Anytime Algorithms for Non-Ending Computations

C. S. Calude[1], D. Desfontaines[2]

[1],University of Auckland, NZ
[2]École Normale Supérieure, Paris, France

CDMTCS-463
June 2014

International Journal of Foundations of Computer Science
© World Scientific Publishing Company

ANYTIME ALGORITHMS FOR NON-ENDING COMPUTATIONS

CRISTIAN S. CALUDE

Department of Computer Science, University of Auckland
Private Bag 92019, Auckland, New Zealand
cristian@cs.auckland.ac.nz

DAMIEN DESFONTAINES

École Normale Supérieure
45 rue d'Ulm, 75005 Paris, France
damien@desfontain.es

Received (26 June 2014)
Revised (December 12, 2014)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

A program which eventually stops but does not halt "too quickly" halts at a time which is algorithmically compressible. This result—originally proved in [4]—is proved in a more general setting. Following Manin [11] we convert the result into an anytime algorithm for the halting problem and we show that the stopping time (cut-off temporal bound) cannot be significantly improved.

Keywords: Anytime algorithm, algorithmically compressible number, halting problem, density

1. Introduction

Anytime algorithms exchange execution time for quality of results [8]. Anytime algorithms can be executed in two modes: either by being given a contract time (a set amount of time to execute), or an interruptible method. To improve the solution, anytime algorithms can be continued after they have halted. Instead of correctness, an anytime algorithm returns a result 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.

Standard anytime algorithms eventually stop, albeit in a prohibitively long time. Following Manin [11] we use a more general form of anytime algorithm as an approximation for a computation which may not end. The proposed anytime algorithm for the halting problem works in the following way: to test whether a program eventually stops we first compute a temporal bound—the interruptible (stopping) condition—and execute 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 the program never stops and evaluate the error probability. By running the program a longer time we can improve its performance either by getting to the halting time or by improving the probability error.

The essence of the algorithm proposed in this paper is based on the fact that programs which take a long time to halt stop at times with a specific property, namely algorithmically compressibility—a machine can generate such a time from a "smaller" input.

In the following we will denote by $\mathbb{Z}^+$ the set of positive integers $\{1, 2, \cdots\}$ and let $\overline{\mathbb{Z}^+} = \mathbb{Z}^+ \cup \{\infty\}$. The domain of a partial function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is denoted $\mathrm{dom}\,(F)$: $\mathrm{dom}\,(F) = \{x \in \mathbb{Z}^+ \mid F\,(x) \neq \infty\}$. All logarithms (log) are implicitly binary. We denote by $\#S$ the cardinality of the set S. We assume familiarity with elementary algorithmic information theory, see [10, 1, 7].

2. A glimpse of algorithmic complexity

In this section we present a few elementary results in algorithmic information theory in an unconventional framework, i.e. for positive integers instead of strings.

2.1. *Algorithmic complexity*

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 $\nabla_F\,(x) = \infty$. That is, the algorithmic complexity of x is the smallest description/encoding of x with respect to the interpreter/decoder F, or infinity if F cannot produce x.

The algorithmic complexity is similar to the complexities studied in [10, 6, 5, 4, 11]; the plain Kolmogorov complexity is about the logarithm of the algorithmic complexity. While the Kolmogorov complexity is optimal up to an additive constant, the optimality of ∇ is up to a multiplicative constant. As, in contrast with Kolmogorov complexity, the complexity ∇_F is injective, there exist at most N positive integers with complexity bounded by N.

Proposition 1. *Let F be a partially computable function. The following are true:*

(1) For all $x \in \mathrm{dom}(\nabla_F)$, $F(\nabla_F(x)) = x$.
(2) The algorithmic complexity is injective.
(3) For every $N \geq 1$, $\#\{i \in \mathbb{Z}^+ \mid \nabla_F(i) \leq N\} \leq N$, hence ∇_F is unbounded if its domain is infinite.
(4) For every $M \geq 1$ there exists $x > M$ such that $\nabla_F(x) > x/2$.

Proof. (1) Follows from the definition of ∇_F. (2) Applying F to ∇_F we get that the algorithmic complexity is injective. (3) Follows from the definition of ∇_F; if the domain of ∇_F is infinite, then ∇_F is unbounded. (4) As ∇_F is injective, for each $N \geq 1$ there exists at most $N/2$ integers $1 \leq x \leq N$ such that $\nabla_F(x) \leq N/2$. Consequently, there are at least $N/2$ integers $1 \leq x \leq N$ such that $\nabla_F(x) > N/2 \geq x/2$, so the property (4) follows. □

Comment. The property (4) above shows the existence of integers with high complexity leading to the definition of algorithmic incompressibility in Section 2.3. Choosing $F(x) = x + 1$ we see that the condition $\nabla_F(x) > x/2$ cannot be replaced with $\nabla_F(x) > x$.

