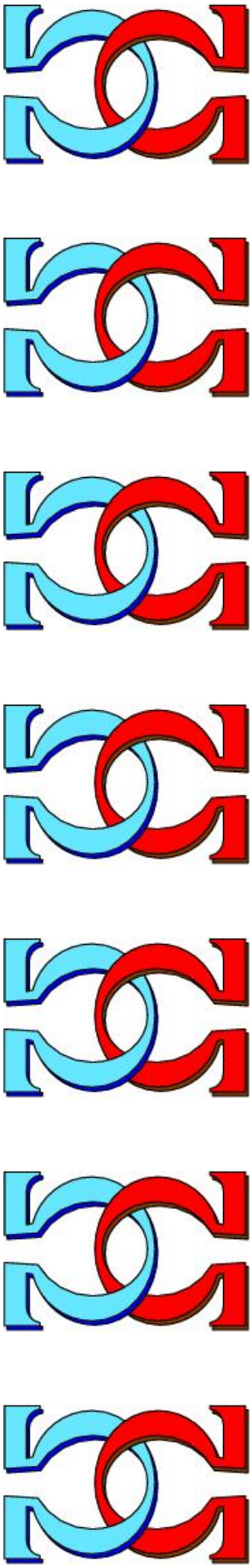

CDMTCS
Research
Report
Series

A Statistical Anytime Algorithm for the Halting Problem

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

[1]University of Auckland, New Zealand
[2]University of Bucharest, Romania

CDMTCS-529
November 2018; revised March 2020

Centre for Discrete Mathematics and
Theoretical Computer Science

Computability 1 (2019) 1–14
DOI
IOS Press

A Statistical Anytime Algorithm for the Halting Problem

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

Monica Dumitrescu
Faculty of Mathematics and Computer Science, Bucharest University, Romania
mdumi@fmi.unibuc.ro
http://goo.gl/txsqpU

Abstract. In a previous paper we used computer running times to define a class of computable probability distributions on the set of halting programs and developed a probabilistic anytime algorithm for the Halting Problem. The choice of a computable probability distribution – essential for the algorithm – can be rather subjective and hard to substantiate.

In this paper we propose and study an efficient statistical anytime algorithm for the Halting Problem. The main advantage of the statistical algorithm is that it can be implemented without any prior information about the running times on the specific model of computation and the cut-off temporal bound is reasonably small. The algorithm has two parts: the pre-processing which is done only once (when the parameters of the quality of solutions are fixed) and the main part which is run for any input program. With a confidence level as large as required, the algorithm produces correct decisions with a probability as large as required. Three implementations of the algorithm are presented and numerically illustrated.

Keywords: Halting Problem, anytime algorithm, running time, order statistics

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 A. Church, and independently A. Turing, proved that there is *no algorithm solving the Halting Problem for all possible program-input pairs*. The Halting Problem has many applications in 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 [2, 6–9, 11, 14, 17, 20, 22, 25].

Anytime algorithms trade execution time for quality of results [13]. 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 quality of the solution, anytime algorithms can be continued after they have halted if the output is not considered acceptable.

Here we use a more general form of anytime algorithm as an approximation for a computation which may never stop (see Manin [22]). Running times play an important role in this problem because halting programs are not uniformly distributed, see [16, 27–29] for experimental work and [8, 9, 14, 15] for theoretical results. Furthermore, every program either stops "quickly" or never stops [9]. This result was used in [6] to design an anytime probabilistic algorithm which simulates the program to be tested up to a threshold stopping time (this cut-off temporal bound is computed from the a priori accepted decision error and the probability distribution of stoping times of halting programs): if the computation still has not terminated by then, the algorithm reports (possibly wrongly) 'The program does not halt!'. This anytime algorithm uses essentially a computable probability distribution on the set of stopping times of halting programs "reflecting" the halting

behaviour of the chosen universal machine. The quantile of this probability distribution is utilised to compute the stoping threshold time, hence, the name "anytime probabilistic algorithm". The probability of a wrong decision is no larger than the accepted error.

In this paper we propose a statistical anytime algorithm for the Halting Problem which improves the anytime probabilistic algorithm developed in [6] using a different strategy. In a pre-processing stage we sample sufficiently many terminating programs in an independent way (see [3, 18]), determine their running times and consider the induced empirical cumulative distribution as approximation to the true, but unknown, cumulative distribution. Using an appropriate statistical framework we construct an anytime statistical algorithm which uses three parameters: the probability of an erroneous decision, the precision of and the confidence level in the estimation. The input program – tested for termination – is run for up to the largest number of steps made by any of the sampled programs. If the computation does not terminate by this time threshold (cut-off), then the (possible wrong) decision is that the program will never stop. With a confidence level as large as required, the anytime statistical algorithm produces correct decisions with a probability as large as required. Three implementations of the anytime algorithm are presented; numerical illustrations show that their time complexities are reasonably small.

The paper is organised as follows. We start with Section 2 on computability and complexity part for the Halting problem; Section 3 presents the probability framework and the probabilistic anytime algorithm for the Halting Problem, Section 4 presents the statistical framework, then Section 5 presents the statistical anytime algorithm for the Halting Problem and the proof of its main properties. Finally, in Section 6 we discuss three possible implementations of the statistical algorithm and present numerical illustrations; the last section is devoted to conclusions and possible extensions.

2. The Halting Problem

We denote by $\mathbb{Z}^+$ the set of positive integers $\{1,2,\dots\}$; $\overline{\mathbb{Z}^+} = \mathbb{Z}^+ \cup \{\infty\}$ and $\mathbb{R}$ is the set of reals. For $\alpha \in \mathbb{R}$, $\lceil \alpha \rceil$ is the ceiling function that maps α to the least integer greater than or equal to α. The domain of a partial function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is denoted by $\mathrm{dom}\,(F)$: $\mathrm{dom}\,(F) = \{x \in \mathbb{Z}^+ : F\,(x) < \infty\}$. We denote by $\#S$ the cardinality of the set S and by $\mathcal{P}(X)$ the power set of X. The indicator (or characteristic) function of a set M is denoted by $\mathbf{1}_M$.

