Real Benefit of Promises and Advice*

Klaus Ambos-Spies[1], Ulrike Brandt[2], and Martin Ziegler[3]

[1] Department of Mathematics and Computer Science, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany
[2] Department of Computer Science, Technische Universität Darmstadt, Darmstadt, Germany
[3] Department of Mathematics, Technische Universität Darmstadt, Darmstadt, Germany

Abstract. Promises are a standard way to formalize partial algorithms; and advice quantifies nonuniformity. For decision problems, the latter is captured in common complexity classes such as $\mathcal{P}/\mathrm{poly}$, that is, with advice growing in size with that of the input. We advertise constant-size advice and explore its theoretical impact on the complexity of classification problems – a natural generalization of promise problems – and on real functions and operators. Specifically we exhibit problems that, without any advice, are decidable/computable but of high complexity while, with (each increase in the permitted size of) advice, (gradually) drop down to polynomial-time.

1 Motivation

The Boolean satisfiability problem SAT is $\mathcal{NP}$–complete; but drops down to $\mathcal{P}$ when restricting to terms in 2-conjunctive normal form (2SAT). More strikingly, restricted to each 'slice' $\{0,1\}^n$ the $\mathcal{NP}$–complete Knapsack problem can be decided by an algorithm $\mathcal{A}_n$ of running time polynomial in n; cf. [2, Corollary 3.27]. In both above examples, a computable problem becomes easier when promising inputs to belong to, and thus permitting algorithms tailored for, a certain subset. Such promises occur frequently in the Theory of Computing [10, 12] formalized as Items a) & f) of

Definition 1. *Fix a subset X of $\{0,1\}^*$ or of $\mathbb{N}$.*

a) A promise problem *over X is a pair (A,B) of disjoint subsets of X. An algorithm* solves *(A,B) within time $t(n)$ if on inputs $\vec{x} \in A$ it reports 0 and 1 on inputs $\vec{x} \in B$, both after at most $\mathcal{O}\big(t(|\vec{x}|)\big)$ steps, where $|\vec{x}|$ denotes the (binary) length of $\vec{x}$. Note that the algorithm may behave arbitrarily, and even diverge, on inputs $\vec{x} \notin A \cup B$.*

* Supported in part by the *Marie Curie International Research Staff Exchange Scheme Fellowship* 294962 within the 7th European Community Framework Programme and by the *German Research Foundation* (DFG) with project Zi 1009/4-1. We acknowledge seminal discussions with Vassilis Gregoriades, Thorsten Kräling, and Hermann K.-G. Walter.

P. Bonizzoni, V. Brattka, and B. Löwe (Eds.): CiE 2013, LNCS 7921, pp. 1–11, 2013.
© Springer-Verlag Berlin Heidelberg 2013

b) A classification problem *over X is a (finite or countable) family $\mathcal{C} = (C_j)_{j\in J}$ of subsets of X. $\mathcal{C}$ is* straight *if the C_j are pairwise disjoint. $\mathcal{C}$ is* total *if X coincides with $\bigcup\mathcal{C} := \bigcup_{j\in J} C_j$.*

c) An algorithm solves *$\mathcal{C}$ if, upon input of $\vec{x} \in \bigcup\mathcal{C}$, it produces some $j \in J$ with $\vec{x} \in C_j$, that is, if it computes (some selection of) the (multivalued, unless $\mathcal{C}$ is straight)* generalized characteristic function $\mathbf{1}_{\mathcal{C}} : \bigcup\mathcal{C} \rightrightarrows J$. *Again, the behaviour on inputs $\vec{x} \notin \bigcup\mathcal{C}$ is unspecified and in particular not necessarily divergent.*

d) A multivalued *mapping $f : X \rightrightarrows Y$ is a relation $f \subseteq X \times Y$ satisfying: $\forall \vec{x} \in X\ \exists \vec{y} \in Y :\ (\vec{x}, \vec{y}) \in f$. We identify such f with the set-valued function $f : X \to 2^Y \setminus \{\emptyset\}$.*

e) Let $\mathcal{C} = (C_j)_{j\in J}$ and $\mathcal{B} = (B_j)_{j\in J}$ denote classification problems over X and fix $Y \subseteq X$. Abbreviate $\mathcal{C} \cap Y := (C_j \cap Y)_{j\in J}$ and $\mathcal{C} \oplus \mathcal{B} := \big((\mathtt{0} \circ C_j) \cup (\mathtt{1} \circ B_j)\big)_{j\in J}$. We write "$\mathcal{B} \subseteq \mathcal{C}$" if it holds $B_j \subseteq C_j$ for every $j \in J$.

