Computability of the entropy of one-tape Turing machines

Emmanuel Jeandel

LORIA, UMR 7503 – Campus Scientifique, Vandoeuvre-lès-Nancy Cedex, France
emmanuel.jeandel@loria.fr

Abstract

We prove that the maximum speed and the entropy of a one-tape Turing machine are computable, in the sense that we can approximate them to any given precision ϵ. This is counterintuitive, as all dynamical properties are usually undecidable for Turing machines. The result is quite specific to one-tape Turing machines, as it is not true anymore for two-tape Turing machines by the results of Blondel et al., and uses the approach of crossing sequences introduced by Hennie.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Turing Machines, Dynamical Systems, Entropy, Crossing Sequences, Automata

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.421

1 Introduction

The Turing machine is probably the most well known of all models of computation. This particular model has many variations, that all lead to the same notion of computability. The simplest model is the Turing machine with just one tape and one head, that we will consider in this paper.

From the point of view of computability, this model is equivalent to all others. From the point of view of *complexity*, however, the situation is very different. Indeed, it is well known [6, 5, 19] that a language (of finite words) accepted by such a Turing machine in *linear* time is always regular. More precisely, it can be proven that if such a Turing machine is in time $O(n)$ on all inputs, then there is a constant k so that, on any input, the machine passes at most k times in any given position.

We will consider in this paper the Turing machine as a *dynamical system*: The execution is starting from *any* given configuration c, i.e. any initial state, and any initial tape, and we will observe the evolution. While the Turing machine is a model of computation, it is however quite important in the study of dynamical systems. It was intensively studied by Kurka [11], and Moore [14, 15] proved that they can be embedded in various "classical" dynamical systems. As an example, the uncomputability of the entropy of a Turing machine, by Blondel et al. [2] can be used to deduce the uncomputability of the entropy of piecewise-affine maps, proven by Koiran [10] in a different way.

However, these undecidability results are usually obtained for Turing machines with *two tapes*; The basic idea is to use one tape to simulate a given Turing machine M and to control the other tape, that will only move its head without doing any computation or reading any symbol. The *computational complexity* of the new Turing machine will come from the first tape, but the *dynamical complexity* will come from the second tape.

There is a reason why these results use Turing machines with two tapes. We will prove that some dynamical quantities for one-tape Turing machines are actually *computable*, in

31st Symposium on Theoretical Aspects of Computer Science (STACS'14).
Editors: Ernst W. Mayr and Natacha Portier; pp. 421–432

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


the sense that there is an algorithm that, given any $\epsilon > 0$, produces an approximation of the quantity up to ϵ. The two quantities we consider are the *speed* and the *entropy* of a Turing machine. While the most theoretically important quantity is the entropy, we will focus our discussion in the introduction to the speed, which is easier to conceive.

The speed of a Turing machine measures how fast the head goes to infinity. Informally, the speed is greater than α if we can find a configuration c for which the Turing machine is roughly in position αn after n units of time. Note that if α is nonzero, this means that it takes a time $n/\alpha = O(n)$ to be in position n. Now, if we recall a previous result, one-tape Turing machines with running time $O(n)$ on all inputs recognize only regular languages. We will prove, using the same techniques, that this also applies to the maximum speed: If the maximum speed is nonzero, hence the running time on *some* infinite configuration is (asymptotically) *linear*, then there is a *regular* (ultimately periodic) configuration that achieves this maximum speed.

This paper is organized as follows. In the first section, we introduce the formal definitions of the speed and entropy of a Turing machine. In the next section, we proceed to prove the three main theorems: The speed and the entropy are computable, and the speed is actually a rational number, achieved by a ultimately periodic configuration.

2 Definitions

We assume the reader is familiar with Turing machines. A (one-tape) Turing machine M is a (total) map $\delta_M : Q \times \Sigma \mapsto Q \times \Sigma \times \{-1, 0, 1\}$ where Q is a finite set called the set of states, and Σ a finite alphabet.

Now, the best way to see it as a dynamical system might seem unorthodox at first. The idea is to consider the Turing Machine as having a *moving tape* rather than a moving head: A *configuration* is then an element of $\mathcal{C} = Q \times \Sigma^{\mathbb{Z}}$, and the map M on $\mathcal{C}$ is defined as follows: $M((q, c)) = (q', c')$ where $\delta_M(q, c(0)) = (q', a, v)$, $c'(-v) = a$ and $c'(i) = c(i + v)$ for all $i \neq -v$. This distinction is particularly important for the definition of the entropy to be technically correct. However it is more convenient to consider the Turing machines as we are used to, and we will say "the Turing machine is in position i" rather than "the tape has moved i positions to the right".

The speed

Given a configuration $c \in \mathcal{C}$, the *speed* of M on c is the average number of cells that are read per unit of time. Formally, let $s_n(c)$ be the number of *different* cells read during the first n steps of the evolution of the Turing Machine M on input c. Note that s_n is subadditive: $s_{n+m}(c) \leq s_n(c) + s_m(M^n(c))$.

