A Statistical Anytime Algorithm for the Halting Problem

C. S. Calude\textsuperscript{1} and M. Dumitrescu\textsuperscript{2}

\textsuperscript{1}University of Auckland, NZ
\textsuperscript{2}Bucharest University, Romania

CDMTCS-529
November 2018
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 \( \{1, 2, \ldots \} \) 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 \( \alpha \) to the least integer greater than or equal to \( \alpha \). The domain of a partial function \( F: \mathbb{Z}^+ \to \overline{\mathbb{Z}}^+ \) is denoted by \( \text{dom}(F) = \{x \in \mathbb{Z}^+ \mid F(x) < \infty \} \). 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 \( 1_M \).

We assume familiarity with elementary computability theory and algorithmic information theory [19, 4, 12].

For a partially computable function \( F: \mathbb{Z}^+ \to \overline{\mathbb{Z}}^+ \) 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 \( \text{Stop}(F, t) = \{x \in \mathbb{Z}^+ \mid F(x)[t] < \infty \} \); and note that

\[
\text{dom}(F) = \bigcup_{t \in \mathbb{Z}^+} \text{Stop}(F, t).
\]

3 Complexity and universality

Informally, the algorithmic complexity of string \( x \) with respect to the interpreter/decoder \( F \) is the smallest \( t \) which produces \( x \) or \( \infty \) if \( F \) cannot produce \( x \). Formally, the algorithmic complexity relative to a partially computable function \( F: \mathbb{Z}^+ \to \overline{\mathbb{Z}}^+ \) is the partial function \( \nabla_F: \mathbb{Z}^+ \to \overline{\mathbb{Z}}^+ \) 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 \( U \) is called universal if for every partially computable function \( F: \mathbb{Z}^+ \to \overline{\mathbb{Z}}^+ \) there exists a constant \( k_{U,F} \) such that for every \( x \in \text{dom}(\nabla_F) \) we have

\[
\nabla_U(x) \leq k_{U,F} \cdot \nabla_F(x),
\]

or equivalently [7], if for every partially computable function \( F: \mathbb{Z}^+ \to \overline{\mathbb{Z}}^+ \) there exists a constant \( c_{U,F} \) such that for every \( x \in \text{dom}(F) \) we have

\[
\nabla_U(F(x)) \leq c_{U,F} \cdot x.
\]

The set \( \text{dom}(U) \) (see (1) for \( U = F \)) is computably enumerable, but not computable (the undecidability of the Halting Problem); its complement \( \text{dom}(U) \) is not computably enumerable, but the sets \( (\text{Stop}(U, t))_{t \geq 1} \) are all computable.

To solve the Halting Problem means to determine for an arbitrary pair \((F, x)\), where \( F \) is a partially computable function and \( x \in \mathbb{Z}^+ \), whether \( F(x) \) stops or not, or equivalently, whether \( x \in \text{dom}(F) \), that is, \( x \in \text{Stop}(F, t) \), for some \( t \in \mathbb{Z}^+ \). Solving the Halting Problem for a fixed universal \( U \) is enough to solve the Halting Problem. From now on we fix a universal \( U \) and study the Halting Problem “For every \( x \in \mathbb{Z}^+ \), does \( U(x) < \infty \)?”.

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 \( \Omega \) and a Borel field of subsets of \( \Omega \), \( \mathcal{B}(\Omega) \subseteq \mathcal{P}(\Omega) \). A probability space is a triple \((\Omega, \mathcal{B}(\Omega), \Pr)\), where \((\Omega, \mathcal{B}(\Omega))\) is a measurable space and
Pr: \( B(\Omega) \rightarrow [0, 1] \) is a probability measure, that is, \( \Pr \) satisfies the following two conditions: a) the probability of a countable union of mutually-exclusive sets in \( B(\Omega) \) is equal to the countable sum of the probabilities of each of these sets, and b) \( \Pr(\Omega) = 1 \). We interpret \( B(\Omega) \) as “the family of events” and \( \Omega \) as “the certain event”.

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

\[
P_X(B) = \Pr(X^{-1}(B)) = \Pr(\{\omega \mid X(\omega) \in B\}), \quad B \in B(A),
\]

which defines the probability space \((A, B(A), P_X)\).

