the set of all existentially quantified polynomial sentences of $S$ decidable or not? This can be seen as a modern restatement of Hilbert's 10th problem.

From the proof of Gödel's first incompleteness theorem [36], it is clear that Gödel only used a finite number of *bounded* universal quantifiers in order to construct his argument. Davis, in [28], shows that a single bounded universal quantifier is sufficient to encode undecidable statements:

$$\exists y \forall z \le y \exists x_1, \ldots, x_n \quad p(y, z, x) = 0, \tag{3.3}$$

where $p$ is an integral polynomial, and $x_1, \ldots, x_n, y, z$ are all in $\mathbb{N}$. In [29], it is shown that there exist undecidable problems that are almost in the desired form:

$$\exists x_1, \ldots, x_n \in \mathbb{N} \quad \eta(x) = 0, \tag{3.4}$$

but where $\eta$ is an *exponential* DE, *i.e.*, a function that can be written using arithmetical operations and exponentiation. In a landmark result, Matiyasevich proved in 1970 [61] that the exponential relationship $a = b^c$ can be expressed by means of a DE (*i.e.*, without exponentiation), thereby settling Hilbert's 10th problem in the negative. This result is now known as the Matiyasevich–Davis–Putnam–Robinson (MDPR) theorem. In the setting of this paper, the MDPR theorem means that

$$\exists x_1, \ldots, x_n \in \mathbb{N} \quad p(x) = 0, \tag{3.5}$$

where $p$ is a polynomial, is generally undecidable. This makes equation (3.2) undecidable, which, by inclusion, makes MINLP undecidable, too.

The MDPR theorem is more general than it may seem at first sight.

(i) A system of DEs such as equation (3.2) is equivalent to a single DE such as equation (3.5): it suffices to write

$$p(x) = \sum_{i \le m} (p_i(x))^2.$$

(ii) The undecidability results relate to $\mathbb{N}$ rather than $\mathbb{Z}$, but this is without loss of generality: we can transform any DE with solutions in $\mathbb{Z}$ to one with solutions in $\mathbb{N}$ by writing $x_i = y_i^+ - y_i^-$ (where $y_i^+, y_i^- \in \mathbb{N}$). Conversely, since every non-negative integer is the sum of four squares, we can express the (non-negative integer) $i$th component $x_i$ of the argument $x$ of $p(x)$ as $x_i = a_i^2 + b_i^2 + c_i^2 + d_i^2$, where $a_i, b_i, c_i, d_i$ are in $\mathbb{Z}$.

(iii) The result also holds for polynomial DEs with coefficients over $\mathbb{Q}$: it suffices to write rational coefficients as fractions, find the least common multiple $L$ of the denominators, and consider $Lp(x)$ instead of $p(x)$.

Finally, it might seem strange to consider at a single polynomial equation such as equation (3.5), and claim that it is undecidable. To a theoretical computer scientist, it sounds like saying that a single instance of a problem is undecidable (this would be absurd: decidability applies to instance classes, not single instances). In fact, the variable vector $x$ occurring as an argument of $p(x)$ is partitioned in two (integer) subsequences

$\alpha = (\alpha_1, \ldots, \alpha_\ell)$ and $y = (y_1, \ldots, y_N)$, so that $\alpha$ encodes the parameters of an "instance", and $y$ the solution. The MDRP result in this setting follows because:

(a) every recursively enumerable set $W$ can be encoded by a single polynomial equation $p^W(\alpha, y) = 0$ such that any string $w$ is encoded in $\alpha$, and $w \in W$ if $p(\alpha, y) = 0$ has a solution in $y$;
(b) there exist recursively enumerable sets that are undecidable.

Recall that a subset of $\mathbb{N}$ is *recursively enumerable* if there exists an algorithm for listing all and only its elements (such sets are also called *semi-decidable*). We remark that a set $W \subseteq \mathbb{N}$ is decidable if both $W$ and its complement $\bar{W}$ w.r.t. $\mathbb{N}$ are semi-decidable. To see this, it suffices to list elements of $W$ alternately with elements of $\bar{W}$: the algorithm will terminate in finite time and produce a YES or NO answer about membership of a given $w$ in $W$.

## 3.3. Universality

As remarked in [49], recursively enumerable sets can be enumerated in some order $W_1, W_2, \ldots$ such that the relation $w \in W_v$ for some $v$ is also recursively enumerable. This implies the existence of a DE

$$U(w, v, y) = 0 \tag{3.6}$$

for some parameters $w$ and $v$ such that $U(\alpha, v, y) = 0$ has an integer solution $y$ iff $w \in W_v$. Such polynomials $U$ are known as *universal Diophantine equations* (UDE). This is equivalent to a universal TM (UTM), which takes as input an encoding of any TM $T_v$ and any instance $w$, and simulates $T_v(w)$. In this setting, $T_v$ is a TM that is supposed to determine whether $w \in W_v$.

UDEs present potential applications, as they allow the encoding of any computer program into a single polynomial. Studying its properties might help shed light on the program itself. Jones [49] conducted a thorough study of two complexity measures for UDEs: their degrees $\delta$ and the number $\nu$ of variables occurring in them.

The minimum known value for $\delta$ is 4, which yields $\nu = 58$ [49]. It suffices to take *any* UDE, and repeatedly replace any degree 2 term by a new variable until we obtain a system of DEs $\forall i \leq m\ (t_i(x) = 0)$ where each $t_i(x)$ consists of monomials of degrees 1 and 2. The equation

$$\sum_{i \leq m} (t_i(x))^2 = 0 \tag{3.7}$$

is therefore a UDE with degree $\delta = 4$. In passing, we also conclude that, in general, systems of (many) quadratic DEs (QDE) are also undecidable. Decreasing $\nu$ unfortunately yields large increases on $\delta$: a pair $(\delta, \nu) = (1.638 \times 10^{45}, 9)$ was found by Matiyasevich [49].

The $(4, 58)$ UDE, suitably modified to minimize the number $B$ basic operations (additions and multiplications) required to evaluate the polynomial, yields $B = 100$ [49]. This implies that for any statement $p$ which can be proved within the formal system $S$, $p$ has a proof which only involves 100 integer additions and multiplications.

Coming back to MINLP, since UDEs are subsets of this class, it follows that MINLP inherits their properties. Thus, there exist universal MINLP formulations, though they may be hard to exploit in practice. The dynamics of a universal register machine have been modelled in [56] using an infinite MINLP which becomes finite, and practically solvable, on *bounded* executions (boundedness ensures termination of the underlying TM).

## 3.4. The cause of MINLP undecidability

Considering Sections 3.1 and 3.2, we can ascribe the undecidability of MINLP to the presence of integer variables. This is apparently in contrast with the fact that any integer can be expressed within a theory encoding polynomials and just continuous variables: given any $a \in \mathbb{Z}$, the equation

$$x - a = 0$$

is polynomial and encodes the fact that the continuous variable $x$ should take the integer value $a$. Taking this further, we can write $x \in \{a_1, \ldots, a_k\}$ for any integer values $a_1, \ldots, a_k$ using the polynomial equation

$$(x - a_1) \cdots (x - a_k) = 0.$$

A different approach uses the length $\ell$ of the finite sum $y + y + \cdots + y$ (say it has $\ell$ occurrences of $y$) to express the integer $\ell$ using the polynomial equation $xy = \sum_\ell y$. Aside from the zero solution, any other solution with $y > 0$ yields $x = \ell$.

However, every finite set $F$ is decidable, at least if its elements are all given explicitly: any total order on $F$ provides an enumeration algorithm. Obviously, the complement $\bar{F}$ of $F$ w.r.t. $\mathbb{N}$ is recursively enumerable: just list $\mathbb{N}$ and verify if the current element is in $F$ or not.

To achieve undecidability, we have to look at infinite subsets of $\mathbb{N}$. Is it possible to encode any such set as solutions over $\mathbb{R}$ of some (multivariate) polynomial equation? The answer is no: supposing there exists a real polynomial system with a countably infinite set of (integer) solutions would yield an infinite number of connected components in the corresponding variety, contrary to [66]. Another way of proving this is that, if it were possible to encode $\mathbb{N}$ as the set of solutions of a real polynomial system in any (even nonstandard) way, then the theory of polynomials over $\mathbb{R}$ would be undecidable, contrary to Tarski's algorithm [81]. It is common knowledge that, in order to encode all integers in a system of nonlinear equations with variables ranging over $\mathbb{R}^n$, one needs at least one periodic function, *e.g.*, the set of solutions of $\sin(\pi x) = 0$ is $\mathbb{Z}$. If the polynomials range over $\mathbb{C}^n$, the exponential function (which is periodic over the complex numbers) is also a candidate.

These arguments suggest that a cause of undecidability is the issue of boundedness *vs.* unboundedness of the decision variables. This is well known in polynomial optimization theory. Indeed, with unbounded variables, even the continuous relaxation of an MILP may need to be reformulated using nonlinear functions for practical usefulness [45].

## 3.5. Undecidability results in MP

Although the theories of polynomial equations over the reals and the natural numbers belongs *de re* to MINLP, since PFP is clearly a subclass of MINLP, the results above are traditionally a part of logic and axiomatic set theory. Here, we focus on results which appeared in the MP literature.

As already mentioned above, [56] gives a practically useful proof of the universality of MINLP, and solves some specific instances of the universal MINLP. Moreover, it also proposes an MP formulation to solve an instantiation (in the interval domain) of an abstract interpretation of the execution of arbitrary programs [25], in view of performing static analysis of computer code.

