COMPLEXITY ISSUES IN GLOBAL OPTIMIZATION: A SURVEY

STEPHEN A. VAVASIS*
Department of Computer Science
Cornell University
Ithaca, NY 14853

1. Introduction

Complexity theory refers to the asymptotic analysis of problems and algorithms. How efficient is an algorithm for a particular optimization problem, as the number of variables gets large? Are there problems for which no efficient algorithm exists? These are the questions that complexity theory attempts to address. The theory originated in work by Hartmanis and Stearns (1965).

By now there is much known about complexity issues in nonlinear optimization. In particular, our recent book Vavasis (1991) contains all the details on many of the results surveyed in this chapter.

We begin the discussion with a look at convex problems in the next section. These problems generally have efficient algorithms. In Section 3 we study the complexity of two nonconvex problems that also have efficient algorithms because of special structure. In Section 4, we look into hardness results (proofs of the nonexistence of efficient algorithms) for general nonconvex problems. Finally, in Section 5 we look at recent developments in "approximation" algorithms.

We follow the notation in this chapter that lower-case boldface letters are vectors, lower-case italic letters are scalars, and upper-case italic letters are sets or matrices. Superscript T indicates matrix transpose, and $\mathbf{a}^T\mathbf{x}$ indicates inner product. The operators '$\geq$' and '$\leq$' are applied componentwise to vectors; we say $\mathbf{x} \geq \mathbf{y}$ if each entry of $\mathbf{x}$ is greater than or equal the corresponding entry of $\mathbf{y}$.

2. Convex problems

Recall that a subset D of $\mathbb{R}^n$ is said to be *convex* if for all $\mathbf{x}, \mathbf{y} \in D$ and for all $\lambda \in [0, 1]$,

$$(1 - \lambda)\mathbf{x} + \lambda\mathbf{y} \in D.$$

* Work supported in part by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550. Also supported in part by an NSF Presidential Young Investigator award.

Convex sets occur very often as the feasible sets for optimization problems. For example, the set given by linear constraints

$$D = \{\mathbf{x} \in \mathsf{IR}^n : A\mathbf{x} \geq \mathbf{b}\},$$

where A is a given $m \times n$ matrix and $\mathbf{b}$ is a given n-vector, is convex. In addition, constraints requiring a vector to be within a certain distance from some given data point, or within a certain distance of another vector, also give rise to convex sets. An important property of convex sets is that the intersection of a collection of convex sets is also convex.

Let D be a nonempty convex set. A function $f : D \rightarrow \mathsf{IR}$ is said to be *convex* if for all $\mathbf{x}, \mathbf{y} \in S$ and for all $\lambda \in [0, 1]$,

$$(1-\lambda)f(\mathbf{x}) + \lambda f(\mathbf{y}) \geq f((1-\lambda)\mathbf{x} + \lambda\mathbf{y}).$$

Convex functions occur very often as the objective functions for optimization problems. For example, a linear function $\mathbf{c}^T\mathbf{x}$, where $\mathbf{c}$ is vector, is convex. Similarly, the quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$ where H is an $n \times n$ symmetric positive semidefinite matrix and $\mathbf{c}$ is an n-vector, is also a convex function. Functions that represent distances are usually convex. In many physical problems, energy is a convex function of the problem variables.

As we shall see, problems with convex feasible sets and convex objective functions have efficient algorithms in general. Such problems are generally called *convex optimization problems.* Thus, linear programming (minimize $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{b}$) is a convex problem. Convex problems have some important theoretical properties; in particular, any local minimum is a global minimum. This means that any algorithm that attempts to find a local minimum (e.g., Newton's method with line-search—see Dennis and Schnabel (1983)) for a convex problem will automatically compute a global minimum.

When a problem is known to be convex, powerful algorithms can be brought to bear on it that are guaranteed to converge to a global minimum efficiently. The first such algorithm is the *ellipsoid* algorithm, invented by Yudin and Nemirovsky (1976). (These authors called this algorithm the "modified method of centers of gravity.") The first key idea in the development of the ellipsoid algorithm is the following theorem, which has a simple proof.