2.2. *Universality*

In this section we give a new characterisation of universality which will be useful for some applications.

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

$$\nabla_U(x) \leq k_{U,F} \cdot \nabla_F(x). \tag{1}$$

A partially computable function U is universal if it allows smaller descriptions for all positive integers (but for a multiplicative constant) than all other partially computable functions.

Theorem 2. *A partially computable function U is universal iff for every partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ there exists a constant $c_{U,F}$ such that for every $x \in \mathrm{dom}(F)$ we have*

$$\nabla_U(F(x)) \leq c_{U,F} \cdot x. \tag{2}$$

Proof. Assume U satisfies the condition (1). Taking F to be the identity we get a constant $k_{U,\mathrm{id}}$ such that for every $z \in \mathbb{Z}^+$

$$\nabla_U(z) \leq k_{U,\mathrm{id}} \cdot \nabla_{\mathrm{id}}(z) = k_{U,\mathrm{id}} \cdot z. \tag{3}$$

Next take F satisfying (1) and $x \in \text{dom}\,(F)$. By definition of ∇_U and the hypothesis, $\nabla_U(x) < \infty$ and we have $U(\nabla_U(x)) = x$, hence $F(U(\nabla_U(x))) = F(x)$.

Let $M_F = F \circ U$. Using in order (1), the inequality $\nabla_{M_F}(F(x)) \leq \nabla_U(x)$ and (3) we deduce (2):

$$\nabla_U(F(x)) \leq k_{U,M_F} \cdot \nabla_{M_F}(F(x)) \leq k_{U,M_F} \cdot \nabla_U(x) \leq k_{U,M_F} \cdot k_{U,\text{id}} \cdot x,$$

hence $c_{U,F} = k_{U,M_F} \cdot k_{U,\text{id}}$.

Conversely, assume F satisfies the condition (2). For every $x \in \mathbb{Z}^+$ with $\nabla_F(x) < \infty$ we deduce in order the relations $\nabla_F(x) \in \text{dom}\,(F)$ and $F(\nabla_F(x)) = x$, hence:

$$\nabla_U(x) = \nabla_U(F(\nabla_F(x))) \leq c_{U,F} \cdot \nabla_F(x).$$

The relation (1) is satisfied for $k_{U,F} = c_{U,F}$. □

Comment. The difference between (1) and (2) is in the role played by F: in the traditional condition (1), F appears through ∇_F (which sometimes can be incomputable), while in (2) F appears as argument of ∇_U making the second member of the inequality always computable.

Comment. A universal partially computable function U "simulates" any other partially computable function F in the following sense: if $x \in \text{dom}\,(F)$, then from (2), one can deduce that $\nabla_U(F(x)) \leq c_{U,F} \cdot x$, hence there exists $y \leq c_{U,F} \cdot x$ in $\text{dom}\,(U)$ such that $U(y) = F(x)$. In particular, $\nabla_U(x) < \infty$, for all $x \in \mathbb{Z}^+$.

Comment. In [11] a partially computable function $U \colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is called *strongly universal* (*programmable universal* in [3]) if for every partially computable function $F \colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ there exists a constant $k_{U,F}$ such that for every $x \in \mathbb{Z}^+$ there exists $y \leq k_{U,F} \cdot x$ with $U(y) = F(x)$. It is easy to prove that a partially computable function U is universal iff it is strongly/programmable universal and the constant $k_{U,F}$ is the same in both definitions.

Comment. If U is a universal partially computable function, then using the identity function $F(x) = x$ we get a constant $k_{U,\text{id}}$ such that for every $x \in \mathbb{Z}^+$, $\nabla_U(x) \leq k_{U,\text{id}} \cdot x$.

Comment. If U_1 and U_2 are universal partially computable functions then there exists a constant c_{U_1,U_2}, c_{U_2,U_1} such that for all $x \in \mathbb{Z}^+$, $c_{U_2,U_1} \leq \frac{\nabla_{U_1}(x)}{\nabla_{U_2}(x)} \leq c_{U_1,U_2}$.

