Complexity of optimizing over the integers

Amitabh Basu[*]

Tuesday 5th July, 2022

Abstract

In the first part of this paper, we present a unified framework for analyzing the algorithmic complexity of any optimization problem, whether it be continuous or discrete in nature. This helps to formalize notions like "input", "size" and "complexity" in the context of general mathematical optimization, avoiding context dependent definitions which is one of the sources of difference in the treatment of complexity within continuous and discrete optimization. In the second part of the paper, we employ the language developed in the first part to study information theoretic and algorithmic complexity of *mixed-integer convex optimization*, which contains as a special case continuous convex optimization on the one hand and pure integer optimization on the other. We strive for the maximum possible generality in our exposition.

We hope that this paper contains material that both continuous optimizers and discrete optimizers find new and interesting, even though almost all of the material presented is common knowledge in one or the other community. We see the main merit of this paper as bringing together all of this information under one unifying umbrella with the hope that this will act as yet another catalyst for more interaction across the continuous-discrete divide. In fact, our motivation behind Part I of the paper is to provide a common language for both communities.

Part I
A general framework for complexity in optimization

1 The setup

This paper deals with theoretical complexity analyses for optimization problems where some or all decision variables are constrained to take integer values. This means that we will look at provable upper and lower bounds on the efficiency of algorithms that solve these problems. We will consider the standard Turing machine model of computation, and we will study algorithms that receive an optimization problem as "input" and we wish to study the efficiency of the algorithm as a function of the "size" of the problem. Moreover, we wish to develop a framework that can seamlessly handle discrete optimization and continuous optimization. In particular, the focus of this paper will be on so-called *mixed-integer convex optimization*, which includes as special cases continuous, nonlinear convex optimization on the one hand and integer linear optimization on the other which can model problems with a combinatorial and discrete nature. Therefore, it is imperative to have a general setup that can formally make sense of "writing down" an optimization problem as "input" to an algorithm and the related "size" of the optimization problem, no matter whether it is combinatorial or numerical in nature.

For instance, most optimization problems with a combinatorial or discrete nature have a well defined, universally accepted notion of "encoding" and therefore of "size" (such as the Matching Problem or Traveling

[*]Department of Applied Mathematics and Statistics, Johns Hopkins University, `basu.amitabh@jhu.edu`. The author gratefully acknowledges support from Air Force Office of Scientific Research (AFOSR) grant FA95502010341 and National Science Foundation (NSF) grants CCF2006587.

Salesperson Problem (TSP) on graphs, or linear programming (LP) over the rationals). Things are more tricky with continuous optimization problems. What is the "size" of a general optimization problem of the form $\min\{f(x) : g_i(x) \leq 0, \;\; i = 1, \ldots, m\}$, where I and g_i are smooth, nonlinear functions defined on $\mathbb{R}^n$? If these functions have a particular algebraic form (e.g., polynomial optimization), then there is usually no controversy because one considers the encoding of the coefficients appearing in the polynomials. What if the functions are given via evaluation and gradient oracles?

In fact, the question can be raised whether the Turing machine model is even appropriate in this setting, because it only allows for computation over a finite set of symbols and numerical/scientific computation problems may require a more general model of computation. Several proposals have been put forward to address this issue; see [1, 24, 27, 70, 94, 101, 116, 126] as a representative list. It turns out that all of the discussion in this Part I of this paper will hold true in any of these different models of computation. For concreteness, we will assume a computational framework that works within the standard framework of the Turing machine model of computation augmented with appropriate oracles [94, 101] and has its roots in the constructive philosophy of mathematics [17, 21]. We include the formal definition here for completeness of this article and the reader is referred to [94, 101] for a fuller discussion.

Definition 1.1. An oracle implementing a real number $\alpha \in \mathbb{R}$ takes as input a rational number $\epsilon > 0$ and outputs a rational number r such that $|r - \alpha| \leq \epsilon$. Moreover, there exists a constant k (independent of ϵ) such that the bit encoding size of r is at most k times the encoding size of ϵ. The *size of the oracle* is k.

If the real number α implemented by an oracle is rational, then a second constant k' is added to the size of the oracle with the guarantee that the bit encoding size of α is at most k'.

A study of general optimization methods in the 70s and 80s led to the insight that all such procedures (whether combinatorial or analytical in nature) can be understood in a unified framework which we present now. Our exposition is a summary of ideas put forth in several works and giving a complete survey of this literature is difficult. With apologies for our omissions, we list two references that we personally found to be most illuminating [108, 125].

The overall idea, roughly speaking, is that one gathers *information* about an optimization problem which tells the optimizer which instance needs to be solved within a problem class, and then computations are performed on the information gathered to arrive at a solution (possibly approximate, with guaranteed bounds on error). Let us formalize this idea in a way that encompasses both discrete and continuous optimization [125].

Definition 1.2. [General optimization problem] An *optimization problem class* is given by a set $\mathcal{I}$ of *instances*, a set G of *possible solutions*, and a *solution operator*

$$S : \mathcal{I} \times \mathbb{R}_+ \to 2^G \cup \{INFEAS\} \cup \{UNBND\},$$

where 2^G denotes the power set of G and the operator S satisfies three properties:

1. $S(I, 0) \neq \emptyset$ for all $I \in \mathcal{I}$, and
2. For any two nonnegative real numbers $\epsilon_1 < \epsilon_2$ and any $I \in \mathcal{I}$, we have $S(I, \epsilon_1) \subseteq S(I, \epsilon_2)$.
3. If $INFEAS \in S(I, \epsilon)$ (or $UNBND \in S(I, \epsilon)$) for some $I \in \mathcal{I}$ and $\epsilon \geq 0$, then $S(I, \epsilon) = \{INFEAS\}$ (or $S(I, \epsilon) = \{UNBND\}$ respectively).
4. If $UNBND \in S(I, 0)$ for some $I \in \mathcal{I}$, then $S(I, \epsilon) = \{UNBND\}$ for all $\epsilon \geq 0$.

The interpretation of the above definition is simple: $\mathcal{I}$ is the set of optimization problem instances we wish to study, G is the space of solutions to the problems, and for any instance $I \in \mathcal{I}$ and $\epsilon \geq 0$, $S(I, \epsilon)$ is the set of ϵ-approximate solutions with the understanding that $S(I, \epsilon) = \{INFEAS\}$ encodes the fact that I is an infeasible instance (with respect to ϵ error; see examples below) and $S(I, \epsilon) = UNBND$ encodes the fact that the objective value for I is unbounded. The advantage of this definition is that there is no need to assume any structure in the set G; for example, it could be some Euclidean space, or it could just as well be some combinatorial set like the set of edges in a graph. Linear Programming in $\mathbb{R}^n$ would set $G = \mathbb{R}^n$ while the Traveling Salesperson Problem on n cities corresponds to setting G to be all tours in the complete graph K_n. It is also not hard to encode optimization problems in varying "dimensions" in this framework, e.g., G is allowed to be $\bigcup_{n\in\mathbb{N}} \mathbb{R}^n$. Also, the notion of "$\epsilon$-approximate" does not require any kind of norm or distance structure on G. Property 2 simply requires that as we allow more error, we obtain more solutions. Thus, Definition 1.2 captures discrete and continuous optimization in a clean, unified framework while allowing for a very flexible notion of an "ϵ-approximate" solution for $\epsilon > 0$.

Example 1.3. We now present some concrete examples.

1. Traveling Salesperson Problem (TSP). For any natural number $n \in \mathbb{N}$, the (symmetric) traveling salesperson problem for n cities seeks to find a tour of minimum length that visits all cities, given pairwise intercity distances. To model this in the above framework, one defines E_n, for any natural number $n \in \mathbb{N}$, to be the set of all unordered pairs in $\{1, \ldots, n\}$. Let $G = \cup_{n\in\mathbb{N}} 2^{E_n}$, i.e., each element of G is a subset of unordered pairs (these are edges in the tour). $\mathcal{I}$ is the family of all TSPs with given intercity distances. We allow any number n of cities; of course, for a particular instance I there is a fixed number of cities. $S(I, \epsilon)$ can be taken to be the set of all tours in the complete graph K_n that are within an additive ϵ error or within a multiplicative $(1 + \epsilon)$ factor of the optimal tour length in I. $\epsilon = 0$ corresponds to the set of all optimal tours.

   One could also fix a natural number $n \in \mathbb{N}$ and simply consider only the problems on n cities. In this case, $G = 2^{E_n}$ and $\mathcal{I}$ would consist of all possible tours on n cities, i.e., where only the intercity distances are changed, but the number of cities is fixed.

2. Mixed-integer linear programming (MILP).

   - (Fixed dimension) Let $n, d \in \mathbb{N}$ be fixed. $G = \mathbb{R}^n \times \mathbb{R}^d$, $\mathcal{I}$ is the set of all mixed-integer linear programs defined by matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{m\times d}$, and vectors $b \in \mathbb{R}^m, c_1 \in \mathbb{R}^n, c_2 \in \mathbb{R}^d$, where $m \in \mathbb{N}$ can be chosen as any natural number (thus, $\mathcal{I}$ contains all MILPs with any number of constraints for $m = 1, 2, \ldots$, but the total number of variables is fixed):

     $$\max\{c_1^T x + c_2^T y \ : \ Ax + By \le b, \ \ x \in \mathbb{Z}^n, y \in \mathbb{R}^d\}.$$

     $S(I, \epsilon)$ may be defined to be all solutions $(x, y) \in G$ to an MILP instance I such that $c_1^T x + c_2^T y$ is within an additive ϵ error of the optimal value for I. Taking $\epsilon = 0$ would mean we are considering the exact optimal solution(s). Alternatively, one may define $S(I, \epsilon)$ to be the set of $(x, y) \in G$ such that there is an optimal solution to I within ϵ distance to (x, y).

   - (Variable dimension) We can consider the family of all MILPs with a fixed number of integer variables, but allowing for any number of continuous variables. Here $n \in \mathbb{N}$ is fixed and $G = \bigcup_{d\in\mathbb{N}} (\mathbb{R}^n \times \mathbb{R}^d)$. Everything else is defined as above. Similarly, we may also allow the number of integer variables to vary by letting $G = \bigcup_{n\in\mathbb{N}, d\in\mathbb{N}} (\mathbb{R}^n \times \mathbb{R}^d)$.

   Fixing $n = 0$ in the above settings would give us *(pure) linear programming.*

3. Nonlinear Optimization. In a similar fashion as above, we may model nonlinear optimization problems of the form

   $$\min\{f(x) : g_i(x) \le 0 \ i = 1, \ldots, m\}. \tag{1.1}$$

   The class $\mathcal{I}$ may restrict the structure of the objective and constraint functions (e.g., convex, twice continuously differentiable, nonsmooth etc.). As before, $S(I, \epsilon)$ may correspond to all solutions that are within an ϵ error of the true optimal value, or solutions within ϵ distance of the set of optimal solutions, or the set of points where the norm of the gradient is at most ϵ in the unconstrained case, or any other notion of ϵ-approximate solutions commonly studied in the nonlinear optimization literature. One may also allow ϵ slack in the constraints, i.e., $g_i(x) \le \epsilon$ for any $x \in S(I, \epsilon)$.

Next, we discuss the notion of an oracle that permits us to figure out which problem instance we need to solve.

Definition 1.4. An *oracle* for an optimization problem class $\mathcal{I}$ is given by a family $\mathcal{Q}$ of possible *queries* and a set H of possible *answers*. Each query $q \in \mathcal{Q}$ is an operator $q : \mathcal{I} \to H$. We say that $q(I) \in H$ is the answer to the query q on the instance $I \in \mathcal{I}$.

Example 1.5. We now consider some standard oracles for the settings considered in Example 1.3.

1. For the TSP, the typical oracle uses two types of queries. One is the dimension query $q_{\dim}$, which returns the number $q_{\dim}(I)$ of cities in the instance I, and the queries $q_{ij}(I)$ which returns the intercity distance between cities i, j (with appropriate error exceptions if i or j are not in the range $\{1, \ldots, q_{\dim}(I)\}$).

2. For MILP, the typical oracle uses the following queries: the dimension queries for n and d (unless one or both of them are fixed and known), a query $q^A_{ij}(I)$ that reports the entry of matrix A in row i and column j for the instance I, and similar queries q^B_{ij}, q^b_i, q^c_j for the matrix B, and vectors b, c (with appropriate error exceptions if the queried index is out of bounds).

3. For Nonlinear Optimization, the most commonly used oracles return function values, gradient/subgradient values, Hessian or higher order derivative values at a queried point. Thus, we have queries such as $q^{f,x}_0(I)$ which returns $f(x)$ for the objective function f in an instance of (1.1) where x is a point in the appropriate domain of f, or the query $q^{f,x}_1(I)$ which returns the gradient $\nabla f(x)$. Similarly, one has queries for the constraints. Often the set version of these oracles are assumed instead, specially in convex optimization, where one is given a separation oracle for the feasible region and the epigraph of the objective function; see [78, 101] for a well developed theory and applications in this setting.

   If the problem class $\mathcal{I}$ has an algebraic form, e.g., polynomial optimization, then the oracle queries may be set up to return the values of the coefficients appearing in the polynomial.

One can seamlessly accommodate oracles with error in the above set-up. For example, the weak separation oracles in [78] can be modeled with no change in the definitions just like strong/exact separation oracles. We will only work with deterministic oracles in this paper. See [28, 108] for a discussion of stochastic oracles in the context of continuous convex optimization. We next recall the notion of an oracle Turing machine.

Definition 1.6. An *oracle Turing machine* with access to an oracle $(\mathcal{Q}, H)$ is a Turing machine that has the enhanced ability to pose any query $q \in \mathcal{Q}$ and use the answer in H in its computations. The queries it poses may depend on its internal states and computations, i.e., it can query adaptively during its processing.

We now have everything in place to define what we mean by an optimization algorithm/procedure. Since we focus on the (oracle) Turing machine model and any algorithm will process elements of the solution set G and set of answers H from an oracle, we assume that the elements of these sets can be represented by binary strings or appropriate real number oracles (see Definition 1.1). If one wishes to adopt a different model of computation, then these sets will need representations within that computing paradigm. This clearly affects what class of problems and oracles are permissible. We will not delve into these subtle questions; rather we will stick to our Turing machine model and assume that the class of problems we are dealing with has appropriate representations for G and H.

Definition 1.7. Let $(\mathcal{I}, G, S)$ be an optimization problem class and let $(\mathcal{Q}, H)$ be an oracle for $\mathcal{I}$. For any $\epsilon \geq 0$, an *ϵ-approximation algorithm for* $(\mathcal{I}, G, S)$ *using* $(\mathcal{Q}, H)$ is an oracle Turing machine with access to $(\mathcal{Q}, H)$ that starts its computations with the empty string as input and, for any $I \in \mathcal{I}$, ends its computation with an element of $S(I, \epsilon)$, when it receives the answer $q(I)$ for any query $q \in \mathcal{Q}$ it poses to the oracle.

If $\mathcal{A}$ is such an ϵ-approximation algorithm, we define the *total complexity* $\mathrm{comp}_{\mathcal{A}}(I)$ to be the number of elementary operations performed by $\mathcal{A}$ during its run on an instance I (meaning that it receives $q(I)$ as the answers to any query q it poses to the oracle), where each oracle query counts as an elementary operation (reading the answer of a query may require more than one elementary operation, depending on its length).

If $\mathcal{A}$ makes queries $q_1, \ldots, q_k$ during its run on an instance I, we say the *information complexity* $\text{icomp}_{\mathcal{A}}(I)$ is $|q_1(I)| + \ldots + |q_k(I)|$, where $|q_i(I)|$ denotes the length of the binary string or size of the real number oracle (see Definition 1.1) representing the answer $q_i(I)$.

Following standard conventions, the *(worst case) complexity* of the algorithm $\mathcal{A}$ for the problem class $(\mathcal{I}, G, S)$ is defined as

$$\text{comp}_{\mathcal{A}} := \sup_{I \in \mathcal{I}} \ \text{comp}_{\mathcal{A}}(I),$$

and the *(worst case) information complexity* is defined as

$$\text{icomp}_{\mathcal{A}} := \sup_{I \in \mathcal{I}} \ \text{icomp}_{\mathcal{A}}(I),$$

Remark 1.8. One can assume that any algorithm that poses a query q reads the entire answer $q(I)$ for an instance I. Indeed, if it ignores some part of the answer, then one can instead consider the queries that simply probe the corresponding bits that are used by the algorithm (and we may assume our oracles to have this completeness property). We will assume this to be the case in the rest of the paper. Such an assumption can also be made without loss of generality in any other reasonable model of computation.

This implies that the information complexity $\text{icomp}_{\mathcal{A}}(I)$ is less than or equal to the total complexity $\text{comp}_{\mathcal{A}}(I)$ of any algorithm $\mathcal{A}$ running on any instance I.

An important notion of complexity that we will not discuss in depth in this paper is that of *space complexity*. This is defined as the maximum amount of information (from the oracle queries) and auxiliary computational memory that is maintained by the algorithm (oracle Turing machine) during its entire run. As in Definition 1.7, one can define the *total space complexity* and the *information space complexity*. Both notions can be quite different from $\text{comp}_{\mathcal{A}}$ and $\text{icomp}_{\mathcal{A}}$ respectively, since one keeps track of only the amount of information and auxiliary data held in memory at any given stage of the computation, as opposed to the overall amount of information or auxiliary memory used. In many optimization algorithms, it is not necessary to maintain all of the answers to previous queries or computations in memory. A classic example of this is the (sub)gradient descent algorithm where only the current function values and (sub)gradients are stored in memory for computations, and they are not needed in subsequent iterations.

2 Oracle ambiguity and lower bounds on complexity

For many settings, especially problems in numerical optimization, a finite number of oracle queries may not pin down the exact problem one is facing. For example, consider $(\mathcal{I}, G, S)$ to be the problem class of the form (1.1) where $f, g_1, \ldots, g_m$ can be any convex, continuously differentiable functions, and suppose the oracle $(\mathcal{Q}, H)$ allows function evaluation and gradient queries. Given any finite number of queries, there are infinitely many instances that give the same answers to those queries.