### *3.5.1. Quadratic integer programming*

To the best of our knowledge, the earliest paper investigating MP and (un)decidability is Jeroslow's 1971 paper [48] about the undecidability of quadratic integer programming. Consider the formulation

$$\left.\begin{array}{rrcl} \min & c^\top x & & \\ \forall i \le m & x^\top Q^i x + a_i^\top x & \le & b_i \\ \forall j \le n & x_j & \in & \mathbb{Z}, \end{array}\right\} \qquad (3.8)$$

where $c \in \mathbb{Q}^n$, $Q^i$ are rational $n \times n$ matrices for each $i \le m$, $A = (a_i \mid i \le m)$ is a rational $m \times n$ matrix, and $b \in \mathbb{Q}^m$. Witzgall's algorithm [88] solves equation (3.8) when each quadratic form $x^\top Q^i x + a_i^\top x$ is *parabolic*, *i.e.*, each $Q^i$ is diagonal with non-negative diagonal entries (for $i \le m$). In contrast, Jeroslow observed that this is a limiting case, since if the quadratic forms are diagonal but are allowed to have some negative entries, then they can encode an undecidable set, via the undecidability theory of DEs. The proof is as follows: given an undecidable DE $p(x) = 0$, we consider the following subclass of equation (3.8):

$$\left.\begin{array}{lrcl} \min & x_{n+1} & & \\ & (1-x_{n+1})p(x) & = & 0 \\ & x_{n+1} & \geq & 0 \\ \forall j \leq n & x_j & \in & \mathbb{Z}. \end{array}\right\} \tag{3.9}$$

Obviously, the minimum of equation (3.9) is 0 if $p(x) = 0$ has a solution, and 1 otherwise. Now we follow the argument given above equation (3.7), and iteratively linearize products occurring in the $k$ monomials of $p(x)$ until $p$ can be reformulated to a linear form $y_1 + \cdots + y_k$, subject to a set of *defining constraints* in the form $y_h = \tau_h(x, y)$, where each $\tau_h$ only involves linear occurrences of variables, or their squares, or bilinear products of two variables. This is already undecidable, since finding the optimum of equation (3.9) would solve the undecidable DE $p(x) = 0$, but Jeroslow notes that one can write each bilinear product of decision variables $uv$ (where $u, v$ are decision variable symbols occurring in either $x$ or $y$), by means of an additional variable $w$, as $w = u + v$ while replacing $uv$ by $\frac{1}{2}(w^2 - u^2 - v^2)$. This yields the required form. Jeroslow also notes that whenever the integer variables are bounded, the problem becomes decidable (since there is only a finite number of possible solutions of $p(x) = 0$).

## 3.5.2. *Polynomial minimization in integers*

The paper [89] focuses on undecidable problems in MINLP and global optimization, as detailed in this and the following subsections.

Consider

$$\left.\begin{array}{lrcl} \min & q(x) & & \\ \forall j \leq n & x_j & \in & \mathbb{Z}, \end{array}\right\} \tag{3.10}$$

where $q$ is a polynomial. The proof given in [89] that equation (3.10) is undecidable is as follows. Let $p(x) = 0$ be an undecidable DE. This can be written as $(p(x))^2 \leq 0$. Let $\rho$ be the constant term in $(p(x))^2$ and $q(x) = (p(x))^2 + \rho$. Then we have $q(x) \leq \rho$. We have reduced the DE to the problem of deciding whether there exists an integral solution to the problem $q(x) \leq \rho$. Since this is a special case of the decision problem corresponding to equation (3.10), as the proof in [89] argues, the result follows.

It appears that this proof lacks a key step. If we want to argue that equation (3.10) is undecidable if its decision version $q(x) \leq \rho$ is, then we have to set $\rho = q^*$, the globally optimal value of equation (3.10) (invoking bisection as an equivalence between decision and optimization problems is generally invalid in the current setting, since bisection calls for bounded variables, whereas undecidability requires the variables to be unbounded). Unfortunately, the proof already fixes the value of $\rho$ to the constant term in $(p(x))^2$, which turns out to be the square of the constant term in $p(x)$. Undecidable DEs are often constructed explicitly as UDEs, for which a common technique is the careful modelling of some UTM equivalent computational framework. If we tamper with the constant term, the modelling may become invalid. This proof could be fixed by exhibiting a specific undecidable DE for which the constant term just happens to be the optimal objective function value of $\min_x q(x)$, but this task looks hard to achieve. Unfortunately, all the other results in the paper use this "equivalence" between decision and optimization.

Fortunately, however, the result is true, and has a very simple proof. We define $q(x) = (p(x))^2$ so that $\min_x q(x) = 0$ iff $p(x) = 0$, since $q$ is a square and its minimum value must be $\geq 0$. If we could compute the minimum of $q(x)$, we could decide on the solvability of $p(x) = 0$ according to whether $\min_x q(x) = 0$ or $> 0$. But this is impossible since $p(x) = 0$ is undecidable.

### *3.5.3. Systems of nonlinear equations*

The paper [89] shows that:

$$\forall i \leq m \quad g_i(x) = 0, \tag{3.11}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$ are nonlinear functions, is undecidable. If $p(x) = 0$ is an undecidable DE, then

$$(p(x))^2 + \sum_{j \leq n} (\sin(\pi x_j))^2 = 0, \tag{3.12}$$

where $x$ ranges in $\mathbb{R}^n$, is precisely a restatement of the DE, and so it is undecidable. Note that equation (3.12) is obviously a subclass of equation (3.11).

### *3.5.4. Unconstrained continuous global optimization*

Consider:

$$\min_x q(x), \tag{3.13}$$

for some nonlinear function $q(x)$. It is shown in [89] that equation (3.13) is undecidable, as follows. Let $p(x) = 0$ be an undecidable DE. Let $q(x) = (p(x))^2 + \sum_{j \leq n} (\sin(\pi x_j))^2$, and suppose $q^* = \min_x q(x)$ is computable. Then if $q^* = 0$ we have $p(x) = 0$ and $x_j \in \mathbb{Z}$ for all $j \leq n$, otherwise we do not, which is equivalent to the decidability of $p(x) = 0$, against assumption.

### *3.5.5. Box-constrained continuous global optimization*

Consider

$$\min_{x^L \leq x \leq x^U} q(x), \tag{3.14}$$

for some nonlinear function $q(x)$ and some variable bound vectors $x^L, x^U$. It is shown in [89] that equation (3.14) is undecidable. Let $p(y) = 0$ be an undecidable DE. Let

$$q(x) = (p(\tan x_1, \ldots, \tan x_n))^2 + \sum_{j \leq n} (\sin(\pi \tan x_j))^2,$$

then we can enforce $-\frac{\pi}{2} \leq x_j \leq \frac{\pi}{2}$ for all $j \leq n$, *i.e.*, $x^L = -\frac{\pi}{2}$ and $x^U = \frac{\pi}{2}$ without changing the values of $q(x)$. The argument runs as above, if we could compute the minimum $q^*$ of $q(x)$ over $x^L \leq x \leq x^U$, we could establish whether $p(y) = 0$ has a solution or not, which is impossible.

## **3.6. Summary**

The MINLP class was shown to contain undecidable formulations: by reduction from UDEs [48], by inclusion of UDEs as pure integer polynomial feasibility problems, and by modelling the dynamics of a UTM [56]. By [89] there also exist undecidable non-polynomial NLPs in continuous variables. On the other hand, Tarski's algorithm establishes that semi-algebraic systems of polynomials in continuous variables are decidable.

## 4. Complexity

In this section, we look at the computational complexity of some MINLP problems. We shall mostly focus on nonconvex NLPs, since the presence of integer variables, even bounded to $\{0, 1\}$, makes it very easy to prove MINLP hardness, as shown in Section 4.2.

### 4.1. Tutorial

We briefly recap some basic notions about computational complexity. This section can be skipped by readers who are already familiar with this material.

#### *4.1.1. Turing machines*

The computational model we use in this section is the TM model, an abstract computational process invented by Turing [83]. It consists of an infinitely long tape divided into cells. Each cell can be read, written on, or ignored by a head operating on the tape, which can store the last read character. An engine can shift the tape one cell forward or backward. Each cell contains a character from a finite alphabet $L$ decided *a priori*. $|L|$ must be at least two, but most usually $L$ consists of at least three symbols: 0, 1 and a space used to separate words. A TM obeys a set of instructions such as "move left", "move right", "read", "write", and "stop", as well as a mechanism for recognizing *states*, which provide the equivalent of "if" and "go to". Sequences of instructions are known as *programs*.

#### *4.1.2. Polytime and exponential problems*

Many TM variants have been proposed in the first years of research on theoretical computer science: larger alphabets, half-infinite or multiple tapes to name a few. For most of these variants, simulation proofs were quickly devised to show that none of them was more powerful than the "basic" TM. Driven by the need for faster computation, the issue of computational complexity became more pressing: given an algorithm and some input data, how many steps would a TM need to perform before termination (the issue of whether the TM would actually stop or not was discussed in Sect. 3)? Given that the same algorithm is routinely run on different inputs, the paradigm of *worst-case complexity* was brought to the attention. Infinite sets of input data of the same type and increasing size, together with a specification of the goal the algorithm is supposed to achieve, were grouped in classes named *problems* (conversely, each member of a given problem is an *instance*). If the task consists in answering a YES/NO question, the problem is a *decision problem*.

Given any problem $P$, if there exists an algorithm taking a number of steps bounded by a polynomial function the instance size $n$ in the worst case, we denote its complexity by $O(n^d)$, where $d$ is the degree of the polynomial, and call the algorithm *polynomial-time* (or simply *polytime*). If we are only able to establish an exponential bound, we denote it by $O(2^n)$, and call the algorithm *exponential-time*.

#### *4.1.3. The class* **P**

The focus of computational complexity rapidly shifted from algorithms to problems. Given a decision problem $P$, the main issue of computational complexity is: what is the worst-case complexity of the *best* algorithm for solving $P$? Edmonds [31] and Cobham [22] remarked around 1965 that problems having a polytime complexity are invariant w.r.t. changes in the TM definitions concerning alphabet size (as long as it contains at least 3 characters), number of tapes, and more. This is the reason why today we partition decision problems into two broad categories called "tractable" and "intractable". The tractable problems are those having polytime complexity. The class of such problems is called **P**.