▶ **Definition 2.1.**

$$\overline{s}(c) = \limsup \frac{s_n(c)}{n}, \qquad \underline{s}(c) = \liminf \frac{s_n(c)}{n}.$$

We give two examples.

- Consider a Turing machine with two states q_1, q_2. On q_1, the Turing machine goes to q_2 without changing the position of the head. On q_2 the Turing machine goes right and changes back to q_1. Then $\overline{s}(c) = \underline{s}(c) = 1/2$ for all c.
- Consider a Turing machine with two states $\{L, R\}$ (for Left and Right) and two symbols $\{a, b\}$. In state q, when the machine reads a symbol a, it goes in the direction q. When the machine reads a symbol b, it writes a symbol a instead and changes direction. On

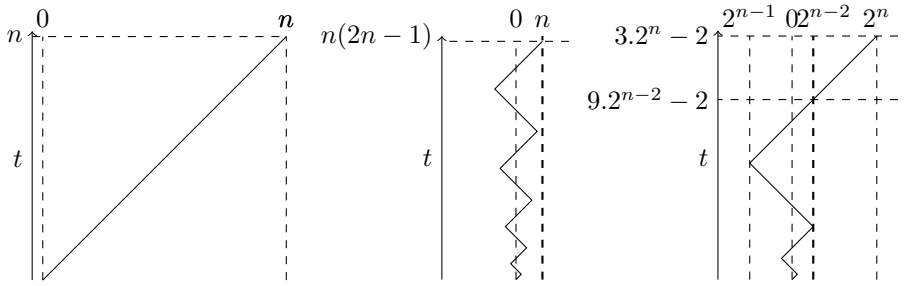

Figure 1 Three different behaviors of the same Turing machine on three different inputs. In the first one, the speed is 1. In the second one the speed is 0. In the third one, the speed is between 1/3 and 1/2. Time goes bottom-up.

input $c = (R, w)$ where w contains only the symbol a, the Turing machine will only go to the right, and $\overline{s}(c) = \underline{s}(c) = 1$. On input $c = (R, w)$ where w contains only the symbol b, the Turing machine will zigzag, and will reach the n-th symbol to the right in time $O(n^2)$, hence will see only $O(\sqrt(n))$ symbols in time n, hence $\overline{s}(c) = \underline{s}(c) = 0$. On input $c = (R, w)$ where w contains b only at all positions $(-2)^i$, the Turing machine will have read (for n even) $2^n + 2^{n-1}$ symbols at time $2^{n+1} + 2^n - 2$ and $\overline{s}(c) = 1/2$, but only $2^{n-1} + 2^{n-2}$ at time $2^{n+1} + 2^{n-2} - 2$, and $\underline{s}(c) = 1/3$. This is illustrated on Figure 1.

Now we define the speed of a Turing machine as the maximum of its average speed on all configurations:

▶ **Definition 2.2.**

$$S(M) = \max_{c\in\mathcal{C}} \underline{s}(c) = \max_{c\in\mathcal{C}} \overline{s}(c) = \lim_n \sup_c \frac{s_n(c)}{n} = \inf_n \sup_c \frac{s_n(c)}{n}\,.$$

The fact that all these definitions are equivalent, and that the maximum speed is indeed a maximum (it is reached by some configuration), is a consequence of the subadditivity of $(s_n)_{n\in\mathbb{N}}$, see [4, Theorem 1.1] or [13] for a more combinatorial proof.

The entropy

The (topological) entropy of a Turing machine is a quantity that measures the complexity of the trajectories. It represents roughly the average number of bits needed to represent the trajectories.

For a configuration c, the *trace* of c is the word $u \in (\Sigma \times Q)^{\mathbb{N}}$ where u_i contains the letter in position 0 of the tape and the state at the i-th step during the execution of M on input c. We note $T(c)$ the trace of c and $T(c)_{|n}$ the first n letters of the trace. Finally, we denote by $T_n = \{T(c)_{|n}, c \in \mathcal{C}\}$

Then the entropy can be defined by

▶ **Definition 2.3.**

$$H(M) = \lim_n \frac{1}{n} \log |T_n| = \inf_n \frac{1}{n} \log |T_n|\,.$$

The limit indeed exists and is equal to the infimum as $(\log |T_n|)_{n\in\mathbb{N}}$ is subadditive. This definition is a specialized version for (moving tape) Turing machines of the general definition of entropy, and was proven equivalent in [16].

We now look again at the examples. In the first case, $T(c)_{|n}$ can take roughly $|\Sigma|^{n/2}$ different values, and $H(M) = 1/2 \log |\Sigma|$. In the second case, the first n letters of $T(c)$ can contain at most $\sqrt{n}$ symbols (b, L) or (b, R) (the maximum is obtained for a configuration with only b). As a consequence, T_n is of size at most $\sum_{i \leq \sqrt{n}} \binom{n}{i} \leq \sqrt{n} \binom{n}{\sqrt{n}}$ so that $H(M) = 0$.