Definition 2.1. Let $(\mathcal{I}, G, S)$ be an optimization problem class and let $(\mathcal{Q}, H)$ be an oracle for $\mathcal{I}$. For any subset $Q \subseteq \mathcal{Q}$ of queries, define an equivalence relation on $\mathcal{I}$ as follows: $I \sim_Q I'$ if $q(I) = q(I')$ for all $q \in Q$. For any instance I, let $V(I, Q)$ denote the equivalence class that I falls in, i.e.,

$$V(I, Q) = \{I' : q(I) = q(I') \ \ \forall q \in Q\}.$$

The above definition formalizes the fact that if one only knows the answers to queries in Q, then one has a course-grained view of $\mathcal{I}$. This is why the notion of an ϵ-approximate solution becomes especially pertinent. The following theorem gives a necessary condition on the nature of queries used by any ϵ-approximation algorithm.

Theorem 2.2. Let $(\mathcal{I}, G, S)$ be an optimization problem class and let $(\mathcal{Q}, H)$ be an oracle for $\mathcal{I}$. If $\mathcal{A}$ is an ϵ-approximation algorithm for this optimization problem for some $\epsilon \geq 0$, then

$$\bigcap_{I' \in V(I, Q(I))} S(I', \epsilon) \neq \emptyset \qquad \forall I \in \mathcal{I},$$

where $Q(I)$ is the set of queries used by $\mathcal{A}$ when processing instance I.

Proof. If $\mathcal{A}$ is a correct ϵ-approximation algorithm, then suppose it returns $x \in S(I, \epsilon)$ for instance I. Since the answers to its queries are the same for all $I' \in V(I, Q(I))$, it will also return x when the answers it receives are $q(I')$ for $q \in Q(I)$ and $I' \in V(I, Q(I))$. Therefore, $x \in S(I', \epsilon)$ for all $I' \in V(I, Q(I))$. $\square$

This leads us to the following definition.

Definition 2.3. Let $(\mathcal{I}, G, S)$ be an optimization problem class and let $(\mathcal{Q}, H)$ be an oracle for $\mathcal{I}$. Let $2^{(\mathcal{Q}\times H)}$ denote the collection of all *finite* sets of pairs $(q, h) \in \mathcal{Q} \times H$.

An *adaptive query strategy* is a function $D : 2^{(\mathcal{Q}\times H)} \to \mathcal{Q}$. The *transcript* $\Pi(D, I)$ *of a strategy* D *on an instance* I is the sequence of query and response pairs $(q_i, q_i(I))$, $i = 1, 2, \ldots$ obtained when one applies D on I, i.e., $q_1 = D(\emptyset)$ and $q_i = D(\{(q_1, q_1(I)), \ldots, (q_{i-1}, q_{i-1}(I))\})$ for $i \geq 2$. $\Pi_k(D, I)$ will denote the truncation of $\Pi(D, I)$ to the first k terms, $k \in \mathbb{N}$. We will use $Q(D, I)$ (and $Q_k(D, I)$) to denote the set of queries in the transcript $\Pi(D, I)$ (and $\Pi_k(D, I)$). Similarly, $R(D, I)$ (and $R_k(D, I)$) will denote the set of responses in the transcript.
The ϵ-*information complexity of an instance* I *for an adaptive strategy* D is defined as

$$\begin{aligned} \operatorname{icomp}_\epsilon(D, I) &:= \inf\left\{\textstyle\sum_{r\in R_k(D,I)} |r| : k \in \mathbb{N} \text{ such that } \bigcap_{I'\in V(I, Q_k(D,I))} S(I', \epsilon) \neq \emptyset\right\} \\ &= \inf\left\{\textstyle\sum_{r\in R_k(D,I)} |r| : k \in \mathbb{N} \text{ such that } \bigcap_{I':\Pi_k(D,I')=\Pi_k(D,I)} S(I', \epsilon) \neq \emptyset\right\} \end{aligned}$$

The ϵ-*information complexity of an adaptive strategy* D *for the problem class* $(\mathcal{I}, G, S)$ is defined as

$$\operatorname{icomp}_\epsilon(D) := \sup_{I\in\mathcal{I}} \ \operatorname{icomp}_\epsilon(D, I)$$

The ϵ-*information complexity of the problem class* $(\mathcal{I}, G, S)$ *with respect to oracle* $(\mathcal{Q}, H)$ is defined as

$$\operatorname{icomp}_\epsilon := \inf_D \ \operatorname{icomp}_\epsilon(D),$$

where the infimum is over all possible adaptive queries.

Remark 2.4. In the definitions above, one could restrict adaptive query strategies to be computable, or even polynomial time computable (in the size of the previous query-response pairs). We are not aware of any existing research where such restrictions have been studied to get a more refined analysis of ϵ-information complexity. The typical lower bounding techniques directly lower bound $\operatorname{icomp}_\epsilon$ defined above. One advantage of this is that one does not have to rely on any complexity theory assumptions such as $P \neq NP$ and the lower bounds are unconditional.

We can now formally state the results for lower bounding algorithmic complexity. Remark 1.8 and Theorem 2.2 imply the following.

Corollary 2.5. Let $(\mathcal{I}, G, S)$ be an optimization problem class and let $(\mathcal{Q}, H)$ be an oracle for $\mathcal{I}$. If $\mathcal{A}$ is an ϵ-approximation algorithm for $(\mathcal{I}, G, S)$ using $(\mathcal{Q}, H)$ for some $\epsilon \geq 0$, then

$$\operatorname{icomp}_\epsilon \leq \operatorname{icomp}_{\mathcal{A}} \leq \operatorname{comp}_{\mathcal{A}}.$$

Remark 2.6. If S, S' are two different solution operators for $\mathcal{I}, G$ such that $S(I, \epsilon) \subseteq S'(I, \epsilon)$ for all $I \in \mathcal{I}$ and $\epsilon \geq 0$, i.e., the operator S is stricter than S', then the complexity measures with respect to S are at least as large as the corresponding measures with respect to S'.

Remark 2.7. In the literature, $\operatorname{icomp}_\epsilon$ is often referred to as the *analytical complexity* of the problem class (see, e.g., [109]). We prefer the phrase *information complexity* since we wish to have a unified framework for continuous and discrete optimization and "analytical" suggests problems that are numerical in nature or involve the continuum. Another term that is used in the literature is *oracle complexity*. This is better, in our opinion, but still has the possibility to suggest the complexity of implementing the oracle, rather than the complexity of the queries. Since $\operatorname{icomp}_\epsilon$ is very much inspired by information theory ideas, we follow the trend [28, 31, 107, 125, 129] of using the term *information complexity (with respect to an oracle)*.

$\operatorname{comp}_{\mathcal{A}}$ is sometimes referred to as *arithmetic complexity* [109] or *combinatorial complexity* [125] of $\mathcal{A}$. We prefer to stick to the more standard terminology of simply *(worst case) complexity* of the algorithm $\mathcal{A}$.

3 What is the size of an optimization problem?

3.1 Size hierarchies

The notions of complexity defined so far are either too fine or too course. At one extreme is the instance dependent notions $\mathrm{comp}_{\mathcal{A}}(I)$, $\mathrm{icomp}_{\mathcal{A}}(I)$ and $\mathrm{icomp}_{\epsilon}(D, I)$, and at the other extreme are the worst case notions $\mathrm{comp}_{\mathcal{A}}$, $\mathrm{icomp}_{\mathcal{A}}$ and $\mathrm{icomp}_{\epsilon}$. It is almost always impossible give a fine tuned analysis of the instance based complexity notions; on the other hand, the worst case notions give too little information, at best as a function of ϵ, and in the worst case, these values are actually ∞ for most problem classes of interest. Typically, a middle path is taken where a countable hierarchy of the problem class is defined and the complexity is analyzed as a function of the levels in the hierarchy.

Definition 3.1. Let $(\mathcal{I}, G, S)$ be an optimization problem class. A *size hierarchy* is a countable, increasing sequence $\mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \mathcal{I}_3 \subseteq \dots$ of subsets of $\mathcal{I}$ such that $\mathcal{I} = \bigcup_{k\in\mathbb{N}} \mathcal{I}_k$. The *size of any instance I with respect to a size hierarchy* is the smallest $k \in \mathbb{N}$ such that $I \in \mathcal{I}_k$.

The (worst case) complexity of any algorithm $\mathcal{A}$ for the problem, with respect to the size hierarchy, is defined naturally as

$$\mathrm{comp}_{\mathcal{A}}(k) := \sup_{I\in\mathcal{I}_k} \mathrm{comp}_{\mathcal{A}}(I).$$

Similarly, the (worst case) information complexity of any algorithm $\mathcal{A}$ for the problem, with respect to the size hierarchy, is defined as

$$\mathrm{icomp}_{\mathcal{A}}(k) := \sup_{I\in\mathcal{I}_k} \mathrm{icomp}_{\mathcal{A}}(I),$$

and the (worst case) ϵ-information complexity of the problem class, with respect to the size hierarchy, is defined as

$$\mathrm{icomp}_{\epsilon}(k) := \inf_D \ \sup_{I\in\mathcal{I}_k} \ \mathrm{icomp}_{\epsilon}(D, I),$$

where the infimum is taken over all adaptive strategies D.

Example 3.2. We review the standard size hierarchies for the problems considered in Example 1.3.

1. In the TSP problem class defined in Example 1.3, the standard "binary encoding" size hierarchy defines $\mathcal{I}_k$ to be all instances such that $\sum_{i,j=1}^{n} \lceil \log(d_{ij}) \rceil \le k$ (so k must be at least n^2), where $d_{ij} \in \mathbb{Z}_+$ are the intercity distances. If one works with real numbers as distances, the size is defined using the sizes of the real number oracles (see Definition 1.1). If one focuses on the so-called *Euclidean TSP* instances, then one can define a different size hierarchy based on the number of bits needed to encode the coordinates of the cities (or sizes of the real number oracles).

   Another alternative is to simply define $\mathcal{I}_k$ to be all instances with at most k cities.

2. For MILPs, the standard "binary encoding" size hierarchy defines $\mathcal{I}_k$ to be all instances such that the total number of bits (or sizes of real number oracles) needed to encode all the entries of A, B, b, c is at most k. Another alternative is to simply define $\mathcal{I}_k$ as those instances where $m(n + d) \le k$, or even simply those instances with $n + d \le k$.

3. For nonlinear optimization problems of the form (1.1), often the notion of "binary encoding" is not meaningful. Consider, for example, the problem of minimizing a linear function $f(x) = c^T x$ over a full-dimensional, compact convex body C given via a separation oracle. A size hierarchy that has been commonly used in this setting defines $\mathcal{I}_k$ as follows: An instance $I \in \mathcal{I}_k$ if there exist rational numbers R, r such that C is contained in the ball of radius R around the origin, C also contains a ball of radius r inside it (the center may not be the origin), and the total number of bits needed to encode R, r and the coordinates of c is at most k. See [78, 101] for a fuller discussion and other variants.

The idea of a size hierarchy is meant to formalize the notion that problems with larger size are "harder" to solve in the sense that it should take an algorithm longer to solve them. This obviously is often a subjective matter, and as discussed in the above examples, different size hierarchies may be defined for the same optimization problem class. The complexity measures as a function of size is very much dependent on this choice (and also on the model of computation because different models measure complexity in ways that are different from the oracle Turing machine model).

Even within the Turing machine model of computation, a classical example of where different size hierarchies are considered is the optimization problem class of *knapsack problems*. Here, one is given n items with weights $w_1, \ldots, w_n \in \mathbb{Z}_+$ and values $v_1, \ldots, v_n \in \mathbb{Z}_+$, and the goal is to find the subset of items with maximum value with total weight bounded by a given budget $W \in \mathbb{Z}$. The standard "binary encoding" size hierarchy defines the level $\mathcal{I}_k$ to be all problems where $\sum_{i=1}^n (\lceil \log w_i \rceil + \lceil \log v_i \rceil) + \lceil \log W \rceil \leq k$. However, one can also stratify the problems by defining $\mathcal{I}_k$ to be all problems where $\sum_{i=1}^n (\lceil \log w_i \rceil + \lceil \log v_i \rceil) + W \leq k$. The well-known dynamic programming based algorithm has complexity $\text{comp}_{\mathcal{A}}(k)$ which is exponential in k with respect to the first size hierarchy, while it is polynomial in k with respect to the second size hierarchy.

The standard "binary encoding" size hierarchies are the most commonly used ones, motivated by the fact that one needs these many bits to "write down the problem" for the Turing machine to solve. As we see above, if one develops a unified theory for discrete and continuous optimization based on oracle Turing machines (or other more flexible models of computation), the "binary encoding" idea loses some of its appeal. And even within the realm of discrete optimization on conventional Turing machines, there is no absolute objective/mathematical principle that dictates the choice of a size hierarchy and, in our opinion, there is subjectivity in this choice. Therefore, complexity measures as a function of the size hierarchy have this subjectivity inherent in them.

3.2 More fine-grained parameterizations

The approach of a size hierarchy as discussed in the previous section parameterizes the instances in a one-dimensional way using the natural numbers. One may choose to stratify instances using more than one parameter and define the complexity measures as functions of these parameters. This is especially useful in continuous, numerical optimization settings where the standard "binary encoding" is unavailable to define a canonical size hierarchy as in the case of discrete optimization problems. For example, consider the problems of the form (1.1) where the functions $f, g_1, \ldots, g_m$ are convex, with the solution operator $S(I, \epsilon)$ consisting of those solutions that satisfy the constraints up to ϵ slack, i.e., $g_i(x) \leq \epsilon$ and have objective value $f(x)$ within ϵ of the optimal value. We also consider access to a first-order oracle for these functions (see Example 1.5, part 3.). We now create a semi-smooth parameterization of the family of instances using three parameters $d \in \mathbb{N}$ and $R, M \in \mathbb{R}$: $\mathcal{I}_{d,R,M}$ are those instances such that 1) the domains of the functions is $\mathbb{R}^d$, 2) the feasible region $\{x \in \mathbb{R}^d : g_i(x) \leq 0 \;\; i = 1, \ldots, m\}$ is contained in the box $\{x \in \mathbb{R}^d : \|x\|_\infty \leq R\}$, and 3) $f, g_1, \ldots, g_m$ are Lipschitz continuous with Lipschitz constant M on this box (a convex function is Lipschitz continuous on any compact set). One can then define the complexity measures $\text{comp}_{\mathcal{A}}(d, R, M)$ and $\text{icomp}_{\mathcal{A}}(d, R, M)$ for any ϵ-approximation algorithm $\mathcal{A}$, and the algorithm independent complexity measure $\text{icomp}_\epsilon(d, R, M)$, as functions of these three parameters (as well as ϵ, of course). As an example one can show $\text{icomp}_\epsilon(d, M, R) \in \Theta(d \log(\frac{MR}{\epsilon}))$; see Section 4.

As in the case of size hierarchies, the goal is to tread a middle path between the two extremes of very fine-grained instance dependent measures, or worst case values over all instances (as a function of ϵ). Parameterizing the problem class with more than one parameter gives a little more information. Several examples of such parameterizations for classes of convex optimization problems is presented in [108]. Our discussion in the next part will involve similar parameterizations of mixed-integer optimization problems. See also the related area of computational complexity theory of *parameterized complexity* and *fixed-parameter tractability (FPT)* [63].

Part II
Complexity of mixed-integer convex optimization

After setting up the framework in Part I, we now derive concrete results for the class of mixed-integer convex optimization problems. More precisely, we will consider problems of the form

$$\inf\{f(x,y) : (x,y) \in C, (x,y) \in \mathbb{Z}^n \times \mathbb{R}^d\}. \tag{3.1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is a convex (possibly nonsmooth) function and $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$ is a closed, convex set, i.e., an instance I is given by f, C. We will consider the solution operator $S(I,\epsilon)$ to be all feasible solutions in C that have value at most ϵ more than the optimal value. One could allow solutions within ϵ distance of C and all of the results given below can be modified accordingly, but we will consider only truly feasible solutions.

The following definition will be useful in what follows.

Definition 3.3. A *fiber box* in $\mathbb{Z}^n \times \mathbb{R}^d$ is a set of the form $\{x\} \times [\ell_1, u_1] \times \ldots [\ell_d, u_d]$ where $x \in \mathbb{Z}^n$ and $\ell_i, u_i \in \mathbb{R}$ for $i = 1, \ldots, d$. The *length* of the box in coordinate j is $u_j - \ell_j$. The *width* of such a fiber box is the minimum of $u_j - \ell_j$, $j = 1, \ldots, d$. If $n = 0$, a fiber box is simply a hypercuboid in $\mathbb{R}^d$. A fiber box is the empty set if $u_i < \ell_i$ for some $i = 1, \ldots, d$. If all $\ell_i = -\infty$ and all $u_i = \infty$, then the set is simply called a *fiber* over x.

4 ϵ-information complexity

In this section, we establish the best-known lower and upper bounds on the ϵ-information complexity of (3.1) in the literature. To get the tightest bounds, we will restrict our attention to problems with bounded feasible regions and therefore a minimum solution exists. Moreover, we will also focus on "strictly feasible" instances.