### *4.1.4. The class* **NP**

Concerning those decision problems for which no-one was able to find a polytime algorithm, it was remarked that a somewhat "magic" TM, one that is able to pursue every test branch in parallel (this is called *nondeterministic* TM), gives another useful partition on the class of all decision problems. Those for which there exists a nondeterministic TM that runs in polytime are all grouped in a class called **NP**. It is not difficult to show that an equivalent definition of **NP** is: all problems for which YES instances come with a (deterministically) polytime checkable *certificate*. Given an instance of a problem in **NP**, a nondeterministic TM will explore all test branches in parallel. Since this TM runs in polytime by definition of **NP**, each branching sequence must have size bounded by a polynomial. If the instance is YES, take the branching sequence which yields the YES: this is a deterministically polytime checkable certificate. Conversely, let $P$ be a problem the instances of which all have polytime checkable YES certificates. Then a nondeterministic TM will explore all polynomially sized test branches in parallel, and find the YES certificate in polytime if it exists. If it does not exist, the work will still be carried out in polytime, since each polynomially sized test branch is checked in polytime, and the TM runs in parallel over all such test branches.

CLIQUE, for example, belongs to **NP**: given a graph $G$ and an integer $k$, does $G$ have a clique of size at least $k$? If the instance is YES, then it has a clique of size at least $k$, and this clique can be supplied as certificate to a verification algorithm which is able to establish in polytime whether the certificate is valid.

While it is easy to show that $\mathbf{P} \subseteq \mathbf{NP}$, no-one was ever able to establish whether **P** is different or equal to **NP** (this is the famous **P** *vs.* **NP** question).

### *4.1.5. Reductions*

The class **NP** is very useful because, in absence of a complement of **P** allowing us to tell problems apart into "easy" and "hard", it provides a good proxy.

Given two problems $P$ and $Q$ in **NP**, suppose we know a polytime algorithm $A$ to solve $P$ but we know of no such algorithm for $Q$. If we can find a polynomial algorithm $R$ which transforms a YES instance of $Q$ into a YES instance of $P$ and, conversely, a NO instance of $Q$ into a NO instance of $P$, then we can compose $R$ and $A$ to obtain a polytime algorithm for $Q$. An algorithm $R$ mapping $Q \to P$ while keeping instance membership in YES and NO classes invariant is called a *polynomial reduction* from $Q$ to $P$.

Technically, this reduction is more precisely known as a *Karp reduction*. A *Cook reduction* occurs from $Q$ to $P$ if $Q$ can be solved by a polynomial number of standard operations and calls to an oracle that solves $P$. A Karp reduction is like a Cook reduction allowing for a single oracle call. Karp reductions allow for a distinction between hard problems where polynomial certificates are available for YES instances and for NO ones — in other words, it allows for the definition of the class co-**NP**.

### *4.1.6. The hardest problem in the class*

This gives an interesting way to define the "hardest problem in **NP**". Let $P$ be a problem in **NP** such that any problem in **NP** polynomially reduces to $P$. Pick any $Q \in \mathbf{NP}$: can $Q$ be harder to solve than $P$? Since there is a polynomial reduction $R : Q \to P$, if we know how to solve $P$ then we know how to solve $Q$, so there is no real additional difficulty in solving $Q$ other than knowing how to solve $P$. In this sense, $P$ is hardest in the class **NP**. The set of such problems are said to be **NP**-*complete*. Since this notion of hardness only depends on polynomial reduction rather than membership in **NP**, we also define as **NP**-*hard* those problems which are as hard as any problem in **NP** without necessarily being members of **NP**. Thus, **NP**-complete problems are a subset of **NP**-hard problems.

The landmark result [24] proves that the class of **NP**-complete problems is non-empty, since it contains SAT. The SAT problem involves a decision about the truth or falsity of sentences defined over variable symbols, the $\neg$ unary operator, the $\wedge$ and $\vee$ binary operators, and the constants 0 and 1 (traditionally, in SAT notation, one writes $\bar{x}$ instead of $\neg x$). A sentence is YES if there is an assignment of 0 and 1 to the variables that evaluates to YES, and NO otherwise. The evaluation uses the common meaning of the boolean operators $\neg, \wedge, \vee$. Cook's result is based on the idea that SAT can be used as a declarative language to model the dynamics of *any* polytime TM, which can be described by a sequence of 0–1 vectors, each component of which represents the value of a different SAT variable for each time step. Since the definition of **NP** involves a polytime-checkable certificate, SAT can be used to model the polytime TM used to verify the certificate. The SAT instance modelling the given TM has finite length because we know that the number of steps of the TM is bounded by a polynomial in the input length.

### *4.1.7. The reduction digraph*

After Cook's result, SAT became known as the "primitive problem" in the theory of **NP**-hardness. While the *first* proof of **NP**-hardness was difficult to devise (since it necessarily has to encode *every* TM, following the definition), the following proofs are of an altogether different nature, as they can rely on the mechanism of reduction. Suppose we know that $P$ is an **NP**-hard problem, and we want to establish the **NP**-hardness of $Q$. Can $Q$ be any easier than $P$? Suppose we find a polynomial reduction $R : P \to Q$: if $Q$ could be solved by an algorithm $A$ that is asymptotically faster than that of $P$, then we could compose $R$ with $A$ to derive a better algorithm for $P$, which is impossible since $P$ was defined to be hardest for **NP**. This idea is used in [51] to prove that many common problems are **NP**-complete. Since then, it has become commonplace in most publications in theoretical computer science that, when presenting a new problem, there should be a proof that either it is in **P** or it is **NP**-hard. Interestingly, some well-known problems (such as GRAPH ISOMORPHISM or SDP FEASIBILITY [55]) are not (yet) known to be either in **P** or **NP**-hard.

Note that polytime reductions define an asymmetric relation on the class **NP**, *i.e.*, two problems $P, Q$ in **NP** are related if there is a reduction $R : P \to Q$. This turns **NP** in an (infinite) directed graph, called the *reduction digraph*, where the problems are represented by nodes, and reductions by arcs. This graph is strongly connected by virtue of Cook's result, which proves that there is a reduction from any problem in **NP** to SAT.

### *4.1.8. Decision vs. optimization*

A problem $P$ is **NP**-complete if it is **NP**-hard and belongs to **NP**. Most theoretical results in computational complexity concern decision problems, but in practice we also need to solve function evaluation and optimization problems. Transforming evaluation and optimization into decision problems does not usually change their computational difficulty.

The "decision version" of the MINLP in equation (1.1) adds a *threshold value* $\phi$ to the input and asks whether the system

$$\left.\begin{array}{rrcl} & f(x) & \le & \phi \\ & g(x) & \in & [g^L, g^U] \\ & x & \in & [x^L, x^U] \\ \forall j \in Z & x_j & \in & \mathbb{Z} \end{array}\right\} \tag{4.1}$$

is feasible. The class of optimization problems which can be reduced to their decision version equation (4.1) is called **NPO** (the "O" stands for "optimization"). It has the following properties: (i) the feasibility of any solution can be established in polytime; (ii) every feasible solution has polynomially bounded size; and (iii) the objective function and constraints can be evaluated in polytime at any feasible solution.

### *4.1.9. When the input is numeric*

For problems involving arrays of rational numbers in the input data, such as vectors or matrices, more notions are used to distinguish complexity in function of the numerical value of the input *vs.* the number of bits of memory required to store it.

An algorithm is *pseudopolynomial* if it is polynomial in the value of the numbers in the input, but not in their size. For example, testing whether a number $n$ is prime by trying to divide it by $2, \ldots, \lfloor\sqrt{n}\rfloor$ takes $\sqrt{n}$

divisions, but if we take the classic TM computational model, the divisions involved take exponential time in the size of the input, which is $\lceil \log_2(n) \rceil$.

In the arithmetic computational model, where elementary operations on numbers take unit time, an algorithm is *strongly polytime* if its worst-case running time is bounded above by a polynomial in function of the *length* of the input arrays, and the worst-case amount of memory required to run it (a.k.a. the *worst-case space*) is bounded by a polynomial in the number of bits required to store their values, a.k.a. the size of the input (*e.g.*, Dijkstra's shortest-path algorithm on weighted graphs is strongly polytime). On the other hand, if the running time also depends on the actual input size, including the bits required to represent the numbers in the arrays, the algorithm is *weakly polytime* (*e.g.*, the ellipsoid method for LP is weakly polytime: whether there exists a strongly polytime algorithm for LP is a major open question).

A problem $P$ is *strongly* **NP**-hard if there is a strongly polytime reduction algorithm $R$ from every problem in **NP** to $P$, or, equivalently, from a single **NP** hard problem to $P$. $P$ is *strongly* **NP**-complete if it is strongly **NP**-hard and also belongs to the class **NP**. An equivalent definition of strong **NP**-hardness is the class of those **NP**-hard problems that remain **NP**-hard even if the input is encoded in unary, *i.e.*, when the value of all of their numerical data is bounded by a polynomial in the input size.

A problem $P$ is *weakly* **NP**-hard when it is **NP**-hard but there exists a pseudopolynomial algorithm that solves it (*e.g.*, often dynamic programming based algorithms search the set of values which a given variable can take, so that the number of iterations might remain polynomial in the value rather than the number of bits required to store it).

When proving **NP**-hardness for a new problem $P$, using reductions from a strongly **NP**-hard problem gives more flexibility, as one can use a pseudopolynomial reduction and still claim **NP**-hardness of $P$. This technique was used in [15].

Lastly, a word of caution about these notions. An algorithm is either strongly or weakly polytime, in the sense that the two definitions are complementary. In the case of problems, strong and weak **NP**-hardness are not complementary in the same sense, as the weak version rests on pseudopolynomial reduction algorithms rather than weakly polytime ones.

## 4.2. Complexity of solving general MINLP

MINLP is not in **NP** because it is not a decision problem. It is **NP**-hard because it includes MILP as a subclass. MILP is **NP**-hard by means of a trivial reduction from SAT. Transform the SAT instance to conjunctive normal form (CNF), write $\bar{x}$ as $1 - x$, $\vee$ as $+$ and let $\wedge$ mark the separation between constraints, which are all set to be $\geq 1$. This yields a set of MILP constraints that are feasible if and only if the SAT instance is YES. We denote these constraints by "SAT $\rightarrow$ MILP" for future reference.

This argument, however, only brushes the surface of the issue, as it is essentially an argument about MILP. We know by Section 3 that nonlinearity makes unbounded MINLP undecidable, whereas finite bounds make it decidable. But what about the continuous variables? We know that PP without integer variables is decidable, but is it **NP**-hard? The answer is yes, and it follows from the fact that polynomials can express a bounded number of integer variables (see the beginning of Sect. 3.4). All that the SAT $\rightarrow$ MILP reduction above needs is a sufficient number of binary (boolean) variables, which can be defined by requiring a continuous variable $x$ to satisfy the polynomial $x(1-x) = 0$.