It is possible to give a definition for the entropy that is very similar to the speed. For this, we use *Kolmogorov complexity*. The (prefix-free) Kolmogorov complexity $K(x)$ of a finite word x is roughly speaking the length of the shortest program that outputs x.

A precise definition of Kolmogorov complexity can be found in [3]. Here we just recall a couple of its properties:

- For any alphabet Σ, there exists constants c and c' so that for all words u over Σ, $K(u) \leq |u| \log |\Sigma| + 2 \log |u| + c$ and for all words u, v, $K(uv) \leq K(u) + K(v) + c'$.
- For any computable function f, there exists a constant c so that $K(f(w)) \leq K(w) + c$ whenever $f(w)$ is defined.

For a trace t, define the *lower* and *upper complexity* of t by $\underline{K}(t) = \liminf \frac{K(t_{|n})}{n}$ and $\overline{K}(t) = \limsup \frac{K(t_{|n})}{n}$.

▶ **Theorem 2.4** ([1, 18]). $H(M) = \max_{c \in \mathcal{C}} \underline{K}(T(c)) = \max_{c \in \mathcal{C}} \overline{K}(T(c))$.

From this definition, it will not be surprising that we can obtain results on both speed and entropy using the same arguments.

3 Computability of the speed and the entropy

We will prove in this section that the speed and the entropy of a TM are computable. The proof goes as follows. By the definition of the speed as an infimum, we can compute a sequence s_n so that $S(M) = \inf s_n$. So it is sufficient to find a (computable) sequence s'_n so that $S(M) = \sup s'_n$ to be able to approximate the speed to any given precision ϵ.

To find such a sequence s'_n, it is sufficient to find configurations c_n of near maximal speed. To do that, we need to better understand configurations of maximal speed.

First, we will establish (Propositions 3.1 and 3.2) that a configuration of maximal speed (entropy) cannot do too many zigzags, and must be only finitely many times at any given position. The idea is that revisiting cells that were already visited is a loss of time (and complexity), so the machine should avoid doing it. In the same vein, we can prove that the zigzags must not be too large (Proposition 3.3): the time of the first and last visit of a given cell must be roughly equivalent ($l_n(c) \sim f_n(c)$ in the notation of this proposition).

All this work allows us to redefine the problem as a graph problem: given a weighted (infinite) graph, find the path of minimum average weight (Proposition 3.5). Using the graph approach, we will then prove (Theorem 3.6) that this average minimum weight can be well approximated by considering only *finite* graphs. Finally, the speed and entropy for finite graphs are easy to compute (Theorems 3.7 and 3.9), which ends the proof.

In each section, the proofs will always be done first for the speed, then for the entropy. We deliberately choose to have similar proofs in both cases, to help to understand the proof for the entropy, which is more complex. In particular, some statements about the speed are probably a bit more elaborate than they need to be.

3.1 Biinfinite tapes are no better

The first step in the proof is to simplify the model: we will prove that to achieve the maximum speed (resp. maximum complexity), we only need to consider configurations that never cross the origin, i.e., that stay always on the same side of the tape. This seems quite natural, as changing from a position $i > 0$ to a position $j < 0$ costs at least $i + (-j)$ steps, and might greatly reduce the average speed of the TM on this configuration.

▶ Proposition 3.1. Let c be a configuration for which $S(M) = \lim_n \frac{s_n(c)}{n}$ and suppose $S(M) > 0$. Then, during the computation on input c, the head of M is only finitely many times in any given position i.

Proof. We prove only the result for $i = 0$, the result for all i follows by considering $M^t(c)$ for some suitable t. We suppose by contradiction that the head of M is infinitely often in position 0.

Let k be an integer. As $S(M) > 0$, there must exist a time t for which the head is in position $\pm k$. Let t be the first time when this happens. We may suppose w.l.o.g that at time t the head is in position $+k$. Now let t'_k be the next time the head was in position 0, and finally let t_k be the time at which the head was at its rightmost position in the first t'_k steps.

First, by definition $s_{t_k}(c) = s_{t'_k}(c)$. Furthermore, $t'_k \geq t_k + s_{t_k}(c)/2$. Indeed by definition of t, the leftmost position in the first t_k steps is at most $-(k-1)$ so the TM went further to the right than to the left in the first t_k steps, so that the rightmost position is at least in position $s_{t_k}(c)/2$. Remark also that $t_k \geq k$ (by definition).

From this we obtain

$$\frac{s_{t'_k}}{t'_k} \leq \frac{s_{t_k}}{t_k + s_{t_k}/2} \leq \frac{\frac{s_{t_k}}{t_k}}{1 + \frac{s_{t_k}}{2t_k}}.$$

By taking a limsup on both sides we obtain

$$S(M) \leq \frac{S(M)}{1 + \frac{S(M)}{2}}.$$

A contradiction. ◀

▶ Proposition 3.2. Let c be a configuration for which $H(M) = \lim_n \frac{K(T(c)_{|n})}{n}$ and suppose $H(M) > 0$. Then for any position i, the head of M is only finitely many times in position i.