Theorem 1 *Let D be a nonempty convex subset of IR^n and $f : D \rightarrow \mathsf{IR}$ a differentiable convex function. Let $\mathbf{x}, \mathbf{y}$ be two points in D such that $f(\mathbf{x}) \leq f(\mathbf{y})$. Then*

$$\nabla f(\mathbf{y})^T(\mathbf{y} - \mathbf{x}) \geq 0.$$

This theorem has the following consequence: If we take $\mathbf{x}$ in the theorem to be the global minimum of f (assuming it exists), then $f(\mathbf{y}) \geq f(\mathbf{x})$ for any $\mathbf{y}$, so $\nabla f(\mathbf{y})^T(\mathbf{y} - \mathbf{x}) \geq 0$ always. This means that, given any point $\mathbf{y}$, we can find a plane containing $\mathbf{y}$ such that the global minimizer is guaranteed to lie on one side of the plane. The condition in the theorem that f be differentiable can be relaxed.

The existence of the separating plane suggests some kind of space-partitioning algorithm. Assume that the problem is to minimize a convex function f over a convex set D. Pick an initial point $\mathbf{x}_1$ somewhere near the middle of D, and find the halfspace through $\mathbf{x}_1$ containing the global minimizer. Now let D_1 be the intersection of D with the halfspace, repeat the process for D_1, etc., each time shrinking the region containing the global minimizer.

The difficulty with the algorithm in the last paragraph is that finding a point "somewhere near the middle" of an arbitrary convex set seems to be a hard problem. Nemirovsky and Yudin solve this problem by surrounding the initial feasible set with an ellipsoid, and repeatedly partitioning the ellipsoid. Recall that an *ellipsoid* E is a subset of $\mathbb{R}^n$ defined as follows:

$$E = \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x}-\mathbf{c})^T M(\mathbf{x}-\mathbf{c}) \leq 1\}.$$

Here, $\mathbf{c}$ is a vector, called the *center* of the ellipsoid, and M is a $n \times n$ symmetric positive definite matrix.

In the ellipsoid method, one starts by computing an large ellipsoid E_1 containing the global minimizer. (The exact method for computing E_1 depends on the format of the problem. For example, in linear programming with rational coefficients one can compute E_1 based on the total size, in digits, of the rational numbers in the problem.) Say that the center of E_1 is $\mathbf{c}_1$. Now we find a halfspace H_1 of the form

$$H_1 = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{d}_1^T(\mathbf{x}-\mathbf{c}_1) \geq 0\}$$

that contains the global minimizer. The method for determining H_1 (i.e., determining $\mathbf{d}_1$) is as follows. If $\mathbf{c}_1$ is a feasible (i.e., it lies in D) then we pick $\mathbf{d}_1$ to be $-\nabla f(\mathbf{c}_1)$ and apply the theorem. If $\mathbf{c}_1$ is not feasible, then we choose a halfspace through $\mathbf{c}_1$ that contains D on one side of it. The exact method for finding such a halfspace depends on how D is presented. For example, if D is a polyhedron presented via linear constraints, i.e.,

$$D = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \geq \mathbf{b}\},$$

then for an infeasible $\mathbf{c}_1$ there must be an index i such that $\mathbf{a}_i^T\mathbf{c}_1 < b_i$, where $\mathbf{a}_i^T$ is the ith row of A. Then we take $\mathbf{d}_1 = \mathbf{a}_i$; clearly every point in D satisfies $d_1^T(\mathbf{x}-\mathbf{c}_1) \geq 0$.

From the way that H_1 is computed, we know that the global minimizer must lie in $E_1 \cap H_1$. The next step of the algorithm is to compute a new ellipsoid E_2 containing $E_1 \cap H_1$. It turns out that the volume of E_2 is strictly smaller than the volume of E_1 by a factor depending on n. Then we continue the algorithm with E_2. An example of E_1, H_1, E_2 is shown in Figure 1.

It has been proved by Yudin and Nemirovsky (1976) that the ellipsoid algorithm always converges at a certain rate to the global optimum. The following theorem is a special case of this type of result; it is proved in Vavasis (1991).

Theorem 2 *Consider the problem of minimizing a convex function f over the unit cube $D = [0,1]^n$. Let p be an upper bound on the difference between the maximum and minimum of f on D. Then in $2n(n+1)(\ln(p/\epsilon)+\ln n)+2$ iterations a point $\mathbf{x}$ can be found such that $f(\mathbf{x}) \leq f(\mathbf{x}^*)+\epsilon$, where $\mathbf{x}^*$ is the global minimum.*

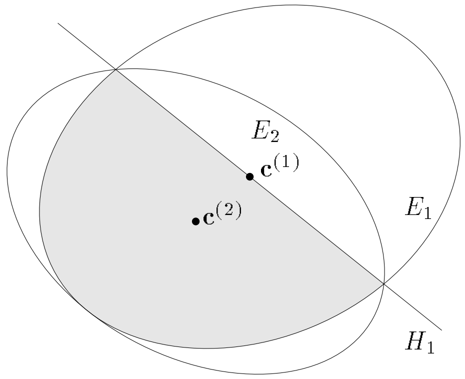

Fig. 1. Illustration of E_2 containing $E_1 \cap H_1$.