In the rest of this section, we shall survey the field of computational complexity of several different MINLP problems, with a particular attention to NLPs involving continuous variables.

## 4.3. Quadratic programming

The general Quadratic Programming (QP) problem is as follows:

$$\left.\begin{array}{rrcl} \min & \frac{1}{2}x^\top Q x & + & c^\top x \\ & Ax & \geq & b. \end{array}\right\} \tag{4.2}$$

### *4.3.1.* **NP***-hardness*

QP was shown in [77] to contain an **NP**-hard subclass of instances (and hence QP itself is **NP**-hard by inclusion). The reduction is from an **NP**-complete problem called SUBSET-SUM: given a list $s$ of non-negative integers $s_1, \ldots, s_n$ and an integer $\sigma$, is there an index set $J \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in J} s_j = \sigma$? Consider the QP formulation:

$$\left.\begin{array}{lrcl} \max & f(x) = \sum\limits_{j \le n} x_j(x_j - 1) & + & \sum\limits_{j \le n} s_j x_j \\ & \sum\limits_{j \le n} s_j x_j & \le & \sigma \\ \forall j \le n & 0 \le x_j & \le & 1. \end{array}\right\} \tag{4.3}$$

This defines a transformation from an instance of SUBSET-SUM to one of QP. Obviously, it can be carried out in polytime. We now prove that it maps YES instances of SUBSET-SUM to QP instances where $f(x) < \sigma$ and NO instances to instances of QP where $f(x) = \sigma$.

Assume $(s, \sigma)$ is a YES instance of SUBSET-SUM with solution $J$. Then setting $x_j = 1$ for all $j \in J$ satisfies all constraints of equation (4.3): in particular, the constraint is satisfied at equality. Solution integrality makes the first sum in the objective function yield value zero, which yields $f(x) = \sigma$. Conversely, assume $(s, \sigma)$ is a NO instance, and suppose first that equation (4.3) has an integer solution: then the constraint must yield $\sum_j s_j x_j < \sigma$; solution integrality again zeroes the first sum in the objective function, so $f(x) < \sigma$. Next, if equation (4.3) has no integer solution, there is at least one $j \le n$ such that $0 < x_j < 1$, which makes the objective function term $x_j(x_j - 1)$ negative, yielding again $f(x) < \sigma$, as claimed.

The paper [77] also contains another proof that shows that an NLP with just one nonlinear constraint of the form $\sum_j x_j(x_j - 1) \ge 0$ is also **NP**-hard.

### *4.3.2. Strong* **NP***-hardness*

A different proof, reducing from SAT, was given in Theorem 2.5 from [85]. It shows that the following QP subclass is **NP**-hard:

$$\left.\begin{array}{ll} \min & f(x) = \sum\limits_{j \le n} x_j(1 - x_j) \\ & \textsc{sat} \to \text{MILP} \\ \forall j \le n & 0 \le x_j \le 1, \end{array}\right\} \tag{4.4}$$

where SAT $\to$ MILP indicates the MILP constraints encoding a SAT instances which were discussed at the beginning of Section 4.2. The proof shows that the given SAT instance is YES iff $f(x) = 0$. Assume first that the SAT instance is YES: then there is an assignment of boolean values to the SAT variables that satisfies the SAT $\to$ MILP constraints. Moreover, since the solution is integral, we get $f(x) = 0$. Conversely, assume the SAT instance is NO, and suppose, to aim at a contradiction, that a solution $x^*$ to equation (4.4) exists, and is such that $f(x^*) = 0$. Since the quadratic form $\sum_j x_j(1 - x_j)$ is zero if each $x_j \in \{0, 1\}$, we must conclude that $x^*$ is integral. Since the SAT $\to$ MILP constraints are feasible if the SAT instance is YES, and $x^*$ satisfies those constraints, it follows that the SAT instance is YES, against assumption. Hence, if the SAT instance is NO, either equation (4.4) is infeasible or $f(x^*) > 0$.

These two arguments prove the same fact, *i.e.*, that QP is **NP**-hard. However, the first reduction is from SUBSET-SUM, while the second is from SAT. This has some consequences, since while SAT is strongly **NP**-hard, SUBSET-SUM is not: namely, the first reduction only proves the weak **NP**-hardness of QP, leaving the possibility of a pseudopolynomial algorithm open. The second reduction rules out this possibility.

### *4.3.3.* **NP***-completeness*

While QP cannot be in **NP** since it is not a decision problem, the decision problem corresponding to QP is a candidate. The question is whether the following decision problem is in **NP**:

$$\left.\begin{array}{rcl} \frac{1}{2}x^\top Qx + c^\top x & \leq & \phi \\ Ax & \geq & b, \end{array}\right\} \tag{4.5}$$

where $\phi$ is a threshold value that is part of the input.

As shown in [85], equation (4.5) turns out to be in **NP**. The proof is long and technical, and consists of many intermediate results that are very important by themselves.

1. If the QP of equation (4.2) is convex, *i.e.*, $Q$ is positive semidefinite (PSD), and has a global optimum, then it has a globally optimal solution vector where all the components are rational numbers — this was shown in [75]. The main idea of the proof is as follows: only consider the active constraints in $Ax \geq b$, then consider the KKT conditions, which consist of linear equations in $x$, and derive a global optimum of the convex QP as a solution of a set of linear equations.
2. Again by [75], there is a polytime algorithm for solving convex QPs (cQP): in other words, cQP is in **P**. This is a landmark result by itself — we note that the algorithm is an interior point method (IPM) and that it is weakly polytime. Specifically, though, this result proves that the size of the rational solution is bounded by a polynomial in the input size. By [85], in page 77, this is enough to prove the existence of a polynomially sized certificate in case equation (4.5) is bounded.
3. The unbounded case is settled in [84].

This allows us to conclude that the decision version of QP is **NP**-complete.

### *4.3.4. Box constraints*

A QP is *box-constrained* if the constraints $Ax \geq b$ in equation (4.2) consist of variable bounds $x^L \leq x \leq x^U$. The hyper-rectangle defined by $[x^L, x^U]$ is also known as a "box". As for the **NP**-hardness proof of QP, there are two reductions from **NP**-complete problems to box-constrained QP: one from SUBSET-SUM ([68], Eq. (3)) and one from SAT ([85], Sect. 4.2). Since reductions from SAT imply strong **NP**-hardness, we only focus on the latter.

We actually reduce from 3SAT in CNF, consisting of a conjunction of $m$ clauses each of which is a disjunction of exactly 3 literals. As before, we write the $i$th literal as $x_i$ or $1 - x_i$ if negated. A disjunction $x_i \vee \bar{x}_j \vee x_k$, for example, is written as a linear constraint $x_h + (1 - x_j) + x_k \geq 1$, yielding

$$\forall i \leq m \quad a_i^\top x \geq b_i \tag{4.6}$$

for appropriate vectors $a_i \in \mathbb{Z}^m$ and $b \in \mathbb{Z}$. By adding a slack variable $v_\ell \in [0,2]$ to each of the $m$ clauses, equation (4.6) becomes $\forall i \leq m \ (a_i^\top x = b_i + v_i)$. Next, we formulate the following box-constrained QP:

$$\left.\begin{array}{rl} \min & f(x,v) = \sum\limits_{j \leq n} x_j(1 - x_j) + \sum\limits_{i \leq m} (a_i^\top x - b_i - v_i)^2 \\ & x \in [0,1]^n \quad v \in [0,2]^m. \end{array}\right\} \tag{4.7}$$

We are going to show that the 3SAT instance is YES iff the globally optimal objective function value of equation (4.7) is zero.

Obviously, if the 3SAT formula is satisfiable, there exists a feasible solution $(x^*, v^*)$ where $x^* \in \{0,1\}^n$ and $v^* \in \{0,1,2\}^m$ that yields a zero objective function value. Conversely, suppose there exists $(x^*, v^*)$ such that

$f(x^*, v^*) = 0$, which yields

$$a_i^\top x^* = b_i + v_i^* \tag{4.8}$$

for all $i \le m$. Supposing some of the $x$ variables take a fractional value, the corresponding term $x_j^*(1 - x_j^*)$ would be nonzero, against the assumption $f(x^*, v^*) = 0$. From integrality of the $x$ variables, equation (4.8) ensures that $v_i^*$ is integer, for all $i \le m$. Since integrality and feasibility w.r.t. equation (4.8) are equivalent to equation (4.6) and therefore encode the clauses of the 3SAT instance, $x^*$ is a YES certificate for the 3SAT instance. Thus, finding the global optimum of the box-constrained QP in equation (4.7) is **NP**-hard.

### *4.3.5. Trust region subproblems*

Trust Region Subproblems (TRS) take their name from the well-known trust region method for black-box optimization. TRSs are essentially instances of QP modified by adding a single (convex) norm constraint $\|x\| \le r$, where $r$ (which is part of the input) is the *radius* of the trust region. If the norm is $\ell_2$ and $Ax \ge b$ has a special structure, then the problem can be solved in polytime [9]. In this case, the solution is generally irrational (due to the nonlinear norm constraint — note that because of that constraint, the TRS is not formally a subclass of QP in general), though for the simple case of $\|x\|_2 \le 1$ and no linear constraints, the technical report [87] states that the decision problem is actually in **P**.

Most TRSs arising in practical black-box optimization problems, however, are formulated with an $\ell_\infty$ norm constraint. This makes the resulting QP easier to solve numerically, to a given $\varepsilon > 0$ approximation tolerance, using local NLP solvers. This is because the $\ell_\infty$ norm yields simple box constraints on the decision variables, and therefore the $\ell_\infty$ TRS does belong to the class QP, specifically it is a box-constrained QP. From a worst-case computational complexity point of view, however, we recall that box-constrained QPs form an **NP**-hard subclass of QP, as mentioned above.

