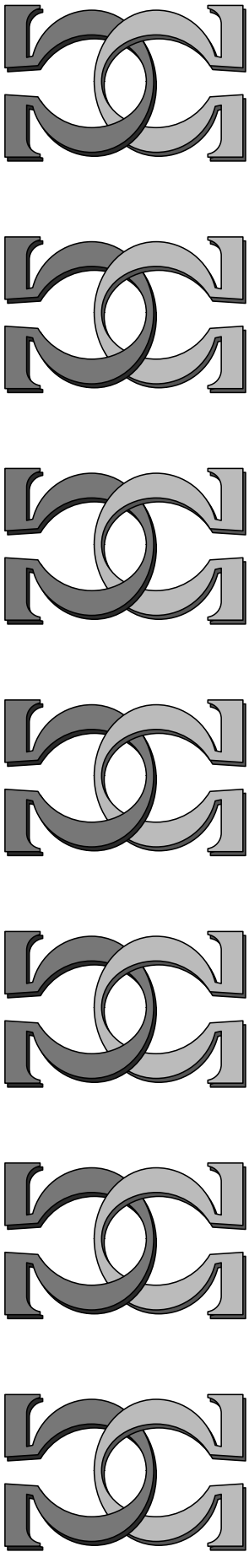

CDMTCS
Research
Report
Series

Two Anytime Algorithms for the Halting Problem

C. S. Calude[1] and M. Dumitrescu[2]

[1]University of Auckland, NZ
[2]Bucharest University, Romania

CDMTCS-493
December 2015

Centre for Discrete Mathematics and
Theoretical Computer Science

Two Anytime Algorithms for the Halting Problem

Cristian S. Calude
Department of Computer Science, University of Auckland
Auckland, New Zealand
www.cs.auckland.ac.nz/~cristian

Monica Dumitrescu
Faculty of Mathematics and Computer Science
Bucharest University, Romania
http://fmi.unibuc.ro/ro/scoala_doctorala_mate/monica_dumitrescu

December 23, 2015

Abstract

The halting problem, the most (in)famous undecidable problem, has important applications in theoretical and applied computer science and beyond, hence the interest in its approximate solutions.

A program which eventually stops but does not halt "quickly" stops at a time which is algorithmically compressible. This result was converted into anytime algorithms for the halting problem for which the stopping time (cut-off temporal bound) is very large and, in general, cannot be significantly improved.

In this paper we propose two classes of anytime algorithms for the halting problem, a theoretical one based on probability theory and another, more practical one, based on order statistic. Previous experimental studies for small Turing machines suggest that some anytime algorithms in these classes may be indeed feasible. Extensive experimental work is needed in this direction.

1 Introduction

The halting problem asks to decide, from a description of an arbitrary program and an input, whether the computation of the program on that input will eventually stop or continue forever. In 1936 Alonzo Church, and independently Alan Turing, proved that (in Turing's formulation) an *algorithm to solve the halting problem for all possible program-input pairs does not exist*; two equivalent models have been used to describe computation by algorithms (an informal notion), the lambda calculus by Church and Turing machines by Turing. The halting problem is historically the first proved undecidable problem; it has many applications in mathematics, logic and theoretical as well as applied computer science, mathematics, physics, biology, etc.

Due to its practical importance approximate solutions for this problem have been proposed for quite a long time, see [15, 14, 7, 19, 24, 6, 5, 2].

Anytime algorithms trade execution time for quality of results [12]. These algorithms can be executed in two modes: either by a given contract time to execute or an interruptible method. Instead of correctness, an anytime algorithm returns a result together with a "quality measure" which evaluates how close the obtained result is to the result that would be returned if the algorithm ran until completion (which may be prohibitively long). To improve the solution, anytime algorithms can be continued after they have halted. That is similar to the use of iterative processes in numerical computing in which, after the process has been halted, if the output is not considered to be acceptable, then it can be refined by resuming the iterative process.

Following Manin [19] we use a more general form of anytime algorithm as an approximation for a computation which may never stop. An anytime algorithm for the halting problem works in the following way: to test whether a program eventually stops on a given input we first *effectively compute a threshold time* – the interruptible (stopping) condition – and then run the program for that specific time. If the computation stops, then the program was proved to halt; if the computation does not stop, then we *declare* that: a) the program will never stop and b) evaluate the error probability, i.e. the probability that the program may eventually stop. The goal is to prove that the *error probability* can be made as small as we wish. By running the program a longer time we can improve its performance either by getting to the halting time or by decreasing the probability error. Another important goal is to develop feasible anytime algorithms for the halting problem.

In [7, 6] anytime algorithms for the halting problem have been developed using the fact – proved in [7] – that programs which take a long time to halt stop at algorithmically "compressible times", i.e. times which a computer can generate from "smaller" inputs. The stopping times obtained using this method are very large and the theoretical bounds cannot be improved, see [6]. However, experimental results reported in [24] for small Turing machines indicate much smaller stopping times.

In this paper we use methods from probability theory and statistics to design two types of anytime algorithms for the halting problem. The first algorithm depends on a computable probability distribution; the second one depends on a sampling method. In this way we obtain two classes of anytime algorithms to choose from in practical applications. The experimental results cited above suggest that some anytime algorithms in these classes may be indeed feasible. Extensive experimental work is needed in this direction.

The paper is organised as follows. We start with a section including the basic notation. Section 3 presents the computability and complexity part while Section 4 is dedicated to probability and statistics. In Section 5 we present the probabilistic framework for our paper. Section 6 constructs a natural computable probability for the set of running-times. Section 7 is the main section: it contains the proposed anytime algorithms and the proofs of their correctness. The last section discusses the power and limits of the proposed algorithms and open problems.

2 Notation

In the following we will denote by $\mathbb{Z}^+$ the set of positive integers $\{1, 2, \dots\}$ and let $\overline{\mathbb{Z}^+} = \mathbb{Z}^+ \cup \{\infty\}$; $\mathbb{R}$ is the set of reals. The domain of a partial function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is denoted by $\mathrm{dom}\,(F)$: $\mathrm{dom}\,(F) = \{x \in \mathbb{Z}^+ \mid F\,(x) < \infty\}$. We denote by $\#S$ the cardinality of the set S.

We assume familiarity with elementary computability theory and algorithmic information theory [18, 3, 11].

