Fractal Dimension Versus Computational Complexity

Joost J. Joosten[*1,4], Fernando Soler-Toscano[†2,4], and Hector Zenil[‡3,4]

[1]Departamento de Lògica, Història i Filosofia de la Ciéncia, Universitat de Barcelona, Barcelona, Spain.
[2]Departamento de Filosofía, Lógica y Filosofía de la Ciencia, Universidad de Sevilla, Seville, Spain.
[3]Unit of Computational Medicine, Center for Molecular Medicine, Karolinska Institutet, Stockholm, Sweden.
[4]Algorithmic Nature Group, LABORES, Paris, France.

Abstract

In this paper we look at small Turing machines (TMs) that work with just two colors (alphabet symbols) and either two or three states. For any particular such machine τ and any particular input x we consider what we call the *space-time* diagram which basically is just the collection of consecutive tape configurations of the computation $\tau(x)$. In our setting it makes sense to define a fractal dimension for a Turing machine as the limiting fractal dimension for the corresponding space-time diagrams. It turns out that there is a very strong relation between the fractal dimension of a Turing machine of the above specified type and its runtime complexity. In particular, a TM with three states and two colors runs in at most linear time iff its dimension is 2, and its dimension is 1 iff it runs in super-polynomial time and it uses polynomial space. If a TM runs in time $\mathcal{O}(x^n)$ we have empirically verified that the corresponding dimension is $\frac{n+1}{n}$, a result that we can only partially prove. We find the results presented here remarkable because they relate two completely different complexity measures: the geometrical fractal dimension on the one side versus the time complexity of a computation on the other side.

Keywords: small Turing machines, Fractal complexity, Hausdorff dimension, Box dimension, space/time complexity, computational complexity.

---

[*]jjoosten@phil.uu.nl

[†]fsoler@us.es

[‡]hector.zenil@algorithmicnaturelab.org

Contents

Part I: Theoretical setting

In the first part of the paper, we shall define the basic notions we work with. In particular, we shall fix on a computational model: small Turing machines with a one-way infinite tape. For these machines, we will define so-called *space-time* diagrams which are a representation of the memory state throughout time. For these diagrams we shall define a notion of fractal dimension. Next, some theoretical results are proven about this dimension.

1 Introduction

Complexity measures are designed as to capture complex behavior and quantify *how* complex that particular behavior is. If a certain phenomenon is genuinely complex this means that it does not all of a sudden becomes simple by just translating the phenomenon to a different setting or framework.

It is in this sense that we expect different good complexity measures from possibly entirely different fields to be related to each other. Now, it would not be realistic to always expect a one-to-one relation between complex phenomena in one field and complex phenomena in another field: if not both fields would be mere translations of each other.

The current paper presents a relation between two rather different complexity measures. On the one hand we have the geometrical framework. Within that framework we measure the complexity of spatial objects by their fractal dimension. On the other hand we have the framework of computational complexity. Here, the complexity of algorithms is measured by the amount of time and memory they take to be executed.

The relation we have between both frameworks is as follows. We start in the framework of computations and algorithms and for simplicity assume that they can be modeled as using discrete time steps. Now, suppose we have some computer τ that performs a certain task $\tau(x)$ on input x. We can assign a spatial object to the computation corresponding to $\tau(x)$ as follows.

We look at the spatial representation σ_0 of the memory when τ starts on input x. Next we look at σ_1: the spatial representation of the memory after one step in the computation and so forth for $\sigma_2, \sigma_3, \ldots$. Then we 'glue' these spatial objects together into one big object $\Sigma(\tau, x)$ in a simple way by just putting the one next to the other: $\langle \sigma_0, \sigma_1, \sigma_2 \ldots \rangle$. Each σ_i can be seen as a slice of $\Sigma(\tau, x)$ of the memory at one particular time i in the computation. This is why we call $\Sigma(\tau, x)$ the space-time diagram of $\tau(x)$. It is of these spatial objects and in particular the limit for x going to infinity that we can sometimes compute or estimate the fractal dimension $d(\tau)$.

One can set this up in such a way that $d(\tau)$ becomes a well defined quantity. Thus, we have a translation from the computational framework to the geometrical framework. Next, one can then investigate the relation between these two frameworks, and in particular, if complex algorithms (in terms of time and space complexity) get translated to complex (in the sense of fractal dimension)

space-time diagrams.

It is this main question that is being investigated in this paper. The computational model that we choose is that of Turing machines. In particular we look at small one-way infinite Turing machines (TMs) with just two or three states and a binary tape alphabet.

For these particular machines we define a notion of dimension along the lines sketched above. In exhaustive computer experiments we compute the dimensions of all machines with at most three states. Among the various relations that we uncover is that such a TM runs in at most linear time iff the corresponding dimension is 2. Likewise, if a TM (in general) runs in super-polynomial time and uses polynomial space, we see that the corresponding dimension is 1.

Admittedly, the way in which fractal geometry measures complexity is not entirely clear and one could even sustain the view that fractal geometry measures something entirely else. Nonetheless, dimension is clearly related to degrees of freedom and as such related to an amount of information storage.

In [46] space-time diagrams of Turing machines and one-dimensional cellular automata were investigated in the context of algorithmic information theory. Notably an uncompressibility test on the space-time diagrams led to a classification of the behaviour of CAs and TMs thereby identifying non-trivial behaviour [50]. The same type of space-time diagrams were also investigated in connection to two other seminal measures of complexity [36, 51, 2] connected to Kolmogorov complexity, namely Solomonoff's algorithmic probability [50, 48] and Bennett's logical depth [11, 47].

The results presented in this paper were found by computer experiments and proven in part. To the best of our knowledge it is the first time that a relation is studied between computational complexity and fractal geometry, of a nature as presented here.

Outline: The current paper naturally falls apart into three parts. In the first part (Sections 2—4) we define the ideas and concepts and prove various theoretical results. In the second part, Sections 5—6, we describe our experiment and its results to investigate those cases where non of our theoretical results would apply. Finally in the third part, we present a literature study where we mention various results that link fractal dimension to other complexity notions.

More in detail: In Section 2 we describe the kind of TMs we shall work with. This paper can be seen as part of a larger project where the authors mine and study the space of small TMs. As such, various previous results and data could be re-used in this paper and in Section 2 we give an adequate description of these used data and results.

In Section 3 we revisit the box-counting dimension and define a suitable similar notion of fractal dimension $d(\tau)$ for TMs τ. We prove that $d(\tau) = 2$ in case τ runs in time at most linear in the size of the input. Next, in Section 4 we prove an upper and a lower bound for the dimension of Turing machines. The Upper Bound Conjecture is formulated to the effect that the proven upper bound is actually always attained. For special cases this can be proved. Moreover, under some additional assumptions this can also be proven in general. In our

4

experiment we test if in our test-space the sufficient additional assumptions were also necessary ones and they turn out to be so.

Section 5 describes how we performed the experiment, what difficulties we encountered, how they were overcome, and also some preliminary findings are given. The main findings are presented in Section 6.

We conclude the paper with Section 7 where we present various results from the literature that link different notions of complexity to put our results within this panorama.

2   The small Turing Machine database

As mentioned before, this paper forms part of a larger project where the authors exhaustively mine and investigate the collection of small Turing machines. In this section, we shall briefly describe the raw data that was used for the experiments in this paper and refer for details to the relevant sources.

