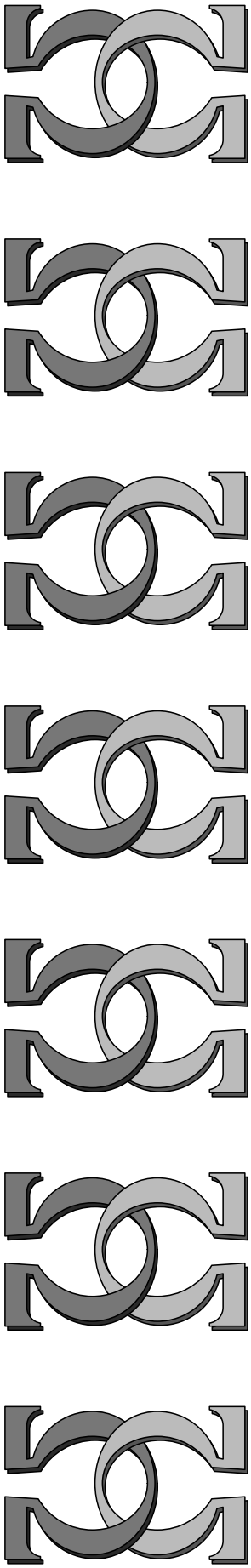

CDMTCS
Research
Report
Series

A Statistical Anytime Algorithm for the Halting Problem

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

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

CDMTCS-529
November 2018

Centre for Discrete Mathematics and
Theoretical Computer Science

A Statistical Anytime Algorithm for the Halting Problem

Cristian S. Calude* Monica Dumitrescu†

November 13, 2018

Abstract

As the Halting Problem, the most (in)famous undecidable problem, has important applications in theoretical and applied computer science and beyond, there is a growing interest in its approximate solutions.

Running times play an important role in the study of this problem because halting programs are not uniformly distributed, a fact experimentally confirmed on various models of computation, and a program, which eventually halts but does not halt "quickly", stops at a time which is algorithmically compressible.

In a previous paper we used 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 cut-off temporal bound of this algorithm can be very large.

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 reasonable 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 us run for any input. 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. The algorithm can be naturally programmed as a faster hybrid classical-quantum algorithm for classes of instances of the Halting Problem.

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 (in Turing's formulation) an *algorithm to solve the Halting Problem for all possible program-input pairs does not exist.* 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 [18, 16, 14, 8, 11, 20, 7, 6, 3, 24, 5].

Anytime algorithms trade execution time for quality of results [13]. These algorithms can be executed in two modes: either by a given execution contract time 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

*Department of Computer Science, University of Auckland, Auckland, New Zealand.
†Faculty of Mathematics and Computer Science, Bucharest University, Romania.

after they have halted. This procedure is similar to iterative processes in numerical computing in which, after the process has been halted, if the output is not considered acceptable, then it can be refined by resuming the iterative process.

Following Manin [20] 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 probability of error, i.e. the probability that the program may eventually stop. The goal is to prove that the *probability of error* can be made as small as we wish. By running the program a longer time we can improve its performance either by reaching the halting time or by decreasing the probability of error of declaring that the program will never stop when, in fact, it eventually stops.

Running times play an important role in the study of this problem because halting programs are not uniformly distributed, a fact experimentally confirmed on various models of computation [27, 28, 26] and by theoretical results in [15, 14]. In [8, 7] anytime algorithms for the Halting Problem have been developed using the fact – proved in [8] – 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 in general [7].

A better anytime algorithm for the Halting Problem has been developed in [5] using a class of computable probability distributions on the set stopping times of halting programs: each computable probability distribution induces a running time probability space on the set of halting programs. The algorithm has good features, the cut-off temporal bound does not depend on programs, the *a priori* class of computable probabilities distributions are not arbitrarily pre-imposed as they reflect the halting behaviour of the chosen universal machine (model of computation) and one can test empirically the choice of the computable probability distribution and sampling, hence adopt parameters suitable for different universal machines. However, the probability distribution on the set of halting programs has to be known and the cut-off temporal bound could be very large in contrast with some experimental results in [28] for small Turing machines that seem to indicate much smaller stopping times.