Proof. It's exactly the same proof. Note that $K(T(c)_{t'_k}) \leq K(T(c)_{t_k}) + O(\log t'_k)$ (The first t'_k bits of $T(c)$ can be recovered if we know only the first t_k bits, and the number of bits we want to recover), and $t'_k \geq t_k + K(T(c)_{t_k})/(2\log|\Sigma|) + O(\log t_k)$ (Indeed $K(T(c)_{t_k}) \leq n\log|\Sigma| + O(\log t_k)$ where $n = s_{t_k}(c)$ is the number of bits read during times $t \leq t_k$, and $t'_k \geq t_k + n/2$), from which we get the same contradiction. ◀

These two propositions state that we only have to deal with configurations that never reach the position $i = 0$ once they leave it at $t = 0$ (replace c by $M^p(c)$ for a suitable p).

If we deal with the disjoint union of the Turing machine M and its mirror (exchange left and right) $\tilde{M}$, we may now assume, and we do in the rest of this section, that the maximum speed and complexity is reached with a configuration that never goes to negative positions $i < 0$ and, if $S(M) > 0$ (resp. $H(M) > 0$), that passes only finitely many times to any given position.

3.2 A reformulation

Recall that we suppose in the following sections that the maximum speed is obtained for a configuration that never goes to negative positions.

Let us call $f_n(c)$ (f for first) the first time the TM reaches position n. Then the average speed on a configuration c (for which the Turing machine never goes in negative positions) can be defined equivalently as $\lim_n \frac{n}{f_n(c)}$. We prove now a stronger statement.

Let us call $l_n(c)$ the *last* time the TM reaches position n. If the TM does not reach position $\pm n$, or if it reaches it infinitely often, let $l_n(c) = \infty$.

▶ Proposition 3.3.

$$S(M) = \max_c \limsup \frac{n}{l_n(c)} = \max_c \liminf \frac{n}{l_n(c)}\,,$$

$$H(M) = \max_c \limsup \frac{K(c_{|n})}{l_n(c)} = \max_c \liminf \frac{K(c_{|n})}{l_n(c)}\,.$$

If the speed (resp. entropy) is nonzero, the maximum is reached for some configuration c for which $l_n(c)$ is never infinite. In particular, for this configuration, $l_n(c) \sim f_n(c)$

Proof. It is clear that $S(M)$ and $H(M)$ are upper bounds, as $n \leq s_{f_n(c)}(c)$ and $K(c_{|n}) \leq K(T(c)_{|f_n(c)}) + O(\log n)$. In particular the result is true if $S(M) = 0$ (resp. $H(M) = 0$).

We first deal with the speed. Let c be a configuration of maximum speed. By the previous subsection, we may suppose that c never reaches negative positions.

Let $t_n = l_n(c)$. Let p be the rightmost position the head reaches before t_n and t'_n the first time this position is reached. Note that $s_{t_n}(c) = s_{t'_n}(c) = p$ (no negative position is ever reached)

From this we get $\lim \frac{t_n}{t'_n} = \lim \frac{t_n}{s_{t_n}(c)} \frac{s_{t'_n}(c)}{t'_n} = 1$.

Note also that $t'_n \geq n$ and $t_n \geq t'_n + s_{t_n} - n$. (The TM is at position $s_{t'_n} = s_{t_n}$ at time t'_n and at position n at time t_n.)

Hence

$$\frac{n}{t_n} \geq \frac{t'_n - t_n}{t_n} + \frac{s_{t_n}}{t_n}\,.$$

From which the result follows.

For the entropy, the proof is almost the same. From $K(T(c)_{t_n}) = K(T(c)_{t'_n}) + O(\log t_n)$, we get again that $\lim_n \frac{t'_n}{t_n} = 1$.

Now $K(T(c)_{t_n}) \leq K(c_n) + (t_n - t'_n)\log|\Sigma| + O(\log t_n)$ (the first t_n bits of $T(c)$ can be recovered if we know t_n and the first p bits of c, hence if we know the first n bits of c and the $p - n \geq t_n - t'_n$ next bits), from which the result follows again. ◀

3.3 Crossing sequences

First denote by $\mathcal{C}^+$ the set of configurations c on which:

- The Turing machine never reaches any positions $i < 0$.
- The Turing machine never reaches the position 0 again once it leaves it at $t = 0$.
- For any $i > 0$, the head of the Turing is only finitely many times in position i.

In the previous section we proved that we only have to deal with configurations in $\mathcal{C}^+$.

The core of the proof is based on *crossing sequences*, introduced by Hennie [6] to obtain complexity lower bounds for one-tape TM.

Let c be a configuration in $\mathcal{C}^+$. The *crossing sequence* at boundary i is the sequence of states of the machine when its head crosses the boundary between the i-th cell and the $i+1$-th cell. We denote by $C_i(c)$ the crossing sequence at boundary i. Note that $C_0(c)$ consists of a single state, which is the initial state of c (the machine never reaches the position 0 anymore) and $C_i(c)$ is finite for $i > 0$.