As mentioned earlier, linear programming is a special case of convex optimization. Therefore, the ellipsoid method can be applied to linear programming. Khachiyan (1979) showed that this yields a polynomial-time algorithm for linear programming. An algorithm is said to be *polynomial-time* if the number of steps is bounded by a polynomial in the length of the input. Usually, the number of steps means the number required for a *Turing machine*, a theoretical model of computation. The Turing machine is discussed a bit more in Section 4. The *length of the input* refers to the total number of symbols needed to represent the input. For example, in the case of linear programming, we assume that the input (A,$\mathbf{b}$ and $\mathbf{c}$) are matrices and vectors of rational numbers, and the size of the input is the total number of digits in the numerators and denominators of all the rational numbers.

After the discovery of this first polynomial time algorithm for linear programming, the next major advance was the discovery of *interior point methods* by Karmarkar (1984). These are polynomial-time algorithms efficient in practice for linear programming. Recently, Nesterov and Nemirovsky (1989) have shown how to generalize interior point methods to a variety of convex optimization problems via *self-concordant functions.*

3. Two nonconvex problems with efficient algorithms.

In this section we examine two nonconvex problems that have efficient algorithms. The first is fractional linear programming, and the second is sphere-constrained quadratic minimization. As we shall see in the next section, most nonconvex problems do not have efficient algorithms, so the problems in this section should be regarded as unusual.

Fractional linear programming (FLP) is the problem of minimizing

$$(\mathbf{c}^T\mathbf{x} + \gamma)/(\mathbf{d}^T\mathbf{x} + \delta)$$

subject to $A\mathbf{x} \geq \mathbf{b}$. (We must assume that $\mathbf{d}^T\mathbf{x} + \delta$ does not change signs over the domain, otherwise the problem is unbounded. We assume in this section that it is positive).

This problem arises in many conomic applications. The objective function is nonconvex. It has, however, the property of *pseudoconvexity*. A function $f(\mathbf{x})$ on a convex set is said to be *pseudoconvex* if the inequality stated in Theorem 1 holds for all $\mathbf{x}, \mathbf{y}$. A pseudoconvex function may be minimized using the ellipsoid algorithm. When applying the ellipsoid algorithm to a pseudoconvex function one must check that the gradient does not vanish at a non-minimum; this can be verified for FLP.

More efficient algorithms are known for FLP, such as Dinkelbach's algorithm. These algorithms typically introduce a parameter θ, that is a "guess" at the optimum value of the objective function. It is not hard to show that if we define

$$\begin{aligned} f(\theta) = \text{minimum of } & c^T\mathbf{x} + \gamma - \theta(\mathbf{d}^T\mathbf{x} + \delta) \\ \text{subject to } & A\mathbf{x} \geq \mathbf{b} \end{aligned}$$

then the optimal value of the FLP θ^* is corresponds exactly to a root of f. Accordingly, one tries to find a minimum of function f using one-variable rootfinding techniques. See Schaible and Ibaraki (1983).

A second nonconvex problem possessing an efficient algorithm is minimizing a quadratic function over a sphere. Recall that a general quadratic function of n variables has the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$, where H is symmetric. Thus, the problem is:

$$\begin{aligned} \text{minimize } & \tfrac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to } & \mathbf{x}^T\mathbf{x} \leq 1. \end{aligned}$$

(Without loss of generality, we have centered the sphere constraint at the origin and used a sphere of radius 1.) The key solving this problem is the first and second order necessary conditions. First, we make the assumption that the solution $\mathbf{x}^*$ satisfies the constraint as an equation: $\mathbf{x}^{*T}\mathbf{x}^* = 1$. The case when this does not hold occurs only when H is positive definite and is easy to detect. Under the assumption, the first and second order conditions are as follows. There is a μ such that:

$$\begin{aligned} & H\mathbf{x}^* + \mathbf{c} = -\mu\mathbf{x}^*, \\ & \mathbf{x}^{*T}\mathbf{x}^* = 1, \\ & \mu \geq 0, \\ & H + \mu I \text{ is positive semidefinite.} \end{aligned}$$

As observed by Gay (1979) and Sorenson (1982) these conditions turn out to be sufficient for global minimality. This is a highly unusual in optimization theory. For nonconvex problems, conditions like this are usually not sufficient even for local minimality.