### *4.3.6. Continuous quadratic knapsack*

The reduction is from SUBSET-SUM. Consider the following formulation, called *continuous Quadratic Knapsack Problem* (cQKP):

$$\left.\begin{array}{rrcl} \min & x^\top Q x & + & c^\top x \\ & \sum\limits_{j \le n} a_j x_j & = & \gamma \\ & x & \in & [0,1]^n, \end{array}\right\} \tag{4.9}$$

where $Q$ is a square diagonal matrix, $a, c \in \mathbb{Q}^n$, and $\gamma \in \mathbb{Q}$.

We encode a SUBSET-SUM instance $(\{s_1, \ldots, s_n\}), \sigma)$ in equation (4.9) as follows ([85], Sect. 4.2):

$$\left.\begin{array}{rl} \min & f(x) = \sum\limits_{j \le n} x_j(1 - x_j) \\ & \sum\limits_{j \le n} s_j x_j = \sigma \\ & x \in [0,1]^n, \end{array}\right\} \tag{4.10}$$

We are going to show that the instance is YES iff the global optimum of equation (4.10) is zero. If the instance is YES, then there is a subset $J \subset \{1, \ldots, n\}$ such that $\sum\limits_{j \in J} s_j = \sigma$. Let $x_j^* = 1$ for all $j \in J$ and $x_j^* = 0$ for all $j \notin J$: then $x^*$ is feasible in the constraints of equation (4.9), and yields $f(x^*) = 0$. Conversely, if $x^*$ is a feasible solution of equation (4.9) yielding $f(x^*) = 0$, then each term $x_j^*(1 - x_j^*)$ in $f(x)$ has value zero, which implies $x^* \in \{0,1\}^n$. Now let $J = \{j \le n \mid x_j^* = 1\}$ be the support of $x^*$. By construction, $J$ is a YES certificate for the SUBSET-SUM instance. Thus, finding a global optimum of a cQKP is **NP**-hard.

### *4.3.7. Convex QKP*

Since a convex QKP is a subclass of convex QP, it can be solved in polytime — but no strongly polytime algorithm is known. On the other hand, if $Q$ is a diagonal matrix, then the objective function of equation (4.9) is separable. As remarked in Section 3.1 of [85], there is an $O(n \log n)$ algorithm for solving this specific variant of convex QKP [43].

### *4.3.8. The Motzkin–Straus formulation*

In 1965, Motzkin and Straus established an interesting relationship between the maximum clique $C$ in a graph $G = (V, E)$ and the following QP [67]:

$$\left.\begin{array}{rrcl} \max & f(x) & = & \sum\limits_{\{i,j\}\in E} x_i x_j \\ & \sum\limits_{j\in V} x_j & = & 1 \\ \forall j \in V & x_j & \geq & 0\ ; \end{array}\right\} \tag{4.11}$$

namely, that there exists a global optimum $x^*$ of equation (4.11) such that

$$f^* = f(x^*) = \frac{1}{2}\left(1 - \frac{1}{\omega(G)}\right),$$

where $\omega(G)$ is the size of a maximum cardinality clique in $G$. In other words, this gives the following formula for computing the *clique number* of a graph:

$$\omega(G) = \frac{1}{1 - 2f^*}.$$

Moreover, a maximum clique is encoded in a global optimum $x^*$ of equation (4.11), which has the form

$$\forall j \in V \quad x_j^* = \begin{cases} \frac{1}{\omega(G)} & \text{if } j \in C \\ 0 & \text{otherwise.} \end{cases}$$

Equation (4.11) is called the *Motzkin–Straus formulation* for the MAX CLIQUE optimization problem. Its decision version CLIQUE is well known to be **NP**-complete, which makes the Motzkin–Straus formulation an appropriate candidate for reductions from CLIQUE to various problems related to QP and NLP.

We follow the proof given in [2]. Let $x^*$ be a global optimum of equation (4.11) with as many components at zero as possible. Consider $C = \{j \in V \mid x_j^* > 0\}$. We claim that $C$ is a maximum clique in $G$.

- First, we claim $C$ is a clique. Suppose this is false to aim at a contradiction, and assume without loss of generality that $1, 2 \in C$ and $\{1, 2\} \notin E$. For some $\epsilon$ in the interval $[-x_1^*, x_2^*]$, let $x^\epsilon = (x_1^* + \epsilon, x_2^* - \epsilon, x_3^*, \ldots, x_n^*)$. Note that $x^\epsilon$ satisfies the simplex constraint $\sum_j x_j = 1$, as well as the non-negativity constraints. Since the edge $\{1, 2\}$ is not in $E$, the product $x_1 x_2$ does not appear in the objective function $f(x)$. This means that $\epsilon^2$ never occurs in $f(x^\epsilon)$. In particular, $f(x)$ is linear in $\epsilon$. We temporarily look at $f$ as a function of $\epsilon$, parametrized by $x^*$. By the choice of $x^*$ (a global optimum), $f(\epsilon)$ achieves its maximum at $\epsilon = 0$. Since $\epsilon = 0$ is in the interior of its range $[-x_1^*, x_2^*]$, $f(\epsilon)$ is constant over this range. Hence setting $\epsilon = -x_1^*$ and $\epsilon = x_2^*$ yields global optima with a smaller number of nonzero components $x^*$, which is a contradiction as $x^*$ was chosen with the maximum number of zero components as possible. Thus $C$ is a clique.

- Now, we claim $C$ has maximum cardinality $|C| = \omega(G)$. Consider the simplex constraint $\sum_j x_j = 1$ and square both sides:

$$1 = \left(\sum_{j\in V} x_j\right)^2 = 2\sum_{i<j\in V} x_i x_j + \sum_{j\in V} x_j^2. \tag{4.12}$$

  Since by construction of $C$ we have $x_j^* = 0$ if $j \notin C$, the above reduces to

$$\psi(x^*) = 2\sum_{i<j\in C} x_i^* x_j^* + \sum_{j\in C}(x_j^*)^2.$$

  Moreover, $\psi(x) = 1$ for all feasible $x$ by equation (4.12). So, the objective function $f(x) = \sum_{i,j} x_i x_j$ achieves its is maximum when the second term of $\psi(x)$ is minimum, *i.e.*, when $\sum_{j\in C}(x_j^*)$ attains its minimum, which occurs at $x_j^* = \frac{1}{|C|}$. Again by the simplex constraint, we have

$$f(x^*) = 1 - \sum_{j\in C}(x_j^*)^2 = 1 - |C|\frac{1}{|C|^2} = 1 - \frac{1}{|C|} \le 1 - \frac{1}{\omega(G)},$$

  with equality when $|C| = \omega(G)$, as claimed.

By reduction from CLIQUE, it follows therefore that equation (4.11) describes an **NP**-hard subclass of QP. Using the same basic ideas, Vavasis gives a proof by induction on $|V|$ in Lemma 4.1 from [85].

An extension of this work shows that there exists a bijection between the local optima of equation (4.11) and the maximal cliques of $G$ [11].

### *4.3.9. QP on a simplex*

"QP on a simplex", refers to the following formulation:

$$\left.\begin{array}{rrcl} \min & x^\top Q x & + & c^\top x \\ & \sum\limits_{j\le n} x_j & = & 1 \\ \forall j \le n & x_j & \ge & 0, \end{array}\right\} \tag{4.13}$$

where $Q$ is a square $n \times n$ rational matrix and $c \in \mathbb{Q}^n$. Since equation (4.11) describes a subclass of equation (4.13), the latter is **NP**-hard by inclusion.

### *4.3.10. QP with one negative eigenvalue*

So far, we established that convex QPs are in **P**, and that a variety of nonconvex QPs are **NP**-hard (with their decision version being **NP**-complete). Where does efficiency end and hardness begin? In an attempt to provide an answer to this question, Pardalos and Vavasis proved in [71] that an objective function $\min x^\top Q x$ where $Q$ has rank one and has a single negative eigenvalue suffices to make the problem **NP**-hard. The problem of solving a QP with one negative eigenvalue is denoted by QP1NE.

The construction is very ingenious. It reduces CLIQUE to QP1NE in two stages. First, it provides a QP formulation having zero globally optimal objective function value attained if the main decision variable vector $x$ takes optimal values in $\{0, 1\}$. Second, it encodes a clique of given cardinality $k$ in a given graph $G = (V, E)$ by means of the following constraints:

$$\forall \{i, j\} \notin E \quad x_i + x_j \le 1 \tag{4.14}$$

$$\sum_{j \in V} x_j = k \tag{4.15}$$

$$0 \leq x \leq 1. \tag{4.16}$$

The rest of the formulation, which essentially ensures integrality of the decision variables, is as follows:

$$\min \quad z - w^2 \tag{4.17}$$

$$w = \sum_{j \in V} 4^j x_j \tag{4.18}$$

$$z = \sum_{j \in V} 4^{2j} x_j + 2 \sum_{i<j} 4^{i+j} y_{ij} \tag{4.19}$$

$$\forall i < j \in V \quad y_{ij} \geq \max(0, x_i + x_j - 1). \tag{4.20}$$

Equation (4.17) clearly is of rank one and has a single nonzero eigenvalue which is negative. Note that the definitions of $w^2$ (by means of Eq. (4.18)) and $z$ in equation (4.19) almost match: we should have $y_{ij} = x_i x_j$ for them to be equal. Equations (4.16), (4.20), and (4.19) ensure that $z$ cannot be negative, which also holds for $w^2$ since it is a square. A couple of technical lemmata ensure that the QP in equations (4.17)–(4.20) has optimal value zero if the optimal solution is binary. Integrating the constraints equations (4.14)–(4.16) encodes CLIQUE in this QP so that $G$ has a clique of cardinality $k$ if the optimal objective function value is zero, as claimed.

### *4.3.11. Bilinear programming*

The BILINEAR PROGRAMMING PROBLEM (BPP) reads as follows:

$$\left.\begin{array}{ll} \min & \sum_{j \leq n} x_j y_j \\ & Ax + By \geq b. \end{array}\right\} \tag{4.21}$$

The **NP**-hardness of BPP has been established in [7], by a reduction from a problem in computational geometry called 2-LINEAR SEPARABILITY (2LS), consisting in determining whether two sets of points in a Euclidean space can be separated by a piecewise linear curve consisting of two pieces. In turn, 2LS was proven **NP**-complete in [63].