Definition 4.1. We parameterize the instances using five parameters $n, d \in \mathbb{N}$ and $R, M, \rho \in \mathbb{R}$. $\mathcal{I}_{n,d,R,M,\rho}$ are those instances such that

1. The domain of f and C are both subsets of $\mathbb{R}^n \times \mathbb{R}^d$.
2. C is contained in the box $\{z \in \mathbb{R}^n \times \mathbb{R}^d : \|z\|_\infty \leq R\}$, and
3. f is Lipschitz continuous with respect to the $\|\cdot\|_\infty$-norm with Lipschitz constant M on any fiber box of the form $\{x\} \times [-R,R]^d$ with $x \in [-R,R]^n \cap \mathbb{Z}^n$, i.e., for any $(x,y),(x,y')$ with $\|y-y'\|_\infty \leq R$, $|f(x,y) - f(x,y')| \leq M\|y-y'\|_\infty$.
4. If $(x^\star, y^\star)$ is the optimum solution, then there exists $\hat{y} \in \mathbb{R}^d$ and $0 < \rho \leq 1$ such that $\{(x^\star, y) : \|y - \hat{y}\|_\infty \leq \rho\} \subseteq C$, i.e., there is a "strictly feasible" point $(x^\star, \hat{y})$ in the same fiber as the optimum $(x^\star, y^\star)$ with a fiber box of width ρ in $\mathbb{R}^d$ (the continuous space) around $(x^\star, \hat{y})$ contained in C. Note that if $d = 0$ (the *pure integer* case), then this requirement becomes vacuous; consequently, the bounds below in Theorem 4.2 for the pure integer case do not involve ρ. Also, the assumption $\rho \leq 1$ is not restrictive in the sense that if the condition is satisfied for some $\rho > 0$, then it is also satisfied for $\min\{\rho, 1\}$. Thus, one could alternatively leave this condition out, and the stated bounds below will be modified by replacing ρ with $\min\{\rho, 1\}$.

Table 1 gives a synopsis.

In the statement of the result, we will ignore the sizes of the subgradients, function values and separating hyperplanes reported in the answers to oracle queries (which is technically included in our definition of icomp_ϵ). Thus, we will give lower and upper bounds on the *number* of oracle queries only. Taking the sizes of the subgradients and real numbers involved in the answers leads to several interesting questions which, to the best of our knowledge, have not been fully worked out in detail. To keep the discussion aligned with the

Parameter	Meaning
n	number of integer variables
d	number of continuous variables
R	Boundedness parameter for the feasible region
ρ	Strict feasibility parameter for the feasible region
M	Lipschitz constant for the objective function

Table 1: Parameters of the problem instance used to state the complexity bounds

focus in the literature, we leave these subtleties out of this presentation. This obviously has implications for space information complexity as well.

The bounds below for the mixed-integer case $n, d \geq 1$ are minor adaptations of arguments that first appeared in [14, 111]. The main difference is that our presentation here uses the more general information theoretic language developed in Part I, whereas the results in [14, 111] were stated for a certain class of algorithms called *cutting plane algorithms* (see Section 5.3).

Theorem 4.2. Let the oracle access to an instance f, C of for (3.1) in $\mathcal{I}_{n,d,R,M,\rho}$ from Definition 4.1 be through a separation oracle for C, and a first-order oracle for f, i.e., one can query the function value and the subdifferential for f at any point. As a quick legend: n

Lower bounds

- If $n, d \geq 1$,

$$\text{icomp}_{\epsilon}(n, d, R, M, \rho) \in \Omega\left(d2^n \log\left(\frac{R}{\rho}\right)\right).$$

- If $d = 0$,

$$\text{icomp}_{\epsilon}(n, d, R, M, \rho) \in \Omega\left(2^n \log\left(R\right)\right).$$

- If $n = 0$,

$$\text{icomp}_{\epsilon}(n, d, R, M, \rho) \in \Omega\left(d \log\left(\frac{MR}{\rho\epsilon}\right)\right).$$

Upper bounds

- If $n, d \geq 1$

$$\text{icomp}_{\epsilon}(n, d, R, M, \rho) \in O\left((n+d)d2^n \log\left(\frac{MR}{\rho\epsilon}\right)\right).$$

- If $d = 0$

$$\text{icomp}_{\epsilon}(n, d, R, M, \rho) \in O\left(n2^n \log(R)\right).$$

- If $n = 0$

$$\text{icomp}_{\epsilon}(n, d, R, M, \rho) \in O\left(d \log\left(\frac{MR}{\rho\epsilon}\right)\right).$$

Note that when $n = 0$, i.e., we consider continuous convex optimization with no integer variables, we have $\text{icomp}_\epsilon(n, d, R, M, \rho) = \Theta\left(d \log\left(\frac{MR}{\rho\epsilon}\right)\right)$, giving a tight characterization of the complexity. In fact, these results can be obtained for a much broader class of oracles that include first-order/separation oracles as special cases; see [28, 107–109].

For pure integer optimization with $d = 0$, our upper and lower bounds are off by a linear factor in the dimension, which is of much lower order compared to the dominating term of $2^n \log(R)$. Put another way, both bounds are $2^{O(n)} \log(R)$. The lower bounds come from the feasibility question (see the proofs below). Additionally, since the strict feasibility assumption is vacuous and for small enough $\epsilon > 0$, $S(I, \epsilon)$ is the set of exact optimum solutions, M, ϵ and ρ do not play a role in the upper and lower bounds; in particular, they are the bounds for obtaining exact solutions ($\epsilon = 0$) as well.

There seems to be scope for nontrivial improvement in the bounds presented for the mixed-integer case, i.e., $n, d \geq 1$:

1. It would be nice to unify the lower bound for $n = 0$ (the continuous case) and $n \geq 1$ (the truly mixed-integer case). The proof below for $n, d \geq 1$ is based on the feasibility question, which is why M and ϵ do not appear in the lower bound. This is inspired by the proof technique in [14]. We do not see a similar way to incorporate the objective function parameters to match the upper bound. We suspect that one should be able to prove the stronger lower bound of $\Omega\left(d2^n \log\left(\frac{MR}{\rho\epsilon}\right)\right)$, but at present we do not see how to do this and we are not aware of any existing literature that achieves this.

2. When one plugs in $n = 0$ in the mixed-integer upper bound ($n, d \geq 1$), one does not recover the tight upper bound for $n = 0$; instead, the bound is off by a factor of d. We believe this can likely be improved, for example, if Conjecture 4.6 below is proved to be true in the future. Then one would have an upper bound of $O\left((n + d)2^n \log\left(\frac{MR}{\rho\epsilon}\right)\right)$ in the mixed-integer case that more accurately generalizes both the pure continuous ($n = 0$) and pure integer ($d = 0$) upper bounds.

4.1 Proof of the lower bounds in Theorem 4.2

The general strategy is the following: Given any adaptive query sequence D, we will construct two instances $(f_1, C_1), (f_2, C_2) \in \mathcal{I}_{n,d,R,M,\rho}$ such that the transcripts $\Pi_k(D, (f_1, C_1))$ and $\Pi_k(D, (f_2, C_2))$ are equal for any k less than the lower bound, but $S((f_1, C_1), \epsilon) \cap S((f_2, C_2), \epsilon) = \emptyset$.

The mixed-integer case ($n, d \geq 1$). We will show that $\text{icomp}_\epsilon(n, d, R, M, \rho) \geq d2^n \log_2\left(\frac{R}{3\rho}\right)$. We construct $C_1, C_2 \subseteq \mathbb{R}^n \times \mathbb{R}^d$ such that $C_1 \cap C_2 \cap (\mathbb{Z}^n \times \mathbb{R}^d) = \emptyset$, both sets satisfy the strict feasibility condition dictated by ρ, and any separation oracle query from D on C_1 and C_2 has the same answer. Our instances will consist of these two sets as feasible regions and $f_1 = f_2$ as constant functions, thus any first-order oracle query in D will simply return this constant value and 0 as a subgradient. Since there is no common feasible point, $S((f_1, C_1), \epsilon) \cap S((f_2, C_2), \epsilon) = \emptyset$ as required.

The construction of C_1 and C_2 goes as follows. Since the function oracle calls are superfluous, we may assume the k (adaptive) queries $\{q_1, \ldots, q_k\}$ to be all separation oracle queries. Begin with $X_0 = [0, 1]^n \times [0, R]^d$. We create a nested sequence $X_0 \supseteq X_1 \supseteq X_2 \supseteq \ldots \supseteq X_k$ such that $X_i \cap (\{x\} \times \mathbb{R}^d)$ is a fiber box (possibly empty) for any $x \in \{0, 1\}^n$. X_i is defined inductively from X_{i-1}, using the query q_i. For every $\tilde{x} \in \{0, 1\}^n$, we maintain a counter $\#\tilde{x}(i)$ which will keep track of how many q_j, $j \leq i$ queried a point of the form $(\tilde{x}, y)$ inside X_{j-1} for some $y \in \mathbb{R}^d$.

If q_i queries $(x^i, y^i) \notin X_{i-1}$, then we simply report any hyperplane separating (x^i, y^i) from X_{i-1} as the answer to q_i and define $X_i = X_{i-1}$. If q_i queries $(x^i, y^i) \in X_{i-1} \setminus (\mathbb{Z}^n \times \mathbb{R}^d)$ (i.e., $x^i \notin \mathbb{Z}^n$), we define $X_i = X_{i-1}$ and the answer to the query q_i is that (x^i, y^i) is in the set.

Suppose now $(x^i, y^i) \in X_{i-1} \cap (\mathbb{Z}^n \times \mathbb{R}^d)$. If $\#x^i(i-1) \geq d \log_2\left(\frac{R}{3\rho}\right)$ then we report a halfspace H that separates $X_{i-1} \cap (\{x_i\} \times \mathbb{R}^d)$ from the rest of the fibers $X_{i-1} \cap (\{x\} \times \mathbb{R}^d)$ for $x \neq x_i$, and define

$X_i = X_{i-1} \cap H$. If $\#x^i(i-1) < d\log_2\left(\frac{R}{3\rho}\right)$ then select the coordinate $j = (\#x^i(i-1) \mod d)+1$ and define the hyperplane $\{(x,y) \in \mathbb{R}^n \times \mathbb{R}^d : y_j = y^i_j\}$. Let B denote the fiber box $X_{i-1} \cap (\{x^i\} \times \mathbb{R}^d)$. Consider the separation with a halfspace $\hat{H}$ with this hyperplane such that $B \cap \hat{H}$ has length in coordinate j to be at least half of the length B in coordinate j. We now rotate this hyperplane and halfspace $\hat{H}$ to obtain a halfspace H such that $X_{i-1} \cap H$ has the same intersection as X_{i-1} with $(\{x\} \times \mathbb{R}^d)$ for $x \neq x_i$. In other words, all other mixed-integer fibers in X_{i-1} are maintained. Define $X_i = X_{i-1} \cap H$ and H as the separating halfspace for query q_i. Update $\#x^i(i) = \#x^i(i-1)+1$. Note that the above construction ensures inductively that for any $i \in \{1,\dots,k\}$, the set $X_i \cap (\{x\} \times \mathbb{R}^d)$ is a fiber box for $x \in \{0,1\}^n$.

Since $\sum_{x\in\{0,1\}^n} \#x(k) \leq k < 2^n \cdot d\log_2\left(\frac{R}{3\rho}\right)$, we observe that X_k contains a fiber box B of width at least 3ρ. Thus, we can select two fiber boxes $B_1, B_2 \subseteq B$ such that $B_1 \cap B_2 = \emptyset$, and B_1 and B_2 have width ρ. For $i = 1,2$, define C_i to be the convex hull of B_i and all the points queried by D that were reported to be in the set. We observe that $C_i \cap (\mathbb{Z}^n \times \mathbb{R}^d) = B_i$ for $i = 1,2$ and thus we have no common feasible points in C_1, C_2. This completes the proof for $d \geq 1$.

The pure integer case ($d = 0$). The proof proceeds in a similar manner to the mixed-integer case ($n, d \geq 1$) with $X_0 = [0,1]^{n-1} \times [0, \lfloor R \rfloor] \subseteq \mathbb{R}^n$. The "fibers" are now $\{x\} \times \{0,1,\dots,\lfloor R \rfloor\}$. If $k < 2^n \log_2(R)$, one can again construct $C_1, C_2 \subseteq X_0$ such that $C_1 \cap C_2 \cap \mathbb{Z}^n = \emptyset$ by an inductive argument based on the queries from D, and take f_1, f_2 as constant functions.

The pure continuous case ($n = 0$). We omit the proof as this has appeared in many different places in the literature [28, 107–109]. The idea is very similar to what was presented above for the general mixed-integer case. One proves that

$$\begin{array}{rcl} \mathrm{icomp}_\epsilon(d,R,M,\rho) & \geq & \max\left\{d\log_2\left(\frac{R}{3\rho}\right), d\log_2\left(\frac{MR}{8\epsilon}\right)\right\} \\ & \geq & \frac{d\log_2\left(\frac{R}{3\rho}\right)+d\log_2\left(\frac{MR}{8\epsilon}\right)}{2} \\ & \in & \Omega\left(d\log\left(\frac{MR}{\rho\epsilon}\right)\right). \end{array}$$

If $k < d\log_2\left(\frac{R}{3\rho}\right)$ one can appeal to the mixed-integer case above. In fact, there is no rotation of halfspaces necessary as there are no integer fibers. If $k < d\log_2\left(\frac{MR}{8\epsilon}\right)$, one constructs two different convex functions f_1, f_2 while the feasible region can be taken to be $[0,R]^d$ in both cases. The details are a little more complicated than the separation oracle case, since the function values and the subgradients have to be more carefully engineered. We refer the reader to the references cited above for the details. □

4.2 Proof of the upper bounds in Theorem 4.2

The idea of the upper bound hinges on a geometric concept that has appeared in several different areas of mathematics, including convex geometry, statistics and theoretical computer science.

Definition 4.3. For any $S \subseteq \mathbb{Z}^n \times \mathbb{R}^d$ with $d \geq 1$, $\nu(S)$ will denote the *mixed-integer volume* of S, i.e.,

$$\nu(S) := \sum_{x\in\mathbb{Z}^n} \mu_d(S \cap (\{x\} \times \mathbb{R}^d)),$$

where μ_d is the standard Lebesgue measure (volume) in $\mathbb{R}^d$. If $d = 0$, we overload notation and use $\nu(S)$ to denote the number of integer points in S, i.e., the counting measure on $\mathbb{Z}^n$.

Note that if $S = C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ for a compact convex set $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$, then $\nu(S)$ is finite.

Definition 4.4. For any $S \subseteq \mathbb{Z}^n \times \mathbb{R}^d$ and $x \in \mathbb{R}^n \times \mathbb{R}^d$, define

$$h_S(x) := \inf_{\substack{\text{halfspace } H: \\ x \in H}} \nu(S \cap H).$$

The set of *centerpoints of* S is defined as $\mathcal{C}(S) := \operatorname{argmax}_{x \in S} h_S(x)$.

The above concept was first defined in Timm Oertel's Ph.D. thesis [111] and extensions were introduced in [14]. We refer the reader to the thesis and the cited paper, and the references therein for structural properties of h_S and $\mathcal{C}(S)$. For our purposes, we will simply need the following result.

Theorem 4.5. Let $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$ be any compact, convex set and let $S = C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$. Then $\mathcal{C}(S)$ is nonempty and $h_S(\hat{x}) \geq \frac{1}{2^n(d+1)}\nu(S)$ for any centerpoint $\hat{x}$. If $n = 0$, then $h_S(\hat{x}) \geq \left(\frac{d}{d+1}\right)^d \nu(S) \geq \frac{1}{e}\nu(S)$.

The first bound in Theorem 4.5 was first established in [111] and is a special case of a general result involving Helly numbers [14, Theorem 3.3]. The second bound ($n = 0$) is due to Grünbaum [79]. There is clearly a gap in the two cases and the following sharper lower bound is conjectured to be true [14, 111]; a matching upper bound is given by $S = \{0,1\}^n \times \Delta_d$, where Δ_d is the standard d-dimensional simplex.

Conjecture 4.6. Under the hypothesis of Theorem 4.5, $h_S(\hat{x}) \geq \frac{1}{2^n}\left(\frac{d}{d+1}\right)^d \nu(S) \geq \frac{1}{2^n}\frac{1}{e}\nu(S)$ for any $n, d \geq 0$ (both not both 0) for any centerpoint $\hat{x}$.

The final piece we need is the following consequence of a "strict feasibility" type assumption.

Lemma 4.7. Let $1 \leq p \leq \infty$. Let $C \subseteq \mathbb{R}^k$ be a closed, convex set such that $\{z \in \mathbb{R}^k : \|z - a\|_p \leq \rho\} \subseteq C \subseteq \{z \in \mathbb{R}^k : \|z\|_p \leq R\}$, for some $R, \rho \in \mathbb{R}_+$ and $a \in \mathbb{R}^k$. Let $f : \mathbb{R}^k \to \mathbb{R}$ be a convex function that is Lipschitz continuous over $\{z \in \mathbb{R}^k : \|z\|_p \leq R\}$ with respect to the $\|\cdot\|_p$-norm with Lipschitz constant M. For any $\epsilon \leq 2MR$ and for any $z^\star \in C$, the set $\{z \in C : f(z) \leq f(z^\star) + \epsilon\}$ contains an $\|\cdot\|_p$ ball of radius $\frac{\epsilon\rho}{2MR}$ with center lying on the line segment between $z^\star$ and a.

Proof. Since $C \subseteq \{z : \|z\|_p \leq R\}$, we must have $C \subseteq \{z : \|z - z^\star\|_p \leq 2R\}$. By convexity of C and the fact that $\frac{\epsilon}{2MR} \leq 1$, $z^\star + \frac{\epsilon}{2MR}(C - z^\star) \subseteq C$. Hence,

$$z^\star + \frac{\epsilon}{2MR}(C - z^\star) \subseteq \{z \in C : \|z - z^\star\|_p \leq \frac{\epsilon}{M}\} \subseteq \{z \in C : f(z) \leq f^\star + \epsilon\},$$

where the second containment follows from the Lipschitz property of f. Since C contains an $\|\cdot\|_p$ ball of radius ρ centered at a, the set $z^\star + \frac{\epsilon}{2MR}(C - z^\star)$ (i.e., the $\frac{\epsilon}{2MR}$ scaling of C about $z^\star$) must contain a ball of radius $\frac{\epsilon\rho}{2MR}$ centered at a point on the line segment between $z^\star$ and a. □

We now proceed with the proof of the upper bounds in Theorem 4.2.