We assume familiarity with elementary computability theory and algorithmic information theory [5, 12, 21]. For a partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ we denote by $F(x)[t] < \infty$ the statement "the algorithm computing F has stopped on x *exactly* in time t". For $t \in \mathbb{Z}^+$ we consider the computable set $\mathrm{Stop}(F,t) = \{x \in \mathbb{Z}^+ : F(x)[t] < \infty\}$, and note that

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

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 $\nabla_F\,(x) = \inf\{y \in \mathbb{Z}^+ : F\,(y) = x\}$. If $F\,(y) \neq x$ for every $y \geq 1$, then $\nabla_F\,(x) = \infty$. A partially computable function $\mathbf{U}$ is *universal* if 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 $\nabla_{\mathbf{U}}\,(F\,(x)) \leq c_{\mathbf{U},F} \cdot x$, see [8].

The set $\mathrm{dom}(\mathbf{U})$ (see (1) for $\mathbf{U} = F$) is computably 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 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 $t \in \mathbb{Z}^+$. Solving the Halting Problem for a fixed universal $\mathbf{U}$ is enough to

solve the Halting Problem. From now on we fix a universal function $\mathbf{U}$ and study the Halting Problem "For every $x \in \mathbb{Z}^+$, does $\mathbf{U}(x) < \infty$?".

3. The probabilistic anytime algorithm for the Halting Problem

A *probability space* is a triple $(\Omega, \mathcal{B}(\Omega), \Pr)$, where $(\Omega, \mathcal{B}(\Omega))$ is a measurable space and $\Pr\colon \mathcal{B}(\Omega) \longrightarrow [0,1]$ is a probability measure, see [10, 24]. A *random variable* is a measurable function defined on Ω with values in a set of real numbers; its *probability distribution* is denoted P_X. The random variable X has a *discrete probability distribution* if A is at most countable. A *computable probability distribution* P_X is a discrete probability distribution such that the function $x \in A \hookrightarrow P_X(\{x\})$ is computable (in particular, $P_X(\{x\})$ is a computable real for each $x \in A$, see [23, 30]). *The mean* (or *expected value*) of the discrete random variable $X : \Omega \longrightarrow A$ is defined by $E(X) = \sum_{x \in A} x \cdot \Pr(\{\omega \in \Omega : X(\omega) = x\})$, if the series converges.

The *Cumulative Distribution Function* of a random variable X is the function $CDF_X : \mathbb{R} \longrightarrow [0,1]$ defined by $CDF_X(y) = \Pr(X \leq y), y \in \mathbb{R}$. The *Inverse Cumulative Distribution Function* or *Quantile function* of the random variable X with a discrete distribution is the function $\mathbf{q}_X : [0,1] \longrightarrow A$ defined by $\mathbf{q}_X(p) = \inf\{y \in A : p \leq CDF_X(y)\}$. For more details see [1].

Next we present the probability framework introduced in [6]. The *finite running times* of the computations $\mathbf{U}(x)$ are the set of exact stopping times for the halting programs of $\mathbf{U}$: $\mathbf{T}_\mathbf{U} = \{t \in \mathbb{Z}^+ : \text{ there exists } x \in \mathbb{Z}^+ \text{ such that } x \in \text{Stop}(\mathbf{U}, t)\}$. The family of (finite and countable) unions of sets $\text{Stop}(\mathbf{U}, t), t \in \mathbb{Z}^+$, generates the Borel field $\mathcal{B}(\text{dom}(\mathbf{U}))$. A *computable running time probability space* $(\text{dom}(\mathbf{U}), \mathcal{B}(\text{dom}(\mathbf{U})), \Pr_{\rho_\mathbf{U}})$ is defined from a computable probability distribution ρ on $\mathbf{T}_\mathbf{U}$ by setting $\Pr = \Pr_\rho : \mathcal{B}(\text{dom}(\mathbf{U})) \longrightarrow [0,1]$, $\Pr(\text{Stop}(\mathbf{U}, t)) = \rho(t)$, $t \in \mathbf{T}_\mathbf{U}$.

We introduce a probability structure on the set $\mathbf{T}_\mathbf{U}$ via a random variable. Let $\mathcal{B}(\mathbf{T}_\mathbf{U})$ be the family of all subsets of $\mathbf{T}_\mathbf{U}$. The function $RT = RT_\mathbf{U} : \text{dom}(\mathbf{U}) \longrightarrow \mathbf{T}_\mathbf{U}$, $RT(x) = \min\{t > 0 : x \in \text{Stop}(\mathbf{U}, t)\}$ has the property that for every $t \in \mathbf{T}_\mathbf{U}$, $RT^{-1}(\{t\}) = \text{Stop}(\mathbf{U}, t) \in \mathcal{B}(\text{dom}(\mathbf{U}))$. The *random variable RT* – called the *running time associated with* $\mathbf{U}$ – induces the probability space $(\mathbf{T}_\mathbf{U}, \mathcal{B}(\mathbf{T}_\mathbf{U}), P_{RT})$ on $\mathbf{T}_\mathbf{U}$ in which the probability is defined by $P_{RT}(\{t\}) = \Pr(RT^{-1}(\{t\}))$, $t \in \mathbf{T}_\mathbf{U}$. For every $t \in \mathbf{T}_\mathbf{U}$ we have: $P_{RT}(\{t\}) = \Pr(\text{Stop}(\mathbf{U}, t)) = \rho(t)$. For more details see [6].

Inference-based decisions are made using statistical procedures based on 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* and its complement is called the *acceptance region*.

Consider a probability space $(A, \mathcal{B}(A), P_X)$ induced by a random variable X. Consider an acceptance region $D \subset A$ with $D \in \mathcal{B}(A)$. For every observed value $z \in A$, a *hypothesis* H_z is a predicate such that the sets $\{z \in A : H_z \text{ is true}\}$ and $\{z \in A : H_z \text{ is false}\}$ are in $\mathcal{B}(A)$.

An inference-based decision has the following form:

If the observed value $z \in A$ belongs to $A \setminus D$, then decide to reject the hypothesis H_z.

An error occurs if we reject H_x on the basis of $A \setminus D$ when H_x is true. The *probability of error*, that is, the probability of an erroneous decision, is $P_X(\{x \in A \setminus D\} \mid \{H_x \text{ is true}\})$.