For a partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ we denote by $F(x)[t] < \infty$ the statement "F has stopped *exactly* in time t". For $t \in \mathbb{Z}^+$ we consider the computable set

$$\mathrm{Stop}(F, t) = \{x \in \mathbb{Z}^+ \mid F(x)[t] < \infty\},$$

and note that

$$\mathrm{dom}(F) = \bigcup_{t \in \mathbb{Z}^+} \mathrm{Stop}(F, t). \tag{1}$$

3 Complexity and universality

The *algorithmic complexity* relative to a partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is the partial function $\nabla_F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ defined by

$$\nabla_F\,(x) = \inf\left\{y \in \mathbb{Z}^+ \mid F\,(y) = x\right\}.$$

If $F\,(y) \neq x$ for every $y \geq 1$, then $\nabla_F\,(x) = \infty$. That is, the algorithmic complexity of x is the smallest description/encoding of x with respect to the interpreter/decoder F, or infinity if F cannot produce x.

A partially computable function $\mathbf{U}$ is called *universal* if for every partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ there exists a constant $k_{\mathbf{U},F}$ such that for every $x \in \mathrm{dom}\,(\nabla_F)$ we have

$$\nabla_{\mathbf{U}}\,(x) \leq k_{\mathbf{U},F} \cdot \nabla_F\,(x)\,. \tag{2}$$

Theorem 1 ([6]). *A partially computable function* $\mathbf{U}$ *is universal iff for every partially computable function* $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ *there exists a constant* $c_{\mathbf{U},F}$ *such that for every* $x \in \mathrm{dom}\,(F)$ *we have*

$$\nabla_{\mathbf{U}}\,(F\,(x)) \leq c_{\mathbf{U},F} \cdot x. \tag{3}$$

The difference between (2) and (3) is in the role played by F: in the traditional condition (2), F appears through ∇_F (which sometimes may be incomputable), while in (3) F appears as argument of $\nabla_{\mathbf{U}}$, making the second member of the inequality always computable.

A universal partially computable function $\mathbf{U}$ "simulates" any other partially computable function F in the following sense: if $x \in \mathrm{dom}\,(F)$, then from (3), one can deduce that $\nabla_{\mathbf{U}}\,(F\,(x)) \leq c_{\mathbf{U},F} \cdot x$, hence there exists $y \leq c_{\mathbf{U},F} \cdot x$ in $\mathrm{dom}\,(\mathbf{U})$ such that $\mathbf{U}\,(y) = F\,(x)$. In particular, $\nabla_{\mathbf{U}}(x) < \infty$, for all $x \in \mathbb{Z}^+$.

The set $\mathrm{dom}(\mathbf{U})$ – see (1) for $\mathbf{U} = F$ – is computable enumerable, but not computable (the *undecidability of the halting problem*), its complement $\overline{\mathrm{dom}(\mathbf{U})}$ is not computably enumerable, but the sets $(\mathrm{Stop}(\mathbf{U}, t))_{t\geq 1}$ are all computable.

To solve the halting problem means to determine for an arbitrarily pair (F, x), where F is a partially computable function and $x \in \mathbb{Z}^+$, whether $F(x)$ stops or not, or equivalently, whether $x \in \mathrm{dom}(F)$, that is, $x \in \mathrm{Stop}(F, t)$, for some $t \in \mathbb{Z}^+$. In view of (2) or (3) solving the halting problem for a fixed universal partially computable function $\mathbf{U}$ is enough to solve the halting problem. From now on we fix a universal partially computable function $\mathbf{U}$ and study the halting problem $\mathbf{U}(x) < \infty$, for $x \in \mathbb{Z}^+$.

4 A glimpse of probability and statistics

In this section we define the main notions from probability theory and statistics used in this paper. For more details see [1, 8, 22].

A *probability space* is a triple $(\Omega, \mathcal{B}(\Omega), \Pr)$, where Ω is non-empty set, $\mathcal{B}(\Omega)$ is a σ-field of subsets of Ω and $\Pr : \mathcal{B}(\Omega) \longrightarrow [0, 1]$ is a *probability measure*, that is, it satisfies the following two conditions: a) the probability of a countable union of mutually-exclusive sets in $\mathcal{B}(\Omega)$ is equal to the countable sum of the probabilities of each of these sets, and b) $\Pr(\Omega) = 1$. We interpret $\mathcal{B}(\Omega)$ as "the family of events" and Ω as "the certain event".

A function $X \colon \Omega \longrightarrow A$, where $A \subseteq \mathbb{R}$ is equipped with a $\sigma-$field $\mathcal{B}(A)$ such that for every $B \in \mathcal{B}(A)$ we have $X^{-1}(B) \in \mathcal{B}(\Omega)$, is called *random variable*; in this case X induces a probability (called *probability distribution of* X) $P_X : \mathcal{B}(A) \longrightarrow [0, 1]$ defined by

$$P_X(B) = \Pr\left(X^{-1}(B)\right) = \Pr(\{\omega \mid X(\omega) \in B\}), \ B \in \mathcal{B}(A),$$

which identifies the probability space $(A, \mathcal{B}(A), P_X)$.

The random variable X has a *discrete probability distribution* if A is countable. If we denote by $P_X(x)$ the probability of the event $\{X = x\}$, then the discrete probability distribution of X is completely defined by the numbers $P_X(x) \in [0, 1]$, $x \in A$, with $\sum\limits_{x \in A} P_X(x) = 1$.

The *Cumulative Distribution Function* of a random variable X with a discrete distribution is the function $CDF_X : \mathbb{R} \longrightarrow [0, 1]$ defined by

$$CDF_X(y) = \Pr(X \leq y) = \sum_{\{x \in A \,|\, x \leq\ y\}} P_X(x), \ y \in \mathbb{R}.$$

In this case, CDF_X is a stair-function (with piecewise-constant sections). For example, the discrete probability distribution X that expresses the probability of a given number of events occurring in a fixed interval of time and/or space, provided these events occur with a known average rate and independently of the time since the last event, is called the *Poisson distribution*. More precisely, the random variable X defined on $A = \mathbb{Z}^+ \cup \{0\}$ has a Poisson distribution with an average λ if $P_X(x) = \frac{\lambda^x}{x!}\exp(-\lambda)$, $x = 0, 1, 2, \ldots$; in this case, $CDF_X(y) = \sum\limits_{x=0,\ x\leq y}^{\infty} \frac{\lambda^x}{x!}\exp(-\lambda)$, $y \in \mathbb{R}$ and $\lim_{y\to\infty} CDF_X(y) = 1$. In Figure 1 the values of $CDF_X(y)$, with an average $\lambda = 3$ for $0 \leq y < 8$, are pictured.

