2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)

Smoothing the gap between NP and ER

Jeff Erickson
Department of Computer Science
University of Illinois at Urbana-Champaign
Illinois, United States of America
Email: jeffe@illinois.edu

Ivor van der Hoog
Department of Computer Science
Utrecht University
Utrecht, Netherlands
i.d.vanderhog@uu.nl

Tillmann Miltzow
Department of Computer Science
Utrecht University
Utrecht, Netherlands
t.mitzow@googlemail.com

***Abstract*—We study algorithmic problems that belong to the complexity class of the existential theory of the reals (ER). A problem is ER-complete if it is as hard as the problem ETR and if it can be written as an ETR formula. Traditionally, these problems are studied in the real RAM, a model of computation that assumes that the storage and comparison of real-valued numbers can be done in constant space and time, with infinite precision. The complexity class ER is often called a real RAM analogue of NP, since the problem ETR can be viewed as the real-valued variant of SAT. The real RAM assumption that we can represent and compare irrational values in constant space and time is not very realistic. Yet this assumption is vital, since some ER-complete problems have an "exponential bit phenomenon" where there exists an input for the problem, such that the witness of the solution requires geometric coordinates which need exponential word size when represented in binary. The problems that exhibit this phenomenon are NP-hard (since ETR is NP-hard) but it is unknown if they lie in NP. NP membership is often showed by using the famous Cook-Levin theorem which states that the existence of a polynomial-time verification algorithm for the problem witness is equivalent to NP membership. The exponential bit phenomenon prohibits a straightforward application of the Cook-Levin theorem.**

In this paper we first present a result which we believe to be of independent interest: we prove a real RAM analogue to the Cook-Levin theorem which shows that ER membership is equivalent to having a verification algorithm that runs in polynomial-time on a real RAM. This gives an easy proof of ER-membership, as verification algorithms on a real RAM are much more versatile than ETR-formulas.

We use this result to construct a framework to study ER-complete problems under smoothed analysis. We show that for a wide class of ER-complete problems, its witness can be represented with logarithmic input-precision by using smoothed analysis on its real RAM verification algorithm. This shows in a formal way that the boundary between NP and ER (formed by inputs whose solution witness needs high input-precision) consists of contrived input.

We apply our framework to well-studied ER-complete recognition problems which have the exponential bit phenomenon such as the recognition of realizable order types or the Steinitz problem in fixed dimension. Interestingly our techniques also generalize to problems with a natural notion of resource augmentation (geometric packing, the art gallery problem).

***Index Terms*—smoothed analysis, Existential Theory of the Reals, real-RAM, bit-precision, resource augmentation, verification algorithms, robust computations.**

I. Introduction

The RAM is a mathematical model of a computer which emulates how a computer can access and manipulate data. Within computational geometry, algorithms are often analyzed within the real RAM [1], [2], [3] where real values can be stored and compared in constant space and time. By allowing these infinite precision computations, it becomes possible to verify geometric primitives in constant time, which simplifies the analysis of geometric algorithms. Mairson and Stolfi [4] point out that "without this assumption it is virtually impossible to prove the correctness of any geometric algorithms."

Intuitively (for a formal definition, skip ahead to the bottom of page 2) we define the input-precision of a real RAM algorithm A with input I as the minimal word size required to express each input value in I, such that the algorithm executes the same operations in the word RAM as in the real RAM. The downside of algorithm analysis in the real RAM is that it neglects the input-precision required by the underlying algorithms for correct execution, although they are very important in practice. Many algorithms, including some verification algorithms of $\exists\mathbb{R}$-complete problems, inherently require large input-precision [1] and there are even examples which in the worst case require a input-precision exponential in the number of input variables in order to be correctly executed [5], [6].

Often inputs which require exponential input-precision are contrived and do not resemble *realistic* inputs. A natural way to capture this from a theoretical perspective is smoothed analysis, which *smoothly* interpolates between worst case analysis and average case analysis [7]. Practical inputs are constructed inherently with small amount of noise and random perturbation. This perturbation helps to show performance guarantees in terms of the input size and and the magnitude of the perturbation. By now smoothed analysis is well-established, for instance Spielman and Teng received the Gödel Prize for it.

We give a bird's eye view of the paper before providing proper definitions: we study the real RAM and its complexity class $\exists\mathbb{R}$ through the lens of smoothed analysis. We start by proving a result separate of smoothed analysis, which we believe to be of independent interest: we show a real RAM analogue of the famous Cook-Levin theorem as we prove that $\exists\mathbb{R}$ membership is equivalent to the existence of

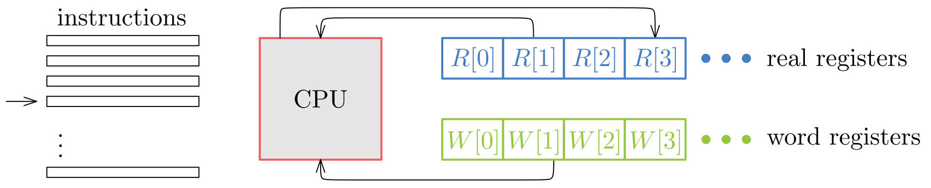

Fig. 1. The dominant model in computational geometry is the real RAM. It consists of a central processing unit, which can operate on real and word registers in constant time, following a set of instructions.

a verification algorithm that runs in polynomial-time on a real RAM. We then use the existence of this verification algorithm to construct a framework with which smoothed analysis can be applied to a wide class of $\exists\mathbb{R}$-complete problems. We show that $\exists\mathbb{R}$-complete recognition problems, with a verification algorithm in the real RAM that has polynomially bounded *arithmetic degree*, have witnesses that can be represented with a logarithmic word size under smoothed analysis. This implies that these problems have polynomial-time verification algorithms on the word RAM that succeed for all but a small set of contrived inputs. Finally, we extend our framework to include $\exists\mathbb{R}$-complete problems that have a natural notion of resource augmentation. This generalizes an earlier result of smoothed analysis of the Art Gallery problem [8].

RAM computations: The Random Access Machine (RAM) is a model of computation for the standard computer architecture. The precise definition of the RAM varies, but at its core the RAM has a number of *registers* and a *central processing unit* (CPU), which can perform operations on register values like reading, writing, comparisons, and arithmetic operations. The canonical model of computation within computer science is the *word RAM*, a variation on the RAM formalized by Hagerup [9] but previously considered by Fredman and Willard [10], [11] and even earlier by Kirkpatrick and Reisch [12]. The word RAM models two crucial aspects of real-life computing: (1) computers must store values with finite precision and (2) computers take more time to perform computations if the input of the computation is longer. Specifically, the word RAM supports constant-time operations on w-bit integers, where the *word size* w is a model parameter.

Many portions of the algorithms community (either explicitly or implicitly) use a different variation of the RAM called the *real RAM*, where registers may contain arbitrary real numbers, instead of just integers; The usage of the real RAM is prevalent in the field of computational geometry but also in probabilistic algorithm analysis where one wants to reason about continuous perturbations of the input. The abstraction offered by the real RAM dramatically simplifies the design and analysis of algorithms, at the cost of working in an physically unrealistic model. Implementations of real RAM algorithms using finite-precision data types are prone to errors, not only because the output becomes imprecise, but because rounding errors can lead the algorithm into inconsistent states. Kettner [13] provides an overview of complications that arise from the unrealistic precision that the real RAM assumes.

Formally modeling real RAM algorithms: The real RAM has been the standard underlying model of computation in computational geometry since the field was founded in the late 1970s [14], [15]. Despite its ubiquity, we are unaware of *any* published definition of the model that is simultaneously precise enough to support our results and broad enough to encompass most algorithms in the algorithm analysis literature. The obvious candidate for such a definition is the real-computation model proposed by Blum, Shub, and Smale [16], [17]; however, this model does not support the integer operations necessary to implement even simple algorithms: even though the real RAM is often presented, either formally or intuitively, as a random access machine that stores and manipulates only exact real numbers, countless algorithms in this model require decisions based on both exact real and finite precision integer values. Consider the following example: given an array of n real values as input, compute their sum. Any algorithm that computes this sum must store and manipulate real numbers; however, the most straightforward algorithm also requires indirect memory access through an *integer* array index. More complex examples include call stack maintenance, discrete symbol manipulation, and multidimensional array indexing and program slicing.