In this paper we propose a statistical anytime algorithm for the Halting Problem in case we don't know the probability distribution on the set of halting programs. The statistical algorithm – which is inspired by the probabilistic one – uses three parameters, namely the probability of an erroneous decision, the precision of and the confidence level in the decision.

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 reasonable small. The pre-processing of the algorithm is done only once (when the parameters of the quality of solutions are fixed) and then used for running the algorithm on any input. 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. The algorithm can be naturally programmed as a faster hybrid classical-quantum algorithm for classes of instances of the Halting Problem.

The paper is organised as follows. We start with a section on basic notation. Section 3 presents the computability and complexity part while Section 4 reviews the main notions and results from probability and statistics needed in the paper, Section 5 presents the probability framework, Section 6 briefly presents the probabilistic anytime algorithm for the Halting Problem, Section 7 presents the statistical framework, then Section 8 presents the statistical anytime algorithm for the Halting Problem and the proof of its main properties. Finally, in Section 9 we discuss three possible implementations of the statistical algorithm and present numerically illustrations; the last section is devoted to conclusions

and possible extensions.

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. 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 $\mathrm{dom}\,(F)$: $\mathrm{dom}\,(F) = \{x \in \mathbb{Z}^+ \mid F\,(x) < \infty\}$. 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 [19, 4, 12].

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 *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

Informally, the algorithmic complexity of string x respect to the interpreter/decoder F is the smallest input y for F which produces x or ∞ if F cannot produce x. Formally, 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 $\nabla_F\,(x) = \inf\{y \in \mathbb{Z}^+ \mid F\,(y) = x\}$. If $F\,(y) \neq x$ for every $y \geq 1$, then $\nabla_F\,(x) = \infty$.

A partially computable function $\mathbf{U}$ is 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

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

or equivalently [7], 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

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

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

4 Probability and statistics

In this section we define the main notions from probability theory and statistics used in this paper. For more details see [9, 23].

A *measurable space* $(\Omega, \mathcal{B}\,(\Omega))$ consists of a non-empty set Ω and a Borel field of subsets of Ω, $\mathcal{B}\,(\Omega) \subseteq \mathcal{P}\,(\Omega)$. 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, that is, Pr 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".

Consider a probability space $(\Omega, \mathcal{B}(\Omega), \Pr)$ and a measurable space $(A, \mathcal{B}(A))$. A *random variable* is a measurable function $X\colon \Omega \longrightarrow A$, that is, for every $B \in \mathcal{B}(A)$ we have $X^{-1}(B) \in \mathcal{B}(\Omega)$. In this case X induces a probability (called *probability distribution of* X) $P_X : \mathcal{B}(A) \longrightarrow [0,1]$ defined by

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

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

The random variable X has a *discrete probability distribution* if A is at most countable. If we denote by $P_X(\{x\})$ the probability of the event $\{X = x\} = \{\omega \in \Omega \mid X(\omega) = x\}$, then the discrete probability distribution of X is completely defined by the numbers $P_X(\{x\}) \in [0,1],\ x \in A$, with $\sum\limits_{x\,\in\,A} P_X(\{x\}) = 1$. *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 $x \in A$ [21, p. 159]; see also [29, 22]).

In what follows we assume that $A \subseteq \mathbb{R}$.

The mean (or *expected value*) of the random variable $X : \Omega \longrightarrow A$ is defined by

$$E(X) = \sum_{x \in A} x \cdot \Pr(\{\omega \in \Omega \mid X(\omega) = x\}),$$

if the series converges. For every event $M \in \mathcal{B}(\Omega)$, $E(\mathbf{1}_M) = 1 \cdot \Pr(M) + 0 \cdot \Pr\left(\overline{M}\right) = \Pr(M)$. If for some $c \in \mathbb{R}$, $X(\omega) = c$ for all ω, then $E(X) = c$ and E is a linear operator, that is for every random variables X, Y and $a \in \mathbb{R}$, $E(X+Y) = E(X) + E(Y)$ and $E(aX) = a \cdot E(X)$.