2.1   Small Turing machines

A TM can be conceived both as a computational device and as a dynamical system. In our studies a TM is represented by a *head* moving over a *tape* consisting of discrete *tape-cells* where the tape extends infinitely in one direction. In our pictures and diagrams we will mostly depict the tape as extending infinitely to the left. Each tape cell can contain a symbol from an *alphabet*. Instead of symbols we speak of *colors* and in the current paper we shall work with just two colors: black and white.

The head of a TM can be in various *states* as it moves over the cells of the tape. We shall refer to the collection of TMs that use n states and k symbols/colors as the (n, k)-space of TMs. We shall always enumerate the states from 1 to n and the colors from 0 to $k - 1$. In this paper we work with just two symbols so that we represent a cell containing a 0 with a white cell, and a cell containing a 1 with a black cell.

A computation of a TM proceeds in discrete time-steps. The tape content at the start of the computation is called the *input*. By definition, our TMs will always start with the head at the position of the first tape cell, that is, the tape cell next to the edge of the tape; In our pictures this is normally the right-most tape. Moreover, by definition, our TMs will always commence their computation in the default start state 1.

A TM τ in (n, k) space is completely specified by its *transition table*. This table tells what *action* the head should perform when it is in State $1 \leq j \leq n$ at some tape cell c and reads there some symbol $0 \leq i < k$. Such an action in turn consists of three aspects: changing to some some state (possibly the same one); the head moving either one cell left or one cell right, but never staying still; writing some symbol at c (possibly the same symbol as was there before). Consequently, each (n, k)-space consists of $(2 \cdot n \cdot k)^{n \cdot k}$ many different TMs. We

number these machines according to Wolfram's enumeration scheme ([45], [19]) which is similar to the lexicographical enumeration.

Clearly, each TM in (n, k) space is also present in (m, k) space for $m \geq n$, by just not using the extra states since they are 'inaccessible' from State 1. Many rules in a (n, k) space are trivially equivalent in the computational sense up to a simple transformation of the underlying geometry, for example, by relabeling states by reflection or complementation, hence for all purposes identical. In the literature machines that have equivalents are sometimes called *amphichiral*, we will sometimes refer to them as machine *twins*.

We say that a TM *halts* when the head "falls off" the tape on the right-hand side; in other words, when the head is at the right-most position and receives an instruction to move right. The tape configuration upon termination of a computation is called the *output*.

We shall refer to the input consisting of the first m tape cells by black on an otherwise white tape as the input m. (This is in slight discrepancy with the convention in [23].) In this context, a *function* is a map sending an input m to some output tape configuration. We call the function where the output is always identical to the input the *tape identity* function.

By Rice's Theorem it is in principle undecidable if two TMs compute the same function. Nonetheless, for spaces $(n, 2)$ with n small, no universal computation is yet present ([33, 28]). In [23] the authors completely classify the TMs in (3,2) space among the functions they compute, taking pragmatic approaches that possibly produce small errors to deal with undecidability and unfeasibility issues.

2.2   Space-time diagrams

As mentioned in the introduction, in this paper a central role is played by so-called *space-time diagrams*. A space-time diagram for some computation is nothing more but the joint collection of consecutive memory configurations. We have included a picture of space-time diagrams for a particular TM for inputs 1 to 14 in Figure 1.

Since these space time diagrams are such a central notion to this paper, let us briefly comment on Figure 1. The top-row of each of these fourteen diagrams always represents the input tape configuration of the TM. We have chosen to depict the space-time diagrams of our TM on inputs 1 to 14. The rightmost cell in the diagram is actually not representing a tape cell. Rather it represents the end of the tape so that we depict it with a different color/grey-tone.

Remember that the computation starts with the head of the TM in State 1 in the rightmost cell. Each lower row represents the tape configuration of a next step in the computation. So, there can at most be one cell of different color between two adjacent rows in a space-time diagram. We see that this particular (2,2) TM with number 346 first moves over the tape input erasing it. Then it gradually moves back to the edge of the tape writing alternatingly black and white cells to eventually fall of the tape, whence it terminates.
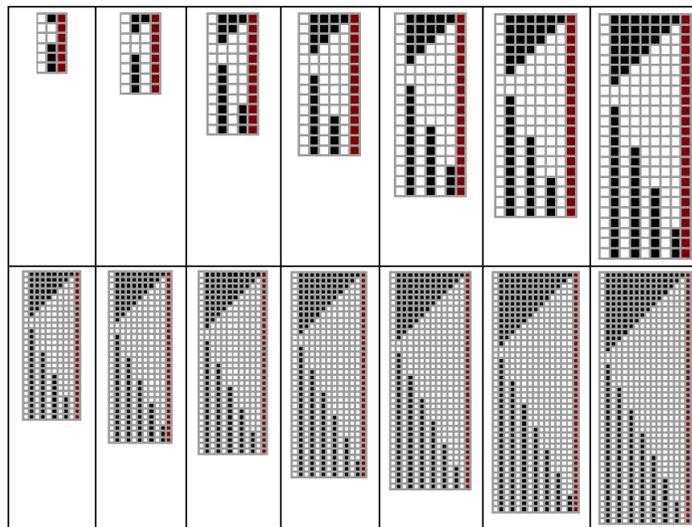
Figure 1: The figure shows a sequence of space-time diagrams corresponding to the TM in (2,2) space with number 346 (according to Wolfram's enumeration scheme [45, 19] for (2,2) space) on inputs 1 up to 14.

Clearly these space-time diagrams define spatial objects by focussing on the black cells. It is of these spatial objects that we wish to measure the geometrical complexity. Subsequently, we wish to see if there is a relation between this geometrical complexity and the computational complexity (space or time usage) of the TM in question.

In the next section we shall see how to assign a measure of geometrical complexity to these space-time diagrams and call this measure the *dimension* of the TM. Various relations between computational complexity of a TM on the one hand and its dimension on the other hand can be proven. Other relations shall be investigated via experiments.

2.3 On our coding convention

Note that for this paper it is entirely irrelevant how to numerically interpret the output tape configuration whence we shall refrain from giving such an interpretation. However, it has been a restrictive choice to represent our input in a unary way. That is to say, the notion of a function in our context only looks at a very restricted class of possible inputs: blocks of n consecutive black cells for $n > 0$. The main reason why we do this is that if we do not do this, our functions all behave in a very awkward and highly undesirable way. In [23] this undesirable behavior is explained in the so-called *Strips Theorem*.

Basically, the Strips Theorem boils down to the following. Let us consider a TM τ in $(n, 2)$ space on input x and suppose $\tau(x)$ is a terminating computation.

If we number the cells on the tape by their distance to the edge, let i be largest cell-number that is visited in the computation of $\tau(x)$. Clearly, any tape input that is equal to x on the first i cells but possibly different on the cells beyond i, will perform exactly the same computation and in this sense in is input-independent.

We have chosen our input-output convention in such a way as to prevent the Strips-Theorem. There are two undesirable side effects of our coding. Firstly, it is clear that any TM that runs in less than linear time, actually runs in constant time. Secondly, the thus defined functions are very fast growing if we were to represent the output in binary. In particular, the tape identity represents an exponentially fast growing numerical function in this way.

A positive feature of our input convention is that the amount of symmetry present in the input coding facilitates various types of analysis and in particular automated function-completion seems to run more smoothly.