An algorithm to globally minimize a quadratic function on a sphere in polynomial time was proposed independently by Ye (1988) and Karmarkar (1989). The algorithm attempts to find μ and $\mathbf{x}^*$ satisfying the above conditions using a binary search. It converges to the solution at a linear rate. Vavasis and Zippel (1990) showed that this algorithm answers the associated decision problem in polynomial time. (See the next section for a description of "decision problems.")

More recently Ye (1992) has argued that Newton's method can be used to obtain μ giving better complexity bounds.

4. General nonconvex problems

In the last section we described two nonconvex problems whose global minima can be found efficiently. Unfortunately, these problems are the exception rather than the rule. General nonconvex problems seem to be intractable. In this section we will describe two kinds of intractability results. The first is for problems where the input is specified as numerical data, including specifically quadratic programming. The second kind of result is for problems in which the input is a "black box."

Recall that quadratic programming (QP) is the problem of minimizing a quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{b}$. If we assume that $H, \mathbf{c}, A, \mathbf{b}$ are specified as sequences of rational numbers, then the problem is in a form that can be addressed by the Turing machine model of computation. We do not define the Turing machine model here; its important features are (1) it manipulates symbols; the list of symbols comes from a finite set (e.g., the digits $0, \ldots, 9$ and punctuation marks); (2) it has an unbounded number of memory cells, and each cell can hold one symbol; and (3) it is controlled by a program whose length is finite. One of the most powerful theories for showing problems are apparently intractable is the theory of **NP**-*completeness*, for which we will now give overview.

First, we have to say more about *problems*. A problem is a mapping $F : I \rightarrow B$ where I, B are both sets of strings of symbols. For example, in the case of quadratic programming, I would consist of quadruples of the form $(H, \mathbf{c}, A, \mathbf{b})$ written out in digits, and B would be vectors written out as rational numbers, and F would map a quadruple $(H, \mathbf{c}, A, \mathbf{b})$ to $\mathbf{x}^*$ that minimizes $\frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{b}$. Elements of I are called *instances* of the problem. We assume that the set I is defined by some fairly simple syntactic property (e.g., it is defined as quadruples $(H, \mathbf{c}, A, \mathbf{b})$ where all the matrices and vectors have compatible sizes).

We say that a Turing machine computes this function F, i.e., solves quadratic programming, if receives $(H, \mathbf{c}, A, \mathbf{b})$ as input and returns the correct $\mathbf{x}^*$ as output. We omit discussion about what the machine should do if there is more than one global minimum, or if the problem is unbounded or infeasible.

A special class of problems is the class of *decision problems*. For a decision problem the set B contains just two elements: $B = \{\mathbf{yes}, \mathbf{no}\}$. Here is an example of a decision problem connected to quadratic programming. The input set I is 5-tuples of the form $(H, \mathbf{c}, A, \mathbf{b}, \zeta)$. This instance is a **yes**-instance if there is a feasible point to $A\mathbf{x} \geq \mathbf{b}$ such that the objective function $f(\mathbf{x})$ achieves a value of ζ or smaller. Otherwise the instance is a **no**-instance. The Turing machine outputs nothing other than **yes** or **no**.

Clearly if we had Turing machine that computed the global minimizer or its value then we could use it to solve the decision problem described in the last paragraph. Conversely, a Turing machine for the decision problem in the last paragraph could be used to actually find the global minimizer by using binary search, and by adding additional constraints to find the entries of $\mathbf{x}^*$.

Traditional complexity theory attempts to classify only decision problems. One important complexity class is **P**; these are decision problems for which there is a Turing machine that solves them, such that the running time of the Turing machine is bounded by a polynomial in the length of the input. Linear programming with rational numbers, when posed as a decision problem, is in **P**.

A decision problem not in **P** is sometimes said to be *intractable*; such a decision problem has no asymptotically efficient algorithm. In general, there are very few decision problems connected with optimization that have been proved to be intractable. The strongest known rigorous result in this direction is the theory of **NP**-completeness.