On the other hand, real and integer operations must be combined with care to avoid unreasonable *discrete* computation power. A model that supports both exact constant-time real arithmetic and constant-time conversion between real numbers and integers, for example using the floor function, would also trivially support arbitrary-precision constant-time *integer* arithmetic. (To multiply two integers, cast them both to reals, multiply them, and cast the result back to an integer.) Including such constant-time operations allows any problem in PSPACE to be solved in polynomial-time [18]; see also [12], [19], [20], [21] for similar results.

To accommodate this mixture of real and integer operations, and to avoid complexity pitfalls, we define the real RAM as an extension of the standard integer word RAM [9] (refer to Figure 1). We define the real RAM in terms of a fixed parameter w, called the *word size*. A *word* is an integer between 0 and $2^w - 1$, represented as a sequence of w bits. Mirroring standard definitions for the word RAM, memory consists of two *random access arrays* $W[0\,..\,2^w - 1]$ and $R[0\,..\,2^w - 1]$, whose elements we call *registers*. Both of these arrays are indexed/addressed by words; for any word i, register $W[i]$ is a word and register $R[i]$ is an exact real number.

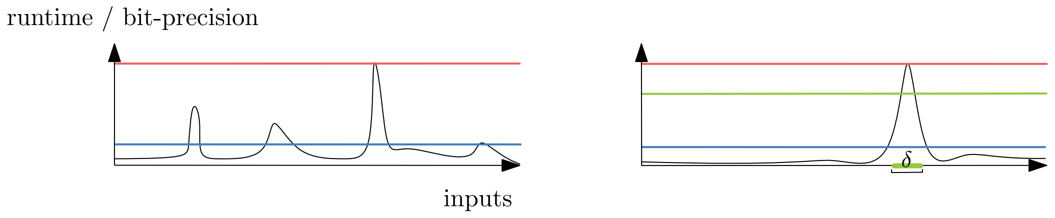

Fig. 2. The x-axis symbolizes all inputs. The red line indicates the worse case cost, the blue line average cost and the green line the cost under smoothed analysis.

A program on the real RAM consists of a fixed, finite indexed sequence of read-only instructions. The machine maintains an integer *program counter*, which is initially equal to 1. At each time step, the machine executes the instruction indicated by the program counter. The goto instruction modifies the program counter directly; the halt and accept and reject instructions halt execution otherwise, the program counter increases by 1 after each instruction is executed.

The input to a real RAM program consists of a pair of vectors $(a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$, for some integers n and m, which are suitably encoded into the corresponding memory arrays before the program begins.[1] To maintain uniformity, we require that neither the input sizes n and m nor the word size w is known to any program at "compile time". The output of a real RAM program consists of the contents of memory when the program executes the halt instruction. The *running time* of a real RAM program is the number of instructions executed before the program halts; each instruction requires one time step by definition. Refer to the full version [22] for a table of the operations and further explanation of the real RAM. Crucially, we consider the real RAM without trigonometric, exponential and logarithmic operations or the square root operator.

Formally defining bit-precision and input-precision: We say two inputs $I = (a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$ and $I' = (a', b) \in \mathbb{R}^n \times \mathbb{Z}^m$ are equivalent with respect to an algorithm A on a real RAM, $(I \cong_A I')$ if every comparison operation gives the same result. In particular, this implies that at every step the program counter is at the same position. For each integer $z \in \mathbb{Z}$, we denote by $\texttt{bit}(z)$ the length of its binary representation, i.e., $\lfloor \log_2 |z| \rfloor + 1$. For each rational number $y = p/q \in \mathbb{Q}$, we denote by $\texttt{bit}(y) := \texttt{bit}(p) + \texttt{bit}(q)$ the length of its binary representation.

First we consider $I = (a, b) \in \mathbb{Q}^n \times \mathbb{Z}^m$ as input for a real RAM algorithm A. We denote by $\texttt{bit}_{\texttt{IN}}(I) = \max_i \max\{\texttt{bit}(a_i), \texttt{bit}(b_i)\}$ the *input bit-length* of I. We denote $C(I)$ as the set of all values of all registers during the execution of A with I, and by $\texttt{bit}(C(I)) = \max_{c \in C(I)} \{\texttt{bit}(c)\}$ as the *execution bit-length* of I.

[1]Following standard practice, we implicitly assume throughout the paper that the integers in the input vector b are actually w-bit *words*; for problems involving larger integers, we take m to be the number of *words* required to encode the integer part of the input.

Now, we are ready to define the bit-precision and input-precision of *real* input. The bit-precision of A with input $I = (a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$ is:

$$\texttt{bit}(I, A) := \min\{\, \texttt{bit}(C(I')) \mid I' \in \mathbb{Q}^n \times \mathbb{Z}^m, I \cong_A I' \,\}.$$

In the same manner, the input-precision is defined as :

$$\texttt{bit}_{\texttt{IN}}(I, A) := \min\{\, \texttt{bit}_{\texttt{IN}}(I') \mid I' \in \mathbb{Q}^n \times \mathbb{Z}^m, I \cong_A I' \,\}.$$

If there is no equivalent input $I' = (a, b) \in \mathbb{Q}^n \times \mathbb{Z}^m$, then we say $\texttt{bit}(I, A) = \texttt{bit}_{\texttt{IN}}(I, A) = \infty$. It is now straightforward to *simulate* an execution of a real RAM algorithm A on input $I = (a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$ on a word RAM with word size $w = O(\texttt{bit}(I, A))$.

Arithmetic degree: Liotta, Preparata and Tamassia [23] studied the required bit-precision for real RAM proximity algorithms. To aid their analysis, they defined the *arithmetic degree* of an algorithm. We express their definition in our real RAM model and add the notion of algebraic dimension for our later analysis. It follows from our definition of real RAM in [22] that at all times during the computation, a real register holds a value which can be described as the quotient of two polynomials $\frac{p}{q}$ of the real input values a. Observe that adding, subtracting, or multiplying two rational functions yields another rational function, possibly of higher degree; for example, $\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1 q_2 + p_2 q_1}{q_1 q_2}$. We say an algorithm A has *arithmetic degree* Δ, if p and q always have total degree at most Δ. Similarly, A has *algebraic dimension* d, if the number of variables in p and q are always at most d. Bounded algebraic dimension and arithmetic degree give an interesting relation between the input-precision and the bit-precision:

Lemma I.1. *Let $I \in (a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$ be some input and A be a real RAM algorithm with $\texttt{bit}_{\texttt{IN}}(I, P) = p$, algebraic dimension d and arithmetic degree Δ. Then its bit-precision is upper bounded by $O(p\Delta^2 \log d)$.*

Proof. If we multiply Δ numbers of bit-length p, the total bit-length is at most Δp. If we sum t numbers of bit-length p, the total bit-length is at most $O(p \log t)$. In a polynomial in d variables, we have at most $O(d^{\Delta})$ monomials. Thus the bit-precision is upper bounded by $O(\log(d^{\Delta})\Delta p) = O(p\Delta^2 \log d)$. □

This allows us to only focus on input-precision in our smoothed analysis which we explain next:

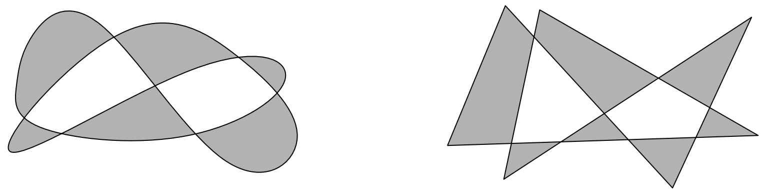

Fig. 3. Given two simple closed curves in the plane it is straightforward to design an algorithm that checks if the two curves are equivalent. But it is not straightforward to describe an ETR formula for it.

Smoothed analysis: In *smoothed analysis*, the performance of an algorithm is studied for worst case input which is randomly perturbed by a magnitude of δ. Intuitively, smoothed analysis interpolates between average case and worst case analysis (Figure 2). The smaller δ, the closer we are to true worst case input. Correspondingly larger δ is closer to the average case analysis. The key difficulty in applying smoothed analysis is that one has to argue about both worst case and average case input. Spielman and Teng explain their analysis by applying it to the simplex algorithm, which was known for a particularly good performance in practice that was seemingly impossible to verify theoretically [24]. Since the introduction of smoothed analysis, it has been applied to numerous problems [25], [26], [27], [28], [29]. Most relevant for us is the smoothed analysis of the art gallery problem [8] and of order types [30], which we generalize in the full version [22].