We shall here describe an alternative way of representing the input a and denote the representation by $\rho(a)$. This representation $\rho(a)$ shall be such that it avoids the Strips Theorem yet does not intrinsically entail exponential growth of the tape identity and similar functions in case we would interpret our output configuration in binary. Although we do not use, nor explore the alternative input coding we find it worth mentioning here and hope that future investigations can take up the new coding.

In order to represent the input a according to ρ, we first write the input a in binary as $\sum_{n=0}^{\infty} a_n 2^n$ with all but finitely many $a_n = 0$. Let us denote the cells on the tape by $c_0, c_1, c_2, \ldots$. Here c_0 is the cell at the edge, c_1 the cell immediately next to it, etc. For $a \neq 0$, let $k = \lfloor \log_2(a) \rfloor + 1$ and $k = 1$ otherwise. That is, $k + 1$ is the number of digits in the binary expansion of a.

For each $0 \leq i \leq k$ we shall represent a_i in $c_{2 \cdot i}$ in the canonical way: we set $c_{2 \cdot i}$ to be one/black whenever $a_i = 1$ and we set $c_{2 \cdot i}$ to be zero/white otherwise. Moreover, we set all odd-labeled cells $c_{2 \cdot i + 1}$ to be zero with the sole exception at cell $2k + 1$ that we define to be one/black.

It is clear that ρ avoids the Strips Theorem. Moreover, if we were to interpret the output as binary, the tape identity defines a function whose growth rate is only in the order of $x \mapsto x^2$.

3   Fractal dimensions

In this section we shall briefly recall the definition of and ideas behind the box-counting dimension which is a particular fractal dimension having the better computational properties whence better suited for applications. In Section 7 we relate the box-dimension to various other notions of fractal dimension and in particular to the well-known Hausdorff dimension.

After revisiting the notion of box-counting dimension, we see how to apply these ideas to Turing machines and their space-time diagrams.

3.1 Box dimension

We shall use the notion of Box dimension. This notion of fractal dimension can be seen as a simplification of the well-known Hausdorff dimension (see [16] and our survey section, Section 7). The Hausdorff dimension is mathematically speaking more robust than the Box dimension. However, the Box-dimension is easier to compute and is known to coincide with the Hausdorff dimension in various situations.

Let us briefly recall the definition of the Box dimension and the main ideas behind it. The intuition is as follows. Suppose we have a mathematical object S of bounded size whose "volume" $V(S)$ we wish to estimate. For example, let us work with a space $\mathbb{R}^n$ that has dimension n large enough as to embed our object S. The idea now is to cover the object S by boxes in $\mathbb{R}^n$ and estimate the "volume" $V(S)$ of S as function of the total number of boxes $N(S)$ needed to cover S. Clearly, the number of boxes $N(S)$ needed to cover S depends on the size of the boxes used. Therefore, in the analysis we will take along the parameter r which denotes the length of the edge of the boxes used, and we will write the number of boxes needed to cover S as $N(S, r)$.

If S is a line, which is a one-dimensional object, the corresponding notion of "volume" $V(S)$ is just the length of the line segment. To estimate the length $V(S)$ we clearly have

$$V(S) = \lim_{r\downarrow 0} \quad rN(S, r)$$

if this is well defined.

If S is a plane, or more in general a two-dimensional manifold, the corresponding notion of "volume" $V(S)$ is just the surface of the plane/manifold segment for which we have

$$V(S) = \lim_{r\downarrow 0} \quad r^2 N(S, r)$$

if well defined.

Likewise, for a three-dimensional object, to estimate its volume we would have

$$V(S) = \lim_{r\downarrow 0} \quad r^3 N(S, r)$$

if well defined, and in general for an d-dimensional object we would obtain

$$V(S) = \lim_{r\downarrow 0} \quad r^d N(S, r). \tag{1}$$

The idea behind the definition of the Box dimension is to take (1) as a *defining* equation of dimension if this makes sense mathematically speaking.

Thus, solving for d in (1) we obtain:

$$
\begin{aligned}
V(S) = \lim_{r\downarrow 0} \quad & r^d N(S,r) && \Longrightarrow \\
V(S) = \lim_{r\downarrow 0} \quad & \left(d\log(r) + \log(N(S,r)) \right) && \Longrightarrow \\
\lim_{r\downarrow 0} \quad & \left(d\log(r) + \log(N(S,r)) - V(S) \right) = 0 && \Longrightarrow \\
\lim_{r\downarrow 0} \quad & \left(d + \frac{\log(N(S,r))}{\log(r)} - \frac{V(S)}{\log(r)} \right) = 0 && \Longrightarrow \\
d \quad = \lim_{r\downarrow 0} \quad & \left(-\frac{\log(N(S,r))}{\log(r)} + \frac{V(S)}{\log(r)} \right) \\
= \lim_{r\downarrow 0} \quad & \left(\frac{\log(N(S,r))}{\log(\frac{1}{r})} + \frac{V(S)}{\log(r)} \right) \\
= \lim_{r\downarrow 0} \quad & \frac{\log(N(S,r))}{\log(\frac{1}{r})}.
\end{aligned}
$$

The last equality is justified in case S is bounded as by assumption and $V(S)$ is finite so that $\lim_{r\downarrow 0} \frac{V(S)}{\log(r)} = 0$.

The above reflections form the main ideas behind the definition of Box dimension that we shall use in this paper.

Definition 3.1 (Box dimension)**.** *Let S be some spatial object that can be embedded in some $\mathbb{R}^n$, let $N(S,r)$ denote the minimal number of boxes of size r needed to fully cover S. The Box dimension of S is denoted by $\delta(S)$ and is defined by*

$$
\delta(S) \quad := \quad \lim_{r\downarrow 0} \quad \frac{\log(N(S,r))}{\log(\frac{1}{r})}
$$

in case this limit is well defined. In all other cases we shall say that $\delta(S)$ is undefined.

3.2 Box dimension for Space-Time diagrams

Let us see how we can adapt the notion of Box dimension to our space-time diagrams. The spatial figure S that we wish to measure will be defined by the black cells in the space-time diagram. Clearly, for each particular input on which the TM halts the corresponding space-time diagram is finite and has dimension $d(S) = 2$: each black cell defines a piece of surface. It gets interesting when we consider limiting behavior of the TM on larger and larger inputs.

A first attempt

Let τ be some TM and let $S(\tau, x)$ denote the space-time diagram corresponding to TM τ on input x if this is well-defined, that is, if τ eventually halts on input x which we shall denote by $\tau(x) \downarrow$. The question is, what is a sensible way to define the dimension $d(\tau)$ of our TM τ? It does not make much sense to define $d(\tau) = \lim_{x\to\infty} \delta(S(\tau, x))$ for a couple of reasons.

Firstly, τ might diverge on various inputs. We can easily bypass that by tacitly understanding $\lim_{x\to\infty}$ as x getting larger and larger among those x for which $\tau(x) \downarrow$ demanding that there are infinitely many such x. In case there are just finitely many x on which τ converges, we could say that $d(\tau)$ is undefined.