f) Let $\bar{X} = (X_k)_{k\in K}$ be a (finite or countable) partition of X. An algorithm $\mathcal{A}$ computes a (possibly multivalued) mapping $f : X \rightrightarrows Y$ with advice $\bar{X}$ *if, upon input of $(\vec{x}, k)$ with $\vec{x} \in X_k$, it produces some $\vec{y} \in f(\vec{x})$. In this case we say that $\mathcal{A}$ computes f* with $|K|$–fold advice.

g) Fix a family $\mathcal{L}$ of subsets of X (such as $\mathcal{P}$ or $\mathcal{RE}$, the recursively enumerable languages) and call a classification problem $\mathcal{C} = (C_1, \ldots, C_J)$ $\mathcal{L}$–separable *if there exist pairwise disjoint $C'_1, \ldots, C'_J \in \mathcal{L}$ such that $C_j \subseteq C'_j$.*

So promise problems are precisely the straight classification problems of cardinality two [9]; and c) describes the computational problem of *providing* the kind of advice employed in f). See also Manifesto 3 below...

1.1 Real Computation and Complexity

Computable Analysis is the theory of real computation by approximation up to guaranteed prescribable absolute error. Initiated by Turing in the very same publication that introduced 'his' machine, it formalizes validated numerics in unbounded precision and more precisely interval computations of arbitrarily prescribable absolute output error; cf., e.g., [7].

Formally, a real number x is considered *computable* if any (equivalently: all) of the following conditions hold [38, §4.1 & Lemma 4.2.1]:

(a1) The (or any) binary expansion $b_n \in \{0, 1\}$ of $x = \sum_{n\geq -N} b_n 2^{-n}$ is recursive.

(a2) A Turing machine can produce dyadic approximations to x up to prescribable error 2^{-n}, that is, compute an integer sequence $a : \mathbb{N} \ni n \mapsto a_n \in \mathbb{Z}$ (output encoded in binary) with $|x - a_n/2^{n+1}| \leq 2^{-n}$.

(a3) A Turing machine can output rational sequences (c_n) and (ϵ_n) with $|x - c_n| \leq \epsilon_n \to 0$.

(a4) x admits a recursive *signed digit expansion*, that is a sequence $s_n \in \{0, -1, +1\}$ with $x = \sum_{n\geq -N} s_n 2^{-n}$.

Moreover the above conditions are, similarly to Shoenfield's Limit Lemma, one Turing jump stronger than the following

(a5) A Turing machine can output rational sequences (c'_n) with $c'_n \to x$.

Under the refined view of complexity, (a2) and (a4) remain uniformly polynomial-time equivalent but not (a1) nor (a3). Let $\mathbb{D}_n := \{a/2^{-n} : a \in \mathbb{Z}\}$ and $\mathbb{D} := \bigcup_n \mathbb{D}_n$ denote the set of dyadic rationals (of precision n). Proceeding from single reals to (possibly partial) functions $f :\subseteq [0;1] \to \mathbb{R}$, the following conditions are known equivalent and thus a reasonable notion of computability [14], [35, §0.7], [38, §6.1], [21, §2.3]:

(b1) A Turing machine can, upon input of every sequence $a_n \in \mathbb{Z}$ with $|x - a_n/2^{n+1}| \leq 2^{-n}$ for $x \in \mathrm{dom}(f)$, output a sequence $b_m \in \mathbb{Z}$ with $|f(x) - b_m/2^{m+1}| \leq 2^{-m}$.

(b2) There exists an oracle Turing machine $\mathcal{M}^?$ which, upon input of each $n \in \mathbb{N}$ and for every (discrete function) oracle $\mathcal{O} = \mathcal{O}(x)$, $x \in \mathrm{dom}(f)$, answering queries "$m \in \mathbb{N}$" with some $a \in \mathbb{D}_{m+1}$ such that $|x - a| \leq 2^{-m}$, prints some $b \in \mathbb{D}_{n+1}$ with $|f(x) - b| \leq 2^{-n}$.

It follows that every (even relatively, i.e. oracle) computable $f :\subseteq [0;1] \to \mathbb{R}$ is necessarily continuous.