Corollary 3. *For every universal partially computable function U, every partially computable function $F : \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ and all $x \in \text{dom}\,(F)$ we have:*

$$\nabla_U(F(x)) \leq k_{U,F\circ U} \cdot \nabla_U(x),$$

where $k_{U,F\circ U}$ comes from (1).

Proof. Applying (1) on $F \circ U$ and $F(x)$ and using the definition of ∇, we get:

$$\nabla_U(F(x)) \leq k_{U,F\circ U} \cdot \nabla_{F\circ U}(F(x)) \leq k_{U,F\circ U} \cdot \nabla_U(x)\,. \qquad \square$$

In what follows we will fix a universal partially computable function U and write ∇ instead of ∇_U.

Theorem 4. *The complexity* ∇ *is incomputable.*

Proof. Assume by contradiction that ∇ is computable. Then the partial function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ defined by $F(x) = \inf\left\{i \in \mathbb{Z}^+ \mid \nabla(i) \geq x^2\right\}$ is partially computable, and, because dom(∇) is infinite and Proposition 1(3), total. Clearly, $\nabla(F(x)) \geq x^2$, for all $x \in \mathbb{Z}^+$.

By the universality condition (2), there exists a constant $c_F = c_{U,F}$ such that for all $x \in \mathbb{Z}^+$ we have: $\nabla(F(x)) \leq c_F \cdot x$, in contradiction with the inequality $\nabla(F(x)) \geq x^2$. $\square$

2.3. *Algorithmic incompressibility (randomness)*

Following [11], an *incompressibility (randomness) cut-off function* is a computable, increasing and divergent function $r\colon \mathbb{Z}^+ \longrightarrow \mathbb{R}^+$ such that the function $x \mapsto \frac{x}{r(x)}$ is increasing and divergent.

Example 5. *The following are incompressibility cut-off functions:*

- $r(x) = \log(x)\,, x > 1,$
- $r(x) = x^\alpha, 0 < \alpha < 1,$
- $r(x) = \frac{x}{\log(x+1)}\cdot$

Let r be an incompressibility cut-off function. An integer $x \in \mathbb{Z}^+$ is said to be r-*(algorithmic) incompressible (random)* if $\nabla(x) \geq \frac{x}{r(x)}\cdot$ In view of Proposition 1(4), infinitely many r-(algorithmic) incompressible (random) integers exist.

Theorem 6. [2] *The set*

$$\text{Incompress}(r) = \left\{x \in \mathbb{Z}^+ \mid \nabla(x) \geq \frac{x}{r(x)}\right\}$$

is immune, i.e. it is infinite and contains no infinite computably enumerable subsets.

Proof. By the definition of r and Proposition 1(4), the set Incompress(r) is infinite. Assume by absurdity that Incompress(r) contains an infinite computably enumerable subset E. Let $e\colon \mathbb{Z}^+ \longrightarrow \mathbb{R}^+$ be a computable one-one function which enumerates E, that is, $E = e(\mathbb{Z}^+)$. The partially computable function $F(x) = e(\inf\{i \in \mathbb{Z}^+ \mid \frac{e(i)}{r(e(i))} \geq x^2\})$ has the following properties:

(1) F is total,
(2) there exists a constant c_F such that $\nabla(F(x)) \leq c_F \cdot x$, for all $x \in \mathbb{Z}^+$,
(3) $\frac{F(x)}{r(F(x))} \geq x^2$, for all $x \in \mathbb{Z}^+$,

hence

$$x^2 \leq \frac{F(x)}{r(F(x))} \leq \nabla(F(x)) \leq c_F \cdot x,$$

for all $x \in \mathbb{Z}^+$, a contradiction. □

Using Proposition 1 we get the following two corollaries.

Corollary 7. *The set $\{x \in \mathbb{Z}^+ \mid \nabla(x) \geq x/2\}$ is immune.*

Proof. The set $\{x \in \mathbb{Z}^+ \mid \nabla(x) \geq x/2\}$ is an infinite subset of the immune set Incompress (r), for any incompressibility cut-off function r. □

Corollary 8. *Let r be an incompressibility cut-off function. Then, for all $N \in \mathbb{Z}^+$ we have:*

$$\lim_{N\to\infty} \frac{\#\left\{1 \leq x \leq N \mid \nabla(x) \geq \frac{x}{r(x)}\right\}}{N} = 1.$$