Formally we define smoothed analysis as follows: let us fix an algorithm A and some $\delta \in [0,1]$, which describes the *magnitude of perturbation*. We denote by $I = (a,b) \in [0,1]^n \times \mathbb{Z}^m$ the input of A. We scale the real-valued input to lie in $[0,1]$, to normalize δ. Unless explicitly stated, we assume that each real number is perturbed *independently uniformly at random* and that the integers stay as they are, since we assume that they already fit into main memory. We denote by $(\Omega_\delta, \mu_\delta)$ the probability space where each $x \in \Omega_\delta$ defines for each instance I a new 'perturbed' instance $I_x = (a+x, b)$. We denote by $\mathcal{C}(I_x)$ the cost of instance I_x. Traditionally in smoothed analysis, this cost is the runtime required for an algorithm in order to compute its solution but in this paper we consider the input-precision and the bit-precision as the cost functions. The expected cost of instance I equals:

$$\mathcal{C}_\delta(I, A) = \mathop{\mathbb{E}}_{x \in \Omega_\delta} [\mathcal{C}(I_x)] = \int_{\Omega_\delta} \mathcal{C}(I_x)\mu_\delta(x)\, \mathrm{d}x.$$

We denote by $\Lambda_{n,m}$ the set of all instances of size $n+m$ (where we implicitly assume that for all integer values b_i, $\mathtt{bit}(b_i) \leq \log\, m$), the smoothed input-precision equals:

$$\mathcal{C}_{\text{smooth}}(\delta, n, A) = \max_{I \in \Lambda_{n,m}} \mathcal{C}_\delta(I, A).$$

This definition formalizes the intuition mentioned before: not only do we require that the majority of instances behave nicely, but actually in every neighborhood (bounded by the maximal perturbation δ) the majority of instances must behave nicely. Following [31], [7] we perceive an algorithm to run in polynomial cost in practice, if the smoothed cost of the algorithm is polynomial in the input size n and in $1/\delta$. If the smoothed cost is small in terms of $1/\delta$ then we have a theoretical verification of the hypothesis that worst case examples are sparse. We defined the input-precision of an algorithm A with input I as the minimal word size required for the correct execution of the algorithm on a word RAM. To model possible disparities between real RAM and word RAM execution we define an operation called *snapping* in the next paragraph.

Snapping: Our real-valued input can be represented as a higher-dimensional point $a \in [0,1]^n$. If we want to express a with only w bits, then the corresponding integer-valued input a' with limited precision is the closest point to a in the scaled integer lattice $\Gamma_\omega = \omega\mathbb{Z}^d$ with $\omega = 2^{-w}$. We call the transformation of a into a' *snapping*. We give a lower bound on the scale factor ω for which $(a,b) \cong_A (a',b)$. This implies an upper bound on the input-precision of A. The algorithms that we study under this snapping operation are algorithms relevant to the complexity class $\exists\mathbb{R}$, which we discuss next.

The Existential Theory of the Reals: It is often easy to describe a potential witness to an NP-hard problem, but the input-precision required to verify the witness is unknown. A concrete example is the recognition of segment intersection graphs: given a graph, can we represent it as the intersection graph of segments? Matoušek [32] comments on this as follows:

Serious people seriously conjectured that the number of digits can be polynomially bounded—but it cannot.

Indeed, there are examples which require an exponential word size in any numerical representation. This *exponential bit phenomenon* occurs not only for segment intersection graphs, but also for many other natural algorithmic problems [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54]. It turns out that all of those algorithmic problems do not accidentally require exponential input-precision, but are closely linked, as they are all complete for a certain complexity class called $\exists\mathbb{R}$. Thus either all of those problems belong to NP, or none of them do. Using our results on smoothed analysis, we show that for many $\exists\mathbb{R}$-hard problems the exponential input-precision phenomenon only occurs for near-degenerate input.

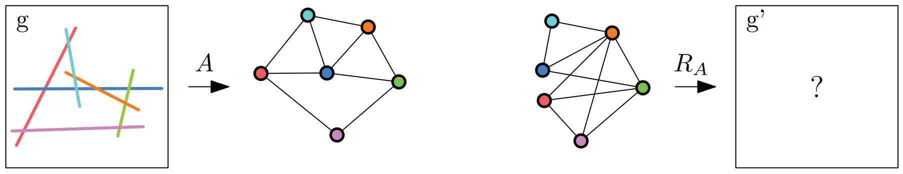

Fig. 4. Left: given a set of segments g, algorithm A constructs segment intersection graph $c = A(g)$. Right: given a graph c', algorithm R_A searches for a set of segments g' such that $A(g') = c'$.

The complexity class $\exists\mathbb{R}$ can be defined as the set of decision problems that are polynomial-time equivalent to deciding if a formula of the *Existential Theory of the Reals* (ETR) is true or not. An ETR formula has the form: $\Psi = \exists x_1, \ldots, x_n \ \ \Phi(x_1, \ldots, x_n)$, where Φ is a well-formed sentence over the alphabet $\Sigma = \{0, 1, x_1, \ldots, +, \cdot, =, \leq, <, \wedge, \vee, \neg\}$. More specifically, Φ is quantifier-free and $x_1, \ldots, x_n$ are all variables of Φ. We say Ψ is true if there are real numbers $x_1, \ldots, x_n \in \mathbb{R}$ such that $\Phi(x_1, \ldots, x_n)$ is true.

II. Algorithmic membership in ER

The complexity class $\exists\mathbb{R}$ is often called a “real analogue” of NP, because it deals with real-valued variables instead of Boolean variables. This is because the real-valued ETR problem plays a role which is analogous to SAT when it comes to hardness and membership. However, the most common way to think about NP-membership is in terms of certificates and verification algorithms.
The seminal theorem of Cook and Levin shows the equivalence of the two perspectives on NP-membership [55], [56]. We show a similar equivalence between ETR-formulas and *real verification algorithms*. Intuitively, a real verification algorithm is an algorithm that runs on the real RAM that accepts as input both an *integer* instance I and a witness, and verifies that the witness describes a valid solution to the instance in polynomial-time.

Formally we say a *discrete decision problem* is any function Q from arbitrary integer vectors to the booleans {TRUE, FALSE}. An integer vector I is called a *yes-instance* of Q if $Q(I) =$ TRUE and a *no-instance* of Q if $Q(I) =$ FALSE. Let $\circ$ denote the concatenation operator. A *real verification algorithm* for Q is formally defined as a real RAM algorithm A that satisfies the following conditions, for some constant $c \geq 1$: (1) A halts after at most N^c time steps given any input of total length N where each value uses word size $\lceil c \log_2 N \rceil$. (2) For every yes-instance $I \in \mathbb{Z}^n$, there is a real vector x and an integer vector z, each of length at most n^c, such that A accepts input $(x, I \circ z)$ and (3) for every no-instance I, for every real vector x and every integer vector z, A rejects input $(x, I \circ z)$. A *certificate* (or *witness*) for yes-instance I is any vector pair (x, z) such that A accepts $(x, I \circ z)$. We show the following theorem:

Theorem II.1. *For any discrete decision problem Q, there is a real verification algorithm for Q $\Leftrightarrow$ $Q \in \exists\mathbb{R}$.*

Our proof in the full version closely follows classical simulation arguments reducing nondeterministic polynomial-time (integer) random access machines to polynomial-size circuits or Boolean formulas, either directly [57] or via nondeterministic polynomial-time Turing machines [58], [21], [59], [56]. We wish to state that the analysis in the full version is broad enough to support the square root operator in the real RAM.

The complexity class $\exists\mathbb{R}$ is known to be equivalent to the discrete portion of the Blum-Shub-Smale complexity class $NP^0_{\mathbb{R}}$—real sequences that can be accepted in polynomial-time by a non-deterministic BSS machine *without constants*, and the equivalence of BSS machines without constants and ETR formulas is already well-known [16], [17]. However as we explained in the introduction, the BSS-machine does not directly support the *integer* computations necessary for common standard programming paradigms such as indirect memory access and multidimensional arrays. The real RAM model originally proposed by Shamos [14], [15] *does* support indirect memory access through integer addresses; however, Shamos did not offer a precise definition of his model, and we are not aware of any published definition precise enough to support our simulation result. We rectify this gap with our precise definition of the real RAM. Our proposal generalizes both the word RAM and BSS models, we believe it formalizes the intuitive model implicitly assumed by computational geometers.