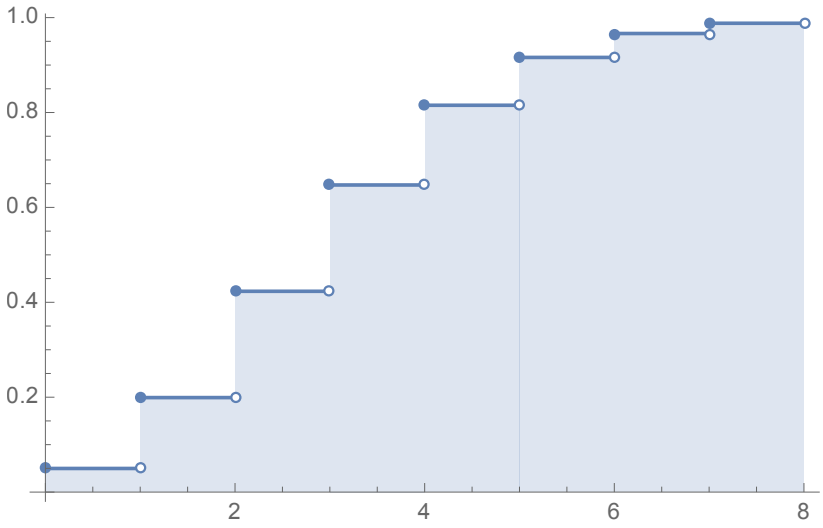

Figure 1: CDF_X for an example of Poisson distribution.

The *Inverse Cumulative Distribution Function* or *quantile function* $\mathbf{q}_X : [0,1] \longrightarrow A$ is defined by the equation $CDF_X(\mathbf{q}_X(p)) = p,\ p \in [0,1]$ whose solution is given by

$$\mathbf{q}_X(p) = \inf\{y \in A \mid p \leq CDF_X(y)\},\ p \in [0,1].$$

For fixed $r \in [0,1]$, the value (number) $\mathbf{q}_X(r)$ is called *the* rth *quantile* of the random variable X. Quantiles are important indicators that give information about the location and clustering of the probability values $\{P_X(x), x \in A\}$. For example, if the data being studied are not actually distributed according to an assumed probability distribution, or if there are outliers far removed from the mean, then quantiles may provide useful information. Beside the classical quartiles – first, second (median), third – the lower and upper εth quantiles, $\mathbf{q}_X(\varepsilon)$ and $\mathbf{q}_X(1-\varepsilon)$, give important informations about the "tails" of the probability distribution (for small $\varepsilon > 0$). For more details see [1].

The notions and results discussed above are theoretical. Can they be reformulated in *an inferential approach*, that is, can we extract information about the probability distribution of the random variable X from observations of the phenomenon described by X? Thus, instead of working with the theoretical CDF_X, can we characterise the probability distribution of a random variable X by means of a long-enough sequence of observations $(x_1, ..., x_N) \in A^N$ that are independent, identically distributed replicates of the random variable X? The answer is affirmative. In what follows $(x_1, ..., x_N)$ will be called an N-*dimensional sample* and its values $x_1, ..., x_N$ *data points*. The *Empirical Cumulative Distribution Function* is defined by

$$ECDF_{X,N}(y) = \frac{\#\{1 \leq i \leq N \mid x_i \leq y\}}{N},\ y \in \mathbb{R}. \tag{4}$$

The rth *sample quantile* of an N-dimensional sample, $\mathbf{q}_{X,N} : [0,1] \longrightarrow \{x_1, ..., x_N\}$, is defined by

$$\mathbf{q}_{X,N}(r) = \inf\{y \in \{x_1, ..., x_N\} \mid r \leq ECDF_{X,N}(y)\},\ r \in [0,1].$$

Suppose that we order increasingly the observed data points and denote the sequence by

$$x_{(1)} \leq x_{(2)} \leq ... \leq x_{(N-1)} \leq x_{(N)}. \tag{5}$$

The *order statistic of rank* k is the kth smallest value in (5).

The following result shows how to compute the rth sample quantile (see [13],[8, Ch. 6]):

Theorem 2. *Let* $(x_1, ..., x_N) \in A^N$ *be* N *independent, identically distributed replicates of the random variable* X*. Then, for an* N*-dimensional sample, the* r*th sample quantile is equal to the order statistic of rank* $\lceil Nr \rceil$*, that is,* $\mathbf{q}_{X,N}(r) = x_{(\lceil Nr \rceil)}$.

In view of Theorem 2, the rth quantile $\mathbf{q}_{\mathbf{X}}(r)$ will be estimated with the rth sample quantile $\mathbf{q}_{X,N}(r)$, that is with $x_{(\lceil Nr \rceil)}$.

Inference-based-decisions are made using statistical procedures using sets of observations. An inference-based-decision of a *hypothesis* results in one of two outcomes: the hypothesis is accepted or rejected. The outcome can be correct or erroneous. The set of observations leading to the decision "reject the hypothesis" is called the *critical region*.

Fix the probability space $(A, \mathcal{B}(A), P_X)$ induced by a random variable X. Consider a critical region $B \subset A, B \in \mathcal{B}(A)$ and an observed value $x \in A$. A *hypothesis* H is a statement such that for every $x \in A$ "H_x is true" $\in \mathcal{B}(A)$ and "H_x is false" $\in \mathcal{B}(A)$. An inference-based-decision has the following form:

If the observed value $x \in A$ belongs to B, then decide to reject the hypothesis H_x.

An error occurs if we reject H_x on the basis of B, when H_x is true. The *error probability*, that is the probability of an erroneous decision, is $P_X(\{x \in B \mid H_x \text{ is true}\})$. Of course, only decisions with (very) low error probability are of genuine practical interest.

5 A probabilistic framework

In this section we describe the probabilistic framework for developing our anytime algorithms. The *finite running-times*[1] of the computations $\mathbf{U}(x)$ are the set of stopping times for the halting programs of $\mathbf{U}$:[2]

$$\begin{aligned}\mathbf{T}_{\mathbf{U}} &= \left\{t \in \mathbb{Z}^+ \mid \text{ there exists } x \in \mathbb{Z}^+ \text{ such that } x \in \text{Stop}(\mathbf{U}, t)\right\} \\ &= \left\{t \in \mathbb{Z}^+ \mid \text{there exists } x \in \mathbb{Z}^+ \text{ such that } \mathbf{U}(x)[t] < \infty\right\}.\end{aligned}$$

Lemma 3. *For every* $M \in \mathbb{Z}^+$ *there is a program* $x \in \text{dom}(\mathbf{U})$ *which stops in time larger than* M*, hence* $\mathbf{T}_{\mathbf{U}}$ *is infinite.*

Proof. The statement in the lemma is true because otherwise all programs would stop in time at most M, hence $\text{dom}(\mathbf{U})$ would be decidable, a contradiction. □

[1] See [10] for modelling running-times.

[2] Recall that $\mathbf{U}(x)[t] < \infty$ means that the computation $\mathbf{U}(x)$ has stopped exactly in time t.