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 \( \sum_{x \in A} P_X(\{x\}) = 1 \). A computable probability distribution \( P_X \) is a discrete probability distribution such that the function \( x \in A \mapsto P_X(\{x\}) \) 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 \( X: \Omega \rightarrow A \) 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 \( M \in B(\Omega) \), \( E(1_M) = 1 \cdot \Pr(M) + 0 \cdot \Pr(\overline{M}) = \Pr(M) \). If for some \( c \in \mathbb{R} \), \( X(\omega) = c \) for all \( \omega \), 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 Hoeffding’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 \((\Omega, B(\Omega), \Pr)\) with values in \([a, b]\) we have:

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

The Cumulative Distribution Function of a random variable \( X \) is the function \( CDF_X: \mathbb{R} \rightarrow [0, 1] \) 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 \( \theta \). 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

\[
CDF_X(y) = \sum_{k=1}^{\infty} (1 - \theta)^{k-1} \cdot \theta, \ y \in \mathbb{R},
\]
and $\lim_{y \to \infty} \text{CDF}_X(y) = 1$.

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

For fixed $r \in [0, 1]$, the value (number) $q_X(r)$ is called the $r$th quantile of the random variable $X$. Quantiles are important indicators that give information about the location and clustering of the probability values $\{\text{CDF}(x), x \in A\}$. 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 (median), third – the lower and upper $\varepsilon$th quantiles, $q_X(\varepsilon)$ and $q_X(1-\varepsilon)$, 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 $P_X(\{x \in A \mid x > q_X(1-\varepsilon)\}) \leq \varepsilon$.

*Proof.* Indeed, we have: $P_X(\{x \in A \mid x > q_X(1-\varepsilon)\}) = 1 - P_X(\{x \in A \mid x \leq q_X(1-\varepsilon)\}) = 1 - \text{CDF}_X(q_X(1-\varepsilon)) \leq \varepsilon$. □

The notions and results discussed above are theoretical and are based on the assumption that the probability $\text{Pr}$ 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 $\text{CDF}_X$, 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 $(\Omega, \mathcal{B}(\Omega), \text{Pr})$, the random variable $X : \Omega \to A$ 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 $k$th 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 $B \subseteq A, B \in \mathcal{B}(A)$ 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 $P_X(\{x \in B \mid \text{“}H_x \text{is true”}\})$. 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 $U(x)$ are the set of exact stopping times for the halting programs of $U$: $T_U = \{ t \in \mathbb{Z}^+ \mid \text{there exists } x \in \mathbb{Z}^+ \text{ such that } x \in \text{Stop}(U, t) \} = \{ t \in \mathbb{Z}^+ \mid \text{there exists } x \in \mathbb{Z}^+ \text{ such that } U(x)[t] < \infty \}$. Let us consider the family of (finite and countable) unions of sets $\text{Stop}(U, t), t \in \mathbb{Z}^+$. This family includes $\text{dom}(U)$ and is closed under complement $\text{Stop}(U, t) = \bigcup_{t' \in T_U \setminus \{t\}} \text{Stop}(U, t')$, and countable unions; accordingly, it is a Borel field of subsets of $\text{dom}(U)$, which we denote by $\mathcal{B}(\text{dom}(U))$. To define the discrete probability measure on the measurable set $(\text{dom}(U), \mathcal{B}(\text{dom}(U)))$ we fix a computable probability distribution $\rho$ on $T_U$ and put $\Pr = \Pr_{\rho} : \mathcal{B}(\text{dom}(U)) \rightarrow [0, 1]$, $\Pr(\text{Stop}(U, t)) = \rho(t)$, $t \in T_U$.

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

A computable probability space $(\text{dom}(U), \mathcal{B}(\text{dom}(U)), \Pr_{\rho_U})$ 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 $U$, $\{\rho(t) \mid t \in T_U\}$. 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 $U$ and to decide, for an arbitrary program $x$, whether $U(x) < \infty$ or $U(x) = \infty$.

The probabilistic anytime algorithm tests for an arbitrary $z \in \mathbb{Z}^+$, the hypothesis $H_z : \{U(z) < \infty\}$ against the alternative $H'_z : \{U(z) = \infty\}$. 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 $(T_U, \mathcal{B}(T_U), P_{RT})$ 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 $U$. If the computation $U(z)$ does not stop in time less than or equal to $q_{RT}(1 - \varepsilon)$, then declare that $U(z) = \infty$.