Concerning complexity, (b1) and (b2) have turned out as polynomial-time equivalent; cf. [21, §8.1] and [38, Theorem 9.4.3]. Here the running time is measured in terms of the output precision parameter n in (b2); and for (b1) in terms of the time until the n-th digit of the infinite binary output string appears: in both cases uniformly in (i.e. w.r.t. the worst-case over all) $x \in \mathrm{dom}(f)$. (Efficient) computability is tied to (quantitative) topological properties; specifically we record from [21, Theorem 2.19]:

Fact 2. *If $f : [0;1] \to \mathbb{R}$ is computable in time $t(n)$, then $\mu(n) := t(n+2)$ constitutes a modulus of continuity for f in the sense that it holds*

$$\forall x, y \in [0;1] : \quad |x - y| \leq 2^{-\mu(n)} \;\Rightarrow\; |f(x) - f(y)| \leq 2^{-n} \quad .$$

For further details and prototype results see, e.g., [20, 29, 30, 33, 21, 31, 22, 6, 34, 17, 23], and [38, §7]. We also emphasize the close relation of the above notions to actual implementations of exact real arithmetic [32].

Multivalued mappings (Definition 1d) arise in Computable Analysis both for structural reasons [4, §1.4] and from practical applications. For instance the Fundamental Theorem of Algebra yields to every monic degree-d polynomial p a d-tuple of complex roots including multiplicities — up to permutation [36], that is, a *multivalued* mapping

$$\mathbb{C}^d \ni (a_0, \ldots a_{d-1}) \Rightarrow (\lambda_1, \ldots \lambda_d) \text{ s.t. } a_0 + a_1 z + \cdots a_{d-1} z^{d-1} + z^d = \prod_{j=1}^{d} (z - \lambda_j).$$

Or when diagonalizing a given real symmetric matrix, one is interested in *some* basis of eigenvectors, not a specific one. It is thus natural to consider computations which, given (some encoding of) x, intensionally choose and output *some*

value $y \in f(x)$. Indeed, a multifunction may well be computable yet admit no computable single-valued *selection*; cf. e.g. [38, Exercise 5.1.13] or [26].

1.2 Advice in Computer Science, Analysis, and Logic

Theoretical computer science regularly employs advice to formalize nonuniformity. Here an algorithm is presented not only with the input instance $\vec{x}$ but also with some additional integer $k = k(\vec{x})$. $\mathcal{P}/\mathrm{poly}$ for example consists precisely of those languages $L \subseteq \{0,1\}^*$ decidable in polynomial-time with advice k

- depending only on the binary length $|\vec{x}|$ of $\vec{x}$
- and of 'size' $\log k$ polynomially bounded in $|\vec{x}|$;

equivalently: languages L decidable by polynomial-size circuit families, i.e., one separate circuit for each input length n [13, §3.1]. These classes have received recent attention for instance in connection with the *Isomorphism Conjecture* [1, §2.7].

Manifesto 3. *We are interested in the benefit of constant-size advice (that is, with $k \in K$ for a fixed finite K) but permit full dependence on $\vec{x}$ (rather than on $|\vec{x}|$ only—which would not make sense for $\vec{x}$ being real numbers, anyway), recall Definition 1f) and see Section 2.2 below. In practice this means that the input $\vec{x}$ is accompanied with some integer $k \in K$ known from the application that generated $\vec{x}$ but not necessarily computable from said $\vec{x}$ alone. Put differently, $|K|$ separate algorithms $(\mathcal{A}_1, \ldots, \mathcal{A}_{|K|})$ may 'jointly' solve a problem in the sense that, on each input $\vec{x}$, at least one of them (namely $\mathcal{A}_{k(\vec{x})}$) terminates and produces the correct output. This is motivated by many problems, mostly over real numbers where finite advice makes the difference between computability and incomputability; see for instance Example 5 below.*

Indeed, already in the discrete realm $\mathcal{P}/\mathrm{const}$ contains undecidable problems. The class $\mathcal{P}_{\mathbb{R}}/\mathrm{const}$ in algebraic complexity theory [2, 28] means algorithms having stored (a constant number of) real constants and is unrelated to advice in our sense.