The second objection is more serious. As for each x with $\tau(x) \downarrow$ we have that $\delta(S(\tau, x)) = 2$ we see that all limits converge to the value 2 if they are well defined. This of course is highly undesirable. We can overcome this objection by scaling the length of each $S(\tau, x)$ to some figure $\mathsf{scale}(S(\tau, x))$ whose length has unit size. Thus, the black areas in $\mathsf{scale}(S(\tau, x))$ become more and more fine-grained so that it seems to make sense to define $d(\tau) = \delta\Big(\lim_{x \to \infty} \mathsf{scale}(S(\tau, x))\Big)$.

A second attempt and formal definition

The new candidate $d(\tau) = \delta\Big(\lim_{x \to \infty} \mathsf{scale}(S(\tau, x))\Big)$ has many good properties. However, for this new candidate we again see two main objections.

The first objection is that $\lim_{x \to \infty} \mathsf{scale}(S(\tau, x))$ need not exist at all and stronger still, is likely not to be well defined in most cases. We could try to remedy this by working with with subsequences for which the limit is defined but it all seems very hairy.

The second objection is that this new definition seems hard to numerically approximate at first glance. We shall see how to overcome the second objection which will yield us automatically a solution to the first objection.

As we mentioned before, we cannot first approximate $\lim_{x \to \infty} \mathsf{scale}(S(\tau, x))$ and then compute the corresponding δ as this would always yield the answer 2. However, what we *can* do is simultaneously approximate both δ and $\lim_{x \to \infty} \mathsf{scale}(S(\tau, x))$. There is a lot of choice in how we approximate and in how fast we approximate δ and how fast we approximate $\lim_{x \to \infty} \mathsf{scale}(S(\tau, x))$ relatively to the approximation of δ.

There seems to be a canonical choice though. The approximation of the dimension δ is dependent on the size r of the boxes. It seems very natural to take the size of our boxes to be exactly the size of one tape-cell. The size of one tape-cell is naturally determined by $\mathsf{scale}(S(\tau, x))$. Let us determine r as dictated by $\mathsf{scale}(S(\tau, x))$. In order to facilitate our discussion we first fix some notation.

Definition 3.2. *For τ a TM and x an input so that $\tau(x)\downarrow$, we denote by $t(\tau, x)$ the amount of time-steps τ needed on input to terminate. Likewise, $s(\tau, x)$ denotes the amount of space-cells used by the computation of τ on input x. Thus, $s(\tau, x)$ measures the distance between the edge of the tape and the furthest tape cell visited by the head during the computation.*

We shall sometimes write $t_\tau(x)$ or even just $t(x)$ if the context allows us to, and similarly for the space-usage function $s(\tau, x)$.

By the nature of our input-output protocol, there exist no TMs whose runtime is sub-linear but not constant. Let us first concentrate on the TMs that run in at least linear time and deal with the constant time TMs later. If a TM halts in non-constant time, the least it should do is read all the input, do some calculations and then go back to the beginning of the tape. Thus, clearly

$t(\tau, x) > s(\tau, x)$, whence the scaling of the figure $S(\tau, x)$ is best done by resizing the runtime to get length 1. Consequently, the size of r scales to $r = \frac{1}{t(\tau, x)}$.

Recall that $N(\Sigma, r)$ denotes the minimal number of boxes of size r needed to cover the spatial object Σ. Now that we have determined the size of r, we can write $N(\tau, x)$ instead of $N(S(\tau, x), \frac{1}{t(\tau, x)})$ and it is clear that $N(\tau, x)$ is just the number of black cells in the space time diagram of τ on input x. Thus, the second attempt of defining $d(\tau)$ then translates to

$$d(\tau) := \lim_{x \to \infty} \frac{\log\big(N(\tau, x)\big)}{\log\big(t(\tau, x)\big)}. \tag{2}$$

In this definition we could address the issue of undefinedness by replacing lim by lim sup or lim inf. Notwithstanding the theoretical correctness of this move, it seems hardly possible to sensibly compute the lim sup or lim inf in the general setting.

In the current paper however, we have only considered TMs with either two or three states and just two colors. It turned out that in this setting we could determine both lim inf and lim sup. In all cases that we witnessed where the limit (2) outright was not well-defined, we were able to identify different subsequences where the limit (2) did converge so that we could choose to either go with the lim sup or with the lim inf. It turns out that in general, the lower the dimension the more interesting the corresponding TM so that we decided to work with lim inf.

For TMs with constant runtime, we know that only a constant number of cells will be visited and possibly changed color. For these TMs the figure $S(\tau, x)$ can only be sensibly scaled by using the input size. By doing so, we see that in the limit we just get a black line whose dimension is clearly equal to one. However, as we consider constant runtime TMs as a degenerate case so to say, we shall for convenience define the dimension of such machines to be equal to 2. We do so in order to have them more like linear time TMs (see Lemma 3.4). All these considerations and reflections lead us to the following definition.

Definition 3.3 (Box dimension of a Turing machine). *Let τ be a TM that converges on infinitely many input values x. In case $\tau(x) \downarrow$, let $N(\tau, x)$ denote the number of black cells in the space-time diagram of τ on input x and let $t(\tau, x)$ denote the number of steps needed for τ to halt on x.*

We will define the Box dimension of a TM τ and denote it by $d(\tau)$. In case $t(\tau, x)$ is constant from some x onwards, we define $d(\tau) := 2$. Otherwise, we define

$$d(\tau) := \liminf_{x \to \infty} \frac{\log\big(N(\tau, x)\big)}{\log\big(t(\tau, x)\big)}.$$

Note that our definition of dimension can readily be generalized to non-terminating computations. Also, restricting to computational models with discrete time steps is not strictly necessary.

3.3 Linear time Turing machines

For certain TMs τ, we can actually compute their Box dimension. Let us reconsider TM 346 again whose space-time diagrams were displayed in Figure 1. Due to the extreme regularity in the space-time diagrams we can see that TM 346 runs in linear time. That is to say, linear in the length of the representation of the input.

Thus, after scaling each space-time diagram so that the vertical time-axis is rescaled to 1, we will always have a little surface in the shape of a black triangle in the scaled space-time diagram. The Box dimension of a triangle is of course 2. We may conclude that $d(2, 2\text{-TM } 346) = 2$. Of course the only important feature used here is the linear-time performance of 2,2-TM 346. We can summarize this observation in a lemma.

Lemma 3.4. *Let τ be a TM that runs at most linear time. Then, $d(\tau) = 2$.*

Proof. We fix some TM τ that runs at most linear time. Our input/output convention is such that τ is either constant time or τ is linear time. In case τ runs in constant time, we have that $d(\tau) = 2$ by definition.

Let us consider the case that τ runs in linear time. It must be the case that the head goes all the way to the end of the tape input, if not, τ would run in constant time from some input y onwards. Input x is represented by $x + 1$ consecutive black cells. In the worst case (the fewest amount of black cells), at all the first steps the input is erased and replaced by a white cell as is the case in Figure 1.

However, τ runs in linear time, say, for any x, the machine τ runs at most $a \cdot (x + 1)$ many steps with $2 \leq a < \infty$. After scaling, the input will get size $\frac{x+1}{a \cdot (x+1)} = \frac{1}{a}$. Thus in the worst case, the upper triangle has size $\frac{1}{2a^2}$ which is independent on x whence non-vanishing. Clearly, the Box dimension of a triangle of whatever size equals 2. Thus $d(\tau) = 2$ as was what we wanted to see. $\qquad\square$

4 The Space-Time Theorem and applications

Above we saw that for linear time TMs we can actually compute the corresponding dimension. However, for non-linear TMs we can only prove an upper bound on the Box dimension.