Proof. We have:

$$\begin{aligned}
\frac{\#\left\{1 \leq x \leq N \mid \nabla(x) \geq \frac{x}{r(x)}\right\}}{N} &= 1 - \frac{\#\left\{1 \leq x \leq N \mid \nabla(x) < \frac{x}{r(x)}\right\}}{N} \\
&\geq 1 - \frac{\#\left\{1 \leq x \leq N \mid \nabla(x) < \frac{N}{r(N)}\right\}}{N} \\
&\geq 1 - \frac{1}{r(N)} \underset{N\to\infty}{\longrightarrow} 1,
\end{aligned}$$

where the first equality is obtained by taking the complement, the second by set inclusion and the third by injectivity of ∇. □

3. Incompressibility cut-off

In this section we generalise a result proved by Manin [11] which gives a sufficient condition that the value of a partially computable function F in a point x from its domain is r–compressible.

Theorem 9. *Let $F : \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ be a partially computable function and $x \in dom(F)$. Assume that*

$$\frac{F(x)}{r(F(x))} > k_F \cdot \nabla(x), \tag{4}$$

where k_F comes from (1). Then, $F(x)$ is r-compressible.

Proof. *Using (1) we get:* $\nabla(F(x)) \leq k_F \cdot \nabla(x) < \frac{F(x)}{r(F(x))}$. □

Example 10. ***[Manin's incompressibility cut-off]*** *Assume that F is a partially computable function satisfying the following two conditions for some $x \in \mathrm{dom}(F)$ and $\varepsilon > 0$:*

1) $F(x) > \nabla(x)^{1+\varepsilon}$,

2) $\frac{\nabla(x)^\varepsilon}{(1+\varepsilon)log(\nabla(x))} \geq k_F$.

Then, $F(x)$ is $\log$–compressible.

Proof. We have:

$$\frac{F(x)}{\log(F(x))} \geq \frac{\nabla(x)^{1+\varepsilon}}{(1+\varepsilon)\log(\nabla(x))} \geq k_F \cdot \nabla(x),$$

so by Theorem 9:

$$\nabla(F(x)) < \frac{F(x)}{\log(F(x))}.$$

□

The bound (4) used in Theorem 9 depends on $\nabla(x)$—an incomputable quantity. This choice is due to the fact that by (3), $\nabla(x) = O(x)$, so a bound of the form $g(\nabla(x))$ is better than the bound $g(x)$. These bounds are asymptotically (up to a multiplicative constant) the same if x is r–incompressible, but the first one can be significantly smaller if $\nabla(x) \ll x$. The disadvantage of bound (4) comes from its incomputability. We can get a computable bound in the following way:

Corollary 11. *Let $F \colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ be a partially computable function and $x \in dom(F)$. Assume that*

$$\frac{F(x)}{r(F(x))} > c_F \cdot x, \tag{5}$$

where c_F comes from (2). Then, $F(x)$ is r–compressible.

Proof. Using (2) we have: $\nabla(F(x)) \leq c_F \cdot x < \frac{F(x)}{r(F(x))}$. □

4. Temporal bounds

Theorem 9 and Corollary 11 are general results in the sense that they apply to every partially computable function. In this section we will illustrate the use of Corollary 11 for a special partially computable function, the time complexity. This will lead to an anytime algorithm for the halting problem.

Let Steps: $\mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ be the partially computable function such that $U(x) < \infty$ iff $U(\mathrm{Steps}(\mathrm{x})) < \infty$, and if $U(x) < \infty$, then $U(x)$ stops in Steps (x) steps.

If we apply Theorem 9 and Corollary 11 to Steps we get a similar result to the main theorem of [4], where the bound can be expressed with or without $\nabla(x)$.

Theorem 12. *Assume that $U(x)$ halts in t_x steps, with t_x such that $\frac{t_x}{r(t_x)} > k_{\mathrm{Steps}} \cdot \nabla(x)$ or $\frac{t_x}{r(t_x)} > c_{\mathrm{Steps}} \cdot x$. Then, t_x is r–compressible.*

To get the entire power of Theorem 12 we need to use it in conjunction with the following result stating that the r–incompressible times (at which a computation can halt) is a "small" set of positive integers. To this aim we will work with the *(natural) density* on $\mathcal{P}(\mathbb{Z}^+)$. The natural density is not a probability in Kolmogorov's sense (no such probability can be defined for all subsets of positive integers). However, if a positive integer is "randomly" selected from the set $\{1, 2, \ldots, m\}$, then the probability that it belongs to a given set $A \subset \mathbb{Z}^+$ is

$$p_m(A) = \frac{\#(\{1, \ldots, m\} \cap A)}{m}.$$

If $\lim_{N \longrightarrow \infty} p_m(A)$ exists and is equal to δ, then the set $A \subset \mathbb{Z}^+$ has density $d(A) = \delta$.