We can reformulate the Halting Problem as an inference-based decision in which we test, for an arbitrary $z \in \mathbb{Z}^+$, the hypothesis $H_z =$ '$\mathbf{U}(z) < \infty$' (remember H_z is a predicate) against the alternative $H'_z =$ '$\mathbf{U}(z) = \infty$'. An "erroneous decision" means rejecting H_z when $\mathbf{U}(z) < \infty$.

The construction of the anytime algorithm for the Halting Problem is based on a computable acceptance region $D \subset \text{dom}(\mathbf{U})$. Accordingly, the algorithm rejects the hypothesis H_z if $z \notin D$. This decision is correct if $z \in \mathbb{Z}^+ \setminus \text{dom}(\mathbf{U})$ and it is wrong if $z \in \text{dom}(\mathbf{U}) \setminus D$. In detail, for an arbitrary program $z \in \mathbb{Z}^+$ there are

three possibilities: a) $z \in D$, b) $z \in \mathrm{dom}(\mathbf{U}) \setminus D$, c) $z \notin \mathrm{dom}(\mathbf{U})$. Condition a) is decidable, but conditions b) and c) are undecidable. If $z \in D$ the anytime algorithm gives the correct decision $\mathbf{U}(z) < \infty$; otherwise $z \in (\mathrm{dom}(\mathbf{U}) \setminus D) \cup (\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U}))$, so the anytime algorithm decides – rightly or wrongly – that $\mathbf{U}(z) = \infty$. Furthermore, for every $z \in \mathbb{Z}^+$,

$$\text{rejecting } H_z \text{ on the basis of } D \text{ is an erroneous decision if and only if } z \in \mathrm{dom}(\mathbf{U}) \setminus D. \tag{2}$$

As the right-hand side of (2) is a subset of $\mathrm{dom}(\mathbf{U})$ we don't need to work in $\mathbb{Z}^+$ but in $\mathrm{dom}(\mathbf{U})$, that is, in the probability space $(\mathrm{dom}(\mathbf{U}), \mathcal{B}(\mathrm{dom}(\mathbf{U})), \Pr)$ or, equivalently, in $(\mathbf{T}_\mathbf{U}, \mathcal{B}(\mathbf{T}_\mathbf{U}), P_{RT})$. The goal to minimise the probability of an erroneous decision can be achieved in this space as long as D and $\mathrm{dom}(\mathbf{U}) \setminus D$ are measurable sets. This condition is satisfied by taking the decidable set of the form $D = \bigcup_{t=1}^{T} \mathrm{Stop}(\mathbf{U}, t)$, for some appropriate T.

The anytime probabilistic algorithm proposed in [6] is:

> *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 $\mathbf{q}_{RT}(1-\varepsilon)$, then declare that $\mathbf{U}(x) = \infty$.*

The acceptance program region of the algorithm is

$$D_\mathbf{U}(P_{RT}, \varepsilon) = \{x \in \mathrm{dom}(\mathbf{U}) : RT(x) \leq \mathbf{q}_{RT}(1-\varepsilon)\}, \tag{3}$$

where $\varepsilon \in (0,1)$ is the decision error and the threshold $\mathbf{q}_{RT}(1-\varepsilon)$ is the $(1-\varepsilon)$ −quantile of the probability distribution P_{RT}. The correctness of the algorithm comes from the inequality on the *critical program region* $\mathbf{C}(P_{RT}, \varepsilon) = \mathrm{dom}(\mathbf{U}) \setminus D_\mathbf{U}(P_{RT}, \varepsilon)$: $\Pr(\mathbf{C}(P_{RT}, \varepsilon)) \leq \varepsilon$.

Condition (3) can be equivalently stated in terms of running times as:

$$D_{\mathbf{T}_\mathbf{U}}(P_{RT}, \varepsilon) = \{t \in \mathbf{T}_\mathbf{U} : t \leq \mathbf{q}_{RT}(1-\varepsilon)\}). \tag{4}$$

The critical time region is $\mathbf{B}(P_{RT}, \varepsilon) = \{t \in \mathbf{T}_\mathbf{U} : t > \mathbf{q}_{RT}(1-\varepsilon)\}$ and the correctness comes from $P_{RT}(\mathbf{B}(P_{RT}, \varepsilon)) \leq \varepsilon$.

4. Statistical framework

The notions and results discussed in Section 3 are based on the assumption that the computable probability distribution on the set of finite running times of programs of $\mathbf{U}$ and the random variable RT are known. In case this assumption is not satisfied, can an inferential approach be used to extract information about the true but unknown probability distribution of the random variable RT from observations of the phenomenon described by RT? More precisely, instead of working with the theoretical CDF_X, can we can we approximate the true, unknown probability distribution of the random variable RT by means of a (long-enough) sequence $X_1, \ldots, X_N$ of independent, identically distributed random variables with the same distribution as RT? The answer is affirmative.

The following form of Hoeffding's inequality (see [26]) is essential in what follows:

Theorem 4.1. *Let $N > 0$ be an integer and $a, b \in \mathbb{R}, a < b$. For every $X_1, \ldots, X_N$ independent random variables defined on $(\Omega, \mathcal{B}(\Omega), \Pr)$ with values in $[a, b]$ we have:*

$$\Pr\left(\left\{\omega \in \Omega : \frac{1}{N}\sum_{i=1}^{N} X_i(\omega) - E\left(\frac{1}{N}\sum_{i=1}^{N} X_i\right) \leq \lambda\right\}\right) \geq 1 - \exp\left(-\frac{2N\lambda^2}{(b-a)^2}\right).$$

Consider the probability space $(\Omega, \mathcal{B}(\Omega), \Pr)$, the random variable $X\colon \Omega \longrightarrow A$ and N replicates $X_1, \ldots, X_N$ of X. In what follows $(x_1, \ldots, x_N) \in A^N$ will denote the observed values of a sample of size N corresponding to the random variables $X_1, \ldots, X_N$: $(x_1, \ldots, x_N) = (X_1(\omega), \ldots, X_N(\omega)) \in A^N$. The vector $(x_1, \ldots, x_N)$ will be called an *N-dimensional sample* and its values $x_1, \ldots, x_N$ *data points*. The *Empirical Cumulative Distribution Function* is defined by

$$ECDF_{X,N}(y) = \frac{\#\{1 \leq i \leq N : x_i \leq y\}}{N}, y \in \mathbb{R}. \tag{5}$$

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

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N-1)} \leq x_{(N)}. \tag{6}$$

The *order statistics of rank k* is the kth smallest value in (6): $X_{(k)}(\omega) = x_{(k)}$. See more in [10, Ch. 6]).