4.1 The Space-time Theorem: an upper bound

Theorem 4.1 (Space-time Theorem)**.** *Let us, for a given TM τ, denote by $s(x)$ the amount of cells visited by τ on input x, and let $t(x)$ denote the amount of computation steps it took τ to terminate on input x.*

$$\textit{If } \liminf_{x \to \infty} \ \frac{\log(s(x))}{\log(t(x))} = n \ \textit{ then } d(\tau) \leq 1 + n.$$

Proof. The box dimension is maximal in case all cells under consideration are black. This number is bounded above by $s \cdot t$. Plugging this in the definition of $d(\tau)$ gives us our result:

$$
\begin{aligned}
d(\tau) \;\; &= \;\; \liminf_{x \to \infty} \frac{\log \left(N(\tau, x) \right)}{\log \left(t(\tau, x) \right)} \\
&\leq \;\; \liminf_{x \to \infty} \frac{\log \left(s(x) \cdot t(x) \right)}{\log \left(t(x) \right)} \\
&\leq \;\; \liminf_{x \to \infty} \frac{\log \left(s(x) \right) + \log \left(t(x) \right)}{\log \left(t(x) \right)} \\
&\leq \;\; 1 + \liminf_{x \to \infty} \frac{\log \left(s(x) \right)}{\log \left(t(x) \right)} \\
&\leq \;\; 1 + n.
\end{aligned}
$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\square$

As we shall see, in all cases the upper bound given by the Space-time Theorem is actually attained in our experiment. It is unknown however, if it holds in general.

4.2 A lower bound

The best lower bound that we can currently prove is $d(\tau) \geq 1$ for each Turing machine τ for which $\lim_{x \to \infty} \frac{s(\tau, x)}{t(\tau, x)} = 0$. Before we can prove this we first need a technical lemma.

Lemma 4.2. *Let $f, g : \mathbb{N} \to \mathbb{N}$ be given so that $\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$. Then, we also have that $\lim_{x \to \infty} \frac{\log \left(f(x) \right)}{\log \left(g(x) \right)} = 1$.*

Proof. It suffices to show that for any $\epsilon > 0$ we can find x_0 so that for $x \geq x_0$ we have

$$
\log \left(g(x) \right) - \epsilon < \log \left(f(x) \right) < \log \left(g(x) \right) + \epsilon. \tag{3}
$$

By the assumption that $\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$ we see that for any $\delta > 0$ we can find x_1 so that for $x \geq x_1$ we have $|1 - \frac{f(x)}{g(x)}| < \delta$ whence, in particular

$$
g(x) \cdot (1 - \delta) < f(x) < g(x) \cdot (1 + \delta). \tag{4}
$$

Consequently, $\log \left(g(x) \right) + \log(1 - \delta) < \log \left(f(x) \right) < \log \left(g(x) \right) + \log(1 + \delta)$. Since log is continuous around 1, we can choose δ small enough so that $-\log(1 - \delta), \log(1 + \delta) < \epsilon$. Evidently, for the x_0 corresponding to this particular δ we have indeed (3) as was to be proven. $\qquad\qquad\qquad\square$

Clearly, Lemma 4.2 can be stated and proved in much more general terms. However, it is only this restricted version that we need to prove our lower bound.

Lemma 4.3. *For each TM τ with $\lim_{x \to \infty} \frac{s(\tau, x)}{t(\tau, x)} = 0$, we have that $d(\tau) \geq 1$.*

Proof. Let n be the number of states of the TM τ. We will see that at many time steps, the tape is not entirely blank.

Let us fix a TM τ and assume that

$$\lim_{x \to \infty} \frac{s(x)}{t(x)} = 0. \tag{5}$$

We claim that in the first $t(x) - n \cdot s(x)$ computation steps each consecutive block of n tapes in the space-time diagram contains a tape that is not entirely blank.

For a contradiction, suppose that there were a block of n completely blank tapes in succession in this part of the space-time diagram. But then, since there are only n many states the machine τ would enter in a loop. That is to say, after $m \leq n$ many steps, the head would be in the same state with exactly the same tape and would repeat behavior.

Clearly, if the head did not make a net movement to to the right (recall that we say that the end of the tape is on the right-hand side) during these m many steps, this would contradict the fact that τ halts on input x.

In the worst case, τ just makes a net move of 1 cell to the right during each leap of n many time-steps (when $m = n$). This shift to the right by τ on input x started in the worst case at cell $s(x)$, that is, the cell which is furthest removed from the edge of the tape but which is still visited by τ on input x. Then, after $n \cdot s(x)$ many time steps, the head of τ would drop off the tape contradicting our assumption that the entire block of n consecutive white tapes occurred within the first $t(x) - n \cdot s(x)$ computation steps.

This concludes the proof of our claim that each consecutive block of n tapes in the first $t(x) - n \cdot s(x)$ computation steps contains at least one none-blank tape. A non-blank tape contains at least one black cell, so a very crude lower-bound on the number of black cells is $N(\tau, x) \geq \frac{t(x) - n \cdot s(x)}{n}$. If we plug this into the definition of dimension we obtain the following:

$$
\begin{aligned}
d(\tau) \quad &= \quad \liminf_{x \to \infty} \frac{\log\left(N(\tau, x)\right)}{\log\left(t(\tau, x)\right)} \\[2ex]
&\geq \quad \liminf_{x \to \infty} \frac{\log\left(\frac{t(x) - n \cdot s(x)}{n}\right)}{\log\left(t(x)\right)} \\[2ex]
&\geq \quad \liminf_{x \to \infty} \frac{\log\left(t(x) - n \cdot s(x)\right) - \log(n)}{\log\left(t(x)\right)} \\[2ex]
&\geq \quad \liminf_{x \to \infty} \frac{\log\left(t(x) - n \cdot s(x)\right)}{\log\left(t(x)\right)}.
\end{aligned}
$$

By (5) we see that $\lim_{x \to \infty} \frac{t(x) - n \cdot s(x)}{t(x)} = 1$. Thus, by Lemma 4.2 we also see that $\lim_{x \to \infty} \frac{\log\left(t(x) - n \cdot s(x)\right)}{\log\left(t(x)\right)} = 1$ so that $d(\tau) \geq 1$ as was to be proven. $\qquad \square$

The method in proving the lower bound seems very crude: no blocks of n consecutive entirely white tapes may occur. It seems that more ind-depth analysis could yield sharper lower bounds.

4.3   The Upper and Lower Bound Conjectures

In Lemma 4.3 we proved $d(\tau) \geq 1$ whenever $\lim_{x \to \infty} \frac{s(\tau,x)}{t(\tau,x)} = 0$. For TMs τ that run at most in linear time, we trivially have that $d(\tau) \geq 1$. Moreover, we conjecture that the result of Lemma 4.3 actually holds in full generality.

Conjecture 4.4 (Lower bound conjecture). *For each TM τ we have that $d(\tau) \geq 1$.*

Actually, we conjecture that a stronger condition holds.

Conjecture 4.5 (Space-time ratio conjecture). *For each TM τ which runs in more than linear time, we have that $\displaystyle\lim_{x \to \infty} \frac{s(\tau, x)}{t(\tau, x)} = 0$.*

In certain cases the Space-Time Theorem (Theorem 4.1) and the lower bound as proved in Lemma 4.2 coincide.