In what follows we will work with the *random variable*

$$RT = RT_{\mathbf{U}} : \mathrm{dom}(\mathbf{U}) \longrightarrow \mathbf{T}_{\mathbf{U}},$$

called the *running-time associated with* $\mathbf{U}$, defined by

$$RT\,(x) = \min\{t > 0 \mid \mathbf{U}(x)[t] < \infty\}. \tag{6}$$

The random variable RT is completely specified by a *computable probability distribution on the set of finite running-times of programs of* $\mathbf{U}$,

$$\left\{ P_{RT}\,(t)\,, t \in \mathbf{T}_{\mathbf{U}} \mid P_{RT}\,(t) \in (0,1)\,, \sum_{t\,\in\,\mathbf{T}_{\mathbf{U}}} P_{RT}\,(t) = 1 \right\}, \tag{7}$$

chosen to "reflect in some natural way" the behaviour of the halting programs.[3]

We will consider two cases, depending on whether we work with this distribution directly or indirectly. In the first case the anytime algorithm will use the *probabilistic threshold* $\mathbf{q}_{RT}\,(1-\varepsilon)$. In the second case we will use an N-dimensional sampling of $\mathbf{T}_{\mathbf{U}}$ obtaining a sequence of independent, identically distributed running-times $(t_1, ..., t_N)$ and choose, according to Theorem 2, $t_{(\lceil N(1-\varepsilon)\rceil)}$, the order statistic of rank $(\lceil N(1-\varepsilon)\rceil)$, as the *fitted threshold*.

The cumulative distribution function $CDF_{RT} : \mathbb{R} \longrightarrow [0,1]$ of the discrete random variable $RT\colon\ \mathrm{dom}(\mathbf{U}) \longrightarrow \mathbf{T}_{\mathbf{U}}$ is then defined by the formula:

$$CDF_{RT}\,(y) = P_{RT}(\{t \in \mathbf{T}_{\mathbf{U}} \mid 1 \leq t \leq y\}) = \sum_{t=1,\ t\in\mathbf{T}_{\mathbf{U}}}^{y} P_{RT}(t).$$

In particular, for every $k \in \mathbf{T}_{\mathbf{U}}$, we have $CDF_{RT}\,(k) = \sum_{t=1}^{k} P_{RT}\,(t)\,.$

The inverse cumulative distribution function which produces the quantiles of RT is then equal to

$$\mathbf{q}_{RT}\,(r) = \inf\,\{k \in \mathbf{T}_{\mathbf{U}} \mid CDF_{RT}\,(k) \geq r\}\,,\ r \in (0,1)\,.$$

For $\varepsilon \in (0,1)$ we now use the $(1-\varepsilon)$ −quantile $\mathbf{q}_{RT}\,(1-\varepsilon)$ as a *probabilistic threshold* separating the "the upper $\varepsilon-$tail" of the distribution, i.e. those very large running-times t making the event "$\mathbf{U}(x)[t] < \infty$" negligible according to P_{RT}.

6 A computable discrete probability distribution for finite running-times

In this section we propose a discrete, computable and "natural" probability distribution for (7) $P_{RT} = P_\rho$.

[3] An example will be presented in Section 6.

A computable probability distribution for $\mathbf{T_U}$ is a discrete probability distribution ρ such that $\rho(t)$ is a computable real[4] for each $t \in \mathbf{T_U}$. We construct the computable probability distribution ρ by using the function $f = f_{\mathbf{U}} \colon \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{Z}^+$ defined by

$$f(x,t) = \begin{cases} \frac{2^{-x}}{t}, & \text{if } x \in \text{Stop}(\mathbf{U},t), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The function f is computable because the sets $\text{Stop}(\mathbf{U},t)$ are computable for every $t \in \mathbb{Z}^+$.

Theorem 4. *The real numbers*

$$\upsilon(t) = \upsilon_{\mathbf{U}}(t) = \sum_{x=1}^{\infty} f(x,t), \tag{9}$$

$$\Upsilon = \Upsilon_{\mathbf{U}} = \sum_{t=1}^{\infty} \upsilon(t) \tag{10}$$

are computable and $0 \leq \upsilon < \Upsilon < 1$.

Proof. We first note that because $\text{Stop}(\mathbf{U},t) \subset \mathbb{Z}^+$ we have:

$$\upsilon(t) = \sum_{x=1}^{\infty} f(x,t) = \sum_{x=1,\, x \in \text{Stop}(\mathbf{U},t)}^{\infty} \frac{2^{-x}}{t} < \frac{1}{t}.$$

Next we show that for every $t \in \mathbb{Z}^+$ the real $\upsilon(t) = 0.\upsilon_1\upsilon_2\ldots\upsilon_n\ldots$ is computable. If $x \geq n$ is in $\text{Stop}(\mathbf{U},t)$, then x contributes to $\upsilon(t)$ with the term $2^{-x}/t \leq 2^{-n}/t \leq 2^{-n}$. Hence, if all $x \geq n$ would be in $\text{Stop}(\mathbf{U},t)$, then their cumulative contribution to $\upsilon(t)$ would be $\sum_{x=n}^{\infty} 2^{-x}/t < 2^{-n}/t \leq 2^{-n}$, hence $\upsilon(t) < 0.\upsilon_1\upsilon_2\ldots\upsilon_n + 2^{-n}$. To determine $\upsilon_1, \upsilon_2, \ldots, \upsilon_n$ we just need to calculate $\sum_{x=1}^{n} 2^{-x}/t$.

A similar argument works for Υ. The solution is to run all non-stopping programs x for enough time such that their cumulative contribution becomes too small to affect the first $n-1$ digits of Υ. Consider the binary expansion $\Upsilon = 0.\gamma_1\gamma_2\ldots$ and $n \geq 2$. If all programs $x \geq n$ would halt on $\mathbf{U}$ in time 1 (an impossibility as there exist programs with arbitrarily large running-times), then their cumulative contribution to Υ would be less than 2^{-n}, that is, $\Upsilon < 0.\gamma_1\gamma_2\ldots\gamma_n + 2^{-n}$. We now run in parallel the programs $x = i, 1 \leq i \leq n$, for at most time $b_i = n \cdot 2^{n-i}$, respectively, add $2^{-i}/t_i$ for each program which has stopped in time $t_i \leq b_i$ and dismiss the ones that have not halted in time b_i. In this way we get all the correct bits $\gamma_1\gamma_2\ldots\gamma_{n-1}$ as the cumulative contribution of the dismissed programs – some of which may eventually stop – is at most $n \cdot 2^n$. As $\sum_{t=1}^{\infty} 2^{-t} = 1$, we have $0 < \Upsilon < 1$; in view of (10), $0 \leq \upsilon < \Upsilon$. □

Using Theorem 4 we construct by normalisation the following *computable probability distribution* ρ on the set of finite running-times $\mathbf{T_U}$:

$$\rho(t) = \frac{\upsilon(t)}{\Upsilon} = \frac{1}{\Upsilon} \sum_{x=1}^{\infty} f(x,t), \;\; t = 1, 2, \ldots \tag{11}$$

[4]That is, there is an algorithm computing the nth binary digit of the binary expansion of $\rho(t)$, see [21, p. 159].