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

\[ B(P_{RT}, \varepsilon) = \{ t \in T_U \mid t > q_{RT}(1 - \varepsilon) \}, \]

\[ \text{B}(P_{RT}, \varepsilon) \text{ is independent of } z. \]
and the \textit{critical program region}

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

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

The set \( C(P_{RT}, \varepsilon) \) contains all halting programs that stop in a time bigger than \( q_{RT}(1 - \varepsilon) \), 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 \( \text{Pr} \) of a program to be in \( C(P_{RT}, \varepsilon) \) is smaller than \( \varepsilon \), so it can be made as small as required.

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

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

\section{Statistical framework}

The statistical anytime algorithm assumes that the \textit{probability distribution of RT} is unknown. Therefore, the cumulative distribution function of \( RT \), \( CDF_{RT}(t) = \text{Pr}(\{ x \in \text{dom}(U) \mid RT(x) \leq t \}) \), 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:

\[ \left( \text{dom}(U)^N, B \left( \text{dom}(U)^N \right), \text{Pr}^N \right). \]

The elements of \( \text{dom}(U)^N \) will be denoted by \( x = (x_1, \ldots, x_N) \). The “projections” \( \{ pr_1, \ldots, pr_N \} \), \( pr_i: \text{dom}(U)^N \rightarrow \text{dom}(U), pr_i(x) = x_i, i = 1, \ldots, N \), are independent random variables. If we denote by \( RT_i = RT \circ pr_i: \text{dom}(U)^N \rightarrow T_U \), \( RT_i(x) = RT(x_i), i = 1, \ldots, N \), then \( \{ RT_1, \ldots, RT_N \} \) are independent, identical distributed random variables. Furthermore, for every \( 1 \leq i \leq N \), we have:

\[ CDF_{RT_i}(t) = \text{Pr}^N \left( \left\{ x \in \text{dom}(U)^N \mid RT(x_i) \leq t, 1 \leq i \leq N \right\} \right) \]

\[ = \text{Pr}^N \left( \text{dom}(U) \times \cdots \times \{ x_i \in \text{dom}(U) \mid RT(x_i) \leq t \} \times \text{dom}(U) \times \cdots \text{dom}(U) \right) \]

\[ = \text{Pr} \left( \text{dom}(U) \right) \cdots \text{Pr} \left( \{ x_i \in \text{dom}(U) \mid RT(x_i) \leq t \} \right) \cdots \text{Pr} \left( \text{dom}(U) \right) \]

\[ = CDF_{RT}(t). \]

For every \( x \in \text{dom}(U)^N \) we put \( RT_i(x) = t_i(x), 1 \leq i \leq N \) and denote the \( N \)-dimensional time sampling space by \( (T_U^N, B(T_U^N), P_{RT}^N) \).

In the following Theorem 4, \( CDF_{RT}(t) \) is estimated by the Empirical Cumulative Distribution Function (2)

\[ ECDF_{RT,N}((RT_1(x), \ldots, RT_N(x)): t) = \frac{\# \{ 1 \leq i \leq N \mid RT_i(x) \leq t \}}{N} = \frac{\# \{ 1 \leq i \leq N \mid t_i(x) \leq t \}}{N}. \]
Theorem 4. For every positive integer $N$, $t \in T_U$ and $\lambda \in (0, 1)$, we have:

$$
\Pr^N \left( \left\{ x \in \text{dom}(U)^N \mid E C D F_{RT,N} ((R T_1(x), \ldots, R T_N(x)); t) - C D F_{RT}(t) \leq \lambda \right\} \right) \geq 1 - \exp(-2N \cdot \lambda^2). \tag{5}
$$

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

$$
E C D F_{RT,N} ((R T_1(x), \ldots, R T_N(x)); t) = \frac{1}{N} \sum_{i=1}^N 1_{\{x \in \text{dom}(U)^N \mid R T_i(x) \leq t\}}.
$$

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

$$
E \left( \frac{1}{N} \sum_{i=1}^N 1_{\{x \in \text{dom}(U)^N \mid R T_i(x) \leq t\}} \right) = \frac{1}{N} \sum_{i=1}^N \Pr^N \left( \{ x \in \text{dom}(U)^N \mid R T_i(x) \leq t \} \right) = \frac{1}{N} \sum_{i=1}^N C D F_{R T_i}(t) = C D F_{R T}(t).
$$

As $R T_i : \text{dom}(U)^N \rightarrow T_U$, $R T_i(x) = R T(x_i)$, $i = 1, \ldots, N$ are independent random variables, for every $t \in T_U$, $1_{\{x \in \text{dom}(U)^N \mid R T_i(x) \leq t\}} : \text{dom}(U)^N \rightarrow [0, 1]$, $i = 1, \ldots, N$ are also independent random variables. Consequently, the inequality (5) follows from Theorem 1 applied to the random variables $\{1_{\{x \in \text{dom}(U)^N \mid R T_i(x) \leq t\}}, i = 1, \ldots, N\}$. 