Bennett and Mangasarian show in [7] that the 2LS reduces to one of three possible BPP formulations, all of which with general inequality constraints and non-negativity constraints on the decision variables.

### *4.3.12. An open question by Vavasis, settled by Matsui*

In page 37 of [86], Vavasis proposes as an open question whether the following QP:

$$\left.\begin{array}{ll} \min & f(x) = (c^\top x + \gamma)(d^\top x + \delta) \\ & Ax \geq b \end{array}\right\} \tag{4.22}$$

is **NP**-hard. Note that although equation (4.22) is a subclass of QP, which is itself an **NP**-hard problem, there is the possibility that this might be a tractable case. Equation (4.22), however, was shown to be **NP**-hard in [62].

The proof is constructed very ingeniously by borrowing the functional form $\min x_1 - x_2^2$ from QP1NE. First, a reduction of the (**NP**-complete) SET PARTITION problem to the formulation

$$\left.\begin{array}{rl}
\min & \sum_{i=1}^{n}\sum_{j=1}^{n} p^{i+j} y_{ij} - \left(\sum_{i=1}^{n} p^i x_i\right)^2 \\
 & Sx = 1 \\
\forall i \neq j \leq n & y_{ij} \leq x_i \\
\forall i \neq j \leq n & y_{ij} \leq x_j \\
\forall i \neq j \leq n & y_{ij} \geq x_i + x_j - 1 \\
\forall i \leq n & y_{ii} = x_i \\
 & x \in [0,1]^n \\
 & y \in [0,1]^{n^2},
\end{array}\right\} \tag{4.23}$$

where $p$ is any positive integer and $Sx = 1$ is a set of constraints modelling SET PARTITION. This is achieved by showing that the objective function is positive if and only if there are fractional feasible solutions $x'$ with $0 < x'_i < 1$ for some $i \leq n$. The crucial point of the proof is that the linearization constraints $x_i + x_j + 1 \leq y_{ij} \leq \min(x_i, x_j)$ and $y_{ii} = x_i$ are exact only for $x \in \{0, 1\}$. A technical (but easy to follow) argument shows that the objective function is positive at the optimum if and only if some fractional solution exists.

This proof is then extended to the generalization of equation (4.23) where the objective function is replaced by a general function $g(x_0, y_0)$ where $x_0 = \sum_i p^i x_i$, $y_0 = \sum_{i,j} p^{i+j} y_{ij}$ and two more conditions on $x_0, y_0$ designed to endow $g$ with positivity/negativity properties similar to that of the objective function of equation (4.23). This new problem also has non-positive objective function value at the optimum if and only if the optimum is binary. This provides the reduction mechanism from SET PARTITION (encoded in the constraints $Sx = 1$).

Finally, the function $g(x_0, y_0) = (y_0 - p + 2p^{4n})^2 - 4p^{4n}x_0^2 - 4p^{8n}$ is shown to satisfy the requirements on $g$ and also to factor as a product of two linear forms plus a constant, which provides the necessary form shown in equation (4.22).

The same proof mechanism can also prove **NP**-hardness of the related problems

$$\min \{x_1 x_2 \mid Ax \geq b\} \tag{4.24}$$

$$\min \left\{x_1 - \frac{1}{x_2} \mid Ax \geq b\right\} \tag{4.25}$$

$$\min \left\{\frac{1}{x_1} - \frac{1}{x_2} \mid Ax \geq b\right\}. \tag{4.26}$$

## *4.3.13. Establishing local minimality*

The problem of deciding whether a given $x$ is a local minimum of a QP or not has attracted quite a lot of attention in the late 1980s and early 1990s. There are at least three distinct proofs of this fact in the literature: in Problem 1 from [68], in Section 3 from [70], and in Section 5.1 from [85]. The first [68] reduces SUBSET-SUM to the problem of deciding whether a quadratic form over a translated simplex is negative, shown to be equivalent to unboundedness of a constrained quadratic form, and to the problem of deciding whether a given point is not a local optimum of a given QP. Moreover, [68] also shows that deciding copositivity of a given matrix is co-**NP**-hard; it further proves co-**NP**-hardness of verifying that a point is not a local optimum in unconstrained minimization (of a polynomial of degree 4). The second [70] presents a QP where a certain point is not a local minimum iff a given 3SAT instance is YES. The third [85] is based on an unconstrained variant of the Motzkin–Straus formulation, where the zero vector is a local minimum iff a given CLIQUE instance is NO.

These proofs all show that it is as hard to prove that a given point is *not* a local minimum of a QP as to show that a given **NP**-hard problem instance is YES. Implicitly, using the reduction transformation, this provides certificates for the NO instances of the QP LOCAL MINIMALITY problem (QPLOC) rather than for the YES instances. In computational complexity, this shows that QPLOC is co-**NP**-hard (see Sect. 4.1.5) rather than **NP**-hard.

Currently, no-one knows whether **NP** = co-**NP**. Suppose we could prove that **NP** $\neq$ co-**NP** (†). We know that **P** = co-**P** (the whole polynomially long trace of the algorithm provides a polynomially long certificate of both YES and NO instances). If **P** = **NP** then it would follow from **P** = co-**P** that **NP** = co-**NP**, a contradiction with (†), which would prove **P** $\neq$ **NP**, the most famous open question in computer science and one of the most famous in mathematics.

It is interesting that many **NP**-hard problems have QP formulations, which proves that solving QPs is **NP**-hard. However, verifying whether a given point is a local optimum is co-**NP**-hard. Looking in more depth, most of the **NP**-hardness reductions about QP actually reduce to deciding whether a given QP has optimal objective function value equal to zero or not. On the other hand, the co-**NP**-hardness reductions in [68, 70, 85] all establish local optimality of a given solution vector. The former setting (testing the optimal objective function value) sometimes allows for clear dichotomies: for example, for the box-constrained QP in equation (8) from [68], it is shown that there is a finite gap between the optimal value being zero or less than a finite negative value (this implies that the corresponding decision problem is actually in **NP**). In practice, a sufficiently close approximation to zero can be rounded to zero exactly, which means that a certificate for "equal to zero" can be exhibited. Moreover, a YES instance of a problem might have multiple certificates, each of which is mapped (through the reduction) to a different solution of the corresponding QP having the same objective function value. The latter setting (testing local optimality) is about a specific point. Since QP involves continuous functions of continuously varying variables, it is harder to find cases where gap separations are possible. Encoding a YES instance (possibly with multiple certificates) in the verification of optimality of a single point, moreover, intuitively seems to be a harder task.

## 4.4. General nonlinear programming

The general NLP formulation is like equation (1.1) where $Z = \varnothing$. It obviously contains QP, so NLP is **NP**-hard in general. Murty and Kabadi's proof of co-**NP**-hardness of verifying whether a given point is a local minimum (see previous section) uses the following formulation

$$\min_u \left(u^2\right)^\top Q u^2,$$

where $u^2$ denotes the vector $(u_1^2, \ldots, u_n^2)$. This is a minimization of a quartic polynomial, and so it is not, strictly speaking, quadratic (although it is readily seen to be equivalent to the quadratic problem $\min\limits_{x \geq 0} x^\top Q x$).

In Section 3.5, we discussed *undecidable* NLP problems, so the hardness issue is somewhat moot. It is worth mentioning two hardness results in NLP: deciding whether a given function is convex, and optimization over the copositive cone. These two problems are related by the notion of convexity – which, according to Rockafellar, is the real barrier between easy and difficult optimization problems. Given any function, can we efficiently decide whether it is convex? If so, then we may hope to optimize it efficiently using a special-purpose algorithm for convex functions. Rockafellar's belief was doubly shattered by relatively recent results. On the one hand, the **NP**-hardness of deciding convexity of polynomials of 4th degree was proved in [1]. On the other hand, many **NP**-hard problems were shown to have natural copositive formulations [30]: copositive programs are convex NLPs where one of the constraints involves a description of the copositive cone.

### *4.4.1. Verifying convexity*

The paper [72] presents a list of open questions in the field of computational complexity in numerical optimization problems. Problem 6, due to Shor, reads as follows.

> Given a degree-4 polynomial of $n$ variables, what is the complexity of determining whether this polynomial describes a convex function [in their whole range]?

Polynomials of degree smaller than 4 are easy to settle: degree-1 is linear and hence convex, degree-2 has constant Hessian which can be computed in polynomial time, and odd-degree polynomials are never convex (although they may be pseudo- or quasi-convex, see [52], Thm. 13.11). Degree-4, the smallest open question, was settled in [1].

The proof provided in [1] reduces from the problem of determining whether a given *biquadratic form*

$$\sum_{\substack{i\le j\\ k\le \ell}} \alpha_{ijk\ell} x_i x_j y_k y_\ell$$

is non-negative over all its range is strongly **NP**-hard by reduction from CLIQUE [57] by way of the Motzkin–Straus formulation equation (4.11). It shows how to construct a (variable) Hessian form from any biquadratic form such that the former is non-negative iff the latter is. This Hessian form allows the construction of a quartic polynomial that is convex over its range if the original biquadratic form is non-negative.

This result is extended also to deciding other forms of convexity: strict and strong convexity, as well as pseudo- and quasi-convexity.

## *4.4.2. The copositive cone*

First, a square symmetric matrix $A$ is *copositive* if $x^\top Ax \ge 0$ for all $x \ge 0$. If we removed the non-negativity condition $x \ge 0$, we would retrieve a definition of $A$ being PSD: while every PSD matrix is copositive, the converse is not true. Establishing copositivity of non-PSD matrices is **NP**-hard, as it involves verifying whether the optimal objective function value of the QP $P \equiv \min\{x^\top Ax \mid x \ge 0\}$ is $\ge 0$, or either $< 0$ or unbounded [68].

We consider a variant of equation (4.13) where $c$ is the zero vector (so the objective function is purely quadratic). This variant is called STANDARD QUADRATIC PROGRAMMING (StQP). It is **NP**-hard by inclusion of equation (4.11). The reformulation of the (**NP**-hard) StQP to a *convex* continuous NLP, reported here below, is a surprising result.