The statistical anytime algorithm assumes that the *probability distribution of RT is unknown.* Therefore, the cumulative distribution function of RT, $CDF_{RT}(t) = \Pr(\{x \in \mathrm{dom}(\mathbf{U}) : RT(x) \leq t\})$, is also unknown and has to be estimated. For evaluating the quality of the approximation we fix a positive integer N and consider the true, unknown $N-$dimensional *program sampling space* $\left(\mathrm{dom}(\mathbf{U})^N, \mathcal{B}\left(\mathrm{dom}(\mathbf{U})^N\right)\right)$.

The elements of $\mathrm{dom}(\mathbf{U})^N$ will be denoted by $\mathbf{x} = (x_1, \ldots, x_N)$. Projections $\{pr_1, \ldots, pr_N\}$, $pr_i\colon \mathrm{dom}(\mathbf{U})^N \longrightarrow \mathrm{dom}(\mathbf{U}), pr_i(\mathbf{x}) = x_i$, $i = 1, \ldots, N$, are independent random variables. If we denote by $RT_i = RT \circ pr_i\colon \mathrm{dom}(\mathbf{U})^N \longrightarrow \mathbf{T}_\mathbf{U}$, $RT_i(\mathbf{x}) = RT(x_i)$, $i = 1, \ldots, N$, then $\{RT_1, \ldots, RT_N\}$ are independent, identical distributed random variables. Furthermore, for every $1 \leq i \leq N$, we have:

$$\begin{aligned} CDF_{RT_i}(t) &= \Pr^N\left(\left\{\mathbf{x} \in (\mathrm{dom}(\mathbf{U}))^N : RT(x_i) \leq t, 1 \leq i \leq N\right\}\right) \\ &= \Pr^N(\mathrm{dom}(\mathbf{U}) \times \cdots \times \{x_i \in \mathrm{dom}(\mathbf{U}) : RT(x_i) \leq t\} \times \mathrm{dom}(\mathbf{U}) \times \cdots \times \mathrm{dom}(\mathbf{U})) \\ &= \Pr(\mathrm{dom}(\mathbf{U})) \cdots \Pr(\{x_i \in \mathrm{dom}(\mathbf{U}) : RT(x_i) \leq t\}) \cdot \Pr(\mathrm{dom}(\mathbf{U})) \cdots \Pr(\mathrm{dom}(\mathbf{U})) \\ &= CDF_{RT}(t). \end{aligned}$$

For every $\mathbf{x} \in \mathrm{dom}(\mathbf{U})^N$ we put $RT_i(\mathbf{x}) = t_i(\mathbf{x})$, $1 \leq i \leq N$ and denote the $N-$dimensional *time sampling space* by $\left(\mathbf{T}_\mathbf{U}^N, \mathcal{B}\left(\mathbf{T}_\mathbf{U}^N\right), P_{RT}^N\right)$.

In the following Lemma 4.2, $CDF_{RT}(t)$ is estimated by the Empirical Cumulative Distribution Function (5)

$$\begin{aligned} ECDF_{RT,N}((RT_1(\mathbf{x}), \ldots, RT_N(\mathbf{x})); t) &= \frac{\#\{1 \leq i \leq N : RT_i(\mathbf{x}) \leq t\}}{N} \\ &= \frac{\#\{1 \leq i \leq N : t_i(\mathbf{x}) \leq t\}}{N}. \end{aligned} \tag{7}$$

Lemma 4.2. *For every positive integer N, $t \in \boldsymbol{T}_\mathbf{U}$ and $\lambda \in (0, 1)$, we have:*

$$\Pr^N\left(\left\{\mathbf{x}\in \mathrm{dom}\,(\mathbf{U})^N : ECDF_{RT,N}\left(\left(RT_1(\mathbf{x}),\ldots,RT_N(\mathbf{x})\right);t\right) - CDF_{RT}(t) \leq \lambda\right\}\right) \geq 1-\exp\left(-2N\cdot\lambda^2\right). \tag{8}$$

Proof. On one hand, from (7) we have:

$$ECDF_{RT,N}\left(\left(RT_1(\mathbf{x}),\ldots,RT_N(\mathbf{x})\right);t\right) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}_{\{\mathbf{x}\in \mathrm{dom}\,\mathbf{U}^N:\, RT_i(\mathbf{x})\,\leq\, t\}}.$$

On the other hand, using the linearity of the operator E we have:

$$\begin{aligned} E\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}_{\{\mathbf{x}\in \mathrm{dom}(\mathbf{U})^N:\, RT_i(\mathbf{x})\,\leq\, t\}}\right) &= \frac{1}{N}\sum_{i=1}^{N}E\left(\mathbf{1}_{\{\mathbf{x}\in \mathrm{dom}(\mathbf{U})^N:RT_i(\mathbf{x})\,\leq\, t\}}\right)\\ &= \frac{1}{N}\sum_{i=1}^{N}\Pr^N\left(\left\{\mathbf{x}\in \mathrm{dom}(\mathbf{U})^N : RT_i(\mathbf{x}) \;\leq\; t\right\}\right)\\ &= CDF_{RT}(t). \end{aligned}$$

As $RT_i\colon\ \mathrm{dom}\,(\mathbf{U})^N \longrightarrow \mathbf{T_U},\ RT_i(\mathbf{x}) = RT(x_i),\ i=1,\ldots,N$ are independent random variables, for every $t\in\mathbf{T_U}$, $\mathbf{1}_{\{\mathbf{x}\in \mathrm{dom}(\mathbf{U})^N:RT_i(\mathbf{x})\,\leq\, t\}}\colon\ \mathrm{dom}(\mathbf{U})^N \longrightarrow [0,1],\ i=1,\ldots,N$ are also independent random variables. Consequently, the inequality (8) follows from Theorem 4.1 applied to the random variables $\left\{\mathbf{1}_{\{\mathbf{x}\in \mathrm{dom}(\mathbf{U})^N:RT_i(\mathbf{x})\,\leq\, t\}},\ i=1,\ldots,N\right\}$.