The mixed-integer case with $n, d \geq 1$. We consider the following adaptive search strategy. For any finite subset $T \subseteq \mathcal{Q} \times H$ (possibly empty), where $\mathcal{Q}$ is the set of all possible first-order or separation oracle queries in $[-R, R]^{n+d}$ and H is the set of possible responses to such queries, define $D(T)$ as follows. Let $z_1, \ldots, z_q$ be the points queried in T where either a first-order oracle call to a function was made, or a separation oracle call was made that returned a separating hyperplane (i.e., the point is not in the set queried). Let h_j be the subgradient or normal vector to the separating hyperplane returned at z_j, $j = 1, \ldots, q$. Define $v_{\min}$ to be the minimum function value seen so far ($+\infty$ if no first order query exists in T).

The next query for D will be at the centerpoint $\hat{z}$ of the set

$$\left\{ z \in \mathbb{Z}^n \times \mathbb{R}^d : \begin{array}{ll} \langle h_i, z - z_i \rangle \leq 0 & i = 1, \ldots, q, \\ \|x\|_\infty \leq R & \end{array} \right\}$$

If T is the transcript on an instance f, C, then the "search space" polyhedron P defined by the above inequalities contains $C \cap \{z : f(z) \leq v_{\min}\}$. D now queries the centerpoint of $P \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ so that any separating hyperplane or subgradient inequality can remove a guaranteed fraction of the mixed-integer volume of the current search space. More formally, D first queries the separation oracle for C at $\hat{z}$. If the separation oracle says $\hat{z}$ is in C, then D queries the first-order oracle for f at $\hat{z}$.

Consider any instance $I = (f, C) \in \mathcal{I}_{n,d,R,M,\rho}$ and any natural number

$$k \geq 2 \cdot \left(\log_b \left(\left(\frac{2R+1}{\rho} \right)^{n+d} \right) + \log_b \left(\left(\frac{M(2R+1)}{\epsilon} \right)^{n+d} \right) \right),$$

where $b = \frac{2^n(d+1)}{2^n(d+1)-1}$. We claim that at least one first-order oracle query appears in the transcript $\Pi_k(D, I)$ and $z_{\min} \in S(I', \epsilon)$ for every instance I' such that $\Pi_k(D, I') = \Pi_k(D, I)$, where $z_{\min}$ is a point queried in the transcript $\Pi_k(D, I)$ with the minimum function value amongst all points queried with a first-order query on f in $\Pi_k(D, I)$. In other words, for any instance $I' = (f', C')$ such that $\Pi_k(D, I') = \Pi_k(D, I)$, we have $z_{\min} \in C'$ and $f'(z_{\min}) - OPT \leq \epsilon$ where OPT is the minimum value of f' on C'. This will prove the result since

$$\begin{aligned}
2 \cdot \left(\log_b \left(\left(\tfrac{2R+1}{\rho} \right)^{n+d} \right) + \log_b \left(\left(\tfrac{M(2R+1)}{\epsilon} \right)^{n+d} \right) \right) &= 2(n+d) \log_b \left(\tfrac{M(2R+1)^2}{\rho\epsilon} \right) \\
&= 2(n+d) \ln \left(\tfrac{M(2R+1)^2}{\rho\epsilon} \right) / \ln(b) \\
&\leq 2(n+d) 2^n (d+1) \ln \left(\tfrac{M(2R+1)^2}{\rho\epsilon} \right) \\
&\in O\left((n+d) d 2^n \log \left(\tfrac{MR}{\rho\epsilon} \right) \right)
\end{aligned}$$

First, let k' be the number of queries in $\Pi_k(D, I)$ that were either first-order oracle queries on f or separation oracle queries on C that returned a separating hyperplane, i.e., we ignore the separation oracle queries on points inside C. Observe that $k' \geq k/2 \geq \log_b \left(\left(\frac{2R+1}{\rho} \right)^{n+d} \right) + \log_b \left(\left(\frac{M(2R+1)}{\epsilon} \right)^{n+d} \right)$ since a query on any point in C is immediately followed by a first order query on the same point. Theorem 4.5 implies that each of these k' queries reduces the mixed-integer volume of the current search space by at least $1/b$. Recall that we start with a mixed-integer volume of at most $(2R+1)^{n+d}$ and C contains a fiber box of mixed-integer volume at least $\rho^d \geq \rho^{n+d}$ (since $\rho \leq 1$). Thus, at most $\log_b \left(\left(\frac{2R+1}{\rho} \right)^{n+d} \right)$ queries can be separation oracle queries and we have at least $\log_b \left(\left(\frac{M(2R+1)}{\epsilon} \right)^{n+d} \right)$ first-order queries to f at points inside $C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$. Let k'' denote the number of such queries, queried at $z_1, \ldots, z_{k''}$ with responses $h_1, \ldots, h_{k''}$ as the subgradients and $v_1, \ldots, v_{k''}$ as the function values. Let $v_{\min}$ be the minimum of these function values, corresponding to the query point $z_{\min}$. Since $z_{\min}$ is feasible to C, if f', C' is any other instance with the same responses to all queries in $\Pi_k(D, I)$, then $z_{\min}$ is feasible to C' as well. In fact, all the points $z_1, \ldots, z_{k''}$ are in C'. We now verify that $f'(z_{\min}) \leq OPT + \epsilon$ where OPT is the minimum value of f' on $C' \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ attained at, say $z^\star = (x^\star, y^\star)$.

Let $C'' = C' \cap (\{x^\star\} \times \mathbb{R}^d)$ be the intersection of C' with the fiber containing $z^\star$. Consider the polyhedron

$$\tilde{P} := \{z : \langle h_j, z - z_j \rangle \leq 0 \;\; j = 1, \ldots, k''\}.$$

Since we have been reducing the mixed-integer volume at a rate of $1/b$, $C' \cap \tilde{P}$ has mixed-integer volume at most $(2R+1)^{n+d}/b^{k'}$ and therefore $C'' \cap \tilde{P}$ has d-dimensional volume at most $(2R+1)^{n+d}/b^{k'}$. Since $k' \geq \log_b \left(\left(\frac{2R+1}{\rho} \right)^{n+d} \right) + \log_b \left(\left(\frac{M(2R+1)}{\epsilon} \right)^{n+d} \right)$, we must have $b^{k'} \geq \left(\frac{2R+1}{\rho} \right)^{n+d} \cdot \left(\frac{M(2R+1)}{\epsilon} \right)^{n+d}$. Thus, $C'' \cap \tilde{P}$ has d-dimensional volume at most $\left(\frac{\rho\epsilon}{M(2R+1)} \right)^{n+d}$. We may assume $\frac{\epsilon}{2MR} \leq 1$, otherwise any feasible solution is an ϵ approximate solution, and so is $z_{\min}$. Since $\rho \leq 1$ as well, this means $\frac{\rho\epsilon}{M(2R+1)} \leq 1$. Therefore,

$\left(\frac{\rho\epsilon}{M(2R+1)}\right)^{n+d} \leq \left(\frac{\rho\epsilon}{M(2R+1)}\right)^{d} < \left(\frac{\rho\epsilon}{2MR}\right)^{d}$. From Lemma 4.7, $\{z \in C'' : f'(z) \leq f'(z^\star) + \epsilon\}$ has volume at least $\left(\frac{\rho\epsilon}{2MR}\right)^{d}$. Thus, at least one point $\hat{z}$ in $\{z \in C'' : f'(z) \leq OPT + \epsilon\}$ must be outside $C'' \cap \tilde{P}$. Such a point must violate one of the subgradient inequalities defining $\tilde{P}$, say corresponding to index $\tilde{j}$. In other words, $\langle h_{\tilde{j}}, \hat{z} - z_{\tilde{j}} \rangle > 0$. This means $f'(\hat{z}) \geq f'(v_{\tilde{j}}) + \langle h_{\tilde{j}}, \hat{z} - z_{\tilde{j}} \rangle > f'(v_{\tilde{j}})$. Thus, $f'(z_{\min}) \leq f'(v_{\tilde{j}}) < f'(\hat{z}) \leq OPT + \epsilon$.

The pure integer case with $d = 0$. The proof proceeds in a very similar manner except that one can stop when we have at most one integer point left in the polyhedral search space. Thus, we start from the box $[-R, R]^n$ containing $(2R+1)^n$ integer points and end with at most a single integer point, removing at least $\frac{1}{2^n}$ fraction of integer points every time by Theorem 4.5.

The pure continuous case with $n = 0$. The proof is very similar and the only difference is that we can use the stronger bound on the centerpoints due to Grünbaum from Theorem 4.5. In other words, b can be taken to be the Euler's constant while mimicking the proof of the $n, d \geq 1$ case above. □

Remark 4.8. The upper and lower bounds achieved above are roughly a consequence of the concept of Helly numbers [7, 18, 62, 81, 85, 122]. For any subset $S \subseteq \mathbb{R}^k$, we say $K_1, \ldots, K_t$ is a *critical family of convex sets* with respect to S (of size t) if $K_1 \cap \ldots \cap K_t \cap S = \emptyset$, but for any $i \in \{1, \ldots, t\}$, $\cap_{j \neq i} K_j \cap S \neq \emptyset$. The *Helly number of* S is the size of the largest critical family with respect to S (possibly $+\infty$). It turns out that the Helly number of $\mathbb{Z}^n \times \mathbb{R}^d \subseteq \mathbb{R}^{n+d}$ is $2^n(d+1)$ and there exists a critical family of halfspaces $H_1, \ldots, H_{2^n(d+1)}$ of this size [7, 85]. Now consider the family of $2^n(d+1)$ polyhedra $\cap_{j \neq i} H_j$ for $i = 1, \ldots, 2^n(d+1)$, along with the polyhedron $\cap_{j=1}^{2^n(d+1)} H_j$. If one makes less than $2^n(d+1)$ separation oracle queries, then every time we can simply report the halfspace H_j that does not contain a mixed-integer query point (such a halfspace exists since $\cap_{j=1}^{2^n(d+1)} H_j \cap (\mathbb{Z}^n \times \mathbb{R}^d) = \emptyset$), and if the query point is not in $\mathbb{Z}^n \times \mathbb{R}^d$, we truthfully report if it is in $\cap_{j=1}^{2^n(d+1)} H_j$ or not. The intersection of these reported halfspaces still contains a point from $\mathbb{Z}^n \times \mathbb{R}^d$ since it is a critical family and we have less than $2^n(d+1)$ queries. Therefore, we are unable to distinguish between the case $\cap_{j=1}^{2^n(d+1)} H_j$ which has no point from $\mathbb{Z}^n \times \mathbb{R}^d$ and the nonempty case. This gives a lower bound of $2^n(d+1)$. As we saw in the proof of the upper bound above, the key result is Theorem 4.5 which is based on Helly numbers again [14, 79, 111].

5 Algorithmic complexity

The upper bounds on ϵ-information complexity presented in Section 4 do not immediately give upper bounds on algorithmic complexity, unless we can provide an algorithm for computing centerpoints. This is computationally extremely challenging [14, 111]. The best known algorithms for mixed-integer convex optimization do not match the ϵ-information complexity bounds presented, even in terms of the informational bound $\mathrm{icomp}_{\mathcal{A}}$ (see Definition 1.7). We will present an ϵ-approximation algorithm for mixed-integer convex optimization in this section whose information complexity bound is the closest known to the algorithm independent ϵ-information complexity bound from the previous section. In the case of pure continuous optimization, i.e., $n = 0$, the algorithm's information complexity is larger than the corresponding ϵ-information complexity bound in Theorem 4.2 by a factor that is linear[1] in the dimension d. In the case of pure integer optimization, i.e., $d = 0$, the algorithm's information complexity bound is $2^{O(n \log n)} \log(R)$. Compared to the ϵ-information complexity bound of $2^{O(n)} \log(R)$ from Theorem 4.2, there seems to be a significant gap. It remains a major open question in integer optimization whether the gap between $2^{O(n)}$ and $2^{O(n \log n)}$ can be closed or not by designing a better algorithm.

The overall complexity (including the computational complexity) of the algorithm (see Definition 1.7) will be seen to be a low degree polynomial factor larger than its information complexity.

[1]There is also a factor of $\log(d)$ which shows up due to technical reasons of using $\|\cdot\|_2$ instead of $\|\cdot\|_\infty$.

5.1 Enumeration and cutting planes

Algorithms for mixed-integer convex optimization are based on two main ideas. The first one, called *branching*, is a way to systematically explore different parts of the feasible region. The second aspect, that of cutting planes, is useful when one is working with a relaxation (superset) of the feasible region and uses separating hyperplanes to remove parts of the relaxation that do not contain feasible points.

Definition 5.1. A *disjunction* for $\mathbb{Z}^n \times \mathbb{R}^d$ is a union of polyhedra $D = Q_1 \cup \ldots \cup Q_k$ such that $\mathbb{Z}^n \times \mathbb{R}^d \subseteq D$.

For any set X in some Euclidean space, a *cutting plane* for X is a halfspace H such that $X \subseteq H$. If X is of the form $C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$, then the cutting plane is *trivial* if $C \subseteq H$, while it is said to be *nontrivial* otherwise.

The words "trivial" and "nontrivial" are used here in a purely technical sense. For a complicated convex set C, we may have a simple polyhedral relaxation $R \supseteq C$ such as those used in the proofs of upper bounds in Theorem 4.2, and the separation oracle for C can return *trivial* cutting planes that shave off parts of R. But if the oracle is difficult to implement, there may be nothing trivial about obtaining such a cutting plane. Our terminology comes from settings where C has a simple description and separating from C is not a big deal; rather, the interesting work is in removing parts of C that do not contain any point from $X = C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$. We hope the reader will indulge us in our purely technical use of the terms *trivial* and *nontrivial* cutting planes.

Example 5.2.

1. A well-known example of disjunctions for $\mathbb{Z}^n \times \mathbb{R}^d$ is the family of *split disjunctions* that are of the form $\{x \in \mathbb{R}^{n+d} : \langle \pi, x\rangle \leq \pi_0\} \cup \{x \in \mathbb{R}^{n+d} : \langle \pi, x\rangle \geq \pi_0 + 1\}$, where $\pi \in \mathbb{Z}^n \times \{0\}^d$ and $\pi_0 \in \mathbb{Z}$. When the first n coordinates of π correspond to a standard unit vector, we get *variable disjunctions*, i.e., disjunctions of the form $\{x : x_i \leq \pi_0\} \cup \{x : x_i \geq \pi_0 + 1\}$, for $i = 1, \ldots, n$. Several researchers in this area have also considered the intersection of t different split disjunctions to get a disjunction [46, 47, 99]; these are known as t-branch split disjunctions.

2. As mentioned above, for any convex set C contained in a polyhedron P, the separation oracle for C can return trivial cutting planes if a point from $P \setminus C$ is queried. Examples of nontrivial cutting planes for sets of the form $C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ include *Chvátal-Gomory cutting planes* [123, Chapter 23] and *split cutting planes* [39]. These will be discussed in more detail below.

5.2 The "Lenstra-style" algorithm

Cutting plane based algorithms were designed in continuous convex optimization quite early in the development of the subject [109]. In the 80s, these ideas were combined with techniques from algorithmic geometry of numbers and the idea of branching on split disjunctions to design algorithms for the mixed-integer case as well [78, 90, 98]. There has been a steady line of work since then with sustained improvements; see [41, 80, 83, 92] for a representative sample. We will present here an algorithm based on these ideas whose complexity is close to the best known algorithmic complexity for the general mixed-integer case.

We first introduce some preliminary concepts and results.

Definition 5.3. Given a positive definite matrix $A \in \mathbb{R}^{k\times k}$, the *norm defined by* A on $\mathbb{R}^k$ is $\|x\|_A := \sqrt{x^T A^{-1} x}$. The unit ball of this norm $E_A := \{x : x^T A^{-1} x \leq 1\}$ is called an *ellipsoid defined by* A. The orthonormal eigenvectors of A are called the *principal axes*.

The following result, due to Yudin and Nemirovski [129], is a foundational building block for the algorithm.

Theorem 5.4. [78, Lemma 3.3.21] Let $A \in \mathbb{R}^{k\times k}$ be a positive definite matrix. For any halfspace H and any $0 \leq \beta < \frac{1}{k}$ such that H does not contain βE_A, there exists another positive definite matrix A' and $c \in \mathbb{R}^k$ such that $E_A \cap H \subseteq c + E_{A'}$ and

$$\mathrm{vol}(E_{A'}) \leq e^{-\frac{(1-\beta k)^2}{5k}}\,\mathrm{vol}(E_A),$$

where $\mathrm{vol}(\cdot)$ denotes the k-dimensional volume. Moreover, c and A' can be computed from A by an algorithm with complexity $O(k^2)$.

We will also need the following fundamental result in geometry of numbers.

Theorem 5.5. [Khinchine's flatness theorem for ellipsoids][8, 9, 121] Let $E \subseteq \mathbb{R}^k$ be an ellipsoid and $c \in \mathbb{R}^k$ such that $(c+E) \cap \mathbb{Z}^k = \emptyset$. Then there exists $w \in \mathbb{Z}^k \setminus \{0\}$ such that

$$\max_{v \in E} \langle w, v\rangle - \min_{v \in E} \langle w, v\rangle \leq k.$$

The final piece we will need is the following algorithmic breakthrough achieved at the beginning of the previous decade [2, 3, 104, 105]. We state the result in a way that will be most convenient for us.

Theorem 5.6. Let $A \in \mathbb{R}^{k \times k}$ be a positive definite matrix. Then there exist algorithms with worst case complexity $2^{O(k)}\,\mathrm{poly}(\mathrm{size}(A))$ that solve the following optimization problems:

$$\min_{v \in \mathbb{Z}^k \setminus \{0\}} \|v\|_A \qquad \text{(Shortest Vector Problem (SVP))}$$

and for any given vector $c \in \mathbb{R}^k$,

$$\min_{v \in \mathbb{Z}^k} \|v - c\|_A \qquad \text{(Closest Vector Problem (CVP))}$$

We are now ready to describe our algorithm. We begin with a feasibility algorithm before discussing optimization. Given a closed, convex set, the algorithm either correctly computes a mixed-integer point in the convex set, or reports that there is no mixed-integer point "deep inside" the set. Thus, the algorithm is not an exact feasibility algorithm. Nevertheless, this will suffice to design an ϵ-approximation algorithm for the problem class (3.1) parameterized by n, d, M, R, ρ, as studied in Section 4. However, since we work with ellipsoids, the parameters R, ρ and the Lipschitz constant M will all use the $\|\cdot\|_2$ norm instead of the $\|\cdot\|_\infty$ norm as in Section 4. Moreover, M is defined with respect to the full space $\mathbb{R}^n \times \mathbb{R}^d$, as opposed to just $\mathbb{R}^d$.