We start with the complexity class **NP**. This class is somewhat harder to define than **P**. A decision problem $F : I \rightarrow \{\mathbf{yes}, \mathbf{no}\}$ is in **NP** if there is a set C of strings defined by some simple syntactic property, and another decision problem $G : I \times C \rightarrow \{\mathbf{yes}, \mathbf{no}\}$ such that:

- G is in **P**, and
- If $F(x) = \mathbf{yes}$, then there is at least one $y \in C$, such that y has length at most polynomial in x, and such that $G(x, y) = \mathbf{yes}$. String y is called a *certificate* of x.
- If $F(x) = \mathbf{no}$, then for all $y \in C$, $G(x, y) = \mathbf{no}$.

Note that $\mathbf{P} \subset \mathbf{NP}$ in this definition; if F were in **P** we could take C to contain a single string, the empty string, and we could take $G = F$.

An example of an **NP** problem is the well-known *subset sum problem*. An instance of the subset-sum problem consists of a sequence of integers $a_1 \ldots, a_m$ and another integer γ. This instance is defined to be a **yes**-instance if there exists a subset $J \subset \{1, \ldots, m\}$ such that

$$\sum_{i \in J} a_i = \gamma.$$

This problem is in **NP**; the argument is as follows. We take the set C in the definition of **NP** to be all finite subsets J of the positive integers. The augmented decision problem G is defined to have as input $(a_1, \ldots, a_m, \gamma, J)$. This augmented instance is a **yes**-instance of G if the above equation holds. Clearly G can be solved in polynomial time; the Turing machine for G merely has to add up a sequence of integers and compare the answer to γ.

Thus, a decision problem is in **NP** all of its **yes** instances can be easily "certified." This class includes a very large number of interesting combinatorial and optimization-related decision problems.

A subclass of **NP** is the set of **NP**-complete problems. A decision problem F in **NP** is said to be **NP**-*complete* if every problem in **NP** can be reduced in polynomial time to F. We do not go into detail about what form this reduction must take, but we provide an example. Consider the *partition* problem. The instances of this problem are sequences of nonnegative integers $(b_1, \ldots, b_m)$. An instance is a **yes**-instance if there is a subset $J \subset \{1, \ldots, m\}$ such that

$$\sum_{i \in J} b_i = \sum_{i \notin J} b_i.$$

Clearly this problem is in **NP**. Furthermore, an instance of the partition problem can easily be reduced to an instance of the subset sum problem; just take $(a_1, \ldots, a_m)$ in the subset sum problem to be $(b_1, \ldots, b_m)$, and take $\gamma = (b_1 + \cdots + b_m)/2$. Then the instance of the partition problem is a **yes**-instance if and only if the instance of the subset-sum problem is a **yes**-instance.

It is nearly as simple to demonstrate a reduction in the opposite direction: An instance of the subset-sum problem can be transformed to an instance of the partition problem. In particular, the reader can check that starting from an instance $(a_1, \ldots, a_m, \gamma)$ of the subset-sum problem, we can transform it to the instance

$$(a_1, \ldots, a_m, a_1 + \cdots + a_m - 2\gamma)$$

of the partition problem.

Since every problem in **NP** can be reduced efficiently to an **NP**-complete problem, the **NP**-complete problem is thus the "hardest" problem in **NP**. There are thousands of problems now known to be **NP**-complete, including the subset-sum and partition problems. Many people believe that **NP** contains some intractable problems, in which case all **NP**-complete problems would also be intractable. However, no problem in **NP** has ever been proved to lie outside of **P**.

For a general background on the complexity classes introduced here, see Garey and Johnson (1979). See Vavasis (1991) for information about how these classes relate to optimization. The theory of **NP**-completeness originates with Cook (1971) and Karp (1972).

A very important result for the field of global optimization is the following theorem.

Theorem 3 *Quadratic programming, when stated as a decision problem (as above), is* **NP***-complete.*

The proof that quadratic programming is **NP**-hard is due to Sahni (1974). A problem F is said to be "**NP**-hard" if any problem in **NP** can be reduced to F. Thus, **NP**-hardness is a weaker condition than **NP**-completeness. (For F to be **NP**-complete, we additionally require that F itself must lie in **NP**. Indeed, **NP**-hard problems are not even necessarily decision problems.) The result in previous theorem (a strengthening of Sahni's result) is due to Vavasis (1990).