Indeed, from (11) and (10) we have:

$$\sum_{t=1}^{\infty} \rho(t) = \sum_{t=1}^{\infty} \frac{1}{\Upsilon} \sum_{x=1}^{\infty} f(x,t) = 1.$$

Using ρ we define the computable discrete probability space

$$(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_\rho),$$

where $\mathcal{B}(\mathbf{T_U})$ is the set of all subsets of $\mathbf{T_U}$ and $P_\rho(t) = \rho(t)$.

The series (9) is a computable semi-measure [17, Section 4] with a computable sum[5] – by Theorem 4, (10) – an essential property for the computability of the probability P_ρ.

The above probability space – inspired from [7] – is "natural" because the discrete probability distribution combines the construction of the halting probability Ω number, see [3], with the time complexity of halting programs (normalised by the computable number Υ, see (10)). In detail, the function (12) biases the programs x – assumed to be uniformly distributed – by dividing 2^{-x} to the program's stopping time t: according to [7], the longer t is, the smaller the halting probability becomes, so if the program never halts, that is $t = \infty$, the probability is zero.

The contribution of the running time t – which is essential for assuring the computability of Υ, see the proof of Theorem 4 – decreases significantly the probability P_ρ posing a challenge for the implementation of the corresponding anytime algorithms presented in the next section. The value of t can be decreased to $\log(t+1)$ or, more generally, to $g(t)$, where g is a non-decreasing, unbounded, computable function. A more substantial improvement can be obtained using Proposition 1.5.2 in [20].

To further justify the "naturalness" of the probability P_ρ we need to show that it reflects the behaviour of *both* halting and non-halting programs. To this aim we use the series (9) to define a variation of ρ, namely a computable probability distribution r on the set of all running times, finite or infinite, $\mathbf{T_U} \cup \{\infty\}$ as follows:

$$r(t) = \begin{cases} \upsilon(t), & \text{if } t \in \mathbf{T_U}, \\ 1 - \Upsilon, & t = \infty. \end{cases} \tag{12}$$

As by (10) and (11)

$$\sum_{t=1}^{\infty} r\left(t\right) + r\left(\infty\right) = \sum_{t=1}^{\infty} \sum_{x=1}^{\infty} f\left(x,t\right) + r\left(\infty\right) = 1,$$

we can define $P_r(t) = r(t)$ for $t \in \mathbf{T_U} \cup \{\infty\}$ to obtain the computable probability space $(\mathbf{T_U} \cup \{\infty\}, \mathcal{B}(\mathbf{T_U} \cup \{\infty\}), P_r)$, where $\mathcal{B}(\mathbf{T_U} \cup \{\infty\})$ is the set of all subsets of $\mathbf{T_U} \cup \{\infty\}$.

In contrast with P_ρ – which deals only with finite running times – P_r handles also the infinite running time, the running time of non-halting programs. The normalisation factor Υ makes P_ρ "reflect" the behaviour of non-halting programs too as the restriction of P_r to $\mathbf{T_U}$ is

[5] The sum of a computable semi-measure may be not computable as in Specker theorem [23].

$$P_\rho(t) = \frac{P_r(t)}{\Upsilon},\ t \in \mathbf{T_U}.$$

Of course, there are many other computable probability distributions for $\mathbf{T_U}$.

7 Anytime algorithms for the halting problem

As we mentioned in Section 3, to solve the *halting problem* is enough to fix a universal $\mathbf{U}$ and to decide, for an arbitrary program x, whether $\mathbf{U}(x) < \infty$ or $\mathbf{U}(x) = \infty$.

Our aim is to construct two classes of anytime algorithms for testing the incomputable predicate "$\mathbf{U}(x) < \infty$". The decision to accept/reject the hypothesis "$\mathbf{U}(x) < \infty$" will be based on the running-time of the computation $\mathbf{U}(x)$. A decision made by the anytime algorithm is *erroneous* when it returns the output "$\mathbf{U}(x) = \infty$", when, in fact, $\mathbf{U}(x) < \infty$ ($\mathbf{U}(x)$ eventually stops after a very long time).

The halting problem will be re-formulated within the probabilistic framework presented in Section 5 as follows:

$$\begin{gathered}\text{For arbitrary } x \in Z^+, \text{ test the hypothesis } H_x : \{\mathbf{U}(x) < \infty\} \\ \text{against the alternative } H'_x : \{\mathbf{U}(x) = \infty\}.\end{gathered}$$

The decision of rejecting H_x will be taken on the basis of a *critical time region* B_x. In both proposed anytime algorithms, the critical regions will *not depend on* x, that is, $B = B_x$, for every $x \in \mathbb{Z}^+$.

An erroneous decision occurs when we reject H_x on the basis of B, but H_x is true. The quality of this decision is expressed by the probability of an erroneous decision, i.e. the probability that a halting program x stops in a time $t \in B$.

In what follows we will work with an a priori computable discrete probability space $(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_{RT})$ defined in (7) and the running-time random variable RT defined in (6). Our main example is $P_{RT} = P_\rho$.

7.1 A probabilistic anytime algorithm for the halting problem

In this section we develop our first anytime algorithm based on a fixed computable probability distribution P_{RT}.

Theorem 5. *For every* $\varepsilon \in (0, 1)$ *we have*

$$P_{RT}\left(\{t \in \mathbf{T_U} \mid t > \mathbf{q}_{RT}(1-\varepsilon)\}\right) < \varepsilon, \tag{13}$$

hence $\lim_{\varepsilon \to \infty} P_{RT}\left(\{t \in \mathbf{T_U} \mid t > \mathbf{q}_{RT}(1-\varepsilon)\}\right) = 0$.

Proof. Using the definition of P_{RT} we have:

$$\begin{aligned} P_{RT}\left(\{t \in \mathbf{T_U} \mid t > \mathbf{q}_{RT}\left(1-\varepsilon\right)\}\right) &= 1 - P_{RT}\left(\{t \in \mathbf{T_U} \mid t \leq \mathbf{q}_{RT}\left(1-\varepsilon\right)\}\right) \\ &= 1 - CDF_{RT}\left(\mathbf{q}_{RT}\left(1-\varepsilon\right)\right) \\ &< 1 - (1-\varepsilon) = \varepsilon. \end{aligned}$$

□

We now use the inequality (13) in Theorem 5 to propose the following *probabilistic anytime algorithm* for the halting problem:

> *Fix $\varepsilon = 2^{-M}$ with $M \in \mathbb{Z}^+$. Let x be an arbitrary program for $\mathbf{U}$. If the computation $\mathbf{U}\left(x\right)$ does not stop in time less than or equal to $\mathbf{q}_{RT}\left(1-\varepsilon\right)$, then decide that $\mathbf{U}(x) = \infty$.*

If the computation $\mathbf{U}\left(x\right)$ stops in time less than or equal to $\mathbf{q}_{RT}\left(1-\varepsilon\right)$, then obviously $\mathbf{U}(x) < \infty$. Otherwise, the answer to the question whether $\mathbf{U}\left(x\right) < \infty$ is *unknown* and *algorithmically unknowable*. The above anytime algorithm gives an approximate answer.