Remark 4. *Another superficially similar, but logically independent, relaxation of classical decision problems is* semi-membership*: Here an algorithm receives as input two arguments $(\vec{x}, \vec{y})$, with the promise that at least one of them belongs to L, and has to output $\vec{z} \in \{\vec{x}, \vec{y}\}$ with $\vec{z} \in L$ [15]. In the setting of Definition 1, this corresponds to the classification problem $(L \times X, X \times L)$.*

Note that Definition 1f) applies also to the realm $X = \mathbb{R} = Y$ of Computable Analysis — where, indeed, discrete advice occurs as a natural means against weak ineffectivity: Many practical problems over real numbers are trivially incomputable for continuity reasons in violation of the sometimes so-called *Main Theorem of Computable Analysis* [38, Theorem 4.3.1]. (Seemingly few) others are ineffective in the stronger sense of mapping computable arguments to incomputable ones – such as the differentiation operator [38, Example 6.4.9]. Ineffectivity of the former kind can easily be mended by providing, in addition

to the 'continuous' argument $x \in X$ from the connected space X, some discrete information k from a countable universe, say, $\mathbb{N}$. Such additional data is known in logic as *enrichment* [24, p.238/239]; cf. also [3]. Observe that, indeed, non-constant functions $k : X \to \mathbb{N}$ constitute 'prototypes' of nonuniformly computable but uniformly incomputable mappings; cf. [5]. Now previous work [40] has determined the precise amount of advice (i.e. the size of K) sufficient and necessary in order to render classical tasks in real linear algebra (continuous and) computable:

Example 5. *a) In order to compute, given a singular $d \times d$ matrix A, some non-zero solution $\vec{x}$ to the homogeneous system $A \cdot \vec{x} = 0$ of linear equations, knowing $k := \mathrm{rank}(A) \in \{0, 1, \ldots, d-1\} =: K$ is sufficient* [39, Theorem 11]*; and $|K| \geq d$ also necessary* [40, Theorem 46].

b) In order to compute, given a symmetric real $d \times d$ matrix A, some basis of eigenvectors, knowing $k := \mathrm{Card}\,\sigma(A) \in \{1, \ldots, d\} =: K$ is sufficient [39, Theorem 19]*; and $|K| \geq d$ also necessary* [40, Theorem 47].

c) In order to compute, given a symmetric real $d \times d$ matrix A, some single eigenvector, $\lfloor 1 + \log_2 d \rfloor$-fold advice is sufficient and necessary [40, Theorem 49].

Note that these investigations on quantitative nonuniformity exhibit examples which, with insufficient advice, are not even computable; see also [8]. [40, Example 7] on the other hand has constructed a computable smooth $h_2 : [0; 1] \to [0; 1]$ which, with two-fold advice, can be evaluated within polynomial, but without requires exponential, time in the sense of Section 1.1.

The present work extends this trade off between computational complexity and (the amount of) additional discrete information. More precisely we construct (Theorem 11) for each $d \in \mathbb{N}$ a smooth function $h_d : [0; 1] \to [0; 1]$ computable but with time complexity an exponential tower of height d; which, when providing k–fold advice, drops to height $d - k$ for each $k = 1, 2, \ldots, d$.

2 Discrete Classification Problems and Advice

Concerning trade-offs between advice and computational complexity, we have

Theorem 6. *Let* $\exp_0(n) := n$, $\exp_1(n) := 2^n$, $\exp_2(n) := 2^{2^n}$, *and* $\exp_{j+1}(n) := \exp_j(2^n) = 2^{\exp_j(n)}$ *denote the tower of iterated exponention also known as* tetration. *For each $J \geq 2$ there is a total straight classification problem of size J*

- *solvable in time* $\exp_{J-1}(\mathrm{poly}\, n)$ *but not in time* $\exp_{J-2}(\mathrm{poly}\, n)$
- *with 2–fold advice solvable in time* $\exp_{J-2}(\mathrm{poly}\, n)$ *but not in time* $\exp_{J-3}(\mathrm{poly}\, n)$
- *and more generally with j–fold advice solvable in time* $\exp_{J-j}(\mathrm{poly}\, n)$ *but not in time* $\exp_{J-j-1}(\mathrm{poly}\, n)$, $1 \leq j \leq J$.

2.1 On Hard Cores of Promise and Classification Problems

Nancy Lynch observed that any recursive decision problem L that cannot be solved in polynomial time contains an infinite recursive subset L' such that any algorithm semi-deciding L makes superpolynomially many steps on L' [27, Lemma 1]; cf. [11].