□

If we define the set of "good program samples" by

$$\mathcal{G}_{N,\lambda,t} = \left\{\mathbf{x}\in \mathrm{dom}\,(\mathbf{U})^N : ECDF_{RT,N}\left(\left(RT_1(\mathbf{x}),\ldots,RT_N(\mathbf{x})\right);t\right) - \lambda \leq CDF_{RT}(t)\right\},$$

then by Lemma 4.2 we have

$$\Pr^N(\mathcal{G}_{N,\lambda,t}) \geq 1-\exp(-2N\lambda^2),$$

were λ is the *precision parameter* and $\left(1-\exp\left(-2N\lambda^2\right)\right)$ can be interpreted as the *confidence level* that a program is in $\mathcal{G}_{N,\lambda,t}$, i.e. it is a good program sample.

With this interpretation, Lemma 4.2 says that the probability $\Pr^N$ of the set of programs $\mathbf{x}\in \mathrm{dom}\,(\mathbf{U})^N$ on which $ECDF_{RT,N}\left(\left(RT_1(\mathbf{x}),\ldots,RT_N(\mathbf{x})\right);t\right)$ estimates $CDF_{RT}(t)$ with precision at least λ can be made as "large" as one wishes. To measure the size of this set (according to $\Pr^N$) we introduce the *confidence level* $(1-\delta)$ by the condition

$$\left(1-\exp\left(-2N\cdot\lambda^2\right)\right) \geq (1-\delta), \tag{9}$$

which is equivalent to

$$N \geq N(\lambda,\delta) = \left\lceil \frac{1}{2\lambda^2}\cdot\ln\frac{1}{\delta}\right\rceil. \tag{10}$$

The following result shows that for every $N \geq N(\lambda, \delta)$ the set of good program samples $\mathcal{G}_{N,\lambda,t}$ can be made as "large" as required in probability $\Pr^N$:

Corollary 4.3. *For every* $t \in \boldsymbol{T}_{\mathbf{U}}$, $\lambda \in (0,1)$, $\delta \in (0,1)$ *and* $N \geq \left\lceil \frac{1}{2\lambda^2} \cdot \ln \frac{1}{\delta} \right\rceil$ *we have*

$$\Pr^N \left(\mathcal{G}_{N,\lambda,t} \right) \geq 1 - \delta. \tag{11}$$

5. A statistical anytime algorithm for the Halting Problem

For a fixed confidence level $(1-\delta)$, precision parameter λ and good program sample $\mathbf{x}$ (which produces a "reliable" estimate of CDF_{RT}) we use the critical time region (see (4)) to reject the hypothesis H_z and to measure the probability of an erroneous decision of the anytime algorithm. Accordingly, for $\varepsilon, \lambda \in (0,1)$, the *critical time region* should satisfy the following two conditions:

$$\mathbf{B}\left(RT, \mathbf{x}; \varepsilon, \lambda\right) = \left\{ t \in \mathbf{T}_{\mathbf{U}} : t > threshold(\mathbf{x}, \varepsilon, \lambda) \right\}, \; P_{RT}\left(\mathbf{B}\left(RT, \mathbf{x}; \varepsilon, \lambda\right)\right) \leq \varepsilon. \tag{12}$$

For a sample of programs $\mathbf{x}$ we use the notation $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}))$, where $t_i = RT_i(\mathbf{x})$, $1 \leq i \leq N$. We increasingly order the observed running times t_i and get the values of the corresponding order statistics $t_{(1)}(\mathbf{x}) \leq \cdots \leq t_{(N)}(\mathbf{x})$. As one of these order statistics will be the choice for the threshold, $threshold(\mathbf{x}, \varepsilon, \lambda)$, we must find the smallest number $1 \leq K \leq N$ such that $\mathbf{x} \in \mathcal{G}_{N,\lambda,t_{(K)}(\mathbf{x})}$. In terms of order statistics, $t_{(K)}(\mathbf{x})$ generates a statistical *threshold* $(\mathbf{x}, \varepsilon, \lambda)$ which must satisfy (12). Explicitly these two requirements are:

$$ECDF_{RT,N}\left(\left(t_1(\mathbf{x}), \ldots, t_N(\mathbf{x})\right); t_{(K)}(\mathbf{x})\right) - \lambda \leq CDF_{RT}\left(t_{(K)}(\mathbf{x})\right), \tag{13}$$

$$P_{RT}\left(\left\{ t \in \mathbf{T}_{\mathbf{U}} : t > t_{(K)}(\mathbf{x}) \right\}\right) \leq \varepsilon. \tag{14}$$

As from (7),

$$ECDF_{RT,N}\left(\left(t_1(\mathbf{x}), \ldots, t_N(\mathbf{x})\right); t_{(K)}(\mathbf{x})\right) = \frac{K}{N},$$

both conditions are satisfied if

$$1 - \varepsilon \leq \frac{K}{N} - \lambda \leq CDF_{RT}\left(t_{(K)}(\mathbf{x})\right). \tag{15}$$

Indeed, from the definition of CDF_{RT}, if $\mathbf{x} \in \mathcal{G}_{N,\lambda,t_{(K)}}$, then

$$\frac{K}{N} - \lambda \leq CDF_{RT}\left(t_{(K)}(\mathbf{x})\right),$$

so (13) is satisfied. Furthermore, as

$$P_{RT}\left(\left\{ t \in \mathbf{T}_{\mathbf{U}} : t > t_{(K)}(\mathbf{x}) \right\}\right) = 1 - CDF_{RT}\left(t_{(K)}(\mathbf{x})\right),$$

if $\mathbf{x} \in \mathcal{G}_{N,\lambda,t_{(K)}(\mathbf{x})}$ and $1 - \varepsilon \leq \frac{K}{N} - \lambda$, then (14) is satisfied.