The following form of Hoeffting's inequality (see [25]) will be used in what follows:

Theorem 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:*

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

The *Cumulative Distribution Function* of a random variable X is the function $CDF_X : \mathbb{R} \longrightarrow [0,1]$ defined by $CDF_X(y) = \Pr(X \leq y), y \in \mathbb{R}$. In case X is a random variable with a discrete distribution, CDF_X is the stair-function (with piecewise-constant sections) given by

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

For example, the generally accepted model for "the time-to-the-first-success" is the *geometric distribution* – the discrete probability distribution that expresses the probability that the first occurrence of an event ("success") requires k independent trials, each with the same success probability θ. More precisely, the random variable X that takes values in $A = \mathbb{Z}^+$ has a geometric distribution with the rate of success $\theta \in (0,1)$ if $P_X(k) = (1-\theta)^{k-1} \cdot \theta,\ k \geq 1$. In this case

$$CDF_X(y) = \sum_{k=1,\,k\leq y}^{\infty} (1-\theta)^{k-1} \cdot \theta,\ y \in \mathbb{R},$$

and $\lim_{y\to\infty} CDF_X(y) = 1$.

The *Quantile Function* of the random variable X with a discrete distribution is the function $\mathbf{q}_X : [0,1] \longrightarrow A$ defined by $\mathbf{q}_X(p) = \inf\{y \in A \mid p \leq CDF_X(y)\}$. By definition, $CDF_X(\mathbf{q}_X(p)) \geq p$, for all $p \in [0,1]$.

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 underlying 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 information about the "tails" of the probability distribution (for small $\varepsilon > 0$). For more details see [2].

The following result suggested the probabilistic anytime algorithm for the Halting Problem [5]:

Proposition 2. *For every* $\varepsilon \in (0,1)$ *we have* $P_X(\{x \in A \mid x > \mathbf{q}_X(1-\varepsilon)\}) \leq \varepsilon$.

Proof. Indeed, we have: $P_X(\{x \in A \mid x > \mathbf{q}_X(1-\varepsilon)\}) = 1 - P_X(\{x \in A \mid x \leq \mathbf{q}_X(1-\varepsilon)\}) = 1 - CDF_X(\mathbf{q}_X(1-\varepsilon)) \leq \varepsilon$. □

The notions and results discussed above are theoretical and are based on the assumption that the probability Pr and the random variable X are known. In case this assumption is not satisfied, can an *inferential approach* be used to 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 $X_1, \ldots, X_N$ of independent, identically distributed random variables with the same distribution as X? The answer is affirmative.

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

$$ECDF_{X,N}(y) = \frac{\#\{1 \leq i \leq N \mid x_i \leq y\}}{N},\ y \in \mathbb{R}. \tag{2}$$

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

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N-1)} \leq x_{(N)}. \tag{3}$$

The *order statistics of rank* k is the kth smallest value in (3): $X_{(k)}(\omega) = x_{(k)}$. See more in [9, Ch. 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*.

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 $x \in A$. For every $x \in A$, a *hypothesis* H_x is a statement such that "H_x is true" and "H_x is false" are measurable sets from $\mathcal{B}(A)$.

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 *probability of error*, that is, the probability of an erroneous decision, is $P_X(\{x \in B \mid$ "H_x is true"$\})$. Of course, only decisions with very low probability of error are of interest.

5 Probability framework

Recall that the *finite running times* of the computations $\mathbf{U}(x)$ are the set of exact stopping times for the halting programs of $\mathbf{U}$: $\mathbf{T_U} = \{t \in \mathbb{Z}^+ \mid \text{ there exists } x \in \mathbb{Z}^+ \text{ such that } x \in \text{Stop}(\mathbf{U}, t)\} = \{t \in \mathbb{Z}^+ \mid \text{there exists } x \in \mathbb{Z}^+ \text{ such that } \mathbf{U}(x)[t] < \infty\}$.