Even when restricted to special cases, quadratic programming remains **NP**-hard. Here are some examples of **NP**-hard special cases of quadratic programming.

1. Quadratic knapsack problems. These are problems of the form:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} d_i x_i^2 + c_i x_i \\ \text{subject to} & a_1 x_1 + \cdots + a_n x_n = \gamma, \\ & l_i \le x_i \le u_i, \text{ for } i = 1, \ldots, n. \end{array}$$

   Here, the input data is the list of d_i's, c_i's, α_i's, γ, a_i's, l_i's, and u_i's. This problem was shown to be **NP**-hard by Sahni.

2. Box-constrained problems. These are problems of the form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to} & 0 \le x_i \le 1, \text{ for } i = 1, \ldots, n. \end{array}$$

3. Simplex-constrained problems. This is a problem of the form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to} & x_1 + \cdots + x_n = 1, \\ & x_i \ge 0, \text{ for } i = 1, \ldots, n. \end{array}$$

Pardalos *et al.* (1989) observe that a theorem of Motzkin and Straus (1965) proves that this problem is **NP**-hard.

4. Problems with one negative eigenvalue. Convex quadratic programming is solvable in polynomial time with the ellipsoid method or an interior point method, as described in Section 2. As mentioned above, a quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$ is convex if H is positive semidefinite, i.e., if all the eigenvalues of H are nonnegative. It is interesting to ask what happens if precisely one eigenvalue of H is negative, while the remaining eigenvalues are nonnegative. It turns out, as proved by Pardalos and Vavasis (1991), that such a problem is **NP**-hard.

An interesting open special case is products of linear functions: It is not known whether instances of QP of the form

$$\begin{aligned} \text{minimize } & (c^T\mathbf{x} + \gamma)(\mathbf{d}^T\mathbf{x} + \delta) \\ \text{subject to } & A\mathbf{x} \geq \mathbf{b} \end{aligned}$$

are solvable in polynomial time.

Since quadratic programming is **NP**-hard, problems more general are expected to be even harder. One generalization of quadratic programming is polynomial programming:

$$\begin{aligned} \text{minimize } & q(x_1, \ldots, x_n) \\ \text{subject to } & p_1(x_1, \ldots, x_n) \geq 0 \\ & \vdots \\ & p_m(x_1, \ldots, x_n) \geq 0 \end{aligned}$$

where $q, p_1, \ldots, p_m$ are polynomial functions of n variables.

This problem is clearly **NP**-hard since it generalizes quadratic programming. When posed as a decision problem, it lies in a complexity class called **PSPACE**; this class contains **NP**. It is not known whether the above problem lies in **NP**. On the other hand it is also not known whether the above problem is **PSPACE**-complete. If it were **PSPACE**-complete, this would be strong evidence that it is not in **NP**.

Another way to generalize quadratic programming is with a "black-box" model. This is a model of computation in which we assume that f, the objective function, is provided as an external subroutine (rather than via numerical data as in quadratic programming). The subroutine takes $\mathbf{x}$ and returns $f(\mathbf{x})$. Thus, global information about f is not available to the optimization algorithm. In general, for this model one assumes that algorithms can compute with real numbers (as opposed to Turing machines, in which all computations are done on symbols drawn from a finite list). It is also common to assume that derivatives of f are available as black-box subroutines. For problems of this model, sometimes the constraints are also expected to be black-boxes, and sometimes the constraints are specified numerically.

In the black-box model it is generally not possible to compute an exact global minimum. Therefore, most theorems pertaining to this model refer to computing an approximation to the global minimum.

The ellipsoid algorithm is an example of an algorithm that can work in this model. This is because the ellipsoid algorithm needs only function values and gradient values for the objective function.

The ellipsoid algorithm works, however, only when f is convex. For nonconvex f there are no efficient algorithms in the black-box model. Unlike the quadratic programming case, where no proof of intractability is known, it is possible in the black-box model to demonstrate true exponential lower bounds on the complexity.

For the sake of definiteness, we assume that the feasible set is $D = [0,1]^n$; this will allow us to focus on the objective function as the source of difficulty.

A naive algorithm for (approximately) globally minimizing a general function f over D would be: (1) insert a fine mesh of points, (2) evaluate f at every point, and (3) output the meshpoint with the smallest value of f. The intractability result below says essentially that in the worst case, there is no better algorithm than this one. The following theorem is a special case of a theorem from Nemirovsky and Yudin (1983).