To analyse the quality of the answer produced by this anytime algorithm we choose the *computable* critical time region[6]

$$\mathbf{B}(P_{RT}, \varepsilon) = \{t \in \mathbb{Z}^+ \mid t > \mathbf{q}_{RT}(1-\varepsilon)\},$$

and the *critical program region*

$$\mathbf{C}(P_{RT}, \varepsilon) = \{x \in \mathbb{Z}^+ \mid \mathbf{U}(x)[t] = \infty, \text{ for some } t \in \mathbf{B}(P_{RT}, \varepsilon)\}.$$

Note that

$$\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U}) \subset \mathbf{C}(P_{RT}, \varepsilon) \subset \mathbb{Z}^+.$$

The anytime algorithm may output the answer "$\mathbf{U}(x) = \infty$" when in fact $\mathbf{U}(x) < \infty$. To evaluate the quality of the anytime algorithm we need to "compare" the set $\mathbf{C}(P_{RT}, \varepsilon)$ – which gives the "anytime" answers "$\mathbf{U}(x) = \infty$" – with the exact set $\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U})$ – giving the correct answers "$\mathbf{U}(x) = \infty$". Theorem 5 does this indirectly, by looking at the running-times, that is by evaluating

> for *every* program $x \in \mathbf{C}(P_{RT}, \varepsilon)$, the probability P_{RT} that x stops in a time in $\mathbf{B}(P_{RT}, \varepsilon)$; this probability is less than ε.

The smaller ε is chosen, the less chances (according to P_{RT}) are for any program which does not stop in time smaller than $\mathbf{q}_{RT}\left(1-\varepsilon\right)$ to eventually halt.

This result is stronger than the ones obtained in [7, 6] where the probability and the stopping decision depend on the program x.

[6] $B_{P_{RT},\varepsilon}$ is independent of x.

To implement the anytime algorithm above we need an algorithm to compute

$$\mathbf{q}_{RT}\left(1-2^{-M}\right)=\min\left\{t\in\mathbf{T}_{\mathbf{U}}\mid CDF_{RT}(t)\geq 1-2^{-M}\right\}.$$

As the set $\mathbf{T}_{\mathbf{U}}$ is only computable enumerable, we will not be able to compute exactly $\mathbf{q}_{RT}\left(1-2^{-N}\right)$, but an upper bound for it. To this aim we consider a computably enumeration of $\mathbf{T}_{\mathbf{U}}=\{t_1,t_2,\ldots,t_i,\ldots\}$ and compute the following new bound:

$$\widetilde{\mathbf{q}_{RT}}\left(1-2^{-M}\right)=\min\left\{t\in\mathbb{Z}^{+}\mid\sum_{t=1}^{t}\geq 1-2^{-M}\right\}\geq\mathbf{q}_{RT}\left(1-2^{-M}\right).$$

Obviously, the anytime algorithm will work correctly with the larger bound, but this will increase its time complexity.

7.2 A statistically-fitted anytime algorithm for the halting problem

In what follows we will work in an *abstract, unknown* computable discrete probability space $(\mathbf{T}_{\mathbf{U}},\mathcal{B}(\mathbf{T}_{\mathbf{U}}),P_{RT})$ as in (7) and we will use an inferential approach to construct a statistically-fitted anytime algorithm for the halting problem.

We will sample $\mathbf{T}_{\mathbf{U}}$ to obtain a long sequence of independent, identically distributed running-times $(t_1,...,t_N)$ according to the discrete random variable RT and choose, using Theorem 2, $t_{(\lceil N(1-\varepsilon)\rceil)}$, the order statistic of rank $(\lceil N(1-\varepsilon)\rceil)$, as the *fitted threshold*.

We use the following *sampling algorithm* for the set $\mathbf{T}_{\mathbf{U}}$:

1. *First we run* $\mathbf{U}(x)$ *for many programs* x *according to a dovetailing method, i.e. we run in parallel the programs controlled by their time complexities. In this way we can obtain a very large set of halting programs (all stopping in a time less than a fixed large time) as well as their stopping times.*

2. *Next we perform a random sampling to obtain* N *identically distributed running-times from the generated halting programs.*

In detail: we denote the halting programs by

$$\mathrm{POS}_L=(x_1,\ldots,x_L)$$

and their running-times by

$$\mathbf{T}_{\mathrm{POS}_L}=(\tau_1,\ldots,\tau_L).$$

By construction, the running-times in $\mathbf{T}_{\mathrm{POS}_L}$ are independent.

In the second step we implement a random sampling (see, for example, [16]) to extract N identically distributed running-times from $\mathbf{T}_{POS_L}$,

$$(t_1,...,t_N),$$

which represent N independent, identically distributed replicates of the random variable RT, so Theorem 2 applies.

Next we consider the order statistics $t_{(1)} \leq t_{(2)} \leq \cdots \leq t_{(N)}$ (see (5)). For a small $\varepsilon > 0$ we use Theorem 2, that is,

$$\mathbf{q}_{RT,N}(1-\varepsilon) = t_{(\lceil N(1-\varepsilon)\rceil)},$$

to define the *fitted threshold* as the order statistic of rank $\lceil N(1-\varepsilon)\rceil$. The *computable* critical time region[7] is then

$$\mathbf{B}(RT, N, \varepsilon) = \{t \in \mathbb{Z}^+ \mid t > \mathbf{q}_{RT,N}(1-\varepsilon)\} = \{t \in \mathbb{Z}^+ \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\}.$$

Theorem 6. *Fix $N \in \mathbb{Z}^+$ and $\varepsilon \in (0,1)$. Then, the probability P_{RT} of an erroneous decision based on the critical time region $\mathbf{B}(RT, N, \varepsilon)$ is*

$$P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) = 1 - \sum_{t=1,\ t\in\mathbf{T_U}}^{t_{(\lceil N(1-\varepsilon)\rceil)}} P_{RT}(t). \tag{14}$$

Proof. Let $x \in \mathbb{Z}^+$ be an arbitrary program for which the hypothesis $H_x : \{\mathbf{U}(x) < \infty\}$ *is true*. Then, with respect to the probability space $(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_{RT})$, the probability of an erroneous decision based on $\mathbf{B}(RT, N, \varepsilon)$ is:

$$\begin{aligned} P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) &= 1 - P_{RT}\left(RT(x) \leq t_{([N(1-\varepsilon)])}\right) \\ &= 1 - CDF_{RT}\left(t_{(\lceil N(1-\varepsilon)\rceil)}\right) \\ &= 1 - \sum_{t=1,\ t\in\mathbf{T_U}}^{t_{(\lceil N(1-\varepsilon)\rceil)}} P_{RT}(t). \end{aligned}$$

□

Next we show that the error probability (14) can be made arbitrarily small.