If we define the set of “good program samples” by

$$
G_{N,\lambda,\delta} = \left\{ x \in \text{dom}(U)^N \mid E C D F_{RT,N} ((R T_1(x), \ldots, R T_N(x)); t) - \lambda \leq C D F_{RT}(t) \right\},
$$

then by Theorem 4 we have

$$
\Pr^N(G_{N,\lambda,\delta}) \geq 1 - \exp(-2N \lambda^2),
$$

were $\lambda$ is the precision parameter and $(1 - \exp(-2N \lambda^2))$ can be interpreted as the confidence level that a program is in $G_{N,\lambda,\delta}$, i.e. it is a good program sample. With this interpretation, Theorem 4 says that the set of programs $x \in \text{dom}(U)^N$ on which $E C D F_{RT,N} ((R T_1(x), \ldots, R T_N(x)); t)$ estimates $C D F_{RT}(t)$ with precision at least $\lambda$ 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

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

which is equivalent with

$$
N \geq N(\lambda, \delta) = \left[ \frac{1}{2\lambda^2} \cdot \ln \frac{1}{\delta} \right]. \tag{7}
$$

The following result shows that for every $N \geq N(\lambda, \delta)$ the set of good program samples $G_{N,\lambda,\delta}$ can be made as “large” as required in probability $\Pr^N$:
Corollary 5. For every \( t \in T_U \), \( \lambda \in (0, 1) \), \( \delta \in (0, 1) \) and \( N \geq \left\lceil \frac{1}{2 \lambda} \cdot \ln \frac{1}{\delta} \right\rceil \) we have

\[
\Pr^N (G_{N, \lambda, t}) \geq 1 - \delta.
\] (8)

The probabilistic anytime algorithm in Section 6 suggests (see Proposition 3) that the critical time region should be a measurable set in \( B(T_U) \) 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:

\[
B(RT, x; \varepsilon, \lambda) = \{ t \in T_U \mid t > \text{threshold}(x, \varepsilon, \lambda) \}, \quad P_{RT}(B(RT, x; \varepsilon, \lambda)) \leq \varepsilon.
\] (9)

For a sample of programs \( x \) we use the notation \( t(x) = (t_1(x), \ldots, t_N(x)) \), where \( t_i = RT_i(x) \), \( 1 \leq i \leq N \). We increasingly order the observed running times \( t_i \) and get the values of the corresponding order statistics \( t_{(1)}(x) \leq \cdots \leq t_{(N)}(x) \). As one of these order statistics will be the choice for the statistical threshold, \( \text{threshold}(x, \varepsilon, \lambda) \), we must find the smallest number \( 1 \leq K \leq N \) such that \( x \in G_{N,\lambda,t_{(K)}} \). In terms of order statistics, \( t_{(K)}(x) \) generates a statistical threshold \( \text{threshold}(x, \varepsilon, \lambda) \) which must satisfy (9). Explicitly these two requirements are:

\[
ECDF_{RT,N} \left( (t_1(x), \ldots, t_N(x)) \mid t_{(K)}(x) \right) \leq CDF_{RT} \left( t_{(K)}(x) \right), \quad P_{RT} \left( \{ t \in T_U \mid t > t_{(K)}(x) \} \right) \leq \varepsilon.
\] (10)

As from (4),

\[
ECDF_{RT,N} \left( (t_1(x), \ldots, t_N(x)) \right) = \frac{K}{N},
\]

both conditions are satisfied if

\[
1 - \varepsilon \leq \frac{K}{N} - \lambda \leq CDF_{RT} \left( t_{(K)}(x) \right), \quad (12)
\]

Indeed, from the definition of \( CDF_{RT} \), if \( x \in G_{N,\lambda,t_{(K)}} \), then

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

so (10) is satisfied. Furthermore, as

\[
P_{RT} \left( \{ t \in T_U \mid t > t_{(K)}(x) \} \right) = 1 - CDF_{RT} \left( t_{(K)}(x) \right),
\]

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

From the first inequality in (12) we get \( K \geq N \left( 1 - \varepsilon + \lambda \right) \). 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

\[
\text{threshold}(x, \varepsilon, \lambda) = t_{K(\varepsilon, \lambda, \delta)}(x) = t_{N(1-\varepsilon+\lambda)}(x).
\]