Crossing sequences have the following property: $C_i(c)$ represents all the exchange of information between the positions $j \leq i$ and the positions $j > i$ of the tape. In particular, if $C_i(c) = C_j(c')$ for two configurations c, c', and if we consider the configuration $\tilde{c}$ that is equal to c up to i then equal to c' (shifted by $i-j$ so that the $j+1$-th cell of c' becomes the $i+1$-th cell of $\tilde{c}$), then the Turing machine on $\tilde{c}$ will behave exactly like c on all positions before i, and as c' (shifted) on positions after i. Hence the crossing sequences capture exactly the behavior of the Turing machine.

The main idea is now that the computation of a Turing machine can be seen as a path on a graph of crossing sequences, where vertices represent crossing sequences and edges link consecutive crossing sequences.

To do this, we now consider the following labeled graph (automaton) G: The vertices of G are all finite words over the alphabet Q (all possible crossing sequences), and there is an edge from w to w' labeled by $a \in \Sigma$ if w and w' are *compatible*, in the sense that it seems possible to find a configuration and a position i so that $C_i(c) = w$, $C_{i+1}(c) = w'$ and a is the letter at position $i+1$ in c (said otherwise, w and w' are two consecutive crossing sequences for some configuration c). The exact definition is as follows. We define recursively two subsets L and R of $Q^* \times Q^* \times \Sigma$ as follows:

- $(\epsilon, \epsilon, a) \in L$, $(\epsilon, \epsilon, a) \in R$
- If $\delta(q_1, a) = (q_2, b, -1)$ then $(q_1 q_2 w, w', a) \in L$ iff $(w, w', b) \in L$
- If $\delta(q_1, a) = (q_2, b, +1)$ then $(q_1 w, q_2 w', a) \in L$ iff $(w, w', b) \in R$
- If $\delta(q_1, a) = (q_2, b, -1)$ then $(q_2 w, q_1 w', a) \in R$ iff $(w, w', b) \in L$
- If $\delta(q_1, a) = (q_2, b, +1)$ then $(w, q_1 q_2 w', a) \in R$ iff $(w, w', b) \in R$

Then there is an edge from w to w' labeled a if and only if $(w, w', a) \in L$.

Note that this echoes a similar definition for two-way finite automata given in [7, 2.6] where $(w, w', a) \in L$ is called "w left-matches w'" (The note in Example 2.15 is particularly relevant). The exact definition above is also hinted at in [17].

Let us explain briefly these conditions. Suppose $\delta(q_1, a) = (q_2, b, +1)$, and suppose that the Turing machine at some point arrives in some cell i from the left, in the state q_1 and sees a. Then by definition, the first symbol from $C_i(c)$ must be q_1. By definition of the local rule δ, the Turing machine will enter state q_2 and go right so that the first symbol in $C_{i+1}(c)$ will be q_2. Now, the next time the Turing machine will come into the cell i, it must be coming from the right, and when it does it will see the symbol b. This explains the rule $(q_1 w, q_2 w', a) \in L$ iff $(w, w', b) \in R$, where w and w' represent the crossing sequences after the second time the Turing machine comes to the cell i.

Now it is clear that a configuration c defines a path in this graph G, and that we can recover the speed of the configuration from the graph, as explained in the following.

A *path* in the graph G is a sequence $p = \{(w_i, u_i)\}_{i<N}$ where w_i is a vertex of G and u_i a letter from Σ so that $(w_i, w_{i+1}, u_i) \in L$ for all $i < N-1$. A *valid* path is an infinite path ($N = \infty$) so that w_0 consists of one single letter (state). We denote by $\mathcal{P}(G)$ the set of valid paths of a graph G.

The following facts are obvious:

▶ Fact 3.4. For any $c \in \mathcal{C}^+$, $\{(C_i(c), c_i)\}_{i \geq 0}$ is a valid path in G.

Furthermore, for any valid path $p = \{(w_i, u_i)\}_{i\geq 0}$, there exists a configuration $c \in \mathcal{C}^+$ so that $u_i = c_i$ and $C_i(c)$ is a prefix of w_i.

Note that it is indeed possible for w_i to be strictly larger than $C_i(c)$.

We are now able to redefine the speed and the complexity based on the graph G. If p is a finite path (N is finite), the *length* of p is $|p| = N$, the *weight* of p is $weight(p) = \sum_{i<N} |w_i|$, and the *complexity* of p is $K(p) = K(u_0 \dots u_{N-1})$

If $p = (u_i, w_i)_{i\geq 0}$ is an infinite path, and $p_{|n} = (u_i, w_i)_{i\leq n}$, the *average speed* of p is $\underline{s}(p) = \liminf \frac{|p_{|n}|}{weight(p_{|n})}$ and the *average complexity* of p is $\underline{K}(p) = \liminf \frac{K(p_{|n})}{weight(p_{|n})}$. We define similarly $\overline{s}(p)$ and $\overline{K}(p)$.