Let us consider the family of (finite and countable) unions of sets $\text{Stop}(\mathbf{U}, t), t \in \mathbb{Z}^+$. This family includes $\text{dom}(\mathbf{U})$ and is closed under complement $\overline{\text{Stop}(\mathbf{U}, t)} = \bigcup_{t' \in \mathbf{T_U} \setminus \{t\}} \text{Stop}(\mathbf{U}, t')$, and countable unions; accordingly, it is a Borel field of subsets of $\text{dom}(\mathbf{U})$, which we denote by $\mathcal{B}(\text{dom}(\mathbf{U}))$. To define the discrete probability measure on the measurable set $(\text{dom}(\mathbf{U}), \mathcal{B}(\text{dom}(\mathbf{U})))$ we fix a computable probability distribution ρ on $\mathbf{T_U}$ and put $\Pr = \Pr_\rho : \mathcal{B}(\text{dom}(\mathbf{U})) \longrightarrow [0, 1]$, $\Pr(\text{Stop}(\mathbf{U}, t)) = \rho(t)$, $t \in \mathbf{T_U}$.

Next we introduce a probability structure on the set $\mathbf{T_U}$ via a random variable. Let $\mathcal{B}(\mathbf{T_U})$ be the family of all subsets of $\mathbf{T_U}$. The function $RT = RT_{\mathbf{U}} : \text{dom}(\mathbf{U}) \longrightarrow \mathbf{T_U}$, $RT(x) = \min\{t > 0 \mid x \in \text{Stop}(\mathbf{U}, t)\}$ has the property that for every $t \in \mathbf{T_U}$, $RT^{-1}(\{t\}) = \text{Stop}(\mathbf{U}, t) \in \mathcal{B}(\text{dom}(\mathbf{U}))$. Consequently, RT is a *random variable* – in fact a stopping time – which will be called the *running time associated with* $\mathbf{U}$. As described in Section 4, the random variable RT induces the probability space $(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_{RT})$ on $\mathbf{T_U}$ in which the probability is defined by $P_{RT}(\{t\}) = \Pr(RT^{-1}(\{t\}))$, $t \in \mathbf{T_U}$. For every $t \in \mathbf{T_U}$ we have: $P_{RT}(\{t\}) = \Pr(\text{Stop}(\mathbf{U}, t)) = \rho(t)$.

A computable probability space $(\text{dom}(\mathbf{U}), \mathcal{B}(\text{dom}(\mathbf{U})), \Pr_{\rho_\mathbf{U}})$ of the form defined above will be called a *running time probability space*. The random variable RT is completely specified by a *computable probability distribution on the set of finite running times of programs of* $\mathbf{U}$, $\{\rho(t) \mid t \in \mathbf{T_U}\}$. Examples of computable probability distributions are in [5].

6 The probabilistic anytime algorithm for the Halting Problem

In this section we briefly present the probabilistic anytime algorithm for the Halting Problem from [5]. 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 $\mathbf{U}(x) = \infty$.

The probabilistic anytime algorithm tests for an arbitrary $z \in Z^+$, the hypothesis $H_z : \{\mathbf{U}(z) < \infty\}$ against the alternative $H'_z : \{\mathbf{U}(z) = \infty\}$. The decision of rejecting H_z will be taken on the basis of a *critical time region* B_z. An erroneous decision occurs when we reject H_z on the basis of B_z, but H_z is true. The quality of this decision is expressed by the probability of an erroneous decision, i.e. the probability that a halting program z stops at a time $t \in B_z$. The algorithm works with an a priori running time probability space $(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_{RT})$ and the running time random variable RT defined in Section 5.