Theorem 5.7. Let $R \geq 0$. There exists an algorithm $\mathcal{A}$, i.e., oracle Turing machine, such that for any closed, convex set $C \subseteq \{z \in \mathbb{R}^n \times \mathbb{R}^d : \|z\|_2 \leq R\}$ equipped with a separation oracle that $\mathcal{A}$ can access, and any $\delta > 0$, either correctly computes a point in $C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$, or correctly reports that there is no point $z \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ such that the Euclidean ball of radius δ around z is contained in C.

Moreover,

$$\mathrm{icomp}_{\mathcal{A}}(n, d, R, \delta) \leq 2^{O(n \log(n+d))} \log\left(\tfrac{R}{\delta}\right)$$

and

$$\mathrm{comp}_{\mathcal{A}}(n, d, R, \delta) \leq 2^{O(n \log(n+d))}\,\mathrm{poly}\left(\log\left(\tfrac{R(n+d)}{\delta}\right)\right)$$

Proof. The algorithm uses recursion on the "integer dimension" n.

Let $c_0 = 0$ and $E_0 = \{z : \|z\|_2 \leq R\}$. The algorithm will either iteratively compute $c_i \in \mathbb{R}^n \times \mathbb{R}^d$ and ellipsoid $E_i \subseteq \mathbb{R}^n \times \mathbb{R}^d$ from c_{i-1}, E_{i-1} for $i = 1, 2, \ldots$ such that the invariant $C \subseteq c_i + E_i$ is maintained, or the algorithm will recurse on lower dimensional problems.

If $\mathrm{vol}(E_{i-1})$ less than the volume of a Euclidean ball of radius δ, then we report that there is no point $z \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ such that the Euclidean ball of radius δ around z is contained in C.

Otherwise, we either compute a "test point" $(\hat{x}, \hat{y}) \in \mathbb{Z}^n \times \mathbb{R}^d$ and generate the new c_i, E_i based on properties of this point (Cases 1 and 2a below), or recurse on lower dimensional subproblems (Case 2b below).

Case 1: $n = 0$. Define $(\hat{x}, \hat{y})$ to be c_{i-1}. We query the separation oracle of C at $(\hat{x}, \hat{y})$. If this point is in C, we are done. Else, we obtain a separating halfspace H. Applying Theorem 5.4 with $k = n + d$ and $\beta = 0$, we can construct c_i and E_i such that $(c_{i-1} + E_{i-1}) \cap H \subseteq c_i + E_i$ and $\mathrm{vol}(E_i) \leq e^{-\frac{1}{5(n+d)}}\,\mathrm{vol}(E_{i-1})$. Note that this ensures $C \subseteq c_i + E_i$ since inductively we know $C \subseteq (c_{i-1} + E_{i-1}) \cap H$.

Case 2: $n \geq 1$. We compute the projections c', E' of c_{i-1}, E_{i-1} onto the coordinates corresponding to the integers, i.e., $\mathbb{R}^n$. This is easy to do for c_{i-1} (simply drop the other coordinates) and given the matrix A_{i-1}

defining E_{i-1}, the submatrix A' of A_{i-1} whose rows and columns correspond to the integer coordinates is such that $E_{A'}$ is the projection of E_{i-1}. We now solve the closest vector problem (CVP) for $c' \in \mathbb{R}^n$ and the norm given by $A' \in \mathbb{R}^{n\times n}$ using the algorithm in Theorem 5.6 to obtain $\hat{x} \in \mathbb{Z}^n$.

Case 2a: $\|\hat{x} - c'\|_{A'} \leq \frac{1}{n+d+1}$. In other words, $\hat{x} \in c' + \frac{1}{n+d+1}E'$. Since c', E' are projections of c_{i-1}, E_{i-1} respectively, $c' + \frac{1}{n+d+1}E'$ is the projection of $c_{i-1} + \frac{1}{n+d+1}E_{i-1}$ by linearity of the projection map. Hence, there must be $\hat{y} \in \mathbb{R}^d$ such that $(\hat{x}, \hat{y}) \in c_{i-1} + \frac{1}{n+d+1}E_{i-1}$. Therefore, if we compute $\operatorname{argmin}_{y\in\mathbb{R}^d} \|(\hat{x}, y) - c_{i-1}\|_{A_{i-1}}$ which amounts to computing the minimizer of an explicit convex quadratic function in $\mathbb{R}^d$ (which can be done analytically or via methods like conjugate gradient), we can find a point $(\hat{x}, \hat{y})$ in $c_{i-1} + \frac{1}{n+d+1}E_{i-1}$.

We query the separation oracle of C at $(\hat{x}, \hat{y})$. If this point is in C, we are done. Else, we obtain a separating halfspace H. Since $(\hat{x}, \hat{y})$ is in $c_{i-1} + \frac{1}{n+d+1}E_{i-1}$, this means $c_{i-1} + \frac{1}{n+d+1}E_{i-1}$ is not contained in H. Applying Theorem 5.4 with $k = n + d$ and $\beta = \frac{1}{n+d+1}$, we can construct c_i and E_i such that $(c_{i-1} + E_{i-1}) \cap H \subseteq c_i + E_i$ and $\mathrm{vol}(E_i) \leq e^{-\frac{1}{5(n+d)(n+d+1)^2}} \mathrm{vol}(E_{i-1})$. Note that this ensures $C \subseteq c_i + E_i$ since inductively we know $C \subseteq (c_{i-1} + E_{i-1}) \cap H$.

Case 2b: $\|\hat{x} - c'\|_{A'} > \frac{1}{n+d+1}$. In other words, $\hat{x} \notin c' + \frac{1}{n+d+1}E'$ which implies that $c' + \frac{1}{n+d+1}E'$ has no integer points since $\hat{x}$ is the closest integer point to c' in the norm $\|\cdot\|_{A'}$. Theorem 5.5 implies that there exists $w \in \mathbb{Z}^n \setminus \{0\}$ such that $\max_{x\in E'} \langle w, x\rangle - \min_{x\in E'} \langle w, x\rangle \leq n(n + d + 1)$. Rearranging, this says that $\max_{x,x'\in E'} \langle w, x - x'\rangle \leq n(n + d + 1)$ and therefore

$$\max_{p\in 2E'} \langle w, p\rangle = \max_{p\in E'+E'} \langle w, p\rangle = \max_{p\in E'-E'} \langle w, p\rangle \leq n(n + d + 1),$$

where the equalities follow from the fact that E' is convex and centrally symmetric about the origin. Standard results in convex analysis involving polarity imply that $\|w\|_{\tilde{A}} = \max_{p\in 2E'} \langle w, p\rangle$ where $\tilde{A} := \frac{1}{4}A'^{-1}$. We therefore compute the shortest vector $w^\star \in \mathbb{Z}^n \setminus \{0\}$ by the algorithm in Theorem 5.6 with respect to the norm $\|\cdot\|_{\tilde{A}}$ and we are guaranteed that

$$\max_{x\in E'} \langle w^\star, x\rangle - \min_{x\in E'} \langle w^\star, x\rangle \leq n(n + d + 1).$$

All mixed-integer points must lie on the hyperplanes $\{(x, y) \in \mathbb{R}^n \times \mathbb{R}^d : \langle w^\star, x\rangle \in \mathbb{Z}\}$. Moreover, since $C \subseteq E$ and E' is the projection of E, it suffices to search over the "slices" of C given by $C \cap \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^d : \langle w^\star, x\rangle = m\}$ for $m = \lceil\langle w^\star, c'\rangle - n(n+d+1)\rceil, \lceil\langle w^\star, c'\rangle - n(n+d+1)\rceil + 1, \ldots, \lfloor\langle w^\star, c'\rangle + n(n+d+1)\rfloor$. By a change of coordinates in the integer constrained variables, these slices involve $n - 1$ integer variables and we recurse on these subproblems. We also note that if there exists $z \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ with a ball of radius δ around z contained in C, then the slice containing z will also have the same property. Thus, if the algorithm fails on all the slices, then the algorithm will indeed report correctly that there is no such point in C.[2]

Number of oracle calls and overall complexity. Within any particular level of the recursion, the algorithm makes at most $5(n + d)(n + d + 1)^2 \ln\left(\left(\frac{R}{\delta}\right)^{n+d}\right)$ iterations of constructing new ellipsoids in Case 1 or Case 2a. This is because we start with a ball of radius R, stop after the volume of the ellipsoid gets smaller than the volume of a ball of radius δ, and the volume is reduced by a factor of at least $e^{-\frac{1}{5(n+d)(n+d+1)^2}}$ every time. Thus, at most $5(n + d)(n + d + 1)^2 \ln\left(\left(\frac{R}{\delta}\right)^{n+d}\right)$ oracle calls are made within every level of the recursion. The recursion over $2n(n + d + 1)$ subproblems leads to at most $(2n(n + d + 1))^n = 2^{O(n\log(n+d))}$ subproblems. Putting everything together we obtain the bound stated in the theorem on the number of separation oracle calls.

[2]A subtlety here is to make sure that one has access to a separation oracle for the lower dimensional subproblems. This is not hard to implement given access to a separation oracle for C: given a point in the new space, one maps back to $\mathbb{R}^n \times \mathbb{R}^d$ and queries the separation oracle there.

There are two computation intensive steps beyond the separation oracle calls: 1) Computing c_i, E_i from c_{i-1}, E_{i-1} based on Theorem 5.4, and 2) Solving closest vector and shortest vector problems using the algorithm in Theorem 5.6. The first has complexity $O(n+d)^2$ and the second has complexity $2^{O(n)}$ times a polynomial factor of the sizes of the matrices involved. Both 1) and 2) above may result in irrational numbers if we perform exact computations. Since we wish to remain in the Turing machine model of computation, one has to make sure that we can round these to a polynomial number of bits and the sizes of the numbers do not grow exponentially. We omit these technical details from this presentation and refer the reader to [78]. Once all of this is taken into account, we obtain the bound on the overall complexity of the algorithm stated in the theorem. Using these careful approximation techniques, the space complexity of the algorithm can in fact be bounded by a polynomial in the parameters of the problem (note that the computational (time) complexity is *not* polynomial) [69, 78]. □

Theorem 5.8. Consider the family of problems of the form (3.1) such that if $(x^\star, y^\star) \in \mathbb{Z}^n \times \mathbb{R}^d$ is the optimum solution, then there exists $\hat{y} \in \mathbb{R}^d$ and $\rho > 0$ such that $\{(x,y) : \|(x,y) - (x^\star, \hat{y})\|_2 \leq \rho\} \subseteq C$, i.e., there is a "strictly feasible" point $(x^\star, \hat{y})$ in the same fiber as the optimum $(x^\star, y^\star)$ with a Euclidean ball of radius ρ in $\mathbb{R}^n \times \mathbb{R}^d$ around $(x^\star, \hat{y})$ contained in C. Let $\mathcal{I}_{n,d,R,M,\rho}$ be defined as in Section 4 for this family, except that $\|\cdot\|_2$ is used instead of $\|\cdot\|_\infty$ in the definition of the parameters R, M, ρ, and M is defined with respect to the full space $\mathbb{R}^n \times \mathbb{R}^d$, as opposed to just $\mathbb{R}^d$.

Let the oracle access to an instance f, C in $\mathcal{I}_{n,d,R,M,\rho}$ be through a separation oracle for C and a first-order oracle for f. Then for every $\epsilon > 0$, there exists an ϵ-approximation algorithm $\mathcal{A}$ for this problem class with

$$\text{icomp}_{\mathcal{A}}(n,d,R,M,\rho) \leq 2^{O(n\log(n+d))}\left(\log\left(\frac{MR}{\rho\epsilon}\right)\right)^2$$

and

$$\text{comp}_{\mathcal{A}}(n,d,R,M,\rho) \leq 2^{O(n\log(n+d))}\,\text{poly}\left(\log\left(\frac{MR(n+d)}{\rho\epsilon}\right)\right)$$

Proof. If $\epsilon > 2MR$, then any feasible solution is an ϵ-approximate solution, so we may simply run the feasibility algorithm from Theorem 5.7 with $\delta := \rho$. Thus, we assume that $\frac{\epsilon}{2MR} \leq 1$.

We use a standard binary search technique to reduce the problem to a feasibility problem. In particular, we use the algorithm in Theorem 5.7 to test if $C \cap \{z : f(z) \leq \gamma\} = \emptyset$ for some guess γ of the optimum value OPT. Lemma 4.7 implies that for $\gamma \geq OPT + \frac{\epsilon}{2}$, the set $C \cap \{z : f(z) \leq \gamma\}$ contains a Euclidean ball of radius $\delta := \frac{\rho\epsilon}{4MR}$ centered at a mixed-integer point in $\mathbb{Z}^n \times \mathbb{R}^d$ (note that because of our strict feasbiility assumption both $z^\star$ and a can be taken as mixed-integer points in the same fiber when applying Lemma 4.7). Thus, for $\gamma \in [OPT + \frac{\epsilon}{2}, OPT + \epsilon]$, the algorithm in Theorem 5.7 will compute $\hat{z} \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ with $f(\hat{z}) \leq \gamma \leq OPT + \epsilon$.

Since the difference between the maximum and the minimum values of f over the Euclidean ball of radius R is at most $2MR$, we need to make at most $\log\left(\frac{4MR}{\epsilon}\right)$ guesses for γ in the binary search. The result now follows from the complexity bounds in Theorem 5.7. □

Remark 5.9. For ease of exposition, we first presented a feasibility algorithm in Theorem 5.7 and then reduced the optimization problem to the feasibility problem using binary search in Theorem 5.8. One can do away with the binary search in the following way. If $\epsilon > 2MR$, then any feasible solution is an ϵ-approximate solution, so we may simply run the feasibility algorithm from Theorem 5.7 with $\delta := \rho$. Otherwise, we follow the feasibility algorithm from Theorem 5.7 with $\delta := \frac{\rho\epsilon}{2MR}$. Since $\delta \leq \rho$, the algorithm is guaranteed to visit feasible points in Case 1 or 2a of the proof of Theorem 5.7. Once we find a feasible point, we can query the first-order oracle of the objective function f at this feasible point. Any subgradient inequality/halfspace that shaves off this point satisfies the condition in Theorem 5.4, similar to the analysis of Case 1 or 2a in Theorem 5.7. One appeals to Theorem 5.4 to obtain a new ellipsoid with reduced volume and the algorithm continues with this new ellipsoid. At the end, the algorithm selects the feasible point with the smallest objective value amongst all the feasible points it visits. This is similar to the idea in the proof of the upper bound in Theorem 4.2. Since the binary search is eliminated, one obtains a slightly better information complexity of $\text{icomp}_{\mathcal{A}}(n,d,R,M,\rho) \leq 2^{O(n\log(n+d))}\log\left(\frac{MR}{\rho\epsilon}\right)$.

Remark 5.10. [Pure continuous case] Stronger results can be obtained in the pure continuous case, i.e., $n = 0$. First, in Case 1 of the algorithm, we use $\beta = 0$ instead of $\beta = \frac{1}{n+d+1}$, reducing the volume of the ellipsoid by a factor $e^{-\frac{1}{5(n+d)}}$ every time. Thus we make a factor of $(n+d+1)^2$ less number of iterations in Case 1 of the proof of Theorem 5.7. Moreover, there is no recursion needed and thus, the algorithm's information complexity is $O\left(d^2 \log\left(\frac{MR}{\rho\epsilon}\right)\right)$ with an additional computational overhead of $O(d^2)$ for computing the new ellipsoids. This is the classical *ellipsoid algorithm* for convex optimization. Thus, one obtains an ϵ-approximation algorithm for the optimization problem that differs only by a factor of the dimension d from the ϵ-information complexity bound given in Theorem 4.2[3]. Vaidya [127] designed an algorithm whose information complexity matches Theorem 4.2's ϵ-information complexity bound of $O\left(d \log\left(\frac{MR}{\rho\epsilon}\right)\right)$, with the same overall complexity as the ellipsoid algorithm. See [4, 89, 97] for improvements on the overall complexity of Vaidya's algorithm. Lemma 5.4 with $\beta > 0$ is also used in continuous convex optimization under the name of the *shallow cut ellipsoid method*; see [78] for details.

Remark 5.11. [Pure integer case] For the pure integer case, i.e., $d = 0$ one can strengthen both Theorems 5.7 and 5.8 by removing the "strict feasibility" type assumptions. In particular, one can prove a variant of Theorem 5.7 with an exact feasibility algorithm that either reports a point in $C \cap \mathbb{Z}^n$ or correctly decides that $C \cap \mathbb{Z}^n = \emptyset$. One observes that if the volume of the ellipsoid in Case 2a falls below $\frac{1}{n!}$, one can be sure that all integer points in C lie on a single hyperplane. This is because otherwise there are affinely independent points $x_1, \ldots, x_{n+1} \in C \cap \mathbb{Z}^n$ and the convex hull of these points has volume at least $\frac{1}{n!}$. Thus, we can recurse on the lower dimensional problem. For more details see [41]. Another approach is to simply stop the iterations in Case 2a when the ellipsoid has volume less than 1. Then one can show that there is a translate of this ellipsoid that does not intersect $\mathbb{Z}^n$. Applying Theorem 5.5, one can again find n lower dimensional slices to recurse on. This idea was explored in [112] for polyhedral outer and inner approximations. We thus obtain an exact optimization algorithm with information complexity $2^{O(n \log(n))} \log(R)$ and overall complexity $2^{O(n \log(n))} \text{poly}(\log(nR))$.