Definition 7. *Let $\mathcal{C} = (C_1, \ldots, C_J)$ denote a classification problem over X.*

a) $\mathcal{C}$ is hard *if none of the promise problems (C_i, C_j), $i < j$, can be solved in polynomial time.*
b) $\mathcal{C}$ is a hard core *if all C_j are infinite and if, for every choice of infinite $B_j \subseteq C_j$, $\mathcal{B} := (B_1, \ldots, B_J)$ is hard.*
c) $\mathcal{C}$ contains *a hard core if there exists a $\mathcal{B} \subseteq \mathcal{C}$ that is a hard core.*

[9, Theorem 3.4] establishes that each promise (i.e. 2-fold classification) problem unsolvable in polynomial time contains a superpolynomial-time hard core. We demonstrate that this fails for 3-fold classification problems:

Example 8. *Let $A, B, C \subseteq \{0,1,2\}^*$ denote classical decision problems – i.e. total promise problems $(A, \bar{A})$, $(B, \bar{B})$, and $(C, \bar{C})$, where $\bar{A} := \{0,1,2\}^* \setminus A$ – decidable in exponential but not in polynomial time. Now consider the classification problem*

$$\Big((2 \circ \bar{C}) \cup (0 \circ A),\ (0 \circ \bar{A}) \cup (1 \circ B),\ (1 \circ \bar{B}) \cup (2 \circ C) \Big) \quad :$$

It clearly is hard but does not contain a hard core. □

2.2 Advice versus Circuit Families: Incompressibility

This section clarifies the relation between advice in the sense of Definition 1f) and the classical model of nonuniform computation. To this end recall that $\mathcal{FP}/\mathrm{const}$ consists precisely of functions $f : \{0,1\}^* \to \{0,1\}^*$ such that

- there exists a polynomial-time Turing machine $\mathcal{M}$
- and a sequence $\vec{a}_n$ of strings from a fixed finite domain
- such that, for every $\vec{x} \in \{0,1\}^n$, $\mathcal{M}(\vec{x}, \vec{a}_n)$ outputs $f(\vec{x})$.

Similarly for other classes such as $\mathcal{REC}/\mathrm{poly}$: all languages decidable with the help of strings $\vec{a}_n$ of polynomial length... Here all inputs $\vec{x}$ of the same length n share the same advice – a real restriction compared to Definition 1f) as we now demonstrate using the Incompressibility Method [25]:

Proposition 9. *Let $\bar{\sigma} \in \{0,1\}^\omega$ denote an infinite random sequence in the sense of Martin-Löf and consider $L := \left\{ \vec{x} : \sigma_{\mathrm{bin}(1\vec{x})} = 1 \right\} \subseteq \{0,1\}^*$, its encoding into a binary language. Then L is trivially decidable (even in constant time) with 2-fold advice; but does not belong to $\mathcal{REC}/\mathrm{poly}$.*

The integer Kolmogorov complexity function $C : \mathbb{N} \to \mathbb{N}$ is well-known uncomputable [25, Theorem 2.3.2] – yet its values can be encoded as suitable, though unbounded, advice. In fact its tally version $C' : \{0,1\}^* \ni \vec{x} \mapsto C(|\vec{x}|)$, belongs to $\mathcal{FP}/\log \subseteq \mathcal{FP}/\operatorname{poly}$ via $\operatorname{bin}(\vec{a}_{|\vec{x}|}) := C'(\vec{x}) \leq |\vec{x}|$; whereas we have

Theorem 10. *Neither C nor C' is computable with finite advice.*

Together with Proposition 9 this demonstrates that (e.g. polynomial-time) computation with finite advice lies skewly to the standard nonuniform circuit complexity classes. We wonder whether $C \in \mathcal{FP}/\operatorname{poly}$ holds.

3 Real Function Complexity with Advice

Example 5 has demonstrated how (and how much) finite advice can make a difference between real computability and incomputability. Our main result provides analogous (although admittedly less natural) examples for the refined view of complexity by connecting discrete classification problems to real functions:

Theorem 11. *a) Fix a finite total straight classification problem $\mathcal{C} = (C_1, \ldots c_J)$ over $\mathbb{N}$. There exists a smooth (i.e. C^∞: infinitely often differentiable) function $h : [0;2] \to [0;J]$ such that the following holds for each $K \in \mathbb{N}$ and superlinear nondecreasing $t : \mathbb{N} \to \mathbb{N}$: $\mathcal{C}$ is solvable with K–fold advice within time $\mathcal{O}\big(t(\operatorname{poly} n)\big)$ iff h is computable with K–fold advice within time $\mathcal{O}\big(t(\operatorname{poly} n)\big)$.*

b) For each $J \in \mathbb{N}$ there is a smooth $f : [0;1] \to [0;1]$ computable with j–fold advice in time $\exp_{J-j}(\operatorname{poly} n)$ but not in time $\exp_{J-j-1}(\operatorname{poly} n)$, $1 \leq j \leq J$.