The probabilistic anytime algorithm is:

> *Choose a rational* $\varepsilon \in (0, 1)$. *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* $\mathbf{q}_{RT}(1 - \varepsilon)$, *then declare that* $\mathbf{U}(z) = \infty$.

If the computation $\mathbf{U}(z)$ stops in time less than or equal to $\mathbf{q}_{RT}(1 - \varepsilon)$, then obviously $\mathbf{U}(z) < \infty$. Otherwise, the answer to the question whether $\mathbf{U}(z) < \infty$ is *unknown* and *algorithmically unknowable*. The probabilistic anytime algorithm gives an approximate answer. We choose the *computable* critical time region[1]

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

[1] $\mathbf{B}(P_{RT}, \varepsilon)$ is independent of z.

and the *critical program region*

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

$$=\operatorname{dom}(\mathbf{U})\setminus\{z\in\mathbb{Z}^{+}\mid \mathbf{U}(z)[t]=\infty,\ \text{for some } t\leq \mathbf{q}_{RT}(1-\varepsilon)\}.$$

The set $\mathbf{C}(P_{RT},\varepsilon)$ contains all halting programs that stop in a time bigger than $\mathbf{q}_{RT}(1-\varepsilon)$, i.e. the programs which the probabilistic anytime algorithm wrongly identifies as non-halting. To evaluate the quality of the anytime algorithm we show that the probability according to Pr of a program to be in $\mathbf{C}(P_{RT},\varepsilon)$ is smaller than ε, so it can be made as small as required.

Proposition 3. *For every* $\varepsilon\in(0,1)$, $\Pr(\mathbf{C}(P_{RT},\varepsilon))=P_{RT}(\mathbf{B}(P_{RT},\varepsilon))\leq\varepsilon$.

To implement the anytime algorithm above we need an algorithm to compute $\mathbf{q}_{RT}(1-\varepsilon)=\min\{t\in\mathbf{T}_{\mathbf{U}}\mid CDF_{RT}(t)\geq 1-\varepsilon\}$. As the set $\mathbf{T}_{\mathbf{U}}$ is computably enumerable but not computable, we cannot compute $\mathbf{q}_{RT}(1-\varepsilon)$, but an upper bound for it obtained from a computable enumeration of $\mathbf{T}_{\mathbf{U}}$.

7 Statistical framework

The statistical anytime algorithm assumes that the *probability distribution of RT is unknown.* Therefore, the cumulative distribution function of RT, $CDF_{RT}(t)=\Pr(\{x\in\operatorname{dom}(\mathbf{U})\mid RT(x)\leq t\})$, is also unknown and has to be estimated.

To this aim we fix a positive integer N and consider the $N-$dimensional *program sampling space*:

$$\left(\operatorname{dom}(\mathbf{U})^{N},\mathcal{B}\left(\operatorname{dom}(\mathbf{U})^{N}\right),\Pr^{N}\right).$$

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

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

For every $\mathbf{x}\in\operatorname{dom}(\mathbf{U})^{N}$ 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 Theorem 4, $CDF_{RT}(t)$ is estimated by the Empirical Cumulative Distribution Function (2)

$$\begin{aligned}ECDF_{RT,N}((RT_1(\mathbf{x}),\ldots,RT_N(\mathbf{x}));t)&=\frac{\#\{1\leq i\leq N\mid RT_i(\mathbf{x})\leq t\}}{N}\\&=\frac{\#\{1\leq i\leq N\mid t_i(\mathbf{x})\leq t\}}{N}.\end{aligned}\tag{4}$$

Theorem 4. *For every positive integer N, $t \in \boldsymbol{T}_{\mathbf{U}}$ and $\lambda \in (0,1)$, we have:*

$$\Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\,(\mathbf{U})^N \mid ECDF_{RT,N}\left(\left(RT_1(\mathbf{x}),\ldots,RT_N(\mathbf{x})\right);t\right) - CDF_{RT}(t) \leq \lambda\right\}\right) \tag{5}$$
$$\geq 1 - \exp\left(-2N\cdot\lambda^2\right).$$