Now note that $\sum_{i<n} |C_i(c)|$ is bounded from below by the first time the TM goes to the position n, and from above by the last time the TM goes to position n. So by the previous section

▶ Proposition 3.5.

$$S(M) = \max_{p\in\mathcal{P}(G)} \underline{s}(p) = \max_{p\in\mathcal{P}(G)} \overline{s}(p)\,,$$

$$H(M) = \max_{p\in\mathcal{P}(G)} \underline{K}(p) = \max_{p\in\mathcal{P}(G)} \overline{K}(p)\,.$$

Now to obtain the main theorems, let G_k be the subgraph of G obtained by taking only the vertices of size $|w_i| \leq k$.

▶ **Theorem 3.6.**

$$S(M) = \sup_k \sup_{p\in\mathcal{P}(G_k)} \overline{s}(p)\,,$$

$$H(M) = \sup_k \sup_{p\in\mathcal{P}(G_k)} \overline{K}(p)\,.$$

This means we only have to consider finite graphs to compute the speed (resp. entropy). We will prove in the next section that the speed and the entropy are computable for finite graphs, which will give the result.

Before going to the proof, some intuition. Let p be a path of maximum speed $S(M) > 0$. For the speed to be nonzero, vertices of large weight cannot be too frequent in p. Now the idea is to *bypass* these vertices (by using other paths in the graph G) to obtain a new path p' with almost the same average speed. For the speed, it's actually possible to obtain a path p' of the *same* speed (this will be done in the next section). However, for the entropy, it is likely that these paths were actually of great complexity so that their removal gives us a path of smaller (yet very near) average complexity.

Proof. First, the speed. One direction is obvious by definition. We suppose that $S(M) > 0$, otherwise the result is trivial. Let p be a path of maximum speed.

Let k be any integer so that $1/k < S(M)$. For any vertex w and w' of size less or equal to k so that p goes through w and w' in that order, choose some finite path $P(w, w')$ from w to w'. Now let K be an upper bound on the weights of all those paths.

The idea is now simple: we will change p so that it will not go through any vertices w of size $|w| > K$: Whenever there is a vertex $\tilde{w}$ of size greater than K, we will look at the last vertex w before it of size less or equal to k, and to the first vertex w' after it of size less or equal to k, and we will replace the portion of this path by $P(w, w')$. Let's call p' this new path. Note that there must exist such a vertex w', otherwise all vertices will be of size greater than k after some time, which means the speed on p is less than $1/k$, a contradiction.

Now we prove this construction works. First, p is of average speed $S(M)$, hence there exists an integer n so that for all $m \geq n$

$$\frac{m}{weight(p_{|m})} \geq S(M)/2\,.$$

Now let m so that the vertex w_m of p is of size less than k. We will look at how the m first positions of p were changed into p'. Let m' be the position of the vertex w_m in p' (w_m still appears in p' as we only change vertices of size greater than k).

By the above inequality, it is clear that in the m first position of the path p, there is at most $2m/(kS(M))$ vertices of size greater than k. All other vertices still appear in p', so that $m' \geq m - 2m/(kS(M))$. Furthermore, at each time, we replace a finite path by a path of smaller weight (each path was of weight at least K, and each new one is of weight at most K).

As a consequence, for this new path p' we have

$$\frac{m'}{weight(p'_{|m'})} \geq \frac{m - 2m/(kS(M))}{weight(p_{|m})}\,.$$

Hence

$$\overline{s}(p') \geq S(M) - 2/k\,.$$

We have proven that some path in G_K is at least $2/k$ to the optimal speed, which proves the result.

The proof for the entropy is, as always, very similar. We start from $1/k < H(M)/(\log|\Sigma|))$, which guarantees that infinitely many vertices are of weight less than k. As before, we will choose K greater than all weights, but now also greater than $k|Q|^{k+1}$.

First, $K(p_{|n}) \leq n\log|\Sigma| + O(\log n)$, so $\underline{K}(p) \leq \underline{s}(p)\log|\Sigma|$, so we may choose n so that for all $m \geq n$

$$\frac{m}{weight(p_{|m})} \geq H(M)/(2\log|\Sigma|)\,.$$

Let $\alpha = 2\log|\Sigma|/H(M)$. With this notation, this implies that for every $m \geq n$, there are at most $m\alpha/k$ (resp. $m\alpha/K$) vertices of size at least k (resp K) in the first m positions of p.

We now have to evaluate $K(p'_{|m'})$. $p_{|m}$ can be recovered from $p'_{|m'}$ by deleting some letters and inserting new ones.

First remark that there are at most $m\alpha/K$ vertices of size at least K in p_m, so we did at most $m\alpha/K$ cuts. In each cut, we inserted at most $|Q|^{k+1}$ letters (the maximal length of a path $P(w,w')$), so we deleted at most $|Q|^{k+1}m\alpha/K \leq m\alpha/k$ letters from $p'_{|m'}$. In particular $m' \leq m(1+\alpha/k)$

We only cut vertices of size at least k, and there are at most $m\alpha/k$ such vertices, so we added at most $m\alpha/k$ letters to $p'_{|m'}$. In particular $m' \geq m(1-\alpha/k)$.