From (12) we have

\[
1 - \varepsilon \leq CDF_{RT} (t_{N(1-\varepsilon+\lambda)}(x)).
\] (13)
8 A statistical anytime algorithm for the Halting Problem

We now approach the testing of the hypothesis $H_z = \{U(z) < \infty\}$ against the alternative $H'_z = \{U(z) = \infty\}$, 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 $(\mathbb{T}_U, B(\mathbb{T}_U), P_{RT})$, 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 $x_1, \ldots, x_N \in \text{dom}(U)$ and, by running them till they stop, calculate their respective running times $t_1(x), \ldots, t_N(x) \in \mathbb{T}_U$. Let $x = (x_1, \ldots, x_N)$ and $t(x) = (t_1(x), \ldots, t_N(x))$.

The statistical anytime algorithm is:

**Pre-processing.**

Fix three rational numbers $\varepsilon, \lambda, \delta \in (0, 1)$ with $\lambda < \varepsilon$.

Compute $N = N(\lambda, \text{delta}) = \lceil \frac{1}{\varepsilon \lambda} \cdot \ln \frac{1}{\delta} \rceil$.

Generate $N$ independent programs $x = (x_1, \ldots, x_N) \in \text{dom}(U)^N$ and calculate their respective running times $t(x) = (t_1(x), \ldots, t_N(x))$.

Compute the order statistics of rank $T = t_{(\lfloor N(1-\varepsilon+\lambda)\rfloor)}(x)$.

**Main part.**

Let $z$ be an arbitrary program for $U$.

If the computation $U(z)$ does not stop in time less than or equal to $T$, then declare that $U(z) = \infty$.

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

$$
\mathcal{B}(RT, x; \varepsilon, \lambda) = \{t \in \mathbb{T}_U \mid t > t_{(\lfloor N(1-\varepsilon+\lambda)\rfloor)}(x)\},
$$

$$
\mathcal{C}(RT, x; \varepsilon, \lambda) = \{y \in \text{dom}(U) \mid RT(y) > t_{(\lfloor N(1-\varepsilon+\lambda)\rfloor)}(x)\}.
$$

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

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

**Proof.** We have:

$$
\Pr(\mathcal{C}(RT, x; \varepsilon, \lambda)) = \Pr(\{y \in \text{dom}(U) \mid RT(y) > t_{(\lfloor N(1-\varepsilon+\lambda)\rfloor)}(x)\})
$$

$$
= P_{RT}(\{t \in \mathbb{T}_U \mid t > t_{(\lfloor N(1-\varepsilon+\lambda)\rfloor)}(x)\})
$$

$$
= P_{RT}(\mathcal{B}(RT, x; \varepsilon, \lambda)).
$$

□

10
**Lemma 7.** For every integer $N > 0$, $\varepsilon, \lambda \in (0, 1)$ with $\lambda < \varepsilon$, we have:

$$\Pr^N \left( \left\{ x \in \text{dom} \left( U \right)^N \mid P_{RT} \left( B \left( RT, x; \varepsilon, \lambda \right) \right) \leq \varepsilon \right\} \right) \geq \Pr^N \left( \left\{ x \in \text{dom} \left( U \right)^N \mid CDF_{RT} \left( t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right) \right) \geq 1 - \varepsilon \right\} \right). \quad (15)$$

*Proof.* We only need to prove the implication:

$$CDF_{RT} \left( t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right) \right) \geq 1 - \varepsilon \implies P_{RT} \left( B \left( RT, x; \varepsilon, \lambda \right) \right) \leq \varepsilon.$$

If $T_1 = \{ k \in \mathbf{T}_U \mid 1 \leq k \leq t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right) \}$ and $T_2 = \{ j \in \mathbf{T}_U \mid j > t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right) \}$, then $T_1 \cap T_2 = \emptyset$ and

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

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

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

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

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

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

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

$$\text{threshold} \left( x, \varepsilon, \lambda \right) = t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right),$$

we have

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

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

$$\Pr^N \left( \left\{ x \in \text{dom} \left( U \right)^N \mid Pr \left( C \left( RT, x; \varepsilon, \lambda \right) \right) \leq \varepsilon \right\} \right) \geq \Pr^N \left( \left\{ x \in \text{dom} \left( U \right)^N \mid P_{RT} \left( B \left( RT, x; \varepsilon, \lambda \right) \right) \leq \varepsilon \right\} \right) \geq \Pr^N \left( \left\{ x \in \text{dom} \left( U \right)^N \mid CDF_{RT} \left( t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right) \right) \geq 1 - \varepsilon \right\} \right) \geq \Pr^N \left( \left\{ x \in \text{dom} \left( U \right)^N \mid CDF_{RT} \left( t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right) \right) \geq \frac{\left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor}{N} - \lambda \right\} \right) \geq 1 - \delta.$$