Proof. On one hand, from (4) we have:

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

On the other hand, using the properties of the operator E (see Section 4) we have:

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

As $RT_i\colon \mathrm{dom}\,(\mathbf{U})^N \longrightarrow \mathbf{T}_{\mathbf{U}}$, $RT_i(\mathbf{x}) = RT(x_i)$, $i = 1,\ldots,N$ are independent random variables, for every $t \in \mathbf{T}_{\mathbf{U}}$, $\mathbf{1}_{\{\mathbf{x}\in\mathrm{dom}(\mathbf{U})^N \mid RT_i(\mathbf{x})\ \leq\ t\}}\colon \mathrm{dom}(\mathbf{U})^N \longrightarrow [0,1]$, $i = 1,\ldots,N$ are also independent random variables. Consequently, the inequality (5) follows from Theorem 1 applied to the random variables $\left\{\mathbf{1}_{\{\mathbf{x}\in\mathrm{dom}(\mathbf{U})^N \mid RT_i(\mathbf{x})\ \leq\ t\}},\ i = 1,\ldots,N\right\}$.

□

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

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

then by Theorem 4 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, Theorem 4 says that 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 according to the probability $\Pr^N$. To measure the size of this set (according to $\Pr^N$) we introduce the *confidence level* $(1-\delta)$ by the condition

$$\left(1-\exp\left(-2N\cdot\lambda^2\right)\right) \geq (1-\delta), \tag{6}$$

which is equivalent with

$$N \geq N(\lambda,\delta) = \lceil\frac{1}{2\lambda^2}\cdot\ln\frac{1}{\delta}\rceil. \tag{7}$$

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

$$\Pr^N \left(\mathcal{G}_{N,\lambda,t} \right) \geq 1 - \delta. \tag{8}$$

The probabilistic anytime algorithm in Section 6 suggests (see Proposition 3) that the critical time region should be a measurable set in $\mathcal{B}(\mathbf{T}_{\mathbf{U}})$ that guarantees an upper bound $\varepsilon \in (0,1)$ on 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:

$$\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda) = \{ t \in \mathbf{T}_{\mathbf{U}} \mid t > threshold(\mathbf{x}, \varepsilon, \lambda) \}, \; P_{RT}\left(\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda) \right) \leq \varepsilon. \tag{9}$$

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 statistical 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 (9). Explicitly these two requirements are:

$$ECDF_{RT,N}\left((t_1(\mathbf{x}), \ldots, t_N(\mathbf{x})); t_{(K)}(\mathbf{x}) \right) - \lambda \leq CDF_{RT}\left(t_{(K)}(\mathbf{x}) \right), \tag{10}$$

$$P_{RT}\left(\left\{ t \in \mathbf{T}_{\mathbf{U}} \mid t > t_{(K)}(\mathbf{x}) \right\} \right) \leq \varepsilon. \tag{11}$$

As from (4),

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

both conditions are satisfied if

$$1 - \varepsilon \leq \frac{K}{N} - \lambda \leq CDF_{RT}\left(t_{(K)}(\mathbf{x}) \right). \tag{12}$$

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 (10) is satisfied. Furthermore, as

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

if $\mathbf{x} \in \mathcal{G}_{N,\lambda,t_{(K)}(\mathbf{x})}$ and $1 - \varepsilon \leq \frac{K}{N} - \lambda$, then (11) is satisfied.

From the first inequality in (12) 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 (7) we can take $K = K(\varepsilon, \lambda, \delta) = \lceil N(1 - \varepsilon + \lambda) \rceil$ – the minimum integer $1 \leq K \leq N$ satisfying (12) – hence

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

From (12) we have

$$1 - \varepsilon \leq CDF_{RT}\left(t_{(\lceil N(1-\varepsilon+\lambda) \rceil)}(\mathbf{x}) \right). \tag{13}$$

8 A statistical anytime algorithm for the Halting Problem

We now approach the testing of the hypothesis $H_z = \{\mathbf{U}(z) < \infty\}$ against the alternative $H'_z = \{\mathbf{U}(z) = \infty\}$, for $z \in \mathbb{Z}^+$, under (the more realistic) *assumption that the probability distribution P_{RT} is unknown.* An "erroneous decision" means rejecting H_z when H_z is true. The decision is taken on the basis of $RT(z)$, hence the probability framework is given by the probability space $(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_{RT})$, see Section 5.