Now the letters we deleted from $p'_{|m'}$ can be encoded into a word over $\{0,1\}$ (specifying which letters we deleted) with at most $m\alpha/k$ symbols "1". For each size $l \leq m\alpha/k$, there are at most $\binom{m'}{m\alpha/k}$ words with l symbols "1", so each such word has complexity at most the logarithm of this number (up to a logarithmic factor to specify l), that is $m'E(\alpha/(k-\alpha)) + o(m)$ where $E(p) = -p\log p - (1-p)\log(1-p)$.

We do the same for the letters we add to p', but we also need to know which letters we had, which can be described by a word of size $m\alpha/k$ and we obtain

$$K(p'_{|m'}) \geq K(p_m) - 2m'E(\alpha/(k-\alpha)) - \lceil m\alpha/k \rceil \lceil \log |\Sigma| \rceil + o(m)\,.$$

Now $weight(p'_{|m'}) \leq weight(p_{|m})$ and $weight(p'_{|m'}) \geq m' \geq m(1-\alpha/k)$:

$$\frac{K(p'_{|m'})}{weight(p'_{|m'})} \geq \frac{K(p_m)}{weight(p_m)} - 2E(\alpha/(k-\alpha)) - \frac{\alpha}{k-\alpha}\lceil \log |\Sigma| \rceil + o(1)\,.$$

Now take the limit (superior) as m tends to infinity:

$$\overline{K}(p') \geq H(M) - 2E(\alpha/(k-\alpha)) - \frac{\alpha}{k-\alpha}\lceil \log |\Sigma| \rceil\,.$$

Now the quantity to the right tends to $H(M)$ when k tends to infinity, which proves the result. ◀

3.4 The main theorems

Now we can explain how to use the last result to prove the main theorems. As hinted above, we only have to be able to compute the speed (and the entropy) from below.

▶ **Theorem 3.7.** *There exists an algorithm that, given a Turing machine M and a precision ϵ, computes $S(M)$ to a precision ϵ.*

Proof. We only have to explain how to compute the maximum speed for a finite graph G. First, we may trim G so that all vertices are reachable from a vertex of size 1. It is then obvious that the maximum speed is obtained by a path that goes to then follow a cycle of minimum average weight, so the maximum speed is exactly the inverse of the minimum average weight. This is easily computable, see [9] for an efficient algorithm. ◀

We can say a bit more

▶ **Theorem 3.8.** *The maximum speed of a Turing machine $S(M)$ is a rational number. It is reached by a configuration which is ultimately periodic.*

Proof. We suppose that $S(M) > 0$ otherwise the result is clear. We will prove that the sequence $\sup_{p \in G_k} \overline{s}(p)$ is stationary. Let $k = 1 + \lceil 1/S(M) \rceil$. Let $K = k(k+1)|Q|^{k+1}$.

Now we look at $\sup_{p \in G_{K'}} \overline{s}(p)$ for some $K' \geq K$. The maximum is reached for some path that reach some cycle of minimum average weight.

Note that this cycle cannot be of length greater than $(k+1)|Q|^{k+1}$. Indeed, denote by m the length of this cycle. As there are at most $|Q|^{k+1}$ vertices in this cycle of length at most k, the average speed on this cycle is less than

$$\frac{m}{(k+1)(m-|Q|^{k+1})} \leq 1/k < S(M)\,.$$

Now, there cannot be any vertices in this cycle of length at least $k(k+1)|Q|^{k+1}$. otherwise the average speed would be less than

$$\frac{(k+1)|Q|^{k+1}}{k(k+1)|Q|^{k+1}} \leq 1/k < S(M)\,.$$

Hence this cycle is already in G_K.

Now if we look at the cycle of minimal average weight in G_K that can be reached in G, hence in G_P from some P, then it is clear that $S(M)$ is exactly the inverse of the average weight of this cycle, and it is reached for some path p in G_P that reaches then follows this cycle. ◀

Note that, while the maximum speed is a rational number, there is no algorithm that actually computes this rational number (we are only able to approximate it up to any given precision). This can be proven by an adaptation of the proof of the undecidability of the existence of a periodic configuration in a Turing machine [8].

Now we do the same for the entropy:

▶ **Theorem 3.9.** *There exists an algorithm that, given a Turing machine M and a precision ϵ, computes $H(M)$ to a precision ϵ.*

Proof. We only have to explain how to compute the maximum complexity for a finite graph G. However, we do not know how to do this in the whole generality. We will only prove how to do it for the graphs G_k, that have an additional property, the *diamond* property: given two vertices w,w' and a word u, there is at most one path from w to w' labeled by u.

First we trim G_k so that any vertex of G_k is reachable from a vertex of size 1.

For a given k, we consider a set B_k of infinite words over the alphabet $(Q \times \Sigma) \cup Q$ defined as follows: A word is in B_k if and only if it does not contain more than $k-1$ consecutive letters in Q, more than 1 consecutive letters in $Q \times \Sigma$, and all factors of the form $(a,q)w(b,q')w'(c,q')$ satisfy than there is a edge from qw to $q'w'$ labeled by b.