Remark 5.12. The information or overall complexity bounds presented in Theorems 5.7 and 5.8 are not the best possible ones. There is a general consensus in the discrete optimization community that the right bound is $2^{O(n \log n)} \text{poly}\left(d, \log\left(\frac{MR}{\rho\epsilon}\right)\right)$. Thus, the dependence on the dimensions n (number of integer variables) and d (number of continuous variables) is $2^{O(n \log n)} \text{poly}(d)$ instead of $2^{n \log(n+d)} = (n+d)^{O(n)}$. In other words, the degree of the polynomial function of d is independent of n in the new stated bound.

How can this be achieved? Observe that if one could work with simply the projection of the convex set on to the space of the integer variables, then one can reduce the problem to the pure integer case discussed in Remark 5.11 (assuming one has at least some integer constrained variables; otherwise, one defaults to Remark 5.10 for the continuous case). Indeed, this was the idea originally presented for the mixed-integer *linear* case in Lenstra's paper [98]. In the general nonlinear setting, this can be achieved if one can design a separation oracle for projections of convex sets, given access to separation oracles for the original set, that runs in time polynomial in n, d. This can be done via a result that is colloquially called "equivalence of separation and optimization". This circle of ideas roughly says the following: given access to a separation oracle to a convex set, one can optimize linear functions over it in time that is polynomial in the dimension of the convex set (and the parameters R, ρ, ϵ and the objective vector size), and conversely, if one can optimize over the set one can implement a separation oracle by making polynomially many calls to the optimization oracle. The first part of this equivalence is simply a restatement of Theorems 5.7 and 5.8; in fact, one is only concerned with the continuous case. We refer the reader to [78, 101] for details on the full equivalence. Coming back to projections: using the separation oracle for the original set, one can implement an optimization oracle for it. This optimization oracle gives an optimization oracle over the projection since optimizing a linear function over the projection is the same as optimizing over the original set. Using the

[3] There is a slight discrepancy because of the use of the $\|\cdot\|_\infty$-norm for the information complexity bound (see Theorem 4.2), and the use of $\|\cdot\|_2$-norm here. This adds a $\log(d)$ factor to the complexity of the ellipsoid algorithm, compared to the information complexity bound. We are not aware of any work that resolves this discrepancy.

equivalence, this gives a separation oracle for the projection. Now one can appeal to the arguments in Remark 5.11.

However, all of these arguments are quite delicate and necessarily require very careful approximations. Thus, while these arguments should in principle work, the author is not aware of any source in the literature where all the tedious details have been fully worked out, except in the rational, linear case from Lenstra's original paper [98]. Our exposition here is considerably simpler because it avoids these technicalities, but this is at the expense of the weaker bounds stated in Theorems 5.7 and 5.8. In Lenstra's original way of doing this, the equivalence of separation and optimization is not needed since he works directly with an optimization oracle for (rational) linear programming for which an exact polynomial time algorithm has been known since Khachiyan's work [91], which builds on the ellipsoid algorithm discussed in Remark 5.10. See [98] for more details.

Brief historical comments. The ideas presented in this section are refinements and improvements over seminal ideas of Lenstra [98], and hence our tribute in the title of this section. His original 1983 paper investigated the mixed-integer *linear* case, i.e., when C is a polytope and f is a linear function [98]. His insights were soon extended to handle the general nonlinear case in [78]. Kannan [90] achieved a breakthrough in the complexity bounds – improving from $2^{O(n^3)}$ dependence on the number of integer variables to $2^{O(n\log n)}$ – by modifying the algorithm to recurse over lower dimensional affine spaces, as opposed to hyperplanes as discussed above. We refer to [41, 80, 83, 92] for a representative sample of important papers since then. See also Fritz Eisenbrand's excellent survey chapter in [64]. To the best of the author's knowledge, in the pure integer case the sharpest constant in the exponent of $2^{O(n\log n)}$ is derived in Daniel Dadush's Ph.D. thesis. This requires the use of highly original and technically deep ideas [41].

5.3 Pruning, nontrivial cutting planes and branch-and-cut

The algorithm presented in Section 5.2 utilizes only trivial cutting planes (see Definition 5.1) and solves the optimization problem by reducing to the feasibility problem via binary search. Modern solvers for mixed-integer optimization utilize nontrivial cutting planes and also use a crucial ingredient called *pruning*. We now present the general framework of *branch-and-cut* methods which incorporate both these techniques. The algorithms in Section 5.2 will then be seen to be essentially special instances of such methods.

Definition 5.13. A family $\mathcal{D}$ of disjunctions is called a *branching scheme*. A *cutting plane paradigm* is a map $\mathcal{CP}$ that takes as input any closed, convex set C and $\mathcal{CP}(C)$ is a family of cutting planes for $C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$. $\mathcal{CP}(C)$ may contain trivial and/or nontrivial cutting planes, and may even be empty for certain inputs C.

Example 5.14.

1. *Chvátal-Gomory cutting plane paradigm:* Given any convex set $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$, define

$$\mathcal{CP}(C) := \{H' : H' = \operatorname{conv}(H \cap (\mathbb{Z}^n \times \mathbb{R}^d)),\ H \text{ rational halfspace with } H \supseteq C\}.$$

   $H' \in \mathcal{CP}(C)$ is nontrivial if and only if H is of the form $\{(x,y) \in \mathbb{Z}^n \times \mathbb{R}^d : \langle a, x\rangle \le b\}$ for some $a \in \mathbb{Z}^n$ with relatively prime coordinates and $b \notin \mathbb{Z}$, in which case $H' = \{(x,y) \in \mathbb{Z}^n \times \mathbb{R}^d : \langle a, x\rangle \le \lfloor b \rfloor\}$.

2. *Disjunctive cuts:* Given any family of disjunctions (branching scheme) $\mathcal{D}$, the *disjunctive cutting plane paradigm based on* $\mathcal{D}$ is defined as

$$\mathcal{CP}(C) := \{H' \text{ halfspace} : H' \supseteq C \cap D,\ D \in \mathcal{D}\}.$$

   The collection of halfspaces H' valid for $C \cap D$ are said to be the *cutting planes derived from the disjunction* D. These are valid cutting planes since $\mathbb{Z}^n \times \mathbb{R}^d \subseteq D$ by definition of a disjunction, and therefore $C \cap (\mathbb{Z}^n \times \mathbb{R}^d) \subseteq C \cap D \subseteq H'$. A disjunction D produces nontrivial cutting planes for a compact, convex set C if and only if at least one extreme point of C is not contained in D.

Remark 5.15. We obtain a specific branch-and-cut procedure once we specify the following things in the framework above.

Branch-and-cut framework based on a branching scheme $\mathcal{D}$ and cutting plane paradigm $\mathcal{CP}$

```
Input: A closed, convex set $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$, a convex function $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$, error guarantee $\epsilon > 0$, and
a relaxation $X \subseteq \mathbb{R}^n \times \mathbb{R}^d$ which is closed, convex and contains $C$.
Output: A point $z^\star \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ such that $f(z^\star) \leq OPT + \epsilon$, where $OPT = \inf\{f(z) : z \in C\}$.

1. Initialize a set $L = \{X\}$. Initialize $UB = +\infty$.
2. While $L \neq \emptyset$ do:
      a. [Node selection] Select an element $N \in L$ and update $L := L \setminus \{N\}$.
      b. [Pruning] If it can be verified that $\inf\{f(z) : z \in C \cap N\} \geq UB - \epsilon$, then continue the While loop.
         Else, select a test point $\hat{z}$ in $N$.
      c. If $\hat{z} \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$, obtain a subgradient $h \in \partial f(\hat{z})$ and add the subgradient halfspace
         $H = \{z : \langle h, z - \hat{z} \rangle \leq 0\}$ to all the elements in $L$, i.e., update $N := N \cap H$ for all $N \in L$.
         Additionally, if $f(\hat{z}) < UB$, then update $UB = f(\hat{z})$ and $z^\star = \hat{z}$.
      d. If $\hat{z} \notin C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$, decide whether to BRANCH or CUT.

            If BRANCH, then choose a disjunction $D = Q_1 \cup \ldots \cup Q_k$ in $\mathcal{D}$ such that $\hat{z} \notin D$.
            Select sets (relaxations) $N_1, \ldots, N_2$ such that $N \cap Q_i \subseteq N_i$.
            Update $L := L \cup \{N_1, \ldots, N_k\}$.

            If CUT, then choose a cutting plane $H \in \mathcal{CP}(C \cap N)$ such that $\hat{z} \notin H$.
            Select a set $N'$ such that $N \cap H \subseteq N'$. Update $L := L \cup \{N'\}$.
```

1. In Step 2a., we must decide on a strategy to select an element from L. In the case of the algorithms presented in Section 5.2, this would be the choice of a "slice" to recurse on.

2. In Step 2b., we must decide on a strategy to verify the condition $\inf\{f(z) : z \in N\} \geq UB + \epsilon$. In the case of the algorithms presented in Section 5.2, this is determined by a volume condition on N (which is an ellipsoid). Another common strategy is used in linear integer optimization, where $C \cap N$ is a polyhedron and linear optimization methods like the simplex method or an interior-point algorithm is used to determine $\inf\{f(z) : z \in C \cap N\}$. More generally, one could have a convex optimization subroutine suitable for the class of problems under study.

3. In Step 2b., $\inf\{f(z) : z \in C \cap N\} < UB + \epsilon$ and one must select a test point $\hat{z} \in N$ and one must have a procedure/subroutine for this. In the algorithms presented in Section 5.2, this was chosen as the center of the ellipsoid in Step I, and in Step II it was chosen using the CVP subroutine (and a convex quadratic minimization over the corresponding fiber if the CVP returned a point in the inner ellipsoid). In most solvers, this test point is taken as an optimal or ϵ-approximate solution to the convex optimization problem $\inf\{f(z) : z \in C \cap N\}$ or $\inf\{f(z) : z \in N\}$.

4. In Step 2d., one must have a strategy for deciding whether to branch or to cut, and in either case have a strategy for selecting a disjunction or a cutting plane. The decision to branch might fail because there is no disjunction D in the chosen branching scheme $\mathcal{D}$ that does not contain $\hat{z}$. In such a case, we simply continue the While loop. In the algorithms from Section 5.2, the disjunction family used was the split disjunctions defined in Example 5.2: the "slices" can be seen as branching on the disjunctions $\{(x, y) : \langle w, x\rangle \leq j\} \cup \{(x, y) : \langle w, x\rangle \geq j + 1\}$.

   If the decision is to add a cutting plane, one may add a trivial cutting plane valid for $C \cap N$, as was done in the algorithms in Section 5.2. One may also fail to find a cutting plane that removes $\hat{z}$, because either the cutting plane paradigm can produce no such cutting plane, i.e., $\mathcal{CP}(C \cap N) = \emptyset$, or because the strategy chosen fails to find such a cutting plane in $\mathcal{CP}(C \cap N)$ even though one exists. In such a case, we simply continue the While loop.

Finally, in Step 2d., we must have a strategy to select the relaxations $N_1, \ldots, N_k$ if the decision is to branch, or we must have a strategy to select a relaxation N' if the decision is to cut. In the algorithms in Section 5.2, these relaxations were taken as ellipsoids.

5. If an algorithm based on the branch-and-cut framework above decides never to branch in Step 2d., it is called a *pure cutting plane* algorithm. If an algorithm decides never to use cutting planes in Step 2d., it is called a *pure branch-and-bound* algorithm.

Remark 5.16. In most solvers, the relaxations N_i are simply taken to be $N \cap Q_i$ in a decision to branch, and the relaxation N' is simply taken to be $N \cap H$ in a decision to cut. However, as mentioned above, in the algorithms from Section 5.2 we consider ellipsoidal relaxations of the $N \cap H$ after a (trivial) cutting plane is added, and ellipsoidal relaxations of the "slices".

Does this help? In practice, pruning and nontrivial cutting planes make a huge difference [22, 23, 100]. Turning these off will bring most of the solvers to a grinding halt on even small scale problems. Nevertheless, from a theoretical perspective, researchers have not been able to improve on the $2^{O(n \log(n+d))}$ algorithm from Section 5.2 by utilizing pruning and nontrivial cutting planes for the general problem; see [12, 60, 106] for examples of positive results in restricted settings. Another empirical fact is that if branching is completely turned off and only cutting planes are used, then again the solvers' performance degrades massively. Recently, some results have been obtained that provide some theoretical basis to these empirical observations that the combination of branching and cutting planes performs significantly better than branching alone or using cutting planes alone. We present some of these results now.

The next definition is inspired by the following simple intuition. It has been established that certain branching schemes can be simulated by certain cutting plane paradigms in the sense that for the problem class under consideration, if we have a pure branch-and-bound algorithm based on the branching scheme, then there exists a pure cutting plane algorithm for the same class that has complexity at most a polynomial factor worse than the branch-and-bound algorithm. Similarly, there are results that establish the reverse. See [10, 11, 16, 43, 44, 67], for example. In such situations, combining branching and cutting planes into branch-and-cut is likely to give no substantial improvement since one method can always do the job of the other, up to polynomial factors.

Definition 5.17. Let $\mathcal{I}$ be a family of mixed-integer convex optimization problems of the form (3.1), along with a size hierarchy (see Definition 3.1). To make the following discussion easier, we assume that the objective function is linear. This is without loss of generality since we can introduce an auxiliary variable v, introduce the epigraph constraint $f(z) \leq v$ and use the linear objective "inf v".

A cutting plane paradigm $\mathcal{CP}$ and a branching scheme $\mathcal{D}$ are *complementary for* $\mathcal{I}$ if there is a family of instances $\mathcal{I}_{\mathcal{CP}>\mathcal{D}} \subseteq \mathcal{I}$ such that there is a pure cutting plane algorithm based on $\mathcal{CP}$ that has polynomial (in the size of the instances) complexity and any branch-and-bound algorithm based on $\mathcal{D}$ is exponential (in the size of the instances), and there is another family of instances $\mathcal{I}_{\mathcal{CP}<\mathcal{D}} \subseteq \mathcal{I}$ where $\mathcal{D}$ gives a polynomial complexity pure branch-and-bound algorithm while any pure cutting plane algorithm based on $\mathcal{CP}$ is exponential.

We wish to formalize the intuition that branch-and-cut is expected to be exponentially better than branch-and-bound or cutting planes alone for complementary pairs of branching schemes and cutting plane paradigms. But we need to make some mild assumptions about the branching schemes and cutting plane paradigms. *All known branching schemes and cutting plane methods from the literature satisfy the following conditions.*

Definition 5.18. A branching scheme is said to be *regular* if no disjunction involves a continuous variable, i.e., each polyhedron in the disjunction is described using inequalities that involve only the integer constrained variables.

A branching scheme $\mathcal{D}$ is said to be *embedding closed* if disjunctions from higher dimensions can be applied to lower dimensions. More formally, let n_1, n_2, d_1, $d_2 \in \mathbb{N}$. If $D \in \mathcal{D}$ is a disjunction in $\mathbb{R}^{n_1} \times \mathbb{R}^{d_1} \times \mathbb{R}^{n_2} \times \mathbb{R}^{d_2}$ with respect to $\mathbb{Z}^{n_1} \times \mathbb{R}^{d_1} \times \mathbb{Z}^{n_2} \times \mathbb{R}^{d_2}$, then the disjunction $D \cap (\mathbb{R}^{n_1} \times \mathbb{R}^{d_1} \times \{0\}^{n_2} \times \{0\}^{d_2})$, interpreted as a set in $\mathbb{R}^{n_1} \times \mathbb{R}^{d_1}$, is also in $\mathcal{D}$ for the space $\mathbb{R}^{n_1} \times \mathbb{R}^{d_1}$ with respect to $\mathbb{Z}^{n_1} \times \mathbb{R}^{d_1}$ (note that $D \cap (\mathbb{R}^{n_1} \times \mathbb{R}^{d_1} \times \{0\}^{n_2} \times \{0\}^{d_2})$, interpreted as a set in $\mathbb{R}^{n_1} \times \mathbb{R}^{d_1}$, is certainly a disjunction with respect to $\mathbb{Z}^{n_1} \times \mathbb{R}^{d_1}$; we want $\mathcal{D}$ to be closed with respect to such restrictions).

A cutting plane paradigm $\mathcal{CP}$ is said to be *regular* if it has the following property, which says that adding "dummy variables" to the formulation of the instance should not change the power of the paradigm. Formally, let $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$ be any closed, convex set and let $C' = \{(x, t) \in \mathbb{R}^n \times \mathbb{R}^d \times \mathbb{R} : x \in C, \;\; t = \langle f, x \rangle\}$ for some $f \in \mathbb{R}^n$. Then if a cutting plane $\langle a, x \rangle \leq b$ is derived by $\mathcal{CP}$ applied to C, i.e., this inequality is in $\mathcal{CP}(C)$, then it should also be in $\mathcal{CP}(C')$, and conversely, if $\langle a, x \rangle + \mu t \leq b$ is in $\mathcal{CP}(C')$, then the equivalent inequality $\langle a + \mu f, x \rangle \leq b$ should be in $\mathcal{CP}(C)$.

A cutting plane paradigm $\mathcal{CP}$ is said to be *embedding closed* if cutting planes from higher dimensions can be applied to lower dimensions. More formally, let $n_1, n_2, d_1, d_2 \in \mathbb{N}$. Let $C \subseteq \mathbb{R}^{n_1} \times \mathbb{R}^{d_1}$ be any closed, convex set. If the inequality $\langle c_1, x_1 \rangle + \langle a_1, y_1 \rangle + \langle c_2, x_2 \rangle + \langle a_2, y_2 \rangle \leq \gamma$ is a cutting plane for $C \times \{0\}^{n_2} \times \{0\}^{d_2}$ with respect to $\mathbb{Z}^{n_1} \times \mathbb{R}^{d_1} \times \mathbb{Z}^{n_2} \times \mathbb{R}^{d_2}$ that can be derived by applying $\mathcal{CP}$ to $C \times \{0\}^{n_2} \times \{0\}^{d_2}$, then the cutting plane $\langle c_1, x_1 \rangle + \langle a_1, y_1 \rangle \leq \gamma$ that is valid for $C \cap (\mathbb{Z}^{n_1} \times \mathbb{R}^{d_1})$ should also belong to $\mathcal{CP}(C)$.