In a sense, the density $d(A)$ models "the probability that a randomly chosen integer $x \in \mathbb{Z}^+$ is in A".

A set $A \subset \mathbb{Z}^+$ is said to have *constructive density zero* if there exists a computable function $b\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ such that for every $i \in \mathbb{Z}^+$ we have $p_m(A) < 2^{-i}$ provided $m \geq b(i)$.

Fact 4.1. *For every incompressibility cut-off function r, the following set $\left\{x \in \mathbb{Z}^+ \mid \nabla(x) < \frac{x}{r(x)}\right\}$ has constructive density zero.*

Proof. The map $x \mapsto \frac{x}{r(x)}$ is increasing as r is an incompressibility cut-off function, so we have

$$\left\{1 \leq x \leq N \mid \nabla(x) < \frac{x}{r(x)}\right\} \subseteq \left\{1 \leq x \leq N \mid \nabla(x) < \frac{N}{r(N)}\right\}.$$

Consequently, because the injectivity of ∇,

$$p_N\left(\overline{\text{Incompress}}(r)\right) \le \frac{1}{r\,(N)}, \tag{6}$$

so the limit converges constructively to 0 because r is computable, increasing and divergent. □

Assume that $U\,(x)$ does not stop in time T_x satisfying the second inequality in Theorem 12, i.e.

$$\frac{T_x}{r\,(T_x)} > c_{\text{Steps}} \cdot x.$$

From (6), for every $s \in \mathbb{Z}^+$, if $N \ge M_s^x = \min\left\{n \in \mathbb{Z}^+ \mid n \ge T_x \text{ and } r\,(n) \ge s\right\}$, then

$$p_N\left(\overline{\text{Incompress}}(r)\right) \le \frac{1}{s}\cdot$$

Given $x, s \in \mathbb{Z}^+$, compute M_s^x, and run $U\,(x)$ for the contracted time M_s^x. If the computation doesn't stop in time M_s^x, then either

- $U\,(x)$ eventually halts and the halting time belongs to a set of density smaller than $\frac{1}{s}$, or
- $U\,(x)$ never stops.

We have obtained the following interruptible divergence anytime algorithm:

```
If $U\,(x)$ doesn't stop in time $M_s^x$, then the probability
(according to density) that $U\,(x)$ never stops is larger
than $1 - \frac{1}{s}\cdot$
```

Comment. Theorem 12 was formulated for the time complexity. In fact it works for every abstract Blum complexity measure for U, i.e. for every partially computable function $B\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ with the following two properties: a) $B\,(x) < \infty$ iff $U\,(x) < \infty$; and b) the predicate "$B\,(x) = n$" is computable.

5. Temporal bounds

Let us assume that we have some control over the number of computational steps taken by U. More precisely, we consider the following

Assumption. For any $n \in \mathbb{Z}^+$, there exist $a, b \in \mathbb{Z}^+$ and a computable family of programs $(x_R)_{R\in\mathbb{Z}^+}$ such that $U(x_R)$ halts in exactly $a + b \cdot R$ steps and $n \cdot x_R \leq b \cdot R$. Furthermore, a can be chosen arbitrarily large.

This condition may seem artificial, but it is actually verified by all "reasonable" models of computation. Indeed, one can write a program x_R executing the following instructions:

(1) compute a large number b from a constant c hard-coded in the source code (for example, $b = c^3$);
(2) read the input tape, on which we have placed R and execute a dummy loop b steps;
(3) and halt.

The number corresponding to the program is bounded by $K \cdot c^2 \cdot R$, where K is constant: c needs $2\log(c)$ bits to be stored in the source code, while R needs $\log(R)$ bits to be written on the input tape. So, we have

$$\frac{x_R}{b \cdot R} \leq \frac{K \cdot c^2 \cdot R}{c^3 \cdot R} = \frac{K}{c}.$$

If we make c large enough then we have $x_R \leq \frac{b \cdot R}{n}$.

One can easily verify that this method allows to effectively write a 2-tape Turing machine or a C/C++ program having the desired property.

The Assumption above allows us to show that the bound in Theorem 12 cannot be significantly improved. First we need the following independent more general result.