Lemma 4.6. *In case a TM τ uses polynomial space, and runs super-polynomial time we have that $d(\tau) = 1$.*
More in general, if for a TM τ we have that $\lim_{x \to \infty} \frac{s(\tau,x)}{t(\tau,x)} = 0$, then

$$\liminf_{x \to \infty} \frac{\log\big(s_\tau(x)\big)}{\log\big(t_\tau(x)\big)} = 0 \quad\Longleftrightarrow\quad d(\tau) = 1.$$

Proof. By combining our general lower and upper bound as proven in Lemma 4.3 and Theorem 4.1 respectively, we see that

$$1 \;\leq\; d(\tau) \;\leq 1 + \liminf_{x \to \infty} \frac{\log\big(s_\tau(x)\big)}{\log\big(t_\tau(x)\big)} \leq 1.$$

$\square$

Lemma 4.6 shows us that in certain cases, the upper bound as given in the Space-Time Theorem is actually attained. We shall empirically verify that this is always the case in (3,2) space and conjecture that it holds more in general.

Conjecture 4.7 (Upper Bound Conjecture). *We conjecture that for each $n \in \omega$ and each TM τ in $(n, 2)$ space we have $d(\tau) = 1 + \displaystyle\liminf_{x \to \infty} \frac{\log\big(s_\tau(x)\big)}{\log\big(t_\tau(x)\big)}$.*

Thus, Lemma 4.6 provides a proof of the Upper Bound Conjecture in certain situations. For any TM that performs at most in linear time we have also proven the Upper Bound Conjecture in Lemma 3.4. Below, in Lemma 4.9, we shall prove the Upper Bound Conjecture for some other situations too. In order to prove this, we first need an additional insight.

Lemma 4.8. *For each TM τ there is a constant $c_\tau \in [0,1]$ with*

$$\liminf_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x) \cdot t_\tau(x)} = c_\tau.$$

Proof. Since $N_\tau(x)$ is bounded above by $s_\tau(x) \cdot t_\tau(x)$ (we observed this before in the proof of Theorem 4.1) we get that for each TM τ we have for each x that $\frac{N_\tau(x)}{s_\tau(x) \cdot t_\tau(x)} \in [0,1]$. But then clearly the lim inf is well defined and within the closed interval $[0,1]$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\square$

Lemma 4.9. *In case $\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x) \cdot t_\tau(x)} \neq 0$ we can prove the Upper Bound Conjecture, that is*

$$d(\tau) = 1 + \liminf_{x\to\infty} \frac{\log\big(s_\tau(x)\big)}{\log\big(t_\tau(x)\big)}.$$

Proof. We may assume that τ runs in at least linear time for otherwise, the claim is proved by Lemma 3.4. Thus $\lim_{x\to\infty} \frac{1}{t_\tau(x)} = 0$. Our assumption gives us that $\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x) \cdot t_\tau(x)} = c_\tau$ for some $c_\tau \neq 0$. Note that in this assumption we have a limit and not a liminf so that any subsequence converges to the same limit. Consequently we have

$$
\begin{aligned}
\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(t_\tau(x))} &= \liminf_{x\to\infty} \frac{\log(c_\tau \cdot s_\tau(x) \cdot t_\tau(x))}{\log(t_\tau(x))} \\[2mm]
&= \liminf_{x\to\infty} \frac{\log(c_\tau) + \log(t_\tau(x)) + \log(s_\tau(x))}{\log(t_\tau(x))}
\end{aligned}
$$

which implies the Upper Bound Conjecture provided $c_\tau \neq 0$. $\qquad\qquad\square$

The following proposition provides an almost equivalent formulation of the Upper Bound Conjecture.

Proposition 4.10. *For each TM τ we have that if*

$$\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} = 1,$$

then the Upper Bound Conjecture holds.

Moreover, if the Upper Bound Conjecture holds uniformly for some TM τ, that is $d(\tau) = 1 + \lim_{x\to\infty} \frac{\log(s_\tau(x))}{\log(t_\tau(x))}$, then $\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} = 1$.

Proof. If $\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} = 1$, we also have $\lim_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} = 1$ since $\log(N_\tau(x)) \leq \log(s_\tau(x)t_\tau(x))$ for each x. Consequently,

$$
\begin{aligned}
d(\tau) &= \liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(t_\tau(x))} \\[2mm]
&= \liminf_{x\to\infty} \frac{\log(s_\tau(x)t_\tau(x))}{\log(t_\tau(x))} \\[2mm]
&= \liminf_{x\to\infty} \frac{\log(t_\tau(x)) + \log(s_\tau(x))}{\log(t_\tau(x))} \\[2mm]
&= 1 + \liminf_{x\to\infty} \frac{\log(s_\tau(x))}{\log(t_\tau(x))}.
\end{aligned}
$$

17

For the other direction we assume $d(\tau) = 1 + \lim_{x\to\infty} \frac{\log(s_\tau(x))}{\log(t_\tau(x))}$. Consequently

$$\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(t_\tau(x))} = 1 + \lim_{x\to\infty} \frac{\log(s_\tau(x))}{\log(t_\tau(x))}$$

$$= \lim_{x\to\infty} \left(1 + \frac{\log(s_\tau(x))}{\log(t_\tau(x))}\right)$$

$$= \lim_{x\to\infty} \left(\frac{\log(t_\tau(x))}{\log(t_\tau(x))} + \frac{\log(s_\tau(x))}{\log(t_\tau(x))}\right)$$

$$= \lim_{x\to\infty} \frac{\log(s_\tau(x)t_\tau(x))}{\log(t_\tau(x))}.$$

Using this identity $\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(t_\tau(x))} = \lim_{x\to\infty} \frac{\log(s_\tau(x)t_\tau(x))}{\log(t_\tau(x))}$ we see that for any subsequence $x_n \to \infty$ we have that

$$\lim_{n\to\infty} \left(\frac{\log(N_\tau(x_n))}{\log(t_\tau(x_n))}\right) \Big/ \left(\frac{\log(s_\tau(x_n)t_\tau(x_n))}{\log(t_\tau(x_n))}\right) \geq 1.$$

Consequently,

$$\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} = \liminf_{x\to\infty} \frac{\frac{\log(N_\tau(x))}{\log(t_\tau(x))}}{\frac{\log(s_\tau(x)t_\tau(x))}{\log(t_\tau(x))}} \geq 1.$$

But since $N_\tau(x) \leq s_\tau(x)t_\tau(x)$ the possibility $\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} > 1$ cannot occur and we are done. $\square$

4.4 The Space-Time Theorem and P versus NP

As usual, we denote by P the class of problems that can be solved by a TM which uses an amount of time that is bounded by some polynomial applied to the size of the input (representing an instantiation of the particular problem).

Likewise, we denote by NP the class of problems so that any solution of this problem can be checked to be indeed a solution to this problem by a TM which uses an amount of time that is bounded by some polynomial applied to the size of the input. Here, the N in NP stands for *non-deterministic*. That is to say, a non-deterministic TM would run in polynomial time by just guessing the right solution and then checking that it is indeed a solution. It is one of the major open question in (theoretical) computer science wether P = NP or not.