This (corrects a mistake in, and) generalizes [40, Example 7].

3.1 Benefit of 2-Fold Advice to Function Maximization

The computable functions constructed in Theorem 11b) exhibit the positive effects (in the sense of complexity-reducing) of discrete advice to real computation – but may arguably be considered artificial. For a more natural example, we now consider the generic problem of continuous optimization: computing the maximum of a given (say, non-expansive) $f : [0;1] \to [0;1]$. This constitutes a functional $\operatorname{Max} : \operatorname{Lip}_1 \to [0;1]$, $f \mapsto \max_{0 \leq x \leq 1} f(x)$ on $\operatorname{Lip}_1 := \big\{ f : [0;1] \to [0;1], |f(x) - f(y)| \leq |x - y| \big\}$ computable with respect to any reasonable encoding of Lip_1 into infinite binary strings [38, Corollary 6.2.5 & Lemma 6.1.7]. For instance $f \in \operatorname{Lip}_1$ is uniquely determined via the real sequence $\big(f(d_m)\big)_m$ where

$$(d_m)_m \quad := \quad \big(0, 1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{3}{4}, \tfrac{1}{8}, \tfrac{3}{8}, \tfrac{5}{8}, \tfrac{7}{8}, \tfrac{1}{16}, \tfrac{3}{16}, \ldots\big) \quad \subseteq \quad \mathbb{D}$$

is dense in $\operatorname{dom}(f)$. And, extending (b1) from Section 1.1, such a real sequence $(y_m)_m$ can be 'represented' (in a sense formalized in the *Type-2 Theory of Effectivity*, TTE) as an integer (double) sequence $(a_{\langle n,m \rangle})_{n,m}$ with $|y_m - a_{\langle n,m \rangle}| \leq 2^{-n}$, where $\langle n, m \rangle := n + (n+m) \cdot (n+m+1)/2$ denotes an integer pairing function.

Remark 12. *Note that $f(2^{-k})$ occurs within $(y_m)_m \subseteq \mathbb{R}$ (and its 2^{-1}–approximation within $(a_{\langle n,m\rangle})_{n,m} \subseteq \mathbb{Z}$) at a position exponential in k. In 'classic' TTE with infinite binary strings as codes* [38, §3]*, this requires 'skipping' over exponentially many bits in order to access $f(2^{-k})$ up to error $\frac{1}{2}$; which prevents evaluation $(f,x) \mapsto f(x)$ to be uniformly polynomial-time computable on Lip_1. In computational practice, on the other hand, such f is implemented as an (say, a C++ function) that, given integers n and m, produces a 2^{-n} approximation to $f(d_m)$ directly, that is, without generating $f(d_1), \ldots, f(d_{m-1})$ first. This has led* [18, 19] *to extend TTE with second-order representations, that is, with encodings that permit random (as opposed to sequential) access.*

For our present purpose the precise technical definition of second-order representations may be spared since we rely only on the following

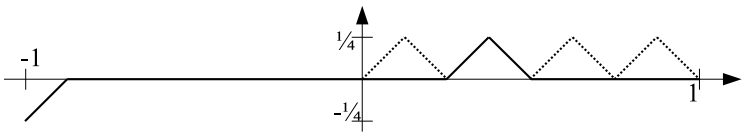
Fact 13. *There exists a (natural and canonical) second-order representation of Lip_1 allowing to recover, given $n \in \mathbb{N}$ and $d \in \mathbb{D}_n \cap [0;1]$, a 2^{-n}–approximation to $f(d)$ of length $\mathcal{O}(n)$ within time polynomial in n.*

We emphasize the similarity to Information-Based Complexity Theory [37, 16] which, based on the complementary, unit-cost model (a.k.a. *realRAM* or *Blum-Shub-Smale Machine*) of real computing, supposes exact black-box evaluation $x \mapsto f(x)$ to take constant time. And in fact, using a refinement of the standard adversary argument to finite precision queries, we can prove