Lemma 7. *For every $\alpha \in (0,1)$, $\lim_{N\to\infty} t_{(\lceil N\alpha\rceil)} = \infty$.*

Proof. The sequence of positive integers $(t_{(\lceil N\alpha\rceil)})_{N\geq 1}$ is non-decreasing and, by Lemma 3, unbounded. □

We can now prove the following analogues of Theorem 5.

Corollary 8. *For every $\varepsilon \in (0,1)$, $\lim_{N\to\infty} P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) = 0$.*

Proof. In view of (14), (7) and Lemma 7 we have:

$$\lim_{N\to\infty} P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) = 1 - \lim_{N\to\infty} \sum_{t=1,\ t\in\mathbf{T_U}}^{t_{(\lceil N(1-\varepsilon)\rceil)}} P_{RT}(t) = 0.$$

□

[7] $B_{\mathbf{U},N,\varepsilon}$ is independent of x.

Corollary 9. *For every* $N \in \mathbb{Z}^+$,

$$\lim_{\varepsilon \to 0} P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) = 1 - \sum_{t=1,\ t \in \mathbf{T_U}}^{t_{(N)}} P_{RT}(t).$$

Proof. Fix an $N \in Z$. If $0 < \varepsilon < \varepsilon'$, then $\mathbf{B}(RT, N, \varepsilon) \supseteq \mathbf{B}(N, \varepsilon')$, hence

$$\lim_{\varepsilon \to 0}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) = \left\{t \in \mathbf{T_U} \mid t > t_{(N)}\right\}.$$

Finally, we have:

$$\begin{aligned}\lim_{\varepsilon \to 0} P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) &= \lim_{\varepsilon \to 0}\left(1 - CDF_{RT}\left(t_{([N(1-\varepsilon)])}\right)\right) \\ &= 1 - CDF_{RT}\left(t_{(N)}\right) \\ &= 1 - \sum_{t=1,\ t \in \mathbf{T_U}}^{t_{(N)}} P_{RT}(t).\end{aligned}$$

□

Using Corollary 8 and Corollary 9 we obtain the following *statistically-fitted anytime algorithm* for the halting problem:

> *Fix* $\varepsilon = 2^{-M}$ *with* $M \in \mathbf{Z}^+$ *and a sample size* $N \in \mathbf{Z}^+$. *Use the sampling algorithm to generate the running-times* $\{t_{(1)}, t_{(2)} \ldots, t_{(N)}\}$. *Let* x *be an arbitrary program for* $\mathbf{U}$. *If the computation* $\mathbf{U}(x)$ *does not stop in time less than or equal to* $t_{(\lceil N(1-\varepsilon)\rceil)}$, *then decide that* $\mathbf{U}(x) = \infty$.

Denoting the *critical programs region* by

$$\mathbf{C}(RT, N, \varepsilon) = \{x \in \mathbb{Z}^+ \mid \mathbf{U}(x)[t] = \infty, \text{ for some } t \in \mathbf{B}(RT, N, \varepsilon)\}$$

we get

$$\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U}) \subset \mathbf{C}(RT, N, \varepsilon) \subset \mathbb{Z}^+.$$

To evaluate the quality of the statistically-fitted anytime algorithm we need to "compare" the set $\mathbf{C}(RT, N, \varepsilon)$ – which gives the "anytime" answers "$\mathbf{U}(x) = \infty$" – with the exact set $\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U})$ – giving the correct answers "$\mathbf{U}(x) = \infty$". We proceed in the same way as for the probabilistic anytime algorithm, i.e. by looking at the running-times. Fix $\delta \in (0, 1)$. Using Corollary 8 with given $\varepsilon \in (0, 1)$ we get an $N \in \mathbb{Z}^+$ such that for *every* program $x \in \mathbf{C}(RT, N, \varepsilon)$, the probability P_{RT} that x stops in a time in the set $\mathbf{B}(RT, N, \varepsilon)$ is smaller than δ. This procedure can be fine-tuned by using Corollary 9.

There are many choices of random sampling algorithms which can be used in the second step of the sampling algorithm. The choices determine how large should be L – the number of halting programs – as a function of N as well as the quality of "randomness" (see [4]).

Let $P_{RT}(t) = P_\rho(t), \varepsilon = 2^{-M}, \delta = 2^{-K}$. The following algorithm – based on Corollary 8 and Corollary 9 – calculates N (in terms of M, K) such that

$$P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon)\rceil)}\right\}\right) < \delta.$$

First calculate $\min\{T \in \mathbb{Z}^+ \mid \sum_{t=1}^{T} \rho(t) > 1-2^{-K}\}$ *and then calculate* $\min\{N \in \mathbb{Z}^+ \mid t_{(\lceil N(1-2^{-M})\rceil)} \geq T\}$ *using the sampling algorithm.*

8 Conclusions

In this paper we have proposed two classes of anytime algorithms for the halting problem, a theoretical one based on probability theory and another, more practical one, based on order statistic.

Our proposed anytime algorithms depend on the computable probability distribution (first algorithm) and sampling method (second algorithm). Making the "right" choices is essential for successful applications. There are a few ways to guide and improve the quality of these choices. One possibility is to test how "natural" is a particular computable probability distribution for some universal $\mathbf{U}$ using the sampling algorithm and, vice–versa, to test how "fit" is the sampling algorithm with respect to some preferred computable probability distribution, for example, P_ρ.

For example we can use the two-sample Kolmogorov-Smirnov goodness-and-fit test (see [9, pp. 309–314]) to test how "natural" is a particular computable probability distribution, say P_ρ, for a given universal $\mathbf{U}$.

Using the sampling algorithm on $\mathbf{U}$ we produce the sample $(t_1, ..., t_N)$ which represents N independent, identically distributed replicates of the random variable $RT = RT_{\mathbf{U}}$ and use the associated Empirical Cumulative Distribution Function (see (4)) $ECDF_{RT,N}$ (that is, $X = RT$). On the other hand, using a simulation technique we generate K independent, identically distributed values $(u_1, ..., u_K)$ from the probability distribution P_ρ and denote the associated Empirical Cumulative Distribution Function by $ECDF_{\rho,N}$.