By PSPACE we denote the class of problems that can be solved by a TM which uses an amount of memory space that is bounded by some polynomial applied to the size of the input. It is well-known that NP $\subseteq$ PSPACE. Thus, by Lemma 4.6 we can state a separation of P and NP in terms of dimensions.

Let Π be some NP-complete problem. If for each PSPACE Turing machine τ that decides Π we have that $d(\tau) = 1$, then P $\neq$ NP.

Clearly, this does not constitute a real strategy since, for one, in general it is undecidable whether $d(\tau) = 1$ [40].

Part II: Experimental setting

In this second part of the paper we describe the experiment we have performed to empirically test if the theoretical results also hold in cases that do not satisfy the necessary requirements for the theoretical results to be applied.

5   The experiment

We have already proven on purely theoretical grounds that there is a relation between runtimes and fractal dimension of the space-time diagrams. However, our theoretical results only apply to a restricted class of TMs.

In the experiment we wanted to also study the fractal dimension of the space-time diagrams in cases where our theoretical results do not apply. Moreover, guided by the first outcomes of our experiment we formulated the Upper Bound Conjecture (Conjecture 4.7) and gathered data so as to investigate if the conjecture holds in $(3, 2)$ space.

5.1   Slow convergence

For TMs τ that run in at most linear time we have proven in Lemma 3.4 that $d(\tau) = 2$. Our aim is to use computer experiments to compute the Box dimension of all TMs τ where $d(\tau)$ is not predicted by any theoretical result.

A substantial complication in this project is caused by the occurrence of logarithms in the definition of $d(\tau)$. As a consequence, increase in precision of $d(\tau)$ requires exponentially larger inputs. This makes direct brute-force computation unfeasible. As an example, let us consider 2,2-TM 346 again whose space time diagrams we saw in Figure 1. By Lemma 3.4 we know that the Box dimension of this Turing machine equals two. However, Figure 2 below shows us how slow the rate of convergence is.
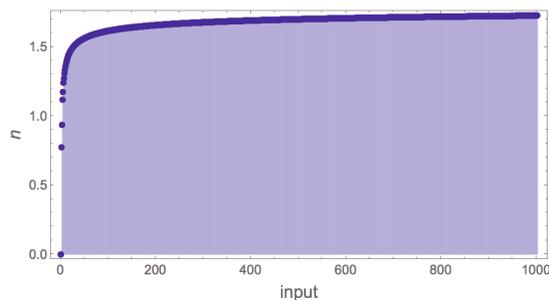

Figure 2: The figure shows an estimate of the Box dimension of 2,2 TM with TM number 346. On the horizontal axis the input is shown and the vertical axis shows the corresponding approximation of the Box dimension. Note that we know that the function converges to 2 when the input tends to infinity.

Our way out here is to apply numerical and mathematical analysis to the functions involved so that we can retrieve their limit behavior. In particular, we were interested in three different functions.

As before, for τ a TM we denote by $t_\tau(x)$ the amount of time-steps needed for τ to halt on input x; by $N_\tau(x)$ we denote the number of black cells in the space-time diagram of τ on input x and by $s_\tau(x)$ the distance between the edge of the tape and the furthest cell visited by τ on input x.

With these functions and knowledge of their asymptotic behavior, we can compute the corresponding dimension $d(\tau)$ and the upper bound $1 + \liminf_{x \to \infty} \frac{s_\tau(x)}{t_\tau(x)}$. The functions are guessed by looking at large enough initial sequences of their outcomes in a mechanized fashion. The few cases that cannot be done in a mechanized version were analyzed by hand.

It is important to bear this process in mind and the fact that we work with guesses that can be wrong in principle. For example, if we speak of a TM τ that performs in time of order n^2 this means in this paper that, by definition, after applying our particular analyzing process, τ was classified as an $\mathcal{O}(n^2)$ time performer. It may well be that in reality τ needs exponential time. However, there are strong indications that our guessing process is rather accurate [49, 23].

5.2 Methodology

In this subsection we shall describe the steps that were performed in obtaining our results. Basically, the methodology consists of the following steps.

1. Each TM that lives in 2,2 space also occurs in (3,2) space so for the final results it suffices to focus on this data-set. The TMs that diverge on all inputs were removed from the initial list of $2\,985\,984$ TMs in the (3,2) space, since for them the dimension is simply not defined. For the remaining TMs we erased all diverging inputs from the sequence to which we were to apply our analysis. Since we are only interested in limit behavior of any subsequences this does not alter our final results.

   We isolated the TMs for which there is no theorem that predicts the corresponding dimension. By Lemma 3.4 and Lemma 4.6 this means that we only needed to pay attention to those TMs which use more than linear time. Moreover, we also removed all simultaneous EXP time and PSPACE performers to finally end up with a collection of TMs. The distribution of the resulting collection is summarized in Table 1 below.

   In addition there are $1\,792$ TMs that perform in exponential time and linear space, but clearly they needed no further analysis since we know on theoretical grounds that their corresponding dimension is 1. All other machines in (3,2) space were very simple in terms of time computational complexity, that is, they perform at most in linear time.

2. Per TM τ, we determined/guessed its function $s_\tau(x)$ corresponding to the space usage of τ on input x. Although this guessing was already performed in [23] we decided to re-do the process. Main reasons to do this were a new

Boxes	Runtime	Space	Machines
$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	3358
$\mathcal{O}(n^4)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	6
$o(\mathsf{P})$	$o(\mathsf{P})$	$o(\mathsf{P})$	14

Table 1: Distribution of those TMs in (3,2) space of which we had to compute the corresponding dimension over their complexity classes. By $o(\mathsf{P})$ we denote the small o notation of the class of polynomials and hereby collect any super-polynomial behavior in one bucket.

release of our analyzing tool *Mathematica* together with the fact that the authors had obtained new insights on how to best perform the analysis. Our results coincided in large part with the ones obtained in [23] but also showed minor discrepancies.

3. Per TM τ, we determined its function $t_\tau(x)$ corresponding to the time usage of τ on input x;

4. Per TM τ, we determined its function $N_\tau(x)$ corresponding to the number of black cells in the space-time diagram of τ on input x;

5. Per TM τ, we computed $\lim_{x\to\infty} \frac{s_\tau(x)}{t_\tau(x)}$;

6. Per TM τ, we checked whether $d(\tau) \geq 1$;

7. Per TM τ, we computed its dimension $d(\tau)$ as $d(\tau) = \liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(t_\tau(x))}$;

8. Per TM τ, we compared its dimension $d(\tau)$ to its theoretical upperbound $1 + \liminf_{x\to\infty} \frac{\log(s_\tau(x))}{\log(t_\tau(x))}$ which we computed separately;

9. Per TM τ, we computed $\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))}$ and $\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x)\cdot t_\tau(x)}$.

5.3 Alternating convergent behavior

Some of the Turing Machines possessed alternating asymptotic behavior. This has been already observed in [23]. Typically the alternation reflects modular properties of the input like being odd or even or of the number of states.

The differences between the alternating subsequences can be rather drastic though. The most extreme example we found is reflected in Figure 3 below.

Figure 3 shows the space-time diagrams for TM τ with number $1\,728\,529$ for inputs 1 to 7. For convenience we have changed the orientation of the diagrams so that time 'goes from left to right' instead of from 'top to bottom'.