Theorem 4 *Let $F(k,p)$ be the class of k-times differentiable functions on D whose kth derivative is bounded by p in the following sense: At any point $\mathbf{x} \in D$ and for any unit vector $\mathbf{u}$,*

$$\left| \frac{d^k}{dt^k} f(\mathbf{x} + t\mathbf{u}) \right| \leq p.$$

Let A be any minimization algorithm that works in the black-box model (evaluating f and its derivatives. Assume that for any function in $F(k,p)$, A is guaranteed to output an $\mathbf{x}$ such that $f(\mathbf{x}) - f(\mathbf{x}^*) \leq \epsilon$. (Here $\mathbf{x}^*$ denotes the global minimum.) Then there is a function $f \in F(k,p)$ such that algorithm A will run for at least

$$c_{n,k} \cdot \left(\frac{p}{\epsilon}\right)^{n/k}$$

steps on f.

For example, in the case that $k = 1$, this theorem states for a differentiable function whose derivative is at most p, the number of steps required is at least a constant multiplied by $(p/\epsilon)^n$. There is an obvious algorithm to achieve this bound, namely insert a mesh of points spaced ϵ/p in each dimension. The above theorem applies also to randomized algorithms under suitable definitions.

Compare the bound in this theorem to the bound for the ellipsoid algorithm: For convex functions, the running time is polynomial in n and polynomial in $\ln(p/\epsilon)$. For nonconvex functions the running time is exponential in n and exponential in $\ln(p/\epsilon)$. Note also that smoothness in f is not much help; smoothness slightly ameliorates the exponential dependence on n but does not change the problem to polynomial.

5. Approximation algorithms in quadratic programming

In the last section we discussed approximate solutions for the black-box model. It is reasonable to inquire into approximate solutions for quadratic programming as well. For combinatorial **NP**-hard problems, there is a well-developed theory of approximate solutions. The definition of approximate solution needs to be revised somewhat (compared to the common definition in the combinatorial literature) in order to work on nonlinear programming.

Consider the problem of minimizing f on a compact set $D \subset \mathbb{R}^n$. We propose the following definition for an *ϵ-approximate solution.* Let p be the difference between the maximum and minimum values of f on D. Then we say that $\mathbf{x}^\diamond$ is an ϵ-approximate solution to the minimization problem if

$$f(\mathbf{x}^\diamond) - f(\mathbf{x}^*) \leq \epsilon p.$$

This definition is sensible only for $\epsilon \in [0, 1]$; if $\epsilon = 0$ then $\mathbf{x}^\diamond$ is a global minimum, and if $\epsilon = 1$ then any feasible point satisfies this condition.

This definition has the desirable properties that it is invariant under translations and scalings of the objective function, and under transformations of the feasible set.

Vavasis (1992a) uses a certain kind of covering and partitioning algorithm to establish the following theorem:

Theorem 5 *Consider the optimization problem of minimizing $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{b}$. Assume that the feasible region $\{\mathbf{x} : A\mathbf{x} \geq \mathbf{b}\}$ is compact. Let t be the number of negative eigenvalues of H. There is an algorithm to find an ϵ-approximate solution to this problem in*

$$O\left(\left\lceil \frac{n(n+1)}{\sqrt{\epsilon}} \right\rceil^t \ell\right)$$

steps. In this formula, ℓ denotes the time to solve a convex quadratic programming problem of the same size as the original problem.

Thus, a good approximate solution may be found efficiently if the number of negative eigenvalues of the quadratic function is not too large.

Unfortunately, for general QP the number of negative eigenvalues could be as large as n, in which case the running-time bound in the preceding theorem grows exponentially fast. Can we hope to get an efficient algorithm for approximating f for general QP? Recent results suggest that such an algorithm, if it exists, could only satisfy weak approximation bounds. Specifically, Bellare and Rogaway (1992) show the following theorem is true for some constant $\delta > 0$.

Theorem 6 *Suppose there were an algorithm to approximate quadratic programming with $\epsilon = \left(2^{(\ln n)^\delta} - 1\right) / \left(2^{(\ln n)^\delta} + 1\right)$. Then any problem in* **NP** *could be solved in quasi-polynomial time, that is, time $O(n^{(\ln n)^k})$.*

This theorem is based partly on complexity results by Feige *et al.* (1991). Since the concluding statement of the theorem is thought to be unlikely, the supposition is probably false. In other words, we cannot hope to approximate QP in polynomial time unless we are willing to accept an approximation factor that tends to 1 asymptotically as the problem gets larger.