A cutting plane paradigm $\mathcal{CP}$ is said to be *inclusion closed*, if for any two closed convex sets $C \subseteq C'$, we have $\mathcal{CP}(C') \subseteq \mathcal{CP}(C)$. In other words, any cutting plane derived for C' can also be derived for a subset C.

Theorem 5.19. [11, Theorem 1.12] Let $\mathcal{D}$ be a regular, embedding closed branching scheme and let $\mathcal{CP}$ be a regular, embedding closed, and inclusion closed cutting plane paradigm such that $\mathcal{D}$ includes all variable disjunctions and $\mathcal{CP}$ and $\mathcal{D}$ form a complementary pair for a mixed-integer convex optimization problem class $\mathcal{I}$. Then there exists a family of instances in $\mathcal{I}$ such that there exists a polynomial complexity branch-and-cut algorithm, whereas any branch-and-bound algorithm based on $\mathcal{D}$ and any cutting plane algorithm based on $\mathcal{CP}$ are of exponential complexity.

The rough idea of the proof of Theorem 5.19 is to embed pairs of instances from $\mathcal{I}_{\mathcal{CP} > \mathcal{D}}$ and $\mathcal{I}_{\mathcal{CP} < \mathcal{D}}$ as faces of a convex set such that a single variable disjunction results in these instances as the subproblems in a branch-and-cut algorithm. On one subproblem, one uses cutting planes and on the other subproblem one uses branching. However, since $\mathcal{CP}$ and $\mathcal{D}$ are complementary, a pure cutting plane or pure branch-and-bound algorithm takes exponential time in processing one or the other of the faces. The details get technical and the reader is referred to [11].

Example 5.20. We now present a concrete example of a complementary pair that satisfies the other conditions of Theorem 5.19. Let $\mathcal{I}$ be the family of mixed-integer linear optimization problems described in point 2. of Example 1.3, with standard "binary encoding" oracles described in point 2. of Example 1.5 and size hierarchy as defined in point 2. of Example 3.2. Let $\mathcal{CP}$ to be the Chvátal-Gomory paradigm (point 1. in Example 5.14) and $\mathcal{D}$ to be the family of variable disjunctions (point 1. in Example 5.2). They are both regular and $\mathcal{D}$ is embedding closed. The Chvátal-Gomory paradigm is also embedding and inclusion closed.

Consider the so-called "Jeroslow instances": For every $n \in \mathbb{N}$, $\max\{\sum_{i=1}^n x_i : \sum_{i=1}^n x_i \leq \frac{n}{2}, \quad x \in [0,1]^n, \quad x \in \mathbb{Z}^n\}$. The single Chvátal-Gomory cut $\sum_{i=1}^n x_i \leq \lfloor \frac{n}{2} \rfloor$ proves optimality, whereas any branch-and-bound algorithm based on variable disjunctions has complexity at least $2^{\lfloor \frac{n}{2} \rfloor}$ [88]. On the other hand, consider the set $T_h \in \mathbb{R}^2$, where $T_h = \text{conv}\{(0,0), (1,0), (\frac{1}{2}, h)\}$ and consider the family of problems for $h \in \mathbb{N}$: $\max\{x_2 : x \in T, \quad x \in \mathbb{Z}^2\}$. Any Chvátal-Gomory paradigm based algorithm has exponential complexity in the size of the input, i.e., every proof has length at least $\Omega(h)$ [123]. On the other hand, a single disjunction on the variable x_1 solves the problem.

Example 5.20 shows that the classical Chvátal-Gomory cuts and variable branching are complementary and thus Theorem 5.19 implies that they give rise to a superior branch-and-cut routine when combined, compared to their stand-alone use. The Chvátal-Gomory cutting plane paradigm and variable disjunctions are the most widely used pairs in state-of-the-art branch-and-cut solvers. We thus have some theoretical basis for explaining the success of this particular combination.

In [10, 11], the authors explore whether certain widely used cutting plane paradigms and branching schemes form complementary pairs. We summarize their results here in informal terms and refer the two papers cited for precise statements and proofs.

1. *Lift-and-project* cutting planes (disjunctive cutting planes based on variable disjunctions – see point 2. in Example 5.14) and variable disjunctions are *not* a complementary pair for 0/1 pure integer convex optimization problems, i.e., $C \subseteq [0,1]^n$. Any branch-and-bound algorithm based on variable disjunctions can be simulated by a cutting plane algorithm with the same complexity[4]. Moreover, there are instances of graph stable set problems where there is a lift-and-project cutting plane algorithm with polynomial complexity, but any branch-and-bound algorithm based on variable disjunctions has exponential complexity. See Theorems 2.1 and 2.2 in [10], and results in [43, 44].

2. Lift-and-project cutting planes and variable disjunctions *do form* a complementary pair for *general* mixed-integer convex optimization problems, i.e., C is not restricted to be in the 0/1 hypercube[5]. See Theorems 2.2 and 2.9 in [10].

3. Split cutting planes (disjunctive cutting planes based on split disjunctions – see point 2. in Example 5.14) and split disjunctions are *not* a complementary pair for general pure integer convex problems. Any cutting plane algorithm based on split cuts can be simulated by a branch-and-bound algorithm based on split disjunctions with the same complexity (up to constant factors)[6]. See Theorem 1.8 in [11].

Connections to proof complexity. Obtaining concrete lower bounds for branch-and-cut algorithms has a long history within the optimization, discrete mathematics and computer science communities. In particular, there is a rich interplay of ideas between optimization and proof complexity arising from the fact that branch-and-cut can be used to certify emptiness of sets of the form $P \cap \{0,1\}^n$, where P is a polyhedron. This question is of fundamental importance in computer science because the *satisfiability* question in logic can be modeled in this way. We provide here a necessarily incomplete but representative sample of references for the interested reader [16, 25, 26, 30, 32–38, 42–45, 61, 66, 67, 72, 73, 77, 86, 96, 117–120].

6 Discussion and open questions

Our presentation above necessarily selected a small subset of results in the vast literature on the complexity of optimization algorithms, even restricted to convex mixed-integer optimization. We briefly discuss three major areas that were left out of the discussion above.

Mixed-integer linear optimization. If we consider the problem class discussed in point 2. of Example 1.3, with algebraic oracles as described in point 2. of Example 1.5, then our ϵ-information complexity bounds from Theorem 4.2 to do not apply anymore. Firstly, we have a much more restricted class of problems. Secondly, the algebraic oracles seem to be more powerful than separation oracles in the following precise sense. Using the standard size hierarchy on this class based on “binary encodings” discussed in point 2. of Example 3.2, a separation/first-order oracle can be implemented easily with the standard algebraic oracle from Example 1.5 in polynomial time, but it is not clear if a separation oracle can implement the algebraic oracle in polynomial time (see point 4. under “Open Problems” below). Moreover, the ϵ-information complexity with respect to the algebraic oracle is bounded by the size of the instance, because once we know all the entries of A, B, b, c, there is no ambiguity in the problem anymore. Nevertheless, it is well-known

[4]There is a technical problem that arises here between the notions of algorithm and *proof*. We have omitted all discussions of cutting plane and branch-and-bound proofs here, which are powerful tools to prove unconditional lower bounds on these algorithms. The precise statement is that any branch-and-bound proof based on variable disjunctions can be replaced by a lift-and-project cutting plane proof of the same size. See [10] for details.

[5]“Lift-and-project cuts” here mean disjunctive cutting planes based on the variable disjunctions (typically the phrase “lift-and-project” is reserved for 0/1 problems).

[6]The same caveat as in point 1. regarding algorithms versus proofs applies.

that the problem class is NP-hard [71]. Therefore, unless $P = NP$, the computational complexity of any algorithm for this class is not polynomial and under the so-called *exponential time hypothesis (ETH)*, it is expected to be exponential.

Nevertheless, several rigorous complexity bounds have been obtained with respect to difference parameterizations of this problem class. We summarize them here with pointers to the references. We note here that the algorithms presented in Section 5.2 are variants of an algorithm that was designed by Lenstra for this problem class of mixed-integer linear optimization [98]. We discuss below approaches that are completely different in nature.

1. *Dynamic programming based algorithms.* We restrict to pure integer instances, i.e., $d = 0$ and assume all the entries of A, b, c are integer. We parameterize such instances by three parameters n, m, Δ, W. $\mathcal{I}_{m,n,\Delta,W}$ are those instances with dimension is at most n, where A has at most m rows, the absolute values of the entries of A are bounded by Δ, and the absolute values of b are bounded by W. In this setting, Papadimitriou designed a dynamic programming based algorithm with complexity $O((n+m)^{2m+2}(m \cdot \max\{\Delta, W\})^{(m+1)(2m+1)})$ [115]. This was improved recently to $O((m\Delta)^m \cdot (n+m)^3 W)$ by Eisenbrand and Weismantel by a clever use of the so-called Steinitz lemma [124]. Subsequent improvements were achieved by Jansen and Rohwedder [87]. Matching lower bounds, subject to the exponential time hypothesis, have been established in [68, 93].

2. *Fixed subdeterminants.* Restricting to pure integer instances with integer data as in the previous point, another paramterization that has been considered is by n, m, Δ, where $\mathcal{I}_{m,n,\Delta}$ are those instances with dimension is at most n, where A has at most m rows, the absolute values of $n \times n$ subdeterminants of A are bounded by Δ. When $\Delta = 1$, a classical result in integer optimization shows that one can simply solve the linear optimization problem to obtain the optimal integer solution [123]. In 2009, Veselov and Chirkov showed that the feasibility problem can be solved in polynomial time when $\Delta = 2$ [128], and the optimization version was resolved using deep techniques from combinatorial optimization by Artmann, Weismantel and Zenkulsen in [6]. See also related results for general Δ in [5, 13, 74–76, 114].

3. *Parameterizations based on the structure of A.* Over the past 25 years, a steady line of research has used algebraic and combinatorial techniques to design algorithms for pure integer optimization that exploit the structure of the constraint matrix A. Several parameters of interest have been defined based on different structural aspects of A. Listing all the literature here is impossible. Instead, we point to [19, 40, 48, 65, 102, 113] and the references therein as excellent summaries and starting points for exploring this diverse field of research activity. An especially intriguing aspect of these algorithms is that they search for the optimal solution by iteratively moving from one feasible solution to a better one. In contrast, the "Lenstra-style" algorithm presented in Section 5.2 approaches the optimal solution "from outside" in a sense by constructing outer (ellipsoidal) approximations. Thus, the newer algebraic and combinatorial algorithms are more in the spirit of *interior point methods* in nonlinear optimization. Exploring the possibility of a unified "interior point style" algorithm for convex mixed-integer optimization would contribute further to bridging the continuous and discrete sides of mathematical optimization.

Mixed-integer polynomial optimization. Instead of linear constraints and objectives as in MILPs, one can consider the problem with polynomial constraints and objective. The standard oracle is algebraic as well in the sense that one can query for the value of a coefficient in a constraint or the objective. The literature on this problem is vast and touches on classical fields like algebraic geometry and diophantine equations. In fact, Hilbert's 10th problem in his famous list of 23 problems presented before the International Congress of Mathematicians in 1900 asks for the construction of an algorithm that solves polynomial equations in integer variables [82]. This problem was proven to be undecidable in a line of work spanning several decades [103]. Thus, while the completely general mixed-integer polynomial optimization problem cannot be solved algorithmically, an enormous literature exists on restricted versions of the problem. For example, with no integer variables we enter the realm of real algebraic geometry or the existential theory of reals where the problem is decidable; see [15] with references to a rich literature. Thus, if one has or can infer finite bounds on the integer constrained decision variables, then the undecidability issues go away since one can enumerate over the integer variables and reduce to the existential theory of reals. The literature is too vast to summarize here; instead, we point the reader to the excellent survey [95] and the textbook expositions in [19, 48, 113], along with the references therein. Some recent progress has appeared in [20, 49–59, 84].

Continuous convex optimization. The information and algorithmic complexity of continuous convex optimization presented in Sections 4 and 5 barely touch the vast literature in this area. For instance, instead of parameterizing by d, M, R, ρ, modern day emphasis is placed on "dimension independent" complexity. For instance, if one focuses on unconstrained minimization instances and parameterizes the class with only two parameters: a parameter M that is usually related to the Lipschitz constant of the objective function or its derivatives (if we further restrict to (twice) differentiable functions), and R is a parameter that says the optimal solution is contained in a Euclidean ball of radius R. It is important to note that the dimension is not part of the parameterization. In the nonsmooth case, one can establish matching upper and lower bounds of $O\left(\frac{MR}{\epsilon^2}\right)$ on the information complexity of a broad class of iterative ϵ-approximation algorithms. These can be improved to $O\left(\frac{MR}{\sqrt{\epsilon}}\right)$ in the smooth case. We will fail to do justice to this enormous and active area of research in this manuscript and instead point the reader to the excellent monographs [29, 110].

Nonconvex continuous optimization. The complexity landspace for nonconvex continuous optimization has recently seen a lot of activity and has reached a level of development paralleling the corresponding development in convex optimization complexity. Here the solution operator $S(I, \epsilon)$ typically is defined as ϵ-approximate stationary points or local minima; for instance, in the smooth unconstrained case, $S(I, \epsilon)$ is the set of all points where the gradient norm is at most ϵ. We recommend the recent thesis of Yair Carmon for a fantastic survey, pointers to literature and several new breakthroughs [31].

Open Questions

1. As mentioned at the beginning of Section 5.2, a major open question in mixed-integer optimization is to bridge the gap of $2^{O(n)}\log(R)$ tight bound on the information complexity and the $2^{O(n\log n)}\log(R)$ algorithmic complexity of Lenstra style algorithms presented in Section 5.2, for pure integer optimization. It seems that radically new ideas are needed and Lenstra style algorithms cannot escape the $2^{O(n\log n)}$ barrier.

2. Closing the gap between the lower and the upper bounds for ϵ-information complexity in Theorem 4.2 for the truly mixed-integer case, i.e., $n, d \geq 1$ seems to be a nontrivial problem. See the discussion after the statement of Theorem 4.2 for the two different sources of discrepancy. In particular, Conjecture 4.6 seems to be a very interesting question in pure convex/discrete geometry. It was first stated in Timm Oertel's thesis [111], and the thesis and [14] contain some partial results towards resolving it. Further, taking into account the sizes of the responses to oracle queries merits further study. To the best of our knowledge, tight lower and upper bounds are not known when the size of the responses are taken into account, i.e., when one uses the strict definition of ϵ-information complexity as in Definition 1.7 of this paper. Finally, we believe that the study of ϵ-information complexity of optimization with respect to oracles that are different from subgradient oracles is worth investigation.

3. The mixed-integer linear optimization problem class (point 2. in Example 1.3) can be accessed through two different oracles: the standard algebraic one described in point 2. of Example 1.5, or a separation oracle. Assuming the standard size hierarchy obtained from the binary encoding sizes, it is not hard to implement a separation oracle using the algebraic oracle with polynomially many algebraic oracle calls. However, it is not clear if the algebraic oracle can be implemented in polynomial time by a separation oracle. Essentially, one has to enumerate the facets of the polyhedral feasible region $P \subseteq \mathbb{R}^k$, which is equivalent to enumerating the vertices of the polar $P^\star$ (after finding an appropriate center in P –

see [101, Lemma 2.2.6]). This can be done if we have an optimization oracle for $P^\star$ since one can iteratively enumerate the vertices by creating increasingly better inner polyhedral approximations. Roughly speaking, one maintains the current list of enumerated vertices $v_1, \ldots, v_t$ of $P^\star$ and also an inequality description of $\text{conv}(\{v_1, \ldots, v_t\})$. Now we optimize over $P^\star$ in the direction of all the inequalities describing $\text{conv}(\{v_1, \ldots, v_t\})$. If $\text{conv}(\{v_1, \ldots, v_t\}) \subsetneq P^\star$, then at least one new vertex will be discovered and we can continue the process after reconvexifying. An optimization oracle for the polar $P^\star$ can be implemented in polynomial time because we have a separation oracle for P [78, 101, 123]. The only issue is that $\text{conv}(\{v_1, \ldots, v_t\})$ may need t^k inequalities in its description, which is not polynomial unless we consider the dimension k to be fixed. It would be good to resolve whether the separation and algebraic oracles for mixed-integer linear optimization are polynomially equivalent.

4. Theorem 5.19 shows that a complementary pair of branching scheme and cutting plane paradigm can lead to substantial gains when combined into a branch-and-cut algorithm, as opposed to a pure branch-and-bound or pure cutting plane algorithm. Is this a characterization of when branch-and-cut is provably better? In other words, if a branching scheme and cutting plane paradigm are such that there exists a family of instances where branch-and-cut is exponentially better than branch-and-bound or cutting planes alone, then is it true that the branching scheme and the cutting plane paradigm are complementary? A result showing that branch-and-cut is superior if and only if the branching scheme and cutting plane paradigm are complementary would be a tight theoretical characterization of this important phenomenon. While this general question remains open, a partial converse to Theorem 5.19 was obtained in [11]:

   Theorem 6.1. [11, Theorem 1.14] Let $\mathcal{D}$ be a branching scheme that includes all split disjunctions and let $\mathcal{CP}$ be any cutting plane paradigm. Suppose that for every pure integer instance and any cutting plane proof based on $\mathcal{CP}$ for this instance, there is a branch-and-bound proof based on $\mathcal{D}$ of size at most a polynomial factor (in the size of the instance) larger. Then for any branch-and-cut proof based on $\mathcal{D}$ and $\mathcal{CP}$ for a pure integer instance, there exists a pure branch-and-bound proof based on $\mathcal{D}$ that has size at most polynomially larger than the branch-and-cut proof.

Acknowledgement

The author benefited greatly from discussions with Daniel Dadush at CWI, Amsterdam and Timm Oertel at FAU, Erlangen-Nürmberg. Comments from two anonymous referees helped the author significantly to consolidate the material, improve its presentation and make tighter connections to the existing literature on the complexity of optimization.