Theorem 2.1 ([22]) not only strengthens the intuitive analogy between NP and $\exists\mathbb{R}$, but also enables much simpler proofs of $\exists\mathbb{R}$-membership in terms of standard geometric algorithms. Our motivation for developing Theorem 2.1 was Erickson's *optimal curve straightening* problem [41]: Given a closed curve γ in the plane and an integer k, is any k-vertex polygon topologically equivalent to γ? (See Figure 3.) The $\exists\mathbb{R}$-hardness of this problem follows from an easy reduction from stretchability of pseudolines, but reducing it directly to ETR proved much more complex; in light of Theorem 2.1, membership in $\exists\mathbb{R}$ follows almost immediately from the standard Bentley-Ottman sweep-line algorithm [60]. The theorem also applies to geometric packing problems [35], where the input is a set of geometric objects and a container and the output is a pairwise disjoint placement of the objects in the container. Its real verification algorithm is straightforward. To further illustrate the power of our technique, we also consider a new topological problem in the full version [22], which we call *optimal unknotted extension*.

Optimal unknotted extension: Given a simple polygonal *path* P in $\mathbb{R}^3$ and an integer k, can we extend P to an *unknotted* closed polygon with at most k additional vertices? In light of Theorem II.1, the proof that this problem is in $\exists\mathbb{R}$ is straightforward: To verify a positive instance, guess the k new vertices and verify that the resulting polygon is unknotted using existing NP algorithms [61], [62].

Corollary II.2. *The following discrete decision problems are in $\exists\mathbb{R}$: The art gallery problem [34], the optimal curve straightening problem [41], geometric packing and, the optimal unknotted extension problem.*

III. Smoothed Analysis of Recognition Problems

In computational geometry, we study many different geometric objects like point sets, polytopes, disks, balls, line-arrangements, segments, and rays. Many algorithms *only* use combinatorial properties of the underlying geometry. A recent impressive example is the EPTAS for the clique problem on disk intersection graphs by Bonamy et al. [63]. In the paper they first derive a set of properties for disk intersection graphs and then they use *only* those properties to find a new EPTAS.

Suppose that a geometric problem can be solved using only precomputed combinatorial properties. Then given the combinatorial structure, we do not need real RAM computations! The crux for this approach is that we first need a family of properties that describe all geometric objects of a certain type. This is the motivation for recognition problems.

Formally (Figure 4), we say A is a *recognition verification algorithm* if it takes some real-valued geometric input $g \in [0,1]^n$ and outputs a combinatorial object $A(g) = c \in \mathbb{Z}^m$ with m polynomial in n. We define a *recognition problem* as a discrete decision problem R_A that takes a combinatorial object c as input, and returns True if there exists a vector $g \in [0,1]^n$ (which we shall call a *witness*) for which $A(g) = c$. For notational convenience, we denote for a given c by $R_A(c)$ *an arbitrary* witness of c (as in the smoothed analysis that is to come, we consider the worst choice over all witnesses for c).

Defining smoothed analysis on recognition problems: Traditionally in smoothed analysis one perturbs the input of an algorithm and measures the expected cost of executing the algorithm with the new input. The real-valued geometric input g offers a straightforward way to perturb it by taking for each $g_i \in [0,1]$ a random offset $x_i \in [\frac{-\delta}{2}, \frac{\delta}{2}]$ uniformly at random. It is not as easy to define a perturbation on the combinatorial (discrete) input that a recognition problem requires. This is why we define the perturbation in terms of the geometric witness: given any input and witness (c, g), we slightly perturb the witness g to a new geometric object $g_x = g + x$ and reconstruct the corresponding combinatorial input c_x (Figure 5). A probability distribution over possible output g_x implies a probability distribution over all input c_x.

We want for an instance c of R_A find a witness g of bounded bit-length. Therefore, our definition is formulated in terms of the input-precision of A:

$$\mathtt{bit}_\delta(g, A) := \mathop{\mathbb{E}}_{x \in \Omega_\delta} \left[\mathtt{bit}_{IN}(g + x, A)\right]$$

$$\mathtt{bit}_\delta(c, R_A) := \mathop{\mathbb{E}}_{x \in \Omega_\delta} \left[\mathtt{bit}_{IN}(g_x, A) \mid g_x = R_A(c) + x\right].$$

Next, we define the smoothed precision. We denote by Λ_m the set of all combinatorial instances c of size m:

$$\mathtt{bit}_{\text{smooth}}(\delta, n, A) = \max_{g \in [0,1]^n} \mathtt{bit}_\delta(g, A)$$

$$\mathtt{bit}_{\text{smooth}}(\delta, m, R_A) = \max_{c \in \Lambda_m} \mathtt{bit}_\delta(c, R_A).$$

Now, we are ready to state our main theorem about smoothed analysis of recognition verification algorithms.

Theorem III.1. *Let A be a polynomial-time recognition verification algorithm with arithmetic degree Δ and algebraic dimension d. Under perturbations of $g \in [0,1]^n$ by $x \in \left[\frac{\delta}{2}, \frac{\delta}{2}\right]^n$ chosen uniformly at random, the algorithm A has a smooth input-precision of at most:*

$$\mathtt{bit}_{\mathit{smooth}}(\delta, n, A) = O\left(d \log \frac{d\Delta n}{\delta}\right).$$

The core idea of this proof (presented at the end of this section and illustrated by Figure 6) is to consider the recognition verification algorithm A with perturbed input $g_x = g + x$, where g is an arbitrary value in $[0,1]^n$ and x is a small perturbation chosen uniformly at random in $\left[\frac{-\delta}{2}, \frac{\delta}{2}\right]^n$. We model the perturbed input g_x as a high-dimensional point which we snap to a fine grid to obtain g' (input which can be described using bounded precision). We then show that for any algorithm A that meets our prerequisites, $\mathtt{bit}_{IN}(g_x, A)$ is low with high probability. For the snapping we consider a sufficiently small ω and we snap the point g_x to a point in $\omega\mathbb{Z}^n$. The Voronoi diagram of the points in $\omega\mathbb{Z}^n$ forms a fine grid in $[0,1]^n$. As we explained in the introduction, the content of a real RAM register for a specific comparison instruction is per assumption the quotient of two polynomials whose variables depend on the input. The core argument is that if the point g_x lies in a Voronoi cell of a point with limited word size which does not intersect the variety of either of the two polynomials, then the comparison instruction will be computed correctly. We upper bound the proportion of Voronoi cells that are intersected by the variety of a polynomial in Theorem III.4.

For the algebraic proof of Theorem III.4 we refer to the full version. Since A is a polynomial-time algorithm, it performs at most a polynomial number of comparison instructions. The union bound over all instructions upper bounds the chance that the execution of A differs for the input g' and g_x. The union bound will show that with high probability, for any perturbed input g_x, $\mathtt{bit}_{IN}(g_x, A)$ is low. We conclude our proof of Theorem III.1 in Section 7 of the full version where we transform that statement into an expected value. Theorem III.1 implies a result of smoothed analysis of recognition problems:

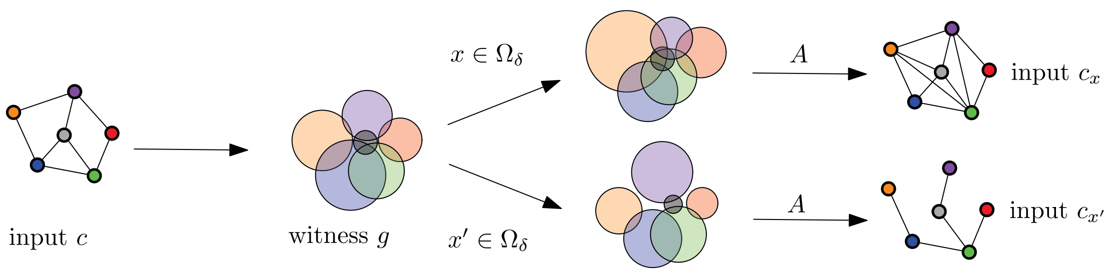

Fig. 5. We define a perturbation on a combinatorial structure c through a witness g. A distribution of witnesses defines a distribution of discrete structures.