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.

We generate N independent halting programs $x_1, \ldots, x_N \in \mathrm{dom}(\mathbf{U})$ and, by running them till they stop, calculate their respective running times $t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}) \in \mathbf{T_U}$. Let $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}))$.

The statistical anytime algorithm is:

Pre-processing.
Fix three rational numbers $\varepsilon, \lambda, \delta \in (0,1)$ with $\lambda < \varepsilon$.
Compute $N = N(\lambda, delta) = \lceil \frac{1}{2\lambda^2} \cdot \ln \frac{1}{\delta} \rceil$.
Generate N independent programs $\mathbf{x} = (x_1, \ldots, x_N) \in \mathrm{dom}(\mathbf{U})^N$ and calculate their respective running times $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}), \ldots, t_N(\mathbf{x}))$.
Compute the order statistics of rank $\mathbf{T} = t_{(\lceil N(1-\varepsilon+\lambda) \rceil)}(\mathbf{x})$.

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 in declaring that $\mathbf{U}(z)$ does not stop when in fact it stops. To this aim we use the statistical threshold $t_{(\lceil N(1-\varepsilon+\lambda) \rceil)}$ and the critical regions

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

Lemma 6. *For every $\mathbf{x} \in \mathrm{dom}(\mathbf{U})^N$, $\varepsilon, \lambda \in (0,1)$ with $\lambda < \varepsilon$, we have:*

$$\Pr(\mathbf{C}(RT, \mathbf{x}; \varepsilon, \lambda)) = P_{RT}(\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda)). \tag{14}$$

Proof. We have:

$$\begin{aligned}
\Pr(\mathbf{C}(RT, \mathbf{x}; \varepsilon, \lambda)) &= \Pr\left(\left\{y \in \mathrm{dom}\,\mathbf{U} \mid RT(y) > t_{(\lceil N(1-\varepsilon+\lambda) \rceil)}(\mathbf{x})\right\}\right) \\
&= P_{RT}\left(\left\{t \in \mathbf{T_U} \mid t > t_{(\lceil N(1-\varepsilon+\lambda) \rceil)}(\mathbf{x})\right\}\right) \\
&= P_{RT}(\mathbf{B}(RT, \mathbf{x}; \varepsilon, \lambda)).
\end{aligned}$$

□

Lemma 7. *For every integer $N > 0$, $\varepsilon, \lambda \in (0,1)$ with $\lambda < \varepsilon$, we have:*

$$\begin{aligned}&\Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\left(\mathbf{U}\right)^N \mid P_{RT}(\mathbf{B}\left(RT, \mathbf{x}; \varepsilon, \lambda\right)) \leq \varepsilon\right\}\right)\\ &\geq \Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\left(\mathbf{U}\right)^N \mid CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) \geq 1-\varepsilon\right\}\right).\end{aligned} \tag{15}$$

Proof. We only need to prove the implication:

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

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

$$CDF_{RT}(t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})) = P_{RT}(T_1), P_{RT}(\mathbf{B}\left(RT, \mathbf{x}; \varepsilon, \lambda\right)) = P_{RT}(T_2).$$

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

Theorem 8. *For every $\varepsilon, \lambda, \delta \in (0,1)$ with $\lambda < \varepsilon$ and $N = N(\lambda, \delta)$ we have:*

$$\Pr^N\left(\left\{\mathbf{x} \in \mathrm{dom}\left(\mathbf{U}\right)^N \mid \Pr(\mathbf{C}\left(RT, \mathbf{x}; \varepsilon, \lambda\right)) \leq \varepsilon\right\}\right) \geq 1-\delta. \tag{16}$$

Proof. From the definition (4) and the choice of the statistical threshold,

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

we have

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

In view of (13), (14) and (15) and (8) we have

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

□

According to (16), 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 $1-\delta$.