(a) We linearize each product $x_ix_j$ occurring in the quadratic form $x^\top Qx$ to a new variable $X_{ij}$. This yields

$$\left.\begin{array}{rrcl} \min & Q\bullet X & & \\ & \sum\limits_{j\le n} x_j & = & 1 \\ \forall j\le n & x_j & \ge & 0 \\ & X & = & xx^\top, \end{array}\right\}$$

where $\bullet$ denotes $\mathsf{trace}(Q^\top X)$. Note that $X = xx^\top$ encodes the whole constraint set

$$\forall i < j \le n \quad X_{ij} = x_i x_j.$$

(b) We square both sides of the simplex constraint $\sum_j x_j = 1$ to obtain $\sum_{i,j} x_ix_j = 1$, which can be written as $\mathbf{1}\bullet X = 1$, where $\mathbf{1}$ is the all-one $n\times n$ matrix.

(c) We replace the (nonconvex) set $C = \{X \mid X = xx^\top \wedge x \ge 0\}$ by its convex hull $\mathcal{C} = \mathsf{conv}(C)$. This yields

$$\left.\begin{array}{rrcl} \min & Q\bullet X & & \\ & \mathbf{1}\bullet X & = & 1 \\ & X & \in & \mathcal{C}. \end{array}\right\} \qquad (4.27)$$

The set $\mathcal{C}$ is a convex combination of rank-one matrices, and as such forms a convex cone.

(d) We write the dual of equation (4.27):

$$\left.\begin{array}{rcl} \max & y & \\ Q - y\mathbf{1} & \in & \mathcal{C}^*, \end{array}\right\} \tag{4.28}$$

where $\mathcal{C}^*$ is the dual cone of $\mathcal{C}$. Unlike the PSD cone, which is self-dual, $\mathcal{C}^* \neq \mathcal{C}$. In fact, $\mathcal{C}^*$ turns out to be the cone of all copositive matrices:

$$\mathcal{C}^* = \{A \mid \forall x \geq 0 \; (x^\top Ax \geq 0)\},$$

see ([41], Thm. 16.2.1) for a detailed proof. Both cones are convex and achieve strong duality. Again, equation (4.28) is a convex NLP.

Convex NLPs are known to be "easy to solve", since every local optimum is also global: thus a local solution algorithm should immediately find the global optimum (this probably motivated Rockafellar to put the barrier of complexity in MP at the frontier of convexity and nonconvexity). Accordingly, equation (4.28) is surprising and unexpected. One might object that convex hulls naturally turn hard problem classes into tractable ones. For example, convex hulls of integer programs allow us to write MILP as LP; in general, however, the resulting LP involves an exponentially long description, thereby yielding an LP which is polytime solvable in its own (exponential) description size, and thus takes exponential time to be solved in function of the description size of the original MILP. Our own practical experience with most convex NLPs suggests that any reasonable local NLP solver would find their global optima efficiently. In view of equation (4.28), the consequences would be formidable: are there efficient algorithms for **NP**-hard problems?

According to [12, 30], no-one knows of an efficient algorithm for solving the convex NLP in equation (4.28), *exactly* because of the copositive cone $\mathcal{C}^*$. The most common convex NLPs are defined over the non-negative orthant (or translations and/or scalings thereof). This cone has a very simple and compact description: $\forall j \leq n \; (x_j \geq 0)$. The PSD cone $\{A \mid A \succeq 0\}$ also has a compact description: it consists of all the square symmetric matrices with non-negative eigenvalues. Since we can test whether a given matrix is PSD in polytime ([79], p. 1152), the polynomially long traces of the checking algorithm provide compact certificates for both YES and NO instances: this is the next best thing to an explicit formulation. Testing copositivity, however, is **NP**-hard, so there is little hope of achieving such compact descriptions for the copositive cone $\mathcal{C}^*$.

A different way to see this issue is to consider that convex NLPs over matrix cones are routinely solved by IPMs, the polytime analysis of which rest on the existence of some functions called "self-concordant barriers", which deviate the search path for the optimum away from the border of the cone. Barrier functions are used as penalty functions to be added to the objective function, which tend to infinity towards the boundary of the cone. Self-concordant barriers can be optimized efficiently using Newton's method. It seems that we know self-concordant barriers for many convex cones (orthants, PSD and more), but not for $\mathcal{C}^*$. In [13], an IPM-based heuristic is proposed to solve copositive programs such as [13].

Yet another interpretation of the complexity inherent in the copositive cone is the classification of extremal matrices of copositive cones of small dimension, pursued in [46]. This paper emphasizes that the description complexity of these extreme rays increases dramatically as the dimension increases.

## 5. Conclusion

We reviewed some MINLP results from the point of view of computability and complexity. We introduced the basic notions of decidability and computational complexity, and then applied them to problems modelled naturally as MINLP or some of its subclasses. While this is by no means a complete survey, we mentioned where to find additional information. We hope this paper will help clarify doubts – but also raise questions and draw attention to some of the theoretical issues related to MINLP.

*Acknowledgements.* I am grateful to the scientific committee of the Aussois Combinatorial Optimization Workshop (C.O.W.) 2017 for inviting me to give the opening tutorial talk (which forced me to study computability and complexity through the Christmas break 2016), as well as to R. Mahjoub for having invited this survey to be published on RAIRO-RO. I am grateful to F. Raimondi and F. Roda for a 20-year long (and counting) discussion on incompleteness and undecidability, and to S. Iwata for bringing the paper [62] to my attention. I am also grateful to two referees for helping me improve the style and presentation of the material: without them, you would have found odd personal comments and anecdotes strewn all around the place, and the whole paper written in the first singular personal pronoun. Apparently, the unwritten standards of scientific writing, which I know and abide by, but thought could be a matter of personal choice in an invited survey, are in fact vigorously enforced (trust me). This paper has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement no. 764759 (ITN "MINOA").

## References

[1] A. Ahmadi, A. Olshevsky, P. Parrilo and J. Tsitsiklis, NP-hardness of deciding convexity of quartic polynomials and related problems. *Math. Program.* **137** (2013) 453–476.
[2] M. Aigner, Turán's graph theorem. *Am. Math. Mon.* **102** (1995) 808–816.
[3] M. Bardet, J.C. Faugère and B. Salvy, On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations, in *Proceedings of International Conference on Polynomial System Solving* (2004).
[4] S. Basu, R. Pollack and M.-F. Roy, Algorithms, in Real Algebraic Geometry. Springer, New York (2006).
[5] N. Beeker, S. Gaubert, C. Glusa and L. Liberti, Is the distance geometry problem in NP?, in Distance Geometry: Theory, Methods, and Applications. Edited by A. Mucherino, C. Lavor, L. Liberti and N. Maculan. Springer, New York (2013) 85–94.
[6] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke and A. Mahajan, Mixed-integer nonlinear optimization. *Acta Numer.* **22** (2013) 1–131.
[7] K. Bennett and O. Mangasarian, Bilinear separation of two sets in $n$-space. *Comput. Optim. Appl.* **2** (1993) 207–227.
[8] D. Bienstock, Computational study of a family of mixed-integer quadratic programming problems. *Math. Program.* **74** (1996) 121–140.
[9] D. Bienstock and A. Michalka, Polynomial solvability of variants of the trust-region subproblem, in Vol. 25 of *SODA. Proceedings of the 25th Annual ACM Symposium on Discrete Algorithms.* ACM, Philadelphia (2014) 380–390.
[10] L. Blum, M. Shub and S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines. *Bull. Am. Math. Soc.* **21** (1989) 1–46.
[11] I. Bomze, Evolution towards the maximum clique. *J. Glob. Optim.* **10** (1997) 143–164.
[12] I. Bomze, Copositive optimization — recent developments and applications. *Eur. J. Oper. Res.* **216** (2012) 509–520.
[13] I. Bomze, M. Dür, E.D. Klerk, C. Roos, A. Quist and T. Terlaky, On copositive programming and standard quadratic optimization problems. *J. Glob. Optim.* **18** (2000) 301–320.
[14] C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi and P. Toth, On the optimal design of water distribution networks: a practical MINLP approach. *Optim. Eng.* **13** (2012) 219–246.
[15] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering. *IEEE Trans. Knowl. Data Eng.* **20** (2008) 172–188.
[16] M. Bruglieri and L. Liberti, Optimal running and planning of a biomass-based energy production process. *Energy Policy* **36** (2008) 2430–2438.
[17] B. Buchberger, Bruno Buchberger's PhD Thesis 1965: an algorithm for finding the basis elements of the residue class ring of a zero-dimensional polynomial ideal. *J. Symb. Comput.* **41** (2006) 475–511.
[18] S. Burer and A. Letchford, Non-convex mixed-integer nonlinear programming: a survey. *Surv. Oper. Res. Manag. Sci.* **17** (2012) 97–106.
[19] S. Cafieri, L. Liberti, F. Messine and B. Nogarede, Optimal design of electrical machines: mathematical programming formulations. *COMPEL* **32** (2013) 977–996.
[20] Y.-J. Chang and B. Wah, *Polynomial Programming Using Gröbner Bases.* Technical Report, University of Illinois at Urbana-Champaign (1994).
[21] D. Cifuentes and P. Parrilo, Exploiting chordal structure in polynomial ideas: a Gröbner basis approach. *SIAM J. Discrete Math.* **30** (2016) 1534–1570.
[22] A. Cobham, The intrinsic computational difficulty of functions, Logic, Methodology and Philosophy of Science, edited by Y. Bar-Hillel. North-Holland, Amsterdam (1965) 24–30.
[23] G. Collins, Quantifier elimination for real closed fields. *ACM SIGSAM Bull.* **8** (1974) 80–90.
[24] S. Cook, The complexity of theorem-proving procedures, in *Proc. of STOC '71 Proceedings of the third annual ACM symposium on Theory of computing.* New York (1971) 151–158.
[25] P. Cousot and R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction of approximations of fixed points. *Princ. Program. Lang.* **4** (1977) 238–252.
[26] C. D'Ambrosio, Application-oriented mixed integer non-linear programming. *4OR* **8** (2010) 319–322.
[27] C. D'Ambrosio and A. Lodi, Mixed-integer nonlinear programming tools: a practical overview. *4OR* **9** (2011) 329–349.