Theorem III.2. *Let R_A be a recognition problem with recognition verification algorithm A. If A is a polynomial-time algorithm with constant arithmetic degree and algebraic dimension then*

$$\mathtt{bit}_{smooth}(\delta, m, R_A) = O(\log(m/\delta)).$$

Proof. This can be shown simply by using the definition and the result of Theorem III.1.

$$\begin{aligned} &\mathtt{bit}_{\text{smooth}}(\delta, m, R_A) = \\ &\max_{c\in\Lambda_m} \mathop{\mathbb{E}}_{x\in\Omega_\delta} [\mathtt{bit}_{IN}(g_x, A) \mid g_x = R_A(c) + x] = \\ &\max_{g:A(g)\in\Lambda_m} \mathop{\mathbb{E}}_{x\in\Omega_\delta} [\mathtt{bit}_{IN}(g_x, A) \mid g_x = g + x] \end{aligned}$$

By definition of a recognition verification algorithm, if $c = A(g)$ has size m then g has size $n = \Theta(m^{const})$ for some fixed constant $const > 0$. Thus Theorem III.1 implies:

$$\begin{aligned} \mathtt{bit}_{\text{smooth}}(\delta, m, R_A) &\leq \max_{g\in[0,1]^n} \mathtt{bit}_\delta(g, A) \\ &= \mathtt{bit}_{\text{smooth}}(\delta, n, A) \\ &= O(\log(n/\delta)) = O(\log(m^{const}/\delta)) \\ &= O(\log(m/\delta)). \qquad \square \end{aligned}$$

Corollary III.3. *The following recognition problems under uniform perturbations of the witness of magnitude δ have a smoothed bit-precision of $O(\log n/\delta)$: recognition of realizable order types [30], disk intersection graphs, segment intersection graphs, ray intersection graphs and the Steinitz Problem in fixed dimension.*

Limitations: We want to point out that we cannot handle all recognition problems. First, not all recognition verification algorithms meet the conditions of Theorem III.1. For example, consider the case that the problem deals with unbounded dimension. We still get some bounds on the input-precision but they may be less desirable. A concrete example is the recognition of ball intersection graphs, where the dimension of the ball is part of the input. Second, perturbing a witness may not be sensible. It does not mean that our theorems do not apply in a mathematical sense, but rather that in reality the result may be less desirable. For example, this limitation applies to problems that rely on degeneracies in one way or another. A concrete example is the point visibility graph recognition problem. Given a set of points P, we define a graph by making two points $p, q \in P$ adjacent if line segment pq contains no other point of P. This in turn defines a recognition problem where we are given a visibility graph G and we look for a point set P_G that realizes this visibility graph. If we perturb the real-valued witness P_G then with probability 1 there are no three collinear points. Thus, the point visibility graph will always be a clique.

Preliminaries for proving Theorem III.1: Per definition, the word RAM has a word size w which allows us to express $2^w = \frac{1}{\omega}$ different values for each coordinate. We consider the recognition verification algorithm A with real-valued input $g \in [0,1]^n$. Since A runs in polynomial-time it can make at most a polynomial number of comparison operations. At every such binary decision the algorithm looks at a real- or integer-value register and verifies if the value at the register is 0 or strictly greater than 0. For every real-valued register, per assumption the value at that register is the result of two d-variate polynomials p_i and q_i with maximum degree Δ whose variables depend only on the values in the input register. Let z be the d-dimensional vector of input variables in g that p_i and g_i depend on.

The evaluation of $p_i(z)/q_i(z)$ depends on the evaluation of the polynomials $p_i(z)$ and $q_i(z)$. During smoothed analysis we perturb our input g into new input $g_x = (g + x)$ with x a value chosen uniformly at random in $[\frac{-\delta}{2}, \frac{\delta}{2}]^m$. Thereafter, we snap g_x to g' and we are interested in the probability that the execution of A under both inputs (g_x and g'_x) is the same, ergo the chance that for all comparison operations, the algorithms give the same answer.

Fix a vector $z = (z_1, \ldots, z_d) \in \mathbb{Z}^d$ with integer coordinates. We denote by C_z the unit (hyper)cube, which has z as its minimal corner: $C_z := [0,1]^d + z$. We denote by $C(d,k) := \{C_z \mid z \in \mathbb{Z}^d \cap [0,k)^d\}$ a $(k \times k \times \ldots \times k)$-grid of unit cubes that cover $[0,k]^d$. Let $C = [0,k]^d$ be a cube partitioned by unit cubes $C(d,k)$. Every facet of C intuitively is a grid of (d-1)-dimensional cubes $C(d-1,k)$. We argue about varieties that intersect cubes in $C(d,k)$ by induction on the dimension.

To that end, we define an equivalence relation. Let C be a d-dimensional cube, partitioned by d-dimensional cubes of equal width. We say C is *equivalent* to $C(d,k)$, denoted by $C \cong C(d,k)$, if there exists an affine transformation τ of C such that there is a one-to-one correspondence between cubes in $\tau(C)$ and $C(d,k)$ where corresponding cubes coincide.

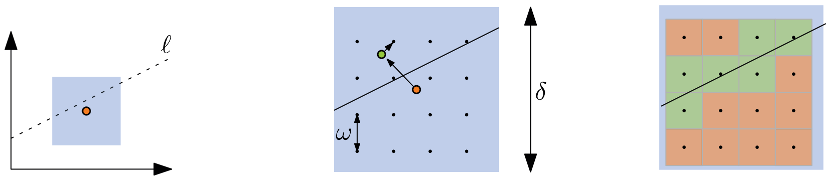

Fig. 6. Given $g = (g_1, g_2) \in [0,1]^2$, we want to decide if the point g (in orange) lies above or below the line ℓ ($y = x/2 + 1$). If g is perturbed slightly to a point g_x, low precision is usually sufficient.

We give to examples of this equivalence relation that is often used in the remainder of the paper: (1) Consider any d, k and any $(d-1)$-dimensional, orthogonal hyperplane H that intersects a d-dimensional cube $C(d,k)$ we have that $C(d,k) \cap H \cong C(d-1,k)$. (2) Consider the d-dimensional grid Γ_ω as defined for the snapping operation, and a cube C which has one corner on the origin and width $k\omega$. The intersection $C \cap \Gamma_\omega$ is equivalent to $C(d,k)$. Using these definitions and a well-known algebraic theorem by Milnor [64] we obtain the following result:

Theorem III.4 (Hitting Cubes)**.** *Let $p \neq 0$ be a d-variate polynomial with maximum degree Δ and let $k \geq 2\Delta + 2$. Then the polynomial p intersects at most $k^{d-1}\Delta 3^d(d+1)!$ unit cubes in $C(d,k)$.*

Proving Theorem III.1: For the smoothed analysis, we *snap* our perturbed real-valued input $g_x = (g+x)$ onto a point with limited precision g', the closest point in $\omega\mathbb{Z}^d \cap [0,1]^d$. Ergo, for all $y \in \omega\mathbb{Z}^d$, all points in the Voronoi cell of y are snapped to y. The Voronoi cells of all these cells (apart from those belonging to grid points on the boundary of $[0,1]^d$) are just d-dimensional cubes and we shall denote a cube centered at y by $C(y)$. We denote $\Gamma_\omega = \{C(y) : y \in \omega\mathbb{Z}^d \cap [0,1]^d\} \cong C(d, 1/\omega)$.

The perturbed point g_x must lie within some cube $C(y)$ in Γ_ω. However the choice of original g and δ limits the cubes in Γ_ω where g_x can lie in. Specifically (Figure 7) the range R of possible locations for g_x is a cube of width δ centered around g. The boundary of this range cube R likely does not align with the boundaries of grid cells in Γ_w. Therefore we make a distinction between two types of cells: the range cube R must contain a cube of cells which is equivalent to $C(d, \lfloor \delta/\omega \rfloor)$ and this sub-cube of R we shall denote by $\Gamma_\omega(g)$. All others cells which are intersected by R but not contained in $\Gamma_\omega(g)$ we call the *perimiter cells*. We can use this observation together with Theorem III.4 to estimate the probability that $\text{sign}(p(g_x)) \neq \text{sign}(p(g'))$:

Lemma III.5. *Let p, q be two d-variate polynomials with maximum degree Δ. Let $g \in \mathbb{R}^d$ be fixed and $x \in \Omega = [-\delta/2, \delta/2]^d$ chosen uniformly at random. Assume $g_x = g + x$ is snapped to a point $g' \in \omega\mathbb{Z}^d \cap [0,1]^d$. Then* $\text{sign}\left(\frac{p(g_x)}{q(g_x)}\right) \neq \text{sign}\left(\frac{p(g')}{q(g')}\right)$ *with probability at most:*

$$(4\,\omega\Delta 3^d(d+1)!)/\delta.$$

Proof. It suffices to show that $\text{sign}(p(g_x)) \neq \text{sign}(p(g'))$ with probability at most:

$$\frac{2\,\omega\Delta 3^d(d+1)!}{\delta}.$$

The statement for q is equivalent and the union bound on these separate probabilities then upper bounds probability that their division does not have the same sign. For any polynomial p and points $x, z \in \mathbb{R}^d$ with $p(x) < 0$ and $p(z) > 0$, there must be a point $y \in \text{line}(x,z)$ for which $p(y) = 0$. It follows that if a cube $C \in \Gamma_\omega$ does not intersect the variety of p, all points in C either have a positive or negative evaluation under p. Let g' be the closest point to g_x in $\omega\mathbb{Z} \cap [0,1]^d$ and let $C(g')$ be its Voronoi cell. If $C(g')$ does not intersect the variety of p then $\text{sign}(p(g_x)) = \text{sign}(p(g'))$. Therefore we are interested in the probability that g_x is contained in a Voronoi cell that is intersected by the variety of p.