9 Implementations of the statistical anytime algorithm

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 $N = \lceil \frac{1}{2\lambda^2} \cdot \ln \frac{1}{\delta} \rceil$; this positive integer is fixed as long as λ, δ are fixed. Then use an algorithm to generate an 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 is fixed as long as $\varepsilon, \lambda, \delta$ are fixed. 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 $1 - \delta$.

Below we illustrate numerically the first implementation with fixed parameters $\varepsilon, \lambda, \delta$.

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

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 (6). We continue with the standard implementation of the statistical anytime algorithm with parameters ε, λ $(\lambda < \varepsilon)$ and

$$\delta(\tilde{N}, \lambda) = \exp(-2\tilde{N} \cdot \lambda^2) \tag{17}$$

calculate the size sample $N(\lambda, \delta(\tilde{N}, \lambda)) = \tilde{N}$. However, the value $\delta(\tilde{N}, \lambda)$ in (17) is not rational, so to preserve the inequality (6) we need to calculate a rational approximation

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

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

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 $\varepsilon, \lambda, \tilde{N}$.

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} (excelent)	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

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.

10 Final comments

A statistical anytime algorithm for the Halting Problem was proposed and investigated. This algorithm is useful in case the probability distribution of the set of halting programs for the specific universal model of computation used is unknown. The statistical algorithm – which is inspired by the probabilistic one – 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 running times. Another advantage is that the 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 prove that with a confidence level as large as required, the algorithm produces correct decisions with a probability as large as required. Finally, three implementations of the algorithm have been presented and numerically illustrated.

It will be interesting to experiment this algorithm with different models of computations in order to understand its practical utility.

Finally, as the algorithm has two parts, the pre-processing – in which the the order statistics of rank $t_{(\lceil N(1-\varepsilon+\lambda)\rceil)}(\mathbf{x})$ is calculated – and the main part, it can naturally be programmed as a faster hybrid

classical-quantum algorithm for classes of instances of the Halting Problem [1]. Indeed, for classes of instances (F, x) of the Halting Problem the main part of the algorithm can be run by efficient quantum algorithms. To fine-tune such a hybrid classical-quantum algorithm one has to re-design the statistical anytime algorithm for classes of instances (F, x) instead of the more general, unique instance $(\mathbf{U}, x)$ because the running of the universal machine is significantly larger than the running times of the machines it simulates.

Data accessibility

This paper has no data.

Competing interests

We have no competing interests.

Authors' contributions

Both authors conceived the mathematical models and wrote the paper. All authors gave final approval for publication.

Acknowledgments

We thank Dr. Ned Allen for many discussions and comments and for motivating one author (CC) to study practical approximate solutions to the Halting Problem.

Funding statement

This work was supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

References

[1] E. H. Allen, C. S. Calude. Quassical computing, *International Journal of Unconventional Computing*, accepted October 2018.

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

[3] 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.

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

[5] C. S. Calude, M. Dumitrescu. A probabilistic anytime algorithm for the Halting Problem, *Computability*, 7 (2018) 259–271. (published online May 2017)

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

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

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

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

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

[11] B. Cook, A. Podelski and A. Rybalchenko. Proving program termination, *Communications ACM*

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

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

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

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

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

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

[22] 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*, http://drops.dagstuhl.de/opus/volltexte/2009/2270/Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2009, Dagstuhl, 185–196.

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

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

[25] C. Scott. Statistical Learning Theory, Topic 3: Hoeffding's Inequality, University of Toronto, 2014, https://www.coursehero.com/file/18068309/03-hoeffding, retrieved 6 October 2018.

[26] 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.

[27] 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.

[28] 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.

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