References

[1] Fred G. Abramson. Effective computation over the real numbers. In *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*, pages 33–37. IEEE Computer Society, 1971.

[2] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling. In *Proceedings of the forty-seventh annual ACM Symposium on Theory of computing (STOC)*, pages 733–742. ACM, 2015.

[3] Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the closest vector problem in 2^n time–the discrete gaussian strikes again! In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 563–582. IEEE, 2015.

[4] Kurt M. Anstreicher. On Vaidya's volumetric cutting plane method for convex programming. *Mathematics of Operations Research*, 22(1):63–89, 1997.

[5] Stephan Artmann, Friedrich Eisenbrand, Christoph Glanzer, Timm Oertel, Santosh Vempala, and Robert Weismantel. A note on non-degenerate integer programs with small sub-determinants. *Operations Research Letters*, 44(5):635–639, 2016.

[6] Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1206–1219. ACM, 2017.

[7] Gennadiy Averkov and Robert Weismantel. Transversal numbers over subsets of linear spaces. *Adv. Geom.*, 12(1):19–28, 2012.

[8] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in R^n II: Application of K-convexity. *Discrete & Computational Geometry*, 16(3):305–311, 1996.

[9] Wojciech Banaszczyk, Alexander E. Litvak, Alain Pajor, and Stanislaw J. Szarek. The flatness theorem for nonsymmetric convex bodies via the local theory of Banach spaces. *Mathematics of Operations Research*, 24(3):728–750, 1999.

[10] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of cutting plane and branch-and-bound algorithms for mixed-integer optimization. To appear in Mathematical Programming, 2019.

[11] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of cutting plane and branch-and-bound algorithms for mixed-integer optimization–II. To appear in Combinatorica, 2020.

[12] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Split cuts in the plane. *SIAM Journal on Optimization*, 31(1):331–347, 2021.

[13] Amitabh Basu and Hongyi Jiang. Enumerating integer points in polytopes with bounded subdeterminants. *SIAM Journal on Discrete Mathematics*, 36(1):449–460, 2022.

[14] Amitabh Basu and Timm Oertel. Centerpoints: A link between optimization and convex geometry. *SIAM Journal on Optimization*, 27(2):866–889, 2017.

[15] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry.* Springer Science & Business Media, 2006.

[16] Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing Planes. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[17] Michael J. Beeson. *Foundations of constructive mathematics: Metamathematical studies*, volume 6. Springer Science & Business Media, 2012.

[18] David E. Bell. A theorem concerning the integer lattice. *Studies in Applied Mathematics*, 56(2):187–188, 1977.

[19] Dimitris Bertsimas and Robert Weismantel. *Optimization over Integers.* Dynamic Ideas, Belmont, MA, May 2005.

[20] Daniel Bienstock, Alberto Del Pia, and Robert Hildebrand. Complexity, exactness, and rationality in polynomial optimization. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 58–72. Springer, 2021.

[21] Errett Bishop. *Foundations of constructive analysis*, volume 5. McGraw-Hill New York, 1967.

[22] Robert E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2012.

[23] Robert E. Bixby, Mary Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling. Mixed integer programming: A progress report. In *The Sharpest Cut*, pages 309–325. MPS-SIAM Series on Optimization, Philadelphia, PA, 2004.

[24] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: W-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc*, 21(1):1–46, 1989.

[25] Alexander Bockmayr, Friedrich Eisenbrand, Mark Hartmann, and Andreas S Schulz. On the Chvátal rank of polytopes in the 0/1 cube. *Discrete Applied Mathematics*, 98(1-2):21–27, 1999.

[26] Maria Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997.

[27] Allan Borodin and Ian Munro. *The computational complexity of algebraic and numeric problems.* American Elsevier, New York, 1975.

[28] Gábor Braun, Cristóbal Guzmán, and Sebastian Pokutta. Lower bounds on the oracle complexity of nonsmooth convex optimization via information theory. *IEEE Transactions on Information Theory*, 63(7):4709–4724, 2017.

[29] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*, 2014.

[30] Samuel R. Buss and Peter Clote. Cutting planes, connectivity, and threshold logic. *Archive for Mathematical Logic*, 35(1):33–62, 1996.

[31] Yair Carmon. *The Complexity of Optimization beyond Convexity.* PhD thesis, Stanford University, August 2020.

[32] Vašek Chvátal. Hard knapsack problems. *Operations Research*, 28(6):1402–1411, 1980.

[33] Vašek Chvátal. *Cutting-plane proofs and the stability number of a graph, Report Number 84326-OR.* Institut für Ökonometrie und Operations Research, Universität Bonn, Bonn, 1984.

[34] Vašek Chvátal, William J. Cook, and Mark Hartmann. On cutting-plane proofs in combinatorial optimization. *Linear algebra and its applications*, 114:455–499, 1989.

[35] Peter Clote. Cutting planes and constant depth frege proofs. In *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 296–307, 1992.

[36] William J. Cook, Collette R. Coullard, and Gy Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

[37] William J. Cook and Sanjeeb Dash. On the matrix-cut rank of polyhedra. *Mathematics of Operations Research*, 26(1):19–30, 2001.

[38] William J. Cook and Mark Hartmann. On the complexity of branch and cut methods for the traveling salesman problem. *Polyhedral Combinatorics*, 1:75–82, 1990.

[39] William J. Cook, Ravindran Kannan, and Alexander Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47:155–174, 1990.

[40] William H. Cunningham and Jim Geelen. On integer programming and the branch-width of the constraint matrix. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 158–166. Springer, 2007.

[41] Daniel Dadush. *Integer programming, lattice algorithms, and deterministic volume estimation.* ProQuest LLC, Ann Arbor, MI, 2012. Thesis (Ph.D.)–Georgia Institute of Technology.

[42] Daniel Dadush and Samarth Tiwari. On the complexity of branching proofs. *arXiv preprint arXiv:2006.04124*, 2020.

[43] Sanjeeb Dash. An exponential lower bound on the length of some classes of branch-and-cut proofs. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 145–160. Springer, 2002.

[44] Sanjeeb Dash. Exponential lower bounds on the lengths of some classes of branch-and-cut proofs. *Mathematics of Operations Research*, 30(3):678–700, 2005.

[45] Sanjeeb Dash. On the complexity of cutting-plane proofs using split cuts. *Operations Research Letters*, 38(2):109–114, 2010.

[46] Sanjeeb Dash, Neil B. Dobbs, Oktay Günlük, Tomasz J. Nowicki, and Grzegorz M. Świrszcz. Lattice-free sets, multi-branch split disjunctions, and mixed-integer programming. *Mathematical Programming*, 145(1-2):483–508, 2014.

[47] Sanjeeb Dash and Oktay Günlük. On t-branch split cuts for mixed-integer programs. *Mathematical Programming*, 141(1-2):591–599, 2013.

[48] Jesús A De Loera, Raymond Hemmecke, and Matthias Köppe. *Algebraic and geometric ideas in the theory of discrete optimization.* SIAM, 2012.

[49] Alberto Del Pia. On approximation algorithms for concave mixed-integer quadratic programming. *Mathematical Programming*, 172(1):3–16, 2018.

[50] Alberto Del Pia. Subdeterminants and concave integer quadratic programming. *SIAM Journal on Optimization*, 29(4):3154–3173, 2019.

[51] Alberto Del Pia, Santanu S Dey, and Marco Molinaro. Mixed-integer quadratic programming is in np. *Mathematical Programming*, 162(1):225–240, 2017.

[52] Alberto Del Pia and Silvia Di Gregorio. On the complexity of binary polynomial optimization over acyclic hypergraphs. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2684–2699. SIAM, 2022.

[53] Alberto Del Pia, Robert Hildebrand, Robert Weismantel, and Kevin Zemmer. Minimizing cubic and homogeneous polynomials over integers in the plane. *Mathematics of Operations Research*, 41(2):511–530, 2016.

[54] Alberto Del Pia and Aida Khajavirad. A polyhedral study of binary polynomial programs. *Mathematics of Operations Research*, 42(2):389–410, 2017.

[55] Alberto Del Pia and Aida Khajavirad. The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization*, 28(2):1049–1076, 2018.

[56] Alberto Del Pia and Aida Khajavirad. The running intersection relaxation of the multilinear polytope. *Mathematics of Operations Research*, 46(3):1008–1037, 2021.

[57] Alberto Del Pia, Aida Khajavirad, and Nikolaos V Sahinidis. On the impact of running intersection inequalities for globally solving polynomial optimization problems. *Mathematical programming computation*, 12(2):165–191, 2020.

[58] Alberto Del Pia and Matthias Walter. Simple odd β-cycle inequalities for binary polynomial optimization. *to appear in Proceedings of IPCO 2022*.

[59] Alberto Del Pia and Robert Weismantel. Integer quadratic programming in the plane. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 840–846, 2014.

[60] Santanu S. Dey, Yatharth Dubey, and Marco Molinaro. Branch-and-bound solves random binary packing IPs in polytime. *arXiv preprint arXiv:2007.15192*, 2020.

[61] Santanu S. Dey, Yatharth Dubey, and Marco Molinaro. Lower bounds on the size of general branch-and-bound trees. *arXiv preprint arXiv:2103.09807*, 2021.

[62] J.-P. Doignon. Convexity in cristallographical lattices. *J. Geometry*, 3:71–85, 1973.

[63] Rodney G. Downey and Michael R. Fellows. *Parameterized complexity.* Springer Science & Business Media, 2012.

[64] Friedrich Eisenbrand. Integer programming and algorithmic geometry of numbers. In M. Jünger, T. Liebling, D. Naddef, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming 1958–2008*. Springer-Verlag, 2010.

[65] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Kouteckỳ, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *arXiv preprint arXiv:1904.01361*, 2019.

[66] Friedrich Eisenbrand and Andreas S. Schulz. Bounds on the Chvátal rank of polytopes in the 0/1-cube. *Combinatorica*, 23(2):245–261, 2003.

[67] Noah Fleming, Mika Göös, Russell Impagliazzo, Toniann Pitassi, Robert Robere, Li-Yang Tan, and Avi Wigderson. On the power and limitations of branch and cut. *arXiv preprint arXiv:2102.05019*, 2021.

[68] Fedor V. Fomin, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Fine-grained complexity of integer programming: The case of bounded branch-width and rank. *arXiv preprint arXiv:1607.05342*, 2016.

[69] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[70] Harvey Friedman. Algorithmic procedures, generalized turing algorithms, and elementary recursion theory. In *Studies in Logic and the Foundations of Mathematics*, volume 61, pages 361–389. Elsevier, 1971.

[71] Michael R. Garey and David S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[72] Andreas Goerdt. Cutting plane versus frege proof systems. In *International Workshop on Computer Science Logic*, pages 174–194. Springer, 1990.

[73] Andreas Goerdt. The cutting plane proof system with bounded degree of falsity. In *International Workshop on Computer Science Logic*, pages 119–133. Springer, 1991.

[74] Dmitry V. Gribanov and Aleksandr Y. Chirkov. The width and integer optimization on simplices with bounded minors of the constraint matrices. *Optimization Letters*, 10(6):1179–1189, 2016.

[75] Dmitry V. Gribanov, Dmitriy S. Malyshev, and Panos M. Pardalos. A note on the parametric integer programming in the average case: sparsity, proximity, and fpt-algorithms. *arXiv preprint arXiv:2002.01307*, 2020.

[76] Dmitry V. Gribanov and Sergey I. Veselov. On integer programming with bounded determinants. *Optimization Letters*, 10(6):1169–1177, 2016.

[77] Dima Grigoriev, Edward A. Hirsch, and Dmitrii V. Pasechnik. Complexity of semi-algebraic proofs. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 419–430. Springer, 2002.

[78] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics: Study and Research Texts*. Springer-Verlag, Berlin, 1988.

[79] Branko Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific J. Math.*, 10:1257–1261, 1960.

[80] Sebastian Heinz. Complexity of integer quasiconvex polynomial optimization. *Journal of Complexity*, 21(4):543–556, 2005.

[81] Eduard Helly. Über mengen konvexer körper mit gemeinschaftlichen punkte. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923.

[82] David Hilbert. Mathematische probleme. *Göttinger Nachrichten*, pages 253–297, 1900.

[83] Robert Hildebrand and Matthias Köppe. A new lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity $2^{O(n \log n)}$. *Discrete Optimization*, 10(1):69–84, 2013.

[84] Robert Hildebrand, Robert Weismantel, and Kevin Zemmer. An fptas for minimizing indefinite quadratic forms over integers in polyhedra. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1715–1723. SIAM, 2016.

[85] Alan J. Hoffman. Binding constraints and Helly numbers. *Annals of the New York Academy of Sciences*, 319:284–288, 1979.

[86] Russell Impagliazzo, Toniann Pitassi, and Alasdair Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 220–228. IEEE, 1994.

[87] Klaus Jansen and Lars Rohwedder. On integer programming, discrepancy, and convolution. *arXiv preprint arXiv:1803.04744*, 2018.

[88] Robert G. Jeroslow. Trivial integer programs unsolvable by branch-and-bound. *Mathematical Programming*, 6(1):105–109, Dec 1974.

[89] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020.

[90] Ravindran Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[91] Leonid Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[92] Leonid Khachiyan and Lorant Porkolab. Integer optimization on convex semialgebraic sets. *Discrete & Computational Geometry*, 23(2):207–224, 2000.

[93] Dusan Knop, Michal Pilipczuk, and Marcin Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Transactions on Computation Theory (TOCT)*, 12(3):1–19, 2020.

[94] Ker-I Ko. *Complexity theory of real functions.* Birkhauser Boston Inc., 1991.

[95] Matthias Köppe. On the complexity of nonlinear mixed-integer optimization. In *Mixed Integer Nonlinear Programming*, pages 533–557. Springer, 2012.

[96] Jan Krajíček. Discretely ordered modules as a first-order extension of the cutting planes proof system. *The Journal of Symbolic Logic*, 63(4):1582–1596, 1998.

[97] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

[98] Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[99] Yanjun Li and Jean-Philippe P Richard. Cook, Kannan and Schrijver's example revisited. *Discrete Optimization*, 5(4):724–734, 2008.

[100] Andrea Lodi. Mixed integer programming computation. In *50 Years of Integer Programming 1958-2008*, pages 619–645. Springer, 2010.

[101] László Lovász. *An algorithmic theory of numbers, graphs, and convexity*, volume 50. SIAM, 1986.

[102] Susan Margulies, Jing Ma, and Illya V Hicks. The cunningham-geelen method in practice: branch-decompositions and integer programming. *INFORMS Journal on Computing*, 25(4):599–610, 2013.

[103] Yuri Matiyasevich. *Hilbert's 10th problem.* MIT Press, 1993.

[104] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations [extended abstract]. In *Proceedings of the 2010 ACM International Symposium on Theory of Computing (STOC)*, pages 351–358. ACM, New York, 2010.

[105] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

[106] Mohammad Javad Naderi, Austin Buchanan, and Jose L Walteros. Worst-case analysis of clique MIPs. *http://www.optimization-online.org/DB_HTML/2021/01/8198.html*, 2021.

[107] Arkadii Nemirovski. Efficient methods in convex programming. *Lecture notes*, 1994.

[108] Arkadii S. Nemirovski and David B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley, 1983.

[109] Yurii E. Nesterov. *Introductory Lectures on Convex Optimization*, volume 87 of *Applied Optimization.* Kluwer Academic Publishers, Boston, 2004.

[110] Yurii E. Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[111] Timm Oertel. *Integer Convex Minimization in Low Dimensions.* PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 22288, 2014.

[112] Timm Oertel, Christian Wagner, and Robert Weismantel. Integer convex minimization by mixed integer linear optimization. *Operations Research Letters*, 42(6–7):424 – 428, 2014.

[113] Shmuel Onn. Nonlinear discrete optimization. *Zurich Lectures in Advanced Mathematics, European Mathematical Society*, 2010.

[114] Joseph Paat, Miriam Schlöter, and Robert Weismantel. The integrality number of an integer program. *Mathematical Programming*, pages 1–21, 2021.

[115] Christos H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

[116] Marian Boykan Pour-El and Ian Richards. Computability and noncomputability in classical analysis. *Transactions of the American Mathematical Society*, 275(2):539–560, 1983.

[117] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.

[118] Pavel Pudlák. On the complexity of the propositional calculus. *London Mathematical Society Lecture Note Series*, pages 197–218, 1999.

[119] Alexander A. Razborov. On the width of semialgebraic proofs and algorithms. *Mathematics of Operations Research*, 42(4):1106–1134, 2017.

[120] Thomas Rothvoß and Laura Sanità. 0/1 polytopes with quadratic Chvátal rank. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 349–361. Springer, 2013.

[121] Mark Rudelson. Distances between non-symmetric convex bodies and the MM^*-estimate. *Positivity*, 4(2):161–178, 2000.

[122] Herbert E Scarf. An observation on the structure of production sets with indivisibilities. *Proceedings of the National Academy of Sciences*, 74(9):3637–3641, 1977.

[123] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, New York, 1986.

[124] Ernst Steinitz. Bedingt konvergente reihen und konvexe systeme. *Journal für die reine und angewandte Mathematik*, 1913.

[125] Joseph Frederick Traub, Grzegorz Włodzimierz Wasilkowski, and Henryk Woźniakowski. *Information, uncertainty, complexity.* Addison-Wesley Reading, MA, 1983.

[126] Alan Mathison Turing. On computable numbers, with an application to the "entscheidungsproblem". a correction. *Proceedings of the London Mathematical Society*, 43(2):544–546, 1937.

[127] Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets. *Math. Program.*, 73(3):291–341, 1996.

[128] Sergey I. Veselov and Aleksandr J. Chirkov. Integer program with bimodular matrix. *Discrete Optimization*, 6(2):220–222, 2009.

[129] David B. Yudin and Arkadii S. Nemirovskii. Informational complexity and efficient methods for the solution of convex extremal problems. *Matekon*, 13(2):22–45, 1976.