From the first inequality in (15) we get $K \geq N(1 - \varepsilon + \lambda)$. As we must have $0 < 1 - \varepsilon + \lambda < 1$, we get $\lambda < \varepsilon$. For $N = N(\lambda, \delta)$ as in (10) we can take $K = K(\varepsilon, \lambda, \delta) = \lceil N(1 - \varepsilon + \lambda) \rceil$ – the minimum integer $1 \leq K \leq N$ satisfying (15) – hence

$$threshold(\mathbf{x}, \varepsilon, \lambda) = t_{K(\varepsilon,\lambda,\delta)}(\mathbf{x}) = t_{\lceil N(1-\varepsilon+\lambda)\rceil}(\mathbf{x}). \tag{16}$$

From (15) we have

$$1 - \varepsilon \leq CDF_{RT}\left(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})\right). \tag{17}$$

The statistical anytime algorithm for the Halting Problem will operate with three parameters: a) a bound $\varepsilon \in (0,1)$ for the decision error, b) a *precision* parameter $1 < \lambda < \varepsilon$ which is a bound on the approximation of CDF_{RT} with $ECDF_{RT,N}$, and c) a *confidence* parameter $1 - \delta \in (0,1)$ which is a probabilistic bound on the confidence in the precision parameter. In detail, the approximation parameter and confidence level control the quality of the approximation of CDF_{RT} by $ECDF_{RT}$: a) the precision parameter λ is the numerical difference between the values of the two functions in a given point, b) the confidence level is the probability that the N sampled programs produce an approximation of CDF_{RT} with a requested precision, c) the decision error is the probability that the decision '$\mathbf{U}(z) = \infty$' is returned when, in reality, $\mathbf{U}(z) < \infty$.

We sample N independent halting programs $x_1, \ldots, x_N \in \mathrm{dom}(\mathbf{U})$ (see [3, 18]) and, by running them till they stop, calculate their respective running times $t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}) \in \mathbf{T}_\mathbf{U}$. Let $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}))$. Randomisation is done according to the probability distribution induced by an injective computable enumeration of the halting programs.

The statistical anytime algorithm is:

Pre-processing.
Fix three rational numbers $\varepsilon, \lambda, \delta \in (0,1)$ *with* $\lambda < \varepsilon$.
Compute $N = N(\lambda, \delta) = \lceil \frac{1}{2\lambda^2} \cdot \ln \frac{1}{\delta} \rceil$.
Sample N *independent programs* $\mathbf{x} = (x_1, \ldots, x_N) \in \mathrm{dom}(\mathbf{U})^N$ *and calculate their respective running times* $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}))$.
Increasingly order the observed running times t_i *and compute the threshold*
$\mathbf{T} = t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})$.
Main part.
Let z *be an arbitrary program for* $\mathbf{U}$.
If the computation $\mathbf{U}(z)$ *does not stop in time less than or equal to* $\mathbf{T}$*, then declare that* $\mathbf{U}(z) = \infty$.

We now evaluate the error the statistical anytime algorithm can make by deciding that $\mathbf{U}(z)$ does not stop when in fact it stops. To this aim we use the threshold $t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}$ and the critical regions

$$\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda) = \{t \in \mathbf{T}_\mathbf{U} : t > t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})\},$$

$$\mathbf{C}(RT, \mathbf{x}; \varepsilon, \lambda) = \{y \in \mathrm{dom}(\mathbf{U}) : RT(y) > t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})\}.$$

Lemma 5.1. *For every* $\mathbf{x} \in \mathrm{dom}(\mathbf{U})^N$, $\varepsilon, \lambda \in (0,1)$ *with* $\lambda < \varepsilon$, *we have:*

$$\Pr(\mathbf{C}(RT, \mathbf{x}; \varepsilon, \lambda)) = P_{RT}(\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda)). \tag{18}$$

Proof. We have:

$$\Pr(\mathbf{C}(RT, \mathbf{x}; \varepsilon, \lambda)) = P_{RT}\left(\left\{t \in \mathbf{T}_\mathbf{U} : t > t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})\right\}\right) = P_{RT}(\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda)).$$

□

Lemma 5.2. *For every integer $N > 0$, $\varepsilon, \lambda \in (0,1)$ with $\lambda < \varepsilon$, we have:*

$$\begin{aligned}&\Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : P_{RT}(\mathbf{B}\,(RT, \mathbf{x}; \varepsilon, \lambda)) \leq \varepsilon\right\}\right)\\ &\quad \geq \Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) \geq 1-\varepsilon\right\}\right).\end{aligned} \tag{19}$$

Proof. We only need to prove the implication:

$$CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) \geq 1-\varepsilon \Longrightarrow P_{RT}(\mathbf{B}\,(RT, \mathbf{x}; \varepsilon, \lambda)) \leq \varepsilon.$$

If $T_1 = \{k \in \mathbf{T_U} : 1 \leq k \leq t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})\}$ and $T_2 = \{j \in \mathbf{T_U} : j > t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})\}$, then $T_1 \cap T_2 = \emptyset$, $CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) = P_{RT}(T_1)$ and $P_{RT}(\mathbf{B}\,(RT, \mathbf{x}; \varepsilon, \lambda)) = P_{RT}(T_2)$.

Consequently, if $P_{RT}(T_1) \geq 1-\varepsilon$, then

$$1 - \varepsilon + P_{RT}(T_2) \leq P_{RT}(T_1) + P_{RT}(T_2) \leq P_{RT}(T_1 \cup T_2) \leq 1,$$

so $P_{RT}(\mathbf{B}\,(RT, \mathbf{x}; \varepsilon, \lambda)) = P_{RT}(T_2) \leq \varepsilon$. □

Theorem 5.3. *For every $\varepsilon, \lambda, \delta \in (0,1)$ with $\lambda < \varepsilon$ and $N = N(\lambda, \delta)$ we have:*

$$\Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : \Pr(\mathbf{C}\,(RT, \mathbf{x}; \varepsilon, \lambda)) \leq \varepsilon\right\}\right) \geq 1-\delta. \tag{20}$$

Proof. From the definition (7) and the choice of the statistical threshold (16) we have

$$ECDF_{RT,N}\left(\left(RT_1(\mathbf{x}), \ldots, RT_N(\mathbf{x})\right); t\right) = \frac{\lceil N(1-\varepsilon+\lambda)\rceil}{N}.$$