[28] M. Davis, Arithmetical problems and recursively enumerable predicates. *J. Symb. Logic* **18** (1953) 33–41.
[29] M. Davis, H. Putnam and J. Robinson, The decision problem for exponential Diophantine equations. *Ann. Math.* **74** (1961) 425–436.
[30] M. Dür, Copositive programming — a survey, in Recent Advances in Optimization and its Applications in Engineering, edited by M. Dür *et al.* Springer, Heidelberg (2010).
[31] J. Edmonds, Paths, trees and flowers. *Can. J. Math.* **17** (1965) 449–467.
[32] C. Floudas, Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, New York (1995).
[33] C. Floudas, Deterministic Global Optimization. Kluwer Academic Publishers, Dordrecht (2000).
[34] T. Franzen, Gödel's Theorem: An Incomplete Guide to I1 Use and Abuse. Peters, Wellesley (2005).
[35] S. Gao, A. Platzer and E. Clarke, Quantifier elimination over finite fields using Gröbner bases, edited by F. Winkler. Algebraic Informatics. Vol. 6742 of *Lect. Note Comput. Sci.* Springer, New York (2011) 140–157.
[36] K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. *Monatshefte Math. Phys.* **38** (1930) 173–198.
[37] I. Grossmann, Mixed-integer programming approach for the synthesis of integrated process flowsheets. *Comput. Chem. Eng.* **9** (1985) 463–482.
[38] I. Grossmann (Ed.), Global Optimization in Engineering Design. Kluwer Academic Publishers, Dordrecht (1996).
[39] I. Grossmann and Z. Kravanja, Mixed-integer nonlinear programming: a survey of algorithms and applications, edited by L. Biegler, T. Coleman, A. Conn and F. Santosa. Large-Scale Optimization with Applications, Part II: Optimal Design and Control. Springer (1997) 73–100.
[40] K. Hägglöf, P. Lindberg and L. Svensson, Computing global minima to polynomial optimization problems using Gröbner bases. *J. Glob. Optim.* **7** (1995) 115–125.
[41] M. Hall, Combinatorial Theory, 2nd edn. Wiley, New York (1986).
[42] I. Harjunkoski, T. Westerlund, R. Pörn and H. Skrifvars, Different transformations for solving nonconvex trim-loss problem by MINLP. *Eur. J. Oper. Res.* **105** (1998) 594–603.
[43] R. Helgason, J. Kennington and H. Lall, A polynomially bounded algorithm for a singly constrained quadratic program. *Math. Program.* **18** (1980) 338–343.
[44] R. Hemmecke, M. Köppe, J. Lee and R. Weismantel, Nonlinear integer programming, 50 Years of Integer Programming, edited by M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi and L. Wolsey. Springer, Berlin (2010) 561–618.
[45] H. Hijazi and L. Liberti, Constraint qualification failure in action. *Oper. Res. Lett.* **44** (2016) 503–506.
[46] R. Hildebrand, The extreme rays of the $5 \times 5$ copositive cone. *Linear Algebra Appl.* **437** (2012) 1538–1547.
[47] D. Hochbaum, Complexity and algorithms for nonlinear optimization problems. *4OR* **3** (2005) 171–216.
[48] R. Jeroslow, There cannot be any algorithm for integer programming with quadratic constraints. *Oper. Res.* **21** (1973) 221–224.
[49] J. Jones, Universal Diophantine equation. *J. Symb. Logic* **47** (1982) 549–571.
[50] J. Kallrath, Cutting circles and polygons from area-minimizing rectangles. *J. Glob. Optim.* **43** (2009) 299–328.
[51] R. Karp, Reducibility among combinatorial problems. Complexity of computer computations, edited by R. Miller and W. Thatcher. Vol. 5 of *IBM Research Symposia.* Plenum, New York (1972) 85–104.
[52] J.-B. Lasserre, An Introduction to Polynomial and Semi-Algebraic Optimization. Cambridge University Press, Cambridge (2015).
[53] J. Lee and S. Leyffer (Eds.), Mixed integer nonlinear programming. Vol. 154 of *IMA*. Springer, New York (2012).
[54] L. Liberti, Reformulations in mathematical programming: definitions and systematics. *RAIRO-RO* **43** (2009) 55–86.
[55] L. Liberti and C. Lavor, Open research areas in distance geometry, Open Problems in Optimization, edited by A. Migalas and P. Pardalos. Springer, New York (2018).
[56] L. Liberti and F. Marinelli, Mathematical programming: turing completeness and applications to software analysis. *J. Comb. Optim.* **28** (2014) 82–104.
[57] C. Ling, J. Nie, L. Qi and Y. Ye, Biquadratic optimization over unit spheres and semidefinite programming relaxations. *SIAM J. Optim.* **20** (2009) 1286–1310.
[58] C. Lizon, C. D'Ambrosio, L. Liberti, M.L. Ravalec and D. Sinoquet, A mixed-integer nonlinear optimization approach for well placement and geometry, in *Proceedings of the 14th European Conference on the Mathematics of Oil Recovery*. Vol. XIV of *ECMOR, Houten.* EAGE (2014) A38.
[59] R. Lyndon, Notes on logic. Number 6 in Mathematical Studies. Van Nostrand, New York (1966).
[60] N. Maculan, P. Michelon and J. MacGregor Smith, *Bounds on the Kissing Numbers in $\mathbb{R}^n$: Mathematical Programming Formulations.* Technical Report, University of Massachusetts, Amherst, USA (1996).
[61] Y. Matiyasevich, Enumerable sets are Diophantine. *Sov. Math. Dokl.* **11** (1970) 354–357.
[62] T. Matsui, NP-hardness of linear multiplicative programming and related problems. *J. Glob. Optim.* **9** (1996) 113–119.
[63] N. Megiddo, On the complexity of polyhedral separability. *Discrete Comput. Geom.* **3** (1988) 325–337.
[64] L. Mencarelli, Y. Sahraoui and L. Liberti, A multiplicative weights update algorithm for MINLP. *EURO J. Comput. Optim.* **5** (2017) 31–86.
[65] F. Messine, B. Nogarede and J.-L. Lagouanelle, Optimal design of electromechanical actuators: a new method based on global optimization. *IEEE Trans. Magn.* **34** (1998) 299–307.

[66] J. Milnor, On the Betti numbers of real varieties. *Proc. Am. Math. Soc.* **15** (1964) 275–280.

[67] T. Motzkin and E. Straus, Maxima for graphs and a new proof of a theorem of Turán. *Can. J. Math.* **17** (1965) 533–540.

[68] K. Murty and S. Kabadi, Some NP-complete problems in quadratic and nonlinear programming. *Math. Program.* **39** (1987) 117–129.

[69] A. Neumaier, Complete search in continuous global optimization and constraint satisfaction. *Acta Numer.* **13** (2004) 271–369.

[70] P. Pardalos and G. Schnitger, Checking local optimality in constrained quadratic programming is NP-hard. *Oper. Res. Lett.* **7** (1988) 33–35.

[71] P. Pardalos and S. Vavasis, Quadratic programming with one negative eigenvalue is NP-hard. *J. Glob. Optim.* **1** (1991) 15–22.

[72] P. Pardalos and S. Vavasis, Open questions in complexity theory for numerical optimization. *Math. Program.* **57** (1992) 337–339.

[73] K. Pruitt, S. Leyffer, A. Newman and R. Braun, A mixed-integer nonlinear program for the optimal design and dispatch of distributed generation systems. *Optim. Eng.* **15** (2014) 167–197.

[74] A. Quist, R.V. Geemert, J. Hoogenboom, T. Illes, E.D. Klerk, C. Roos and T. Terlaky, Finding optimal nuclear reactor core reload patterns using nonlinear optimization and search heuristics. *Eng. Optim.* **32** (1999) 143–176.

[75] J. Renegar and M. Shub, Unified complexity analysis for Newton LP methods. *Math. Program.* **53** (1992) 1–16.

[76] M. Ruiz, J. Maeght, A. Marié, P. Panciatici and A. Renaud, A progressive method to solve large-scale AC optimal power flow with discrete variables and control of the feasibility, in *Proceedings of the Power Systems Computation Conference*. Vol. 18 of *PSCC*, Piscataway. IEEE (2014).

[77] S. Sahni, Computationally related problems. *SIAM J. Comput.* **3** (1974) 262–279.

[78] E. Salgado, A. Scozzari, F. Tardella and L. Liberti, Alternating current optimal power flow with generator selection, Combinatorial Optimization (*Proceedings of ISCO 2018*) Edited by J. Lee, G. Rinaldi and R. Mahjoub. In Vol. 10856 of *Lecture Notes in Computer Science*. Springer, New York (2018) 364–375.

[79] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003).

[80] H. Sherali and W. Adams, A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer Academic Publishers, Dodrecht (1999).

[81] A. Tarski, *A Decision Method for Elementary Algebra and Geometry.* Technical Report R-109, Rand Corporation (1951).

[82] M. Tawarmalani and N. Sahinidis, Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer, Dordrecht (2002).

[83] A. Turing, On computable numbers, with an application to the Entscheidungs problem. *Proc. Lond. Math. Soc.* **42** (1937) 230–265.

[84] S. Vavasis, Quadratic programming is in NP. *Inf. Process. Lett.* **36** (1990) 73–77.

[85] S. Vavasis, Nonlinear Optimization: Complexity Issues. Oxford University Press, Oxford (1991).

[86] S. Vavasis, Complexity issues in global optimization: a survey, edited by R. Horst and P. Pardalos. Vol. 1 of Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht (1995) 27–41.

[87] S. Vavasis and R. Zippel, *Proving Polynomial-Time for Sphere-Constrained Quadratic Programming.* Technical Report 90-1182, Dept. of Comp. Sci., Cornell University (1990).

[88] C. Witzgall, An all-integer programming algorithm with parabolic constraints. *J. Soc. Ind. Appl. Math.* **11** (1963) 855–870.

[89] W. Zhu, Unsolvability of some optimization problems. *Appl. Math. Comput.* **174** (2006) 921–926.