Now it it clear that if $p = \{(u_i, q_i w_i)\}_{i \geq 0}$ is an infinite path in G_k, then the word $(u_0,q_0)w_0(u_1,q_1)w_1\dots$ is a word of B_k. Conversely, any word of B_k, up to the deletion of at most $k+1$ letters at its beginning, represents a path in G_k.

Moreover, $K((u_0,q_0)w_0\dots(u_n,q_n)w_n) = K(u_0\dots u_n) + O(1) = K(p_{|n}) + O(1)$. Indeed, we can recover all the states knowing only w_0 and w_n, as the graph has no diamond. Furthermore, the length of $(u_0,q_0)w_0\dots(u_n,q_n)w_n$ is exactly $weight(p_{|n})$.

This means that the maximum complexity on the graph G_k can be computed as:

$$\sup_{w \in B_k} \limsup \frac{K(w_0\dots w_n)}{n}.$$

And we know how to compute this. Indeed, B_k is what is called a subshift of finite type (it is defined by a finite set of forbidden words), for which the above quantity is exactly the entropy (!) of B_k [1, 18], which is easy to compute, see e.g., [12].

To better understand what we did in this theorem, the intuition is as follows: Computing the entropy of the trace is difficult, but the trace can be approximated by taking into account only configurations for which we cross at most k times the frontier between any two consecutive cells. For this approximation T_k of the trace, we can reorder the letters inside the trace so that transitions corresponding to the same position are consecutive, and this does not change the entropy. However, it makes it easier to compute. ◀

Open Problems

From the point of view of dynamical systems, the entropy and the speed (called the maximal Lyapunov exponent) are among the few well known invariants, and thus the result is quite important in this context. However, from the point of view of computer science, it also makes sense to look at the *average* speed, and we are currently trying to compute this number.

An important open problem is to strengthen the last theorem, and actually characterize the exact numbers that can arise as entropies of Turing machines. It cannot be all nonnegative computable numbers, as an enumeration of Turing machines would give us an enumeration of these numbers, which is impossible by an easy diagonalization argument. We have examples showing that the supremum in the theorem is not always reached, which

means it might be possible to obtain different numbers with Turing machines than with finite graphs (which are well known), but the main question remain open.

Finally, the situation for Turing machines with two tapes is not clear. Of course, we know that the speed (resp. entropy) is not computable [2] (there is no algorithm that given a Turing machine and a precision ϵ computes the speed up to ϵ), but we know of no example where the speed (resp. the entropy) is not a computable number.

Acknowledgements. The author thanks the anonymous referees for various comments that led to noticeable improvements to the quality of this article.

References

1 A. A. Brudno. Entropy and the complexity of the trajectories of a dynamical system. *Transactions of the Moscow Mathematical Society*, 44(2):127–151, 1983.

2 Jean-Charles Delvenne and Vincent D. Blondel. Quasi-periodic configurations and undecidable dynamics for tilings, infinite words and Turing machines. *Theoretical Computer Science*, 319:127–143, 2004.

3 Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Theory and Applications of Computability. Springer, 2010.

4 De-Jun Feng and Weng Hang. Lyapunov Spectrum of Asymptotically Sub-additive Potentials. *Communications in Mathematical Physics*, 297(1):1–43, 2010.

5 J. Hartmanis. Computational Complexity of One-Tape Turing Machine Computations. *Journal of the ACM (JACM)*, 15(2):325–339, 1968.

6 F.C. Hennie. One-Tape, Off-Line Turing Machine Computations. *Information and Control*, 8:553–578, 1965.

7 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

8 Jarkko Kari and Nicolas Ollinger. Periodicity and Immortality in Reversible Computing. In *MFCS 2008*, LNCS 5162, pages 419–430, 2008.

9 Richard M. Karp. A Characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978.

10 Pascal Koiran. The Topological Entropy of Iterated Piecewise Affine Maps is Uncomputable. *Discrete Mathematics and Theoretical Computer Science*, 4(2):351–356, 2001.

11 Petr Kurka. On topological dynamics of Turing machines. *Theoretical Computer Science*, 174:203–216, 1997.

12 Douglas A. Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, New York, NY, USA, 1995.

13 László Máté. On infinite composition of affine mappings. *Fundamenta Mathematicae*, 159:85–90, 1999.

14 Cristopher Moore. Generalized one-sided shifts and maps of the interval. *Nonlinearity*, 4(3):727–745, 1991.

15 Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(2):199–230, 1991.

16 Piotr Oprocha. On entropy and Turing machine with moving tape dynamical model. *Nonlinearity*, 19(10):2475–2487, 2006.

17 Giovanni Pighizzini. Nondeterministic one-tape off-line Turing machines and their time complexity. *Journal of Automata, Languages and Combinatorics*, 14(1):107–124, 2009.

18 Stephen G. Simpson. Symbolic Dynamics: Entropy = Dimension = Complexity. accepted for publication in *Theory of Computung Systems*.

19 B. A. Trakhtenbrot. Turing Computations with Logarithmic Delay. *Algebra i Logika*, 3(4):33–48, 1964. (In russian).