In view of (17), (18) and (19) and (11) we have

$$\begin{aligned}&\Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : \Pr(\mathbf{C}\,(RT, \mathbf{x}; \varepsilon, \lambda)) \leq \varepsilon\right\}\right)\\ &= \Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : P_{RT}(\mathbf{B}\,(RT, \mathbf{x}; \varepsilon, \lambda)) \leq \varepsilon\right\}\right)\\ &\geq \Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) \geq 1-\varepsilon\right\}\right)\\ &\geq \Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N : CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) \geq \frac{\lceil N(1-\varepsilon+\lambda)\rceil}{N}\right\} - \lambda\right)\\ &\geq 1-\delta.\end{aligned}$$

□

According to (20), the probability $\Pr^N$ of the event that the statistical anytime algorithm gives a wrong decision, that is, it declares $\mathbf{U}(z) = \infty$ when there exists $t > t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})$ such that $\mathbf{U}(z)$ stops in time t, is smaller or equal than ε, is larger than $1-\delta$, i.e. the probability of error is smaller than or equal than ε with confidence larger than $1-\delta$.

δ	ε	$\lambda\,(<\varepsilon)$	$N(\lambda,\delta)$	$\lceil N(\lambda,\delta)\cdot(1-\varepsilon+\lambda)\rceil$
$\frac{1}{100}$	$\frac{5}{1000}$	$\frac{1}{1000}$	2.3026×10^6	2.2934×10^6
$\frac{1}{100}$	$\frac{1}{1000}$	$\frac{5}{10000}$	9.2103×10^6	9.2057×10^6
$\frac{5}{1000}$	$\frac{5}{1000}$	$\frac{1}{1000}$	2.6492×10^6	2.6386×10^6
$\frac{5}{1000}$	$\frac{1}{1000}$	$\frac{5}{10000}$	1.0597×10^7	1.0592×10^7
$\frac{1}{1000}$	$\frac{5}{1000}$	$\frac{1}{1000}$	3.4539×10^6	3.4401×10^6
$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{5}{10000}$	1.3816×10^7	1.3809×10^7

Table 1. Numerical illustration of the first implementation of the anytime algorithm

6. Implementations of the statistical anytime algorithm

In this section we present three implementations of the anytime algorithm; numerical illustrations show that their time complexities are reasonably small.

The standard implementation of the statistical anytime algorithm is as follows. Given three rational numbers $\varepsilon,\lambda,\delta\in(0,1)$ with $\lambda<\varepsilon$, first compute the sample size from (10), $N=\lceil\frac{1}{2\lambda^2}\cdot\ln\frac{1}{\delta}\rceil$; this positive integer is fixed as long as λ,δ are fixed. Then use an algorithm to generate a random injective computable enumeration of $\mathrm{dom}(\mathbf{U})$ till N programs $x_1,\ldots,x_N$ and their running times $t_1,\ldots,t_N$ are obtained; again, these programs are fixed with $\varepsilon,\lambda,\delta$. Then, for every program $z\in\mathbb{Z}^+$, if the computation $\mathbf{U}(z)$ does not stop in time $t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})$, then declare that $\mathbf{U}(z)=\infty$. In the latter case the probability of error is smaller than or equal to ε with confidence larger than $1-\delta$.

In Table 1 we illustrate numerically $N(\lambda,\delta)$ and $\lceil N(\lambda,\delta)\cdot(1-\varepsilon+\lambda)\rceil$ for the first implementation with fixed parameters $\varepsilon,\lambda,\delta$ having a few statistically standard values.

In a second implementation we start with two rational numbers $\varepsilon,\lambda\in(0,1)$ with $\lambda<\varepsilon$ and an "affordable" size $\tilde{N}$ of samples (programs and running times), then compute the rational $\delta(\tilde{N},\lambda)\in(0,1)$ satisfying the inequality (9). We continue with the standard implementation of the statistical anytime algorithm with parameters ε,λ ($\lambda<\varepsilon$) and

$$\delta(\tilde{N},\lambda)=\exp(-2\tilde{N}\cdot\lambda^2) \tag{21}$$

to calculate the size sample $N(\lambda,\delta(\tilde{N},\lambda))=\tilde{N}$. However, the value $\delta(\tilde{N},\lambda)$ in (21) is not rational, so to preserve the inequality (9) we need to calculate a rational approximation

$$\delta(\tilde{N},\lambda)\geq\exp(-2\tilde{N}\cdot\lambda^2),$$

ε	$\lambda\,(<\varepsilon)$	$\widetilde{N}$	$\delta\left(\widetilde{N},\lambda\right)$	$\left\lceil \widetilde{N}\,(1-\varepsilon+\lambda)\right\rceil$
$\frac{1}{100}$	$\frac{5}{1000}$	10^5	6.7379×10^{-3} (very good)	9.95×10^4
$\frac{1}{100}$	$\frac{4}{1000}$	10^5	4.0762×10^{-2} (good)	9.94×10^4
$\frac{1}{100}$	$\frac{5}{1000}$	$2\cdot 10^5$	4.54×10^{-5} (excellent)	1.99×10^5
$\frac{1}{100}$	$\frac{4}{1000}$	$2\cdot 10^5$	1.6616×10^{-3} (very good)	1.988×10^5
$\frac{1}{100}$	$\frac{1}{1000}$	10^6	1.3534×10^{-1} (hardly acceptable)	9.91×10^5
$\frac{1}{1000}$	$\frac{5}{10000}$	10^6	6.0653×10^{-1} (unacceptable)	9.995×10^5

Table 2. Numerical illustration of the second implementation of the anytime algorithm

which implies the inequality $N(\lambda,\delta(\tilde{N},\lambda))<\tilde{N}+1$.

The "price" paid working with an affordable, smaller sample size $\tilde{N}$ is a (possibly sharp) decrease in the confidence level; below is a numerical illustration of the second implementation with fixed parameters $\varepsilon,\lambda,\tilde{N}$. Table 2 illustrates the second implementation.