Example 14. *For notational ease consider the class $\widetilde{\mathrm{Lip}_1}$ of non-expansive (a similar effect holds for appropriate smooth) functions $f : [-1;1] \to [-1;1]$;*

$$\mathcal{L} := \left\{ f \in \widetilde{\mathrm{Lip}_1} \;\middle|\; f(x) \le 0 \text{ for } x \le 0,\ f(x) = 0 \text{ for } x \ge 0 \right\}$$
$$\mathcal{K} := \left\{ f \in \widetilde{\mathrm{Lip}_1} \;\middle|\; \mathrm{Max}(f) = -f(-1) \right\}$$

and observe that $\mathcal{L} \cap \mathcal{K} = \{\equiv 0\}$, $\mathrm{Max}(f) = 0$ all $f \in \mathcal{L}$, and $\mathrm{Max}(f) = -f(-1)$ all $f \in \mathcal{K}$. In particular both restrictions $\mathrm{Max}\big|_{\mathcal{L}}$ and $\mathrm{Max}\big|_{\mathcal{K}}$ are polynomial-time computable – while, without such 2-fold advice, $\mathrm{Max}\big|_{\mathcal{L}\cup\mathcal{K}}$ is not. □



4 Conclusion and Perspectives

Recursive Analysis had traditionally focused on the continuous aspects of real number computation as opposed to classical computability theory. Still, discrete advice has over the last years been revealed as essential an ingredient to uniform computability for many natural problems over the reals. We have demonstrated

that, even for problems that are computable without such advice, its presence or absence can have a huge impact in terms of computational complexity.

While our Theorem 11 is entirely artificial, it seems that the underlying concept may be relevant to practical computations. Indeed numerical science generally considers discrete-valued functions like floor or matrix rank efficiently computable – while at the same time at least deprecating the use of tests for equality: a seemingly inherent ambiguity in the foundational semantics. The representation-based theory of real number computation (TTE and its 2nd-order extension) on the other hand regularly extends continuous with discrete information in order to assert the computability of a problem – and provides the canonical interface declaration of an actual implementation in exact real number packages [32].

Still the question remains as for natural examples of real problems where discrete advice makes a difference in complexity but not in computability: in Theorem 11 the function is artificial while its domain $[0;1]$ is easy, in Example 14 the functional is natural while its domain is artificial.

References

1. Agrawal, M.: The Isomorphism Conjecture for $\mathcal{NP}$. In: Cooper, S.B., Sorbi, A. (eds.) Computability in Context, pp. 19–48. World Scientific (2009)
2. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Springer (1997)
3. Beyersdorff, O., Köbler, J., Müller, S.: Proof Systems that Take Advice. Proof Systems that Take Advice 209(3), 320–332 (2011)
4. Brattka, V.: Recursive Characterization of Computable Real-Valued Functions and Relations. Theoretical Computer Science 162, 45–77 (1996)
5. Brattka, V.: Computable Invariance. Theoretical Computer Science 210, 3–20 (1999)
6. Braverman, M.: On the Complexity of Real Functions. In: Proc. 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 155–164
7. Braverman, M., Cook, S.A.: Computing over the Reals: Foundations for Scientific Computing. Notices of the Americal Mathematical Society 53(3), 318–329 (2006)
8. Brattka, V., Pauly, A.M.: Computation with Advice. In: Electronic Proceedings in Theoretical Computer Science, vol. 24 (June 2010)
9. Brandt, U., Walter, H.K.-G.: Cohesiveness in Promise Problems. Presented at the 64th GI Workshop on Algorithms and Complexity (2012)
10. Even, S., Selman, A.L., Yacobi, Y.: The Complexity of Promise Problems with Applications to Public-Key Cryptography. Inform. and Control 61, 159–173 (1984)
11. Even, S., Selman, A.L., Yacobi, Y.: Hard-Core Theorems for Complexity Classes. Journal of the ACM 32(1), 205–217 (1985)
12. Goldreich, O.: On Promise Problems: A Survey. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Shimon Even Festschrift. LNCS, vol. 3895, pp. 254–290. Springer, Heidelberg (2006)
13. Goldreich, O.: Computational Complexity: A Conceptual Perspective. Cambridge University Press (2008)