As we discussed, the set of possible locations of g_x is a cube which contains $\Gamma_\omega(g) \cong C(d, \lfloor \delta/\omega \rfloor)$ together with a collection of perimeter cells.

We upper bound the probability that g_x lies within a cell that is intersected by p in two steps: (1) We upper bound the number of perimeter cells and make the worst case assumption that they are all intersect by the variety of p. (2) We upper bound the number of cells of $\Gamma_\omega(g)$ that intersect the variety of p. These two numbers, divided by the total number of cells in $\Gamma_\omega(g)$ gives the upper bound we are looking for. Note that the width of $\Gamma_\omega(g)$ is equal to $k = \lfloor \frac{\delta}{\omega} \rfloor$ and that the perimeter of $\Gamma_\omega(g)$ contains $2d \cdot k^{d-1}$ cells. Theorem III.4 gives an upper bound on the number of intersected cubes in $C(d,k)$ of:

$$k^{d-1}\Delta 3^d(d+1)! \leq \left\lfloor \frac{\delta}{\omega} \right\rfloor^{d-1} \Delta 3^d(d+1)!$$

There are $k = \lfloor \delta/\omega \rfloor^d$ cubes in $\Gamma_\omega(a)$. If we denote by E the event that $\text{sign}(p(g_x)) \neq \text{sign}(p(g'))$, it follows that:

$$\Pr(E) \leq \frac{(\lfloor \frac{\delta}{\omega} \rfloor)^{d-1}\Delta 3^d(d+1)! + 2d(\lfloor \frac{\delta}{\omega} \rfloor)^{d-1}}{\lfloor \delta/\omega \rfloor^d}$$

$$\Pr(E) \leq \frac{2\omega\Delta 3^d(d+1)!}{\delta}$$

Using this, in conjunction with the union bound finishes the proof. ∎

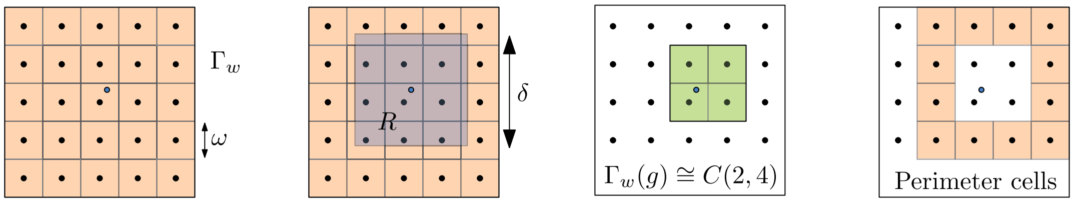

Fig. 7. We depict the range R of width δ around a point g. The block of green cells in $\Gamma_w(g)$ and the remaining cells that are intersected by R are the perimeter cells.

Lemma III.5 upper bounds the probability that snapping the perturbed input g_x changes a single real-valued comparison in A. We can use the union bound to upper bound the probability that for the whole algorithm, any real-valued comparison for g_x and g' is different:

Lemma III.6 (Snapping). *Let $g \in [0,1]^n$ be the input of an algorithm A that makes $T(n)$ real RAM comparison operations. Let x be a perturbation chosen uniformly at random in $[-\delta/2, \delta/2]^n$ and let $g_x = g + x$ be a perturbed instance of the input. For all $\varepsilon \in [0,1]$, if g_x is snapped to a grid of width*

$$\omega \leq \frac{\varepsilon\delta}{3^d(d+1)!\Delta 4T(n)},$$

Then the algorithm A makes for g_x and g' the same decision at each comparison instruction with probability at least $1-\varepsilon$.

Proof. By E_i for $i \in [T(n)]$, we denote the event that in the i'th algorithm step the inputs g_x and g' get a different outcome. Lemma III.5 upper bounds the probability of E_i occuring by:

$$\Pr(E_i) \leq \frac{4\omega\Delta 3^d(d+1)!}{\delta}.$$

The probability that g_x and g' are not equivalent is equal to the probability that for at least one event E_i, the event occurs. In other words if we denote by E the event that g_x and g' are not equivalent then:

$$\Pr(E) = \Pr\left(\cup_{i=1}^{T(n)} E_i\right) \leq T(n) \cdot 4\omega\Delta 3^d(d+1)/\delta.$$

In the antecedent, $Pr(g_x$ and g' not equivalent$) < \varepsilon$ is implied by $T(n) \cdot \frac{4\omega\Delta 3^d(d+1)}{\delta} < \varepsilon$. □

Finally to bound the *expected* input-precision of A, we apply a folklore lemma about swapping the order of integration [65]. We refer to the full version for details. We wish to note that Theorem III.1 only bounds the input-precision of the real-valued input of a recognition verification algorithm A and not the word size needed to store any intermediate results. Lemma I.1 shows then the smoothed bit-precision can be upper bound through the smoothed input-precision if the algebraic dimension and arithmetic degree of A are constant. Through linearity of expectation we obtain that the smoothed bit-precision is upper bound by: $O(\texttt{bit}_{\text{smooth}}(\delta, n, A))$.

IV. Smoothed Analysis of Resource Augmentation

The predominant approach to find decent solutions for hard optimization problems is to compute an approximation. An alternative approach is resource augmentation, where you consider an optimal solution subject to slightly weaker problem constraints. This alternative approach has received considerably less attention in theoretical computer science. We want to emphasize that resource augmentation algorithms find a solution which does not compromise the optimality, in the sense that it gives the optimal solution to the augmented problem. Using smoothed analysis, we argue that in practice some $\exists\mathbb{R}$-complete optimization problems have an optimal solution that can be represented with a logarithmic word size by applying smoothed analysis to the augmentation parameter. The proofs are deferred to the full version [22].

For the art gallery problem, Dobbins, Holmsen and Miltzow [8] showed that augmenting the polygon by so-called edge-inflations, guarding will become easier. This leads to small expected input-precision under smoothed analysis. Their proof consists of a problem specific part and a calculation of probabilities and expectations. We generalize their idea to a widely applicable framework for smoothed analysis.

Definition: Let us fix an algorithmic optimization problem P with real verification algorithm A. For it to be a *resource-augmentation* problem, we require three specific conditions: First, each input consists of an $I \in [0,1]^n \times \mathbb{Z}^m$ together with an *augmentation-parameter* $\alpha \in [0,1]$. Secondly, we assume that there is an implicitly defined *solution space* $\chi_I[\alpha] = \chi[\alpha]$ where each element in $\chi[\alpha]$ is considered a solution for the problem P (with input I and augmented by α) which does not have to be optimal. For example, in the art gallery problem I is the real-valued vertices of a simple polygon and an augmentation can be an inflation of the polygon by α (see [8]). The set $\chi_I[\alpha]$ is the set of all guard placements which guard the inflated polygon. Thirdly, we assume that there exists an evaluation function $f : \chi[\alpha] \to \mathbb{N}$. The aim is to find a solution $x^* \in \chi[\alpha]$ for which $f(x^*)$ is the maximum or minimum denoted by $\text{opt}(\chi[\alpha])$. For notational convenience, we assume that P is a maximization problem.