According to (16), the probability $Pr^N$ of the event that the statistical anytime algorithm gives a wrong decision, that is, it declares $U(z) = \infty$ when there exists $t > t_{\left( \left\lfloor N \left( 1 - \varepsilon + \lambda \right) \right\rfloor \right)} \left( x \right)$ such that $U(z)$ stops in time $t$, is smaller or equal than $\varepsilon$, is larger than $1 - \delta$, i.e. the probability of error is smaller than or equal than $\varepsilon$ with confidence larger than $1 - \delta$. 

11
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 \( N = \lceil \frac{1}{2\lambda^2} \ln \frac{1}{\varepsilon} \rceil \); this positive integer is fixed as long as \( \lambda, \delta \) are fixed. Then use an algorithm to generate an injective computable enumeration of \( \text{dom}(U) \) 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 \( U(z) \) does not stop in time \( \tau = \frac{N(1-\varepsilon+\lambda)}{\varepsilon} \), then declare that \( U(z) = \infty \). In the latter case the probability of error is smaller than or equal to \( \varepsilon \) with confidence larger than \( 1 - \delta \).

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

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>( \varepsilon )</th>
<th>( \lambda (&lt; \varepsilon) )</th>
<th>( N(\lambda, \delta) )</th>
<th>( [N(\lambda, \delta) \cdot (1 - \varepsilon + \lambda)] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{1}{1000} )</td>
<td>( \frac{5}{1000} )</td>
<td>( \frac{1}{1000} )</td>
<td>( 2.3026 \times 10^6 )</td>
<td>( 2.2934 \times 10^6 )</td>
</tr>
<tr>
<td>( \frac{1}{1000} )</td>
<td>( \frac{1}{1000} )</td>
<td>( \frac{5}{10000} )</td>
<td>( 9.2103 \times 10^6 )</td>
<td>( 9.2057 \times 10^6 )</td>
</tr>
<tr>
<td>( \frac{5}{1000} )</td>
<td>( \frac{5}{1000} )</td>
<td>( \frac{1}{1000} )</td>
<td>( 2.6492 \times 10^6 )</td>
<td>( 2.6386 \times 10^6 )</td>
</tr>
<tr>
<td>( \frac{5}{1000} )</td>
<td>( \frac{1}{1000} )</td>
<td>( \frac{5}{10000} )</td>
<td>( 1.0597 \times 10^7 )</td>
<td>( 1.0592 \times 10^7 )</td>
</tr>
<tr>
<td>( \frac{1}{1000} )</td>
<td>( \frac{5}{1000} )</td>
<td>( \frac{1}{1000} )</td>
<td>( 3.4539 \times 10^6 )</td>
<td>( 3.4401 \times 10^6 )</td>
</tr>
<tr>
<td>( \frac{1}{1000} )</td>
<td>( \frac{1}{1000} )</td>
<td>( \frac{5}{10000} )</td>
<td>( 1.3816 \times 10^7 )</td>
<td>( 1.3809 \times 10^7 )</td>
</tr>
</tbody>
</table>

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 \( \varepsilon, \lambda (\lambda < \varepsilon) \) and

\[
\delta(\tilde{N}, \lambda) = \exp(-2\tilde{N} \cdot \lambda^2)
\]

(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 \( \varepsilon, \lambda, \tilde{N} \).

12
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 \( U(z) \) in the statistical anytime algorithm. We then use an injective dovetailing algorithm to generate as many elements of \( \text{dom}(U) \) as possible in time \( T \). In this way we will obtain a sample of \( N(T) \) programs \( x_1 \vdots x_N(T) \) and their respective running times \( t_1 \vdots t_N(T) \) 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 \( \varepsilon \), the precision \( \lambda \) and the confidence level \( \delta \) 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 \( [N(\lambda, \delta) \cdot (1 - \varepsilon + \lambda)] \) 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 order statistics of rank \( t([N(1-\varepsilon+\lambda)])(x) \) 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 \((U, x)\) 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


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