Indeed, it is possible to construct a weak approximate solution in polynomial time, a result due to Vavasis (1992b):

Theorem 7 *There is a polynomial time algorithm to compute a $1 - cn^{-2}$ approximate solution for quadratic programming, where c is a constant.*

The algorithm is based on computing a sphere lying inside the polytope, and then minimizing the objective function over the sphere using the Ye-Karmarkar algorithm.

A stronger approximate solution can be constructed for special cases of quadratic programming. Here is a result due to Vavasis (1992a):

Theorem 8 *For the nonconvex quadratic knapsack problem (see Section 4 for a description of this problem) an ϵ-approximate solution can be computed in time polynomial in n, $1/\epsilon$, and the problem size (in digits) for any $\epsilon \in (0,1)$.*

The algorithm for this theorem is based on *dynamic programming*; its analysis is quite lengthy.

6. Conclusions

A major theme of this chapter is the difference between convex and nonconvex problems: highly effective algorithms exist for convex global minimization. For nonconvex problems, however, no asymptotically efficient algorithms are known. In practice, only nonconvex problems with a small number of variables can be solved. Indeed the correlation between nonconvexity and intractability is quite strongly exhibited by Theorem 5—the running time grows exponentially with t, the degree of nonconvexity.

References

M. Bellare and P. Rogaway. The complexity of approximating a nonlinear program. preprint, 1992.

S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

J. E. Dennis and R. E. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice–Hall, Englewood Cliffs, New Jersey, 1983.

U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Symposium on Foundations of Computer Science*, pages 2–12, 1991.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.

D. M. Gay. Computing optimal locally constrained steps. Technical Report 2013, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1979.

J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Trans. A.M.S.*, 117:285–306, 1965.

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

N. Karmarkar. An interior-point approach to NP-complete problems. preprint, 1989.

R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

L. G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244:1093–1086, 1979. translated in *Soviet Math. Dokl.* 20:191–194.

T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17:533–540, 1965.

A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley and Sons, Chichester, 1983. Translated by E. R. Dawson from *Slozhnost' Zadach i Effektivnost' Metodov Optimizatsii*, 1979, Glavnaya redaktsiya fiziko-matematicheskoi literatury, Izdatelstva "Nauka".

Y. E. Nesterov and A. S. Nemirovsky. Self-concordant functions and polynomial-time methods in convex programming. Book–Preprint, Central Economic and Mathematical Institute, USSR Academy of Science, Moscow, Russia, 1989.

P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *J. Global Optimiz.*, 1:15–22, 1991.

P. M. Pardalos, Y. Ye, and C.-G. Han. Algorithms for the solution of quadratic knapsack problems. preprint, 1989.

S. Sahni. Computationally related problems. *SIAM J. Comp.*, 3:262–279, 1974.

S. Schaible and T. Ibaraki. Fractional programming. *European Journal of Operational Research*, 12:325–338, 1983.

D. C. Sorenson. Newton's method with a model trust region modification. *SIAM J. Numer. Anal.*, 19:409–426, 1982.

S. A. Vavasis and R. Zippel. Proving polynomial-time for sphere-constrained quadratic programming. Technical Report 90-1182, Department of Computer Science, Cornell University, Ithaca, New York, 1990.

S. A. Vavasis. Quadratic programming is in NP. *Info. Proc. Lett*, 36:73–77, 1990.

S. A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.

S. A. Vavasis. Approximation algorithms for indefinite quadratic programming. *Mathematical Programming*, 1992. to appear.

S. A. Vavasis. Polynomial time weak approximation algorithms for quadratic programming. preprint, 1992.

Y. Ye. On the interior algorithms for nonconvex quadratic programming. preprint, 1988.

Y. Ye. A new complexity result on minimization of a quadratic function with a sphere constraint. In C. A. Floudas and P. M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 19–31. Princeton University Press, Princeton, New Jersey, 1992.

D. B. Yudin and A. S. Nemirovsky. Informational complexity and efficient methods for solving complex extremal problems. *Ekonomika i Matematicheskie Metody*, 12:357–369, 1976. Translated in *Matekon: Translations of Russian and East European Math. Economics* 13:25–45, Spring 1977.