Smoothed analysis of resource augmentation: For any x, we intuitively consider the minimal word size required for each coordinate in x such that the verification algorithm A of P can verify if $x \in \chi_I[\alpha]$ on the word RAM. We denote by $\texttt{bit}(\chi_I[\alpha]) = \min\{\texttt{bit}_{IN}(x, A) \mid f(x) = \text{opt}(\chi_I[\alpha]), x \in \chi_I[\alpha]\}$ the minimal precision needed to express an optimal

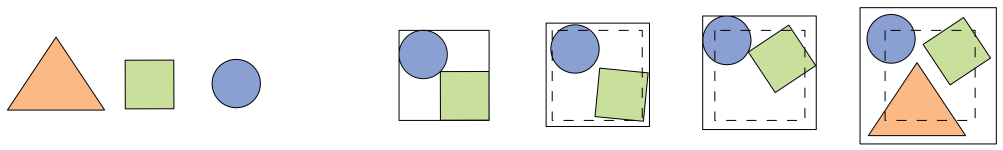

Fig. 8. We augment the container from left to right. This extra space can lead to a better solution. If the optimal solution does not change, the extra space allows for a solution with low input-precision.

solution in the solution space $\chi_I[\alpha]$. For the smoothed analysis of resource augmentation, we choose α uniform at random between $[0, \delta]$ (since we assume that a negative augmentation is not well-defined). Just as in the previous section, we first define the expected cost of a given input: $\texttt{bit}_\delta(I, P) = \underset{\alpha\in\Omega_\delta}{\mathbb{E}} [\texttt{bit}(\chi_I[\alpha])]$ and the smoothed cost:

$$\texttt{bit}_{\text{smooth}}(\delta, n, m, P) = \max_{I\in[0,1]^n\times\mathbb{Z}^m} \texttt{bit}_\delta(I, P).$$

In this paper, we study resource augmentation problems with three additional but natural properties:

- The *monotonous* property, which states that for all inputs I, for all $\alpha, \alpha' \in [0, 1]$ if $\alpha \leq \alpha'$ then $\chi_I[\alpha] \subseteq \chi_I[\alpha']$. Note that this property implies that in a more augmented version of the problem, the optimum is at least as good.
- The *moderate* property requires that for all inputs, there are at most $n^{O(1)}$ breakpoints. Where a breakpoint is defined as an $\alpha \in [0, 1]$ such that $\forall\varepsilon > 0$, $\text{opt}(\chi_I[\alpha - \varepsilon]) \neq \text{opt}(\chi_I[\alpha + \varepsilon])$.
- The *smoothable* property requires that for all x optimal in $\chi_I[\alpha]$ and for all $\varepsilon > 0$, there is an $x' \in \chi_I[\alpha + \varepsilon]$ with input-precision with respect to its real verification algorithm of $\leq c\log(n/\varepsilon)$, for some $c \in \mathbb{N}$, and $f(x) \leq f(x')$. Ergo, given some x we can increase the augmentation parameter by ε, to obtain an equally good solution $x' \in \chi_I[\alpha + \varepsilon]$. Furthermore, x' has low input-precision. Note that x' is not necessarily optimal for $\chi_I[\alpha + \varepsilon]$.

We apply smoothed analysis to resource-augmentation problems with these three properties by choosing uniformly at random the augmentation $\alpha \in [0, \delta]$. In the full version [22] we prove the following theorem:

Theorem IV.1. *Let P be a resource augmentation problem that is monotonous, moderate and smoothable and $m \leq O(n^c)$ then $\texttt{bit}_{\mathit{smooth}}(\delta, n, m, P)$ is upper bounded by $O(\log(n/\delta))$.*

Implications of Theorem IV.1: To illustrate the applicability of our findings, we give the following two corollaries. The first result was already shown in [8]. The art gallery problem has been shown to be $\exists\mathbb{R}$-complete [34], and currently $\exists\mathbb{R}$-completeness of the packing problems is in preparation [35]. Our results imply that apart from near-degenerate conditions the solutions to these problems have logarithmic input-precision.

Corollary IV.2. *Under perturbations of the augmentation of magnitude δ, the following problems have an optimal solution with an expected input-precision of $O(\log(n/\delta))$:*

- *the art gallery problem under perturbation of edge inflation [8].*
- *packing polygonal objects into a square container under perturbation of the container width.*

V. Conclusion

In Section II we strengthened the intuitive analogy between NP and $\exists\mathbb{R}$ by showing that for both of them membership is equivalent to the existence of a polynomial-time verification algorithm in their respective RAM. It is well-known that NP $\subseteq \exists\mathbb{R}$, but it is unknown if the two complexity classes are the same. The gap between the two complexity classes could be formed by inputs for $\exists\mathbb{R}$-complete problems, for which the witness cannot (to the best of our knowledge) be represented using polynomial input-precision. In Sections 3 and 4 of the full version, we show that for a wide class of $\exists\mathbb{R}$-complete problems that have such an "exponential input-precision phenomenon" their witness can, under smoothed analysis, be presented using logarithmic input-precision. Thus, the gap between $\exists\mathbb{R}$ and NP (if it exists) is formed by near-degenerate input.

Acknowledgment

Ivor van der Hoog is supported by the Netherlands Organisation for Scientific Research (NWO); 614.001.504. Tillmann Miltzow is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 016.Veni.192.25

References

[1] D. Salesin, J. Stolfi, and L. Guibas, "Epsilon geometry: building robust algorithms from imprecise computations," in *Proceedings of the fifth annual symposium on Computational geometry*. ACM, 1989, pp. 208–217.

[2] S. Fortune and C. Van Wyk, "Efficient exact arithmetic for computational geometry," in *Proceedings of the ninth annual symposium on Computational geometry*. ACM, 1993, pp. 163–172.

[3] C. Li, S. Pion, and C.-K. Yap, "Recent progress in exact geometric computation," *The Journal of Logic and Algebraic Programming*, vol. 64, no. 1, pp. 85–111, 2005.

[4] H. Mairson and J. Stolfi, "Reporting and counting intersections between two sets of line segments," in *Theoretical Foundations of Computer Graphics and CAD*. Springer, 1988, pp. 307–325.

[5] F. Kammer, M. Löffler, P. Mutser, and F. Staals, "Practical approaches to partially guarding a polyhedral terrain," in *International Conference on Geographic Information Science*. Springer, 2014, pp. 318–332.

[6] J. Goodman, R. Pollack, and B. Sturmfels, "Coordinate representation of order types requires exponential storage," in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '89. New York, NY, USA: ACM, 1989, pp. 405–410.

[7] D. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 385–463, 2004.

[8] M. G. Dobbins, A. Holmsen, and T. Miltzow, "Smoothed analysis of the art gallery problem," *CoRR*, vol. abs/1811.01177, 2018.

[9] T. Hagerup, "Sorting and searching on the word ram," in *Proc. 15th Ann. Symp. Theoretical Aspects Comput. Sci.*, ser. Lecture Notes Comput. Sci., M. Morvan, C. Meinel, and D. Krob, Eds., no. 1373. Springer Berlin Heidelberg, 1998, pp. 366–398.

[10] M. Fredman and D. Willard, "Surpassing the information theoretic bound with fusion trees," *Journal of Computer and System Sciences*, vol. 48, no. 3, pp. 424–436, 1993.

[11] ——, "Trans-dichotomous algorithms for minimum spanning trees and shortest paths," *Journal of Computer and System Sciences*, vol. 48, no. 3, pp. 533–551, 1994.

[12] D. Kirkpatrick and S. Reisch, "Upper bounds for sorting integers on random access machines," *Theor. Comput. Sci.*, vol. 28, no. 3, pp. 263–276, 1984.

[13] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C.-K. Yap, "Classroom examples of robustness problems in geometric computations," *Computational Geometry*, vol. 40, no. 1, pp. 61–78, 2008.

[14] M. I. Shamos, "Computational geometry," Ph.D. dissertation, Yale Unviersity, 1979. [Online]. Available: http://euro.ecom.cmu.edu/people/faculty/mshamos/1978ShamosThesis.pdf

[15] F. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, ser. Texts and Monographs in Computer Science. Springer, 1985.

[16] L. Blum, M. Shub, and S. Smale, "On a theory of computation and complexity over the real numbers: np-completeness, recursive functions and universal machines," *BulletinAmer. Math. Soc.*, vol. 21, no. 1, pp. 1–46, 1989.

[17] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation*. Springer, 1998.

[18] A. Schönhage, "On the power of random access machines," in *Proc. 6th Internat. Colloq. Automata Lang. Program.*, ser. Lecture Notes in Computer Science. Springer-Verlag, 1979, pp. 520–529.

[19] J. Hartmanis and J. Simon, "On the power of multiplciation in random-access machines," in *Proc. 15th Annu. IEEE Sympos. Switching Automata Theory*, 1974, pp. 13–23.

[20] A. Bertoni, G. Mauri, and N. Sabadini, "Simulations among classes of random access machines and equivalence among numbers succinctly represented," *Annals of Discrete Mathematics*, vol. 25, pp. 65–90, 1985.