The two-sample Kolmogorov-Smirnov test compares these two empirical distribution functions in order to accept/reject *the null hypothesis* that the two datasets were drawn from "the same stochastic source". The null hypothesis, denoted by $H_0 : \{P_{RT} = P_\rho\}$, states that the data produced by the sampling algorithm for the particular universal $\mathbf{U}$ fits the computable probability distribution P_ρ. The decision of accepting/rejecting H_0 is taken on the basis of numerical comparison of $ECDF_{RT,N}$ and $ECDF_{\rho,N}$.

Accordingly, the following algorithm can be used to run the probabilistic anytime algorithm:

(1) Choose a computable probability distribution $\{P_{RT}(t), t \in \mathbf{T}_{\mathbf{U}}\}$ *as in* (7).

(2) Choose N *and use the sampling algorithm to generate* $(t_1, \ldots, t_N)$.

(3) Apply a goodness-of-fit test to compare the chosen probability model (7) *with the sample distribution of the data* $(t_1, \ldots, t_N)$.

(4) Choose a particular ε.

- *If the model fits the data, then continue with the probabilistic anytime algorithm.*

- *If the model does not fit the data, then continue with the statistically-fitted anytime algorithm.*

Next we enumerate some basic features of our proposed anytime algorithms. The following are positive features:

(P1) We have proposed two classes of anytime algorithms.

(P2) The cut-off temporal bounds for both classes of anytime algorithms do not depend on programs.

(P3) The a priori computable probability $P_{RT} = P_\rho$ was not arbitrarily chosen: it reflects the halting behaviour of the chosen universal machine. Two methods to construct natural variations of this probability have been presented.

(P4) We can test empirically the choice of the computable probability distribution and sampling, hence adopt parameters suiting different universal machines and classes of programs.

However, the approach has limits:

(L1) Because of (P1) and the use of a universal machine, the cut-off temporal bounds derived for the probabilistic anytime algorithm are expected to be large. This can be mitigated to some extent by (P3).

(L2) The statistically-fitted anytime algorithm is "P_{RT} free", but the evaluation of its quality is not, hence the choice of N is not "P_{RT} free".

(L3) Working with a fixed universal machine and programs x instead of pairs $(program,\ y)$ increases the computational time as the simulation of the computation $program(y)$ on $\mathbf{U}$, that is, $\mathbf{U}(x) = program(y)$, takes longer than running $program(y)$.

(L4) We don't have a computational complexity analysis of concrete examples of our proposed anytime algorithms.

The experimental results in [24] suggest that some of the anytime algorithms in the proposed classes may be indeed feasible. It is a natural important followup to develop theoretical and experimental analyses of various anytime algorithms in both classes.

Acknowledgement

We thank Yu. Manin, L. Staiger and G. Tee for useful comments and suggestions.

References

[1] Barry C. Arnold, N. Balakrishnan and H. N. Nagaraja. *A First Course in Order Statistics*, John Wiley, New York, 2008.

[2] L. Bienvenu, D. Desfontaines, A. Shen. What percentage of programs halt? in M. M. Halldórsson, K. Iwama, N. Kobayashi, B. Speckmann (eds.). *Automata, Languages, and Programming I*, Lecture Notes in Computer Science 9134, Springer, 2015, 219–230.

[3] C. S. Calude. *Information and Randomness: An Algorithmic Perspective*, Springer-Verlag, Berlin, 2002. (2nd edition)

[4] C. S. Calude. Quantum randomness: From practice to theory and back, in S. B. Cooper, M. Soskova (eds.). *The Incomputable: Journeys Beyond the Turing Barrier*, Springer, Berlin, to appear, 2016.

[5] C. S. Calude and D. Desfontaines. Universality and almost decidability, *Fundamenta Informaticae* 138(1-2) (2015), 77–84.

[6] C. S. Calude and D. Desfontaines. Anytime algorithms for non-ending computations, *International Journal of Foundations of Computer Science* 26, 4 (2015), 465–475.

[7] C. S. Calude and M. A. Stay. Most programs stop quickly or never halt, *Advances in Applied Mathematics* 40 (2008), 295–308.

[8] A. DasGupta. *Probability for Statistics and Machine Learning*, Springer, New York, 2011.

[9] W. J. Conover. *Practical Nonparametric Statistics* John Wiley, New York, 1971.

[10] C. A. Furia, D. Mandrioli, A. Morzenti and M. Rossi. *Modeling Time in Computing*, Springer, Berlin, 2012.

[11] R. Downey and D. Hirschfeldt. *Algorithmic Randomness and Complexity*, Springer, Heidelberg, 2010.

[12] J. Grass. Reasoning about computational resource allocation. An introduction to anytime algorithms, *Magazine Crossroads* 3, 1 (1996), 16–20.

[13] P. J. Haas. *Simulation. Quantile Estimation*, Lecture Notes #9, Stanford University, MS & E 223, Spring Quarter 2005–2006, http://web.stanford.edu/class/msande223/handouts/lecturenotes09.pdf.

[14] J. D. Hamkins and A. Miasnikov. The halting problem is decidable on a set of asymptotic probability one, *Notre Dame Journal of Formal Logic* 47 (4) (2006), 515–524.

[15] R. H. Lathrop. On the learnability of the uncomputable, in L. Saitta (ed.). *Proceedings International Conference on Machine Learning*, Morgan Kaufmann, 1996, 302–309.

[16] P. S. Levy and S. Lemeshow. *Sampling of Populations. Methods and Applications*, John Wiley, NJ, 1999. (3rd edition)

[17] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, New York, 2008. (3rd edition)

[18] Yu. I. Manin. *A Course in Mathematical Logic for Mathematicians*, Springer, Berlin, 2010. (2nd edition)

[19] Yu. I. Manin. Renormalisation and computation II: time cut-off and the Halting Problem, *Mathematical Structures in Computer Science* 22 (2012), 729–751.

[20] Y. I. Manin. Zipf's law and L. levin probability distributions. *Functional Analysis and Its Applications*, 48, 2 (2014), 116–127.

[21] M. Minsky. *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc. Englewood Cliffs, NJ, 1967.

[22] P. Olofsson. *Probability, Statistics, and Stochastic Processes*, Wiley-Interscience, New York, 2005.

[23] E. Specker. Nicht konstruktiv beweisbare Sätze der Analysis. *The Journal of Symbolic Logic* 14 (1949), 145–158.

[24] H. Zenil. Computer runtimes and the length of proofs, in M. J. Dinneen, B. Khoussainov, A. Nies (eds.). *Computation, Physics and Beyond*, Lecture Notes in Computer Science 7160, Springer, 2012, 224–240.