In a third implementation we start with two rational numbers $\varepsilon,\lambda\in(0,1)$ with $\lambda<\varepsilon$ and an "affordable" upper bound T on the running time of the computation $\mathbf{U}(z)$ in the statistical anytime algorithm. We then use an injective dovetailing algorithm to generate as many elements of $\mathrm{dom}(\mathbf{U})$ as possible in time T. In this way we will obtain a sample of $N(T)$ programs $x_1,\ldots,x_{N(T)}$ and their respective running times $t_1,\ldots,t_{N(T)}$ such that each program stops in time at least (in fact, much smaller than) T: $t_i\leq T$, for all $1\leq i\leq N(T)$. We then continue with the second implementation with parameters $\varepsilon,\lambda,\tilde{N}=N(T)$.

Both second and third implementations can be improved by increasing the sample size or the time bound, respectively.

Approximations in algorithmic information theory [4], for example, the approximations of Solomonoff universal distribution[1] [19], involve constants; in contrast, all implementations of the statistical anytime algorithm are free from such uncertainty.

7. Final comments

The anytime probabilistic algorithm for the Halting Problem proposed in [6] uses essentially a computable probability distribution on the set of stopping times of halting programs which reflects the halting behaviour of the chosen universal machine. The quantile of this probability distribution is used to compute the stopping threshold time. The probability of a wrong decision is no larger than the accepted error.

[1] Solomonoff universal distribution [19] is an priori incomputable probability distribution over the set of finite binary strings which is different from the computable probability distribution on the set of stopping times discussed in Section 3.

The statistical anytime algorithm for the Halting Problem – which is inspired by the probabilistic one – does not make any assumption on the probability distribution on the halting programs and uses an order statistics to compute the stopping threshold time (the cut-off temporal bound). In a nutshell, this anytime algorithm works on an arbitrary program as follows :

1. "Sample" sufficiently many halting programs independently at random.
2. Determine their running times and consider the induced empirical distribution as approximation to the true but unknown distribution.
3. Simulate the given program for the largest number of steps made by any of the sampled programs: if it still has not terminated by then, report (possibly wrongly) 'The program does not halt!'.

In detail, the statistical algorithm uses three parameters for evaluating the quality of solutions, namely the probability of an erroneous decision ε, the precision λ and the confidence level δ of the statistical approximation. The sample size and critical regions are constructed based on these parameters. The main advantage of the statistical algorithm is that it can be implemented without any prior information about the distribution of running times. Another advantage is that the threshold (cut-off) temporal bound $\lceil N(\lambda, \delta) \cdot (1 - \varepsilon + \lambda) \rceil$ is calculated only once (when $\varepsilon, \lambda, \delta \in (0, 1)$ with $\lambda < \varepsilon$ are fixed) and then used for running the algorithm on any input. We proved that with a confidence level as large as required, the algorithm produces correct decisions with a probability as large as required.

The main advantage of the anytime statistical algorithm is that it can be implemented without any prior information about the running times on the specific model of computation – the choice of a computable probability distribution for the anytime probabilistic algorithm can be rather subjective and hard to substantiate; also, the cut-off temporal bound is reasonably small.

Finally, three implementations of the algorithm have been presented and numerically illustrated. Recent experimental work with Turing machines in [16] shows that "the halting probability of a TM decreases with time and will have a smaller chance of halting at every step it progresses", reflecting the behaviour discovered in [9] which is at the core of both anytime probabilistic and statistic algorithms for the Halting Problem. It will be interesting to experiment these algorithms, particularly the statistical one, with different models of computations in order to understand their practical utility.

Acknowledgments

We thank Dr. Ned Allen for discussions and comments and for motivating one author (CC) to study practical approximate solutions to the Halting Problem. We thank the anonymous referees for useful comments that improved the presentation; we distinctly thank the referee who made many excellent proposals including the description of the statistical anytime algorithm presented in Section 7. This work was supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

References

[1] B. 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*, LNCS 9134, Springer, 2015, 219–230.

[3] K. Bringmann, K. Panagiotou. Efficient sampling methods for discrete distributions *Algorithmica* (2016), 1–25.

[4] C. Calude. *Theories of Computational Complexity*, North Holland, Amsterdam, 1988.

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

[6] C. S. Calude, M. Dumitrescu. A probabilistic anytime algorithm for the Halting Problem, *Computability*, 7 (2018), 259–271.

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

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

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

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

[11] B. Cook, A. Podelski and A. Rybalchenko. Proving program termination, *Communications ACM* 54, 5 (2011), 88–98.

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

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

[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] S. Köhler, C. Schindelhauer, and M. Ziegler. On approximating real-world halting problems, in M. Liskiewicz and R. Reischuk (eds.). *Fundamentals of Computation Theory 2005*, LNCS 3623, Springer, 2005, 454–466.

[16] K. Krzyzanska. Exploring halting times for unconventional halting schemes, *Complex Systems* 27 (1) (2018), 85–99.

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

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

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

[20] N. Lynch. Approximations to the Halting Problem, *Journal of Computer and System Sciences* 9 (1974), 143-150.

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

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

[23] T. Mori, Y. Tsujii, and M. Yasugi. Computability of probability distributions and distribution functions, in A. Bauer, P. Hertling, Ker-I Ko (eds.). *6th International Conference on Computability and Complexity in Analysis*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2009, Dagstuhl, 185–196.

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

[25] A. Rybalov. On the generic undecidability of the halting problem for normalized Turing machines, *Theory of Computing Systems*, (2016), 1–6.

[26] C. Scott. Statistical Learning Theory, Topic 3: Hoeffding's Inequality, University of Toronto, 2014, https://www.coursehero.com/file/18068309/03-hoeffding, retrieved 4 June 2019.

[27] F. Soler-Toscano, H. Zenil, J.-P. Delahaye, and N. Gauvrit. Calculating Kolmogorov complexity from the output frequency distributions of small Turing machines, *PLoS ONE*, 9, 5 (2014):e96223.

[28] H. Zenil and J-P. Delahaye. On the algorithmic nature of the world, in G. Dodig-Crnkovic and M. Burgin (eds.). *Information and Computation. Essays on Scientific and Philosophical Understanding of Foundations of Information and Computation*, World Scientific, Singapore, 2010, 477–499.

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

[30] K. Weihrauch. *Computable Analysis. An Introduction*, Springer, Berlin, 2000.