[21] P. van Emde Boas, "Machine models and similations," in *Algorithms and Complexity*, ser. Handbook of Theoretical Computer Science. Elsevier Science, 1990, ch. 1, pp. 1, 3–66.

[22] J. Erickson, I. van der Hoog, and T. Miltzow, "Smoothing the gap between np and er." *CoRR*, vol. abs/1912.02278, 2019.

[23] G. Liotta, F. P. Preparata, and R. Tamassia, "Robust proximity queries: An illustration of degree-driven algorithm design," *SIAM Journal on Computing*, vol. 28, no. 3, pp. 864–889, 1998.

[24] V. Klee and G. Minty, "How good is the simplex algorithm," Washington Univ. Seattle Dept. of Mathematics, Tech. Rep., 1970.

[25] G. Nemhauser and Z. Ullmann, "Discrete dynamic programming and capital allocation," *Management Science*, vol. 15, no. 9, pp. 494–505, 1969.

[26] R. Beier and B. Vöcking, "Random knapsack in expected polynomial time," in *STOC*. ACM, 2003, pp. 232–241.

[27] D. Arthur and S. Vassilvitskii, "Worst-case and smoothed analysis of the icp algorithm, with an application to the k-means method," in *FOCS 2016*. IEEE, 2006, pp. 153–164.

[28] M. Englert, H. Röglin, and B. Vöcking, "Worst case and probabilistic analysis of the 2-opt algorithm for the tsp," in *SODA*, 2007, pp. 1295–1304.

[29] M. Etscheid and H. Röglin, "Smoothed analysis of local search for the maximum-cut problem," *ACM Transactions on Algorithms (TALG)*, vol. 13, no. 2, p. 25, 2017.

[30] I. van der Hoog, T. Miltzow, and M. van Schaik, "Smoothed analysis of order types," *arXiv:1907.04645*, 2019.

[31] D. Dadush and S. Huiberts, "A friendly smoothed analysis of the simplex method," in *STOC*. ACM, 2018, pp. 390–403.

[32] J. Matoušek, "Intersection graphs of segments and $\exists\mathbb{R}$," 2014, arXiv 1406.2636.

[33] Z. Abel, E. Demaine, M. Demaine, S. Eisenstat, J. Lynch, and T. Schardl, "Who needs crossings? Hardness of plane graph rigidity," in *32nd International Symposium on Computational Geometry (SoCG 2016)*, 2016, pp. 3:1–3:15.

[34] M. Abrahamsen, A. Adamaszek, and T. Miltzow, "The art gallery problem is $\exists\mathbb{R}$-complete," in *STOC*, 2018, pp. 65–73.

[35] M. Abrahamsen, T. Miltzow, and N. Seiferth, "A framework for $\exists\mathbb{R}$-completeness of two-dimensional packing problems," *Foundations of Computer Science*, 2020.

[36] D. Bienstock, "Some provably hard crossing number problems," *Discrete & Computational Geometry*, vol. 6, no. 3, pp. 443–459, 1991.

[37] J. Cardinal, S. Felsner, T. Miltzow, C. Tompkins, and B. Vogtenhuber, "Intersection graphs of rays and grounded segments," in *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 2017, pp. 153–166.

[38] J. Cardinal and U. Hoffmann, "Recognition and complexity of point visibility graphs," *Discrete & Computational Geometry*, vol. 57, no. 1, pp. 164–178, 2017.

[39] M. G. Dobbins, L. Kleist, T. Miltzow, and P. Rzążewski, "$\forall\exists\mathbb{R}$-completeness and area-universality," *ArXiv 1712.05142*, 2017.

[40] M. G. Dobbins, A. Holmsen, and T. Miltzow, "A universality theorem for nested polytopes," *arXiv*, vol. 1908.02213, 2019.

[41] J. Erickson, "Optimal curve straightening is $\exists\mathbb{R}$-complete," *arXiv:1908.09400*, 2019.

[42] J. Garg, R. Mehta, V. V. Vazirani, and S. Yazdanbod, "ETR-completeness for decision versions of multi-player (symmetric) Nash equilibria," in *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP 2015), part 1*, ser. Lecture Notes in Computer Science (LNCS), 2015, pp. 554–566.

[43] R. Kang and T. Müller, "Sphere and dot product representations of graphs," in *27th Annual Symposium on Computational Geometry (SoCG)*. ACM, 2011, pp. 308–314.

[44] L. Kleist, "Planar graphs and faces areas – area-universality," Ph.D. dissertation, Technische Universität Berlin, 2018, phD thesis.

[45] A. Lubiw, T. Miltzow, and D. Mondal, "The complexity of drawing a graph in a polygonal region," in *International Symposium on Graph Drawing and Network Visualization*, 2018.

[46] C. McDiarmid and T. Müller, "Integer realizations of disk and segment graphs," *Journal of Combinatorial Theory, Series B*, vol. 103, no. 1, pp. 114–143, 2013.

[47] N. Mnëv, "The universality theorems on the classification problem of configuration varieties and convex polytopes varieties," in *Topology and geometry – Rohlin seminar*, O. Y. Viro, Ed. Springer-Verlag Berlin Heidelberg, 1988, pp. 527–543.

[48] J. Richter-Gebert and G. M. Ziegler, "Realization spaces of 4-polytopes are universal," *Bulletin of the American Mathematical Society*, vol. 32, no. 4, pp. 403–412, 1995.

[49] M. Schaefer, "Complexity of some geometric and topological problems," in *Proceedings of the 17th International Symposium on Graph Drawing (GD 2009)*, ser. Lecture Notes in Computer Science (LNCS). Springer, 2009, pp. 334–344.

[50] ——, "Realizability of graphs and linkages," in *Thirty Essays on Geometric Graph Theory*, J. Pach, Ed. Springer-Verlag New York, 2013, ch. 23, pp. 461–482.

[51] M. Schaefer and D. Štefankovič, “Fixed points, Nash equilibria, and the existential theory of the reals,” *Theory of Computing Systems*, vol. 60, no. 2, pp. 172–193, 2017.

[52] Y. Shitov, “A universality theorem for nonnegative matrix factorizations,” 2016, journal=ArXiv 1606.09068.

[53] ——, “The complexity of positive semidefinite matrix factorization,” *SIAM Journal on Optimization*, vol. 27, no. 3, pp. 1898–1909, 2017.

[54] P. Shor, “Stretchability of pseudolines is NP-hard,” in *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, ser. DIMACS – Series in Discrete Mathematics and Theoretical Computer Science, P. Gritzmann and B. Sturmfels, Eds. American Mathematical Society and Association for Computing Machinery, 1991, pp. 531–554.

[55] L. A. Levin, “Universal sequential search problems,” *Problemy peredachi informatsii*, vol. 9, no. 3, pp. 115–116, 1973.

[56] S. A. Cook, “The complexity of theorem-proving procedures,” in *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 1971, pp. 151–158.

[57] J. Robson, “An $O(T \log T)$ reduction from RAM computations to satisfiability,” *Theor. Comput. Sci.*, vol. 82, no. 1, pp. 141–149, 1991.

[58] S. Cook and R. Reckhow, “Time bounded random access machines,” *J. Comput. Syst. Sci.*, vol. 7, no. 4, pp. 354–375, 1973.

[59] S. Cook, “Short propositional formulas represent nondeterministic computations,” *Inform. Proc. Lett.*, vol. 26, pp. 269–270, 1987/88.

[60] J. Bentley and T. Ottmann, “Algorithms for reporting and counting geometric intersections,” *IEEE Transactions on Computers*, vol. C-28, no. 9, pp. 643–647, 1979.

[61] J. Hass, J. C. Lagarias, and N. Pippenger, “The computational complexity of knot and link problems,” *J. ACM*, vol. 46, no. 2, pp. 185–211, 1999.

[62] M. Lackenby, “A polynomial upper bound on Reidemeister moves,” *Ann. Math.*, vol. 182, no. 2, pp. 491–564, 2015.

[63] M. Bonamy, E. Bonnet, N. Bousquet, P. Charbit, and S. Thomassé, “EPTAS for max clique on disks and unit balls,” in *FOCS*. IEEE Computer Society, 2018, pp. 568–579.

[64] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in real algebraic geometry. ISBN-10 3-540-33098-4*. Springer, Berlin Heidelberg, 2006.

[65] L. Tonelli, “Sull’integrazione per parti,” *Atti della Accademia Nazionale dei Lincei*, vol. 5, no. 18, pp. 246–253, 1909.