Lemma 13. *Let f be a computable, strictly increasing function such that $\frac{f(x)}{r(f(x))} = o(x)$. Then $a + b \cdot f(R)$ is r-incompressible for infinitely many R.*

Proof.

Let $g(R) = a + b \cdot f(R)$. First we prove that $\frac{g(x)}{r(g(x))} = o(x)$. Indeed, we have:

$$\frac{g(x)}{r(g(x))} = \frac{a + b \cdot f(x)}{r(a + b \cdot f(x))} \leq \frac{a + b \cdot f(x)}{r(f(x))} \leq a + b \cdot \frac{f(x)}{r(f(x))} = o(x).$$

Second, as g is injective, there exists a computable function g^{-1} such that for all $x \in \mathbb{Z}^+$, $g^{-1}(g(x)) = x$. Using the universality of U and Corollary 3, there exists a constant $k_{g^{-1}}$ such that $\nabla\left(g^{-1}(x)\right) \leq k_{g^{-1}} \cdot \nabla(x)$. Using Proposition 1(4) and the fact that $\frac{g(x)}{r(g(x))} = o(x)$, we can choose R so that it satisfies the inequalities:

$$\nabla (R) > R/2 \text{ and } R/2 \cdot k_{g^{-1}} \cdot \frac{g(R)}{r(g(R))}. \tag{7}$$

We then have:

$$k_{g^{-1}} \cdot \frac{g(R)}{r(g(R))} < R/2 < \nabla (R) = \nabla \left(g^{-1}(g(R))\right) \leq k_{g^{-1}} \cdot \nabla (g(R)),$$

hence

$$\nabla (g(R)) \geq \frac{g(R)}{r(g(R))},$$

that is, $g(R)$ is r-incompressible. As we can choose arbitrarily large integers R verifying (7), the proof is concluded. □

Theorem 14. *In Theorem 12, the condition "$\frac{t_x}{r(t_x)} > c_{Steps} \cdot x$" cannot be replaced with the condition "$t_x > f(x)$", where f is an increasing, computable function such that there exist $m, n \in \mathbb{Z}^+$ with the property that for all $x \in \mathbb{Z}^+$ we have $nx < f(x) \leq m + nx$.*

Proof. Assume for the sake of a contradiction that Theorem 12 is true with the new condition. Fix a function f having all properties in the statement of the theorem.

Let (x_R) be the family of programs from the Assumption above, chosen so that

(i) $m < a$,
(ii) $x_R < \frac{b \cdot R}{n}$.

Use Lemma 13 (for the identity function) to get a positive integer R such that

(iii) $a + b \cdot R$ is r-incompressible.

We then have:

$$f(x_R) \leq f\left(\frac{b \cdot R}{n}\right) \leq m + b \cdot R < a + b \cdot R,$$

because of (i) and (ii). Applying Theorem 12 with the new condition implies that $a + b \cdot R$ is r-compressible, thus contradicting (iii). □

6. Final comments

In [4] it was proved that if a program doesn't stop quickly (meaning, before a temporal bound which can be computed from the program), then the program either stops at a time which is algorithmically compressible or never stops. Based on the fact that the set of algorithmically compressible times has constructive density zero (i.e. it is constructively negligible) we can construct an anytime algorithm for solving the halting problem. The analysis in this paper shows two facts: a) a positive one, in which the construction of the anytime algorithm can be done with respect to algorithmically very low compressible times, and b) a negative one, that the temporal bound cannot be significantly lowered.

It will be interesting to see if the same method can be applied to other incomputable problems, in particular, to program testing.

Acknowledgement

The authors have been supported in part by the Quantum Computing Research Initiatives at Lockheed Martin. We thank Marcus Triplett and the anonymous referees for useful comments which lead to a better presentation of the results.

References

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

[2] C. Calude, I. Chiţescu. Strong noncomputability of random strings, *Internat. J. Comput. Math.* 11 (1982), 43–45.

[3] C. S. Calude, D. Desfontaines. Universality and almost decidability, *Fundamenta Informaticae*, 2014. (to appear).

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

[5] C. S. Calude, M. A. Stay. Natural halting probabilities, partial randomness, and Zeta functions, *Information and Computation* 204 (2006), 1718–1739.

[6] C. S. Calude, M. A. Stay. From Heisenberg to Gödel via Chaitin, *International Journal of Theoretical Physics* 44, 7 (2005), 1053–1065.

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

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

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

[10] Yu. I. Manin. *A Course in Mathematical Logic for Mathematicians*, Springer, Berlin, 1977; second edition, 2010.

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