14. Grzegorczyk, A.: On the Definitions of Computable Real Continuous Functions. Fundamenta Mathematicae 44, 61–77 (1957)
15. Hemaspaandra, L.A., Torenvliet, L.: Theory of Semi-Feasible Algorithms. Springer Monographs in Theoretical Computer Science (2003)
16. Hertling, P.: Topological Complexity of Zero Finding with Algebraic Operations. Journal of Complexity 18(4), 912–942 (2002)
17. Kawamura, A.: Lipschitz Continuous Ordinary Differential Equations are Polynomial-Space Complete. Computational Complexity 19(2), 305–332 (2010)
18. Kawamura, A., Cook, S.A.: Complexity Theory for Operators in Analysis. In: Proc. 42nd Ann. ACM Symp. on Theory of Computing (STOC 2010), pp. 495–502 (2010)
19. Kawamura, A., Cook, S.A.: Complexity Theory for Operators in Analysis. ACM Transactions in Computation Theory 4(2), article 5 (2012)
20. Ko, K.-I., Friedman, H.: Computational Complexity of Real Functions. Theoretical Computer Science 20, 323–352 (1982)
21. Ko, K.-I.: Complexity Theory of Real Functions. Birkhäuser (1991)
22. Ko, K.-I.: Polynomial-Time Computability in Analysis. In: Ershov, Y.L., et al. (eds.) Handbook of Recursive Mathematics, vol. 2, pp. 1271–1317 (1998)
23. Kawamura, A., Ota, H., Rösnick, C., Ziegler, M.: Computational Complexity of Smooth Differential Equations. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 578–589. Springer, Heidelberg (2012)
24. Kreisel, G., Macintyre, A.: Constructive Logic versus Algebraization I. In: Troelstra, A.S., van Dalen, D. (eds.) Proc. L.E.J. Brouwer Centenary Symposium, pp. 217–260. North-Holland (1982)
25. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer (1997)
26. Luckhardt, H.: A Fundamental Effect in Computations on Real Numbers. Theoretical Computer Science 5, 321–324 (1977)
27. Lynch, N.: On Reducibility to Complex or Sparse Sets. Journal of the ACM 22(3), 341–345 (1975)
28. Michaux, C.: $\mathcal{P} \neq \mathcal{NP}$ over the Nonstandard Reals Implies $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{R}$. Theoretical Computer Science 133, 95–104 (1994)
29. Müller, N.T.: Subpolynomial Complexity Classes of Real Functions and Real Numbers. In: Kott, L. (ed.) ICALP 1986. LNCS, vol. 226, pp. 284–293. Springer, Heidelberg (1986)
30. Müller, N.T.: Uniform Computational Complexity of Taylor Series. In: Ottmann, T. (ed.) ICALP 1987. LNCS, vol. 267, pp. 435–444. Springer, Heidelberg (1987)
31. Müller, N.T.: Constructive Aspects of Analytic Functions. In: Proc. Workshop on Computability and Complexity in Analysis (CCA), InformatikBerichte FernUniversität Hagen, vol. 190, pp. 105–114 (1995)
32. Müller, N.T.: The iRRAM: Exact Arithmetic in C++. In: Blank, J., Brattka, V., Hertling, P. (eds.) CCA 2000. LNCS, vol. 2064, pp. 222–252. Springer, Heidelberg (2001)
33. Müller, N.T., Moiske, B.: Solving Initial Value Problems in Polynomial Time. In: Proc. 22nd JAIIO-PANEL, pp. 283–293 (1993)
34. Müller, N.T., Zhao, X.: Complexity of Operators on Compact Sets. Electronic Notes Theoretical Computer Science 202, 101–119 (2008)
35. Pour-El, M.B., Richards, J.I.: Computability in Analysis and Physics. Springer (1989)

36. Specker, E.: The Fundamental Theorem of Algebra in Recursive Analysis. In: Dejon, B., Henrici, P. (eds.) Constructive Aspects of the Fundamental Theorem of Algebra, pp. 321–329. Wiley-Interscience (1969)
37. Traub, J.F., Wasilkowski, G.W., Woźniakowski, H.: Information-Based Complexity. Academic Press (1988)
38. Weihrauch, K.: Computable Analysis. Springer (2000)
39. Ziegler, M., Brattka, V.: Computability in Linear Algebra. Theoretical Computer Science 326, 187–211 (2004)
40. Ziegler, M.: Real Computation with Least Discrete Advice: A Complexity Theory of Nonuniform Computability. Annals of Pure and Applied Logic 163(8), 1108–1113 (2012)