This machine runs in linear time for even inputs and exponential time for odd inputs. The runtime is given by:

$$t_\tau(x) = \begin{cases} 2(x-2) + 9 & \text{if } x \text{ is even;} \\ 2(x-1) + 3\,2^{\frac{x-1}{2}+1} + 5 & \text{if } x \text{ is odd.} \end{cases}$$
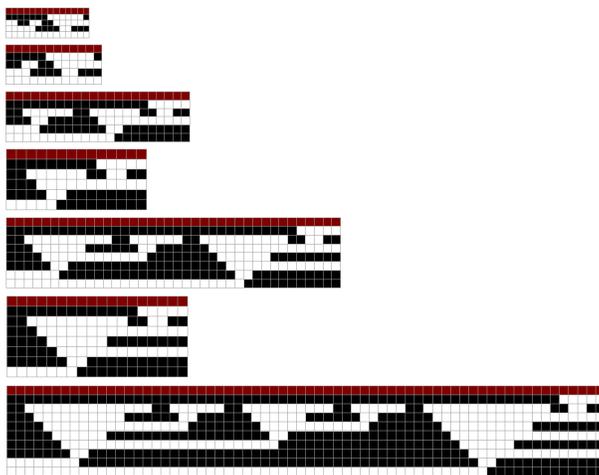
Figure 3: Alternating linear and exponential runtime behavior for TM $1\,728\,529$

The number of black cells $(N_\tau(x))$ in the space-time diagram exhibits the same behavior. Note however, that the space that τ uses is linear in the size of the input and in particular the amount of tape cells used is equal to the size of the output.

Moreover, we note that the sequence of outputs is of a very simple and regular nature. The outputs can be grouped in series of two, where the output on input $2 \cdot n + 1$ consecutive black cells is equal to the output on input $2 \cdot n + 2$ consecutive black cells. So, in a sense this TM incorporates two different algorithms to compute this output: one in linear time, the other, in exponential time.

We have found alternating sequences of periodicity 2, 3 and 6. Like we noted in [23], the periodicity typically reflects either the number of states, the number of colors, or a divisor of their product. Figure 4 shows an example of TM Number $1\,159\,345$ whose corresponding box counting sequence $N_\tau(x)$ has periodicity six.

On the left of Figure 4 we show the points $N_\tau(x)$ on the vertical axis plotted against the input x. On the right of the same figure we estimated a fit from below.

This alternating behavior reflects the richness of what we sometimes refer to as *the micro-cosmos of small Turing machines*. It is this alternating behavior which complicated analyzing the data set in a straight-forward automated fashion.

5.4   Determining the important functions

In this subsection we would mainly like to stress that most of the computational effort for this paper has actually been put into determining/guessing the functions $s_\tau(x), t_\tau(x)$ and $N_\tau(x)$ and computing the corresponding limits.
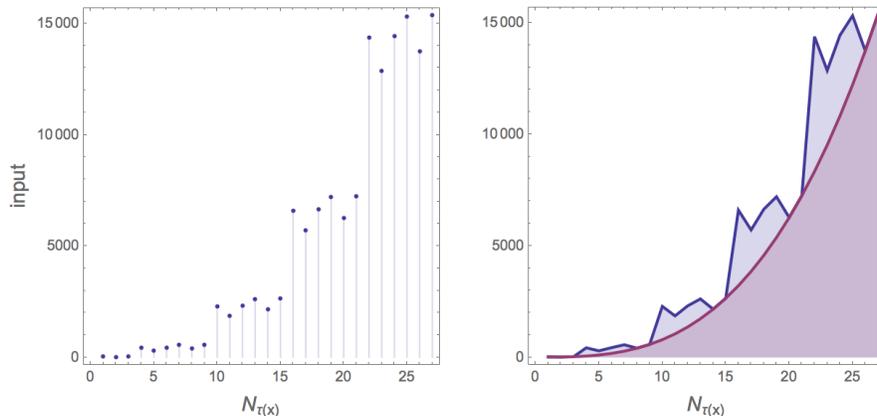
Figure 4: Alternating values for $N_\tau(x)$ with periodicity 6. The depicted values are for TM number $1\,159\,345$. The diagram on the left shows the data-set and on the right we included a fit from below.

As may have become manifest from the previous subsection, it is hard to automatically guess perfect matches for these functions in case there is alternating behavior present. Finally we could deal with all functions in a satisfactory way. Notwithstanding our confidence, it is good to bear in mind that all classifications provided in this paper are classifications given the current methodology.

We shall here briefly describe how we proceeded to guess our functions. The methodology is fairly similar as performed in [23]. However, for this project we used newer tools and a slightly more sophisticated methodology which accounts for possible differences with [23]. Schematically the guessing process can be split into the following steps.

1. We collected the sequences for time-usage $t_\tau(x)$ and space-usage $s_\tau(x)$ from the TM data set as described in Section 2 of this paper.

2. These sequences $t_\tau(x)$ and $s_\tau(x)$ are only given for the first 21 different inputs. We used an initial segment of 15 elements of these sequences to guess in an automated fashion the corresponding function that allegedly generates this sequence. In some cases the beginning of the sequence (up to three elements) was removed because the beginning did not match the general pattern that only occurred later on in the sequence. If we would leave the first values, *Mathematica* was no longer able to find the general pattern. The guessing process was done in *Mathematica* v.8 and 9 using the `FindSequenceFunction` as built-in in this software. In some cases `FindSequenceFunction` came with a solution, in others it did not. The function `FindSequenceFunction` does various standard numerical and algebraic analyses on the sequences but also checks for obvious recurrence patterns. The function, built into the computer algebra system *Mathemat-*

ica, takes a sequence of integer values $\{a_1, a_2, \ldots, a_m\}$ to define a function that yields a sequence $\{a_n\}_{n\in\omega}$ which coincides on the first m values.

`FindSequenceFunction` finds results in terms of a wide range of integer functions such as sums and series coefficients, as well as implicit solutions to difference equations using early elements in the list to find candidate functions, then validates the predicted function by looking at later elements.

3. Thus, we obtain two lists: a list L_1 of TMs where we found a guess and a list L_2 where we did not find any guess. From the initial list of 528 runtime sequences, we could not guess 11, and from the 167 space sequences, we could not guess 15. Note that this number need not be equal since TMs from various different functions had the same space sequence.

   Moreover, 288 runtime sequences and 85 space sequences in L_1 were alternators. *Mathematica* guessed the right function using terms like $(-1)^x$. However, in computing the lim inf, we manually split those sequences into, for example, an even and an odd part, to obtain the corresponding limits.

4. We performed a check on our guesses as collected in L_1 by applying the guessed function to inputs 16–21. In almost all cases our guess turned out to be predictive and coincided with the real values. For those few cases where there was a discrepancy between the guesses and the actual values, we made a new guess based on a larger initial segment, now consisting of the first 18 elements and then testing it once more on new real values. Finally we were able to guess and successfully check all of the sequences –both space and time usage– in L_1.

5. From the list L_1 we deleted all complexities for which we knew the dimension on theoretical grounds so as to obtain a list L_3.

6. For the TMs in L_3 we used the supercomputing resources of CICA (Andalusian Centre for Scientific Computing) to compute the corresponding sequences $N_\tau(x)$ with a C++ TM simulator. To reduce the computational effort, for each set of equivalent TMs (up to a geometrical transformation, such as state mirroring) only one representative was run. We applied the guessing process as described above for $t_\tau(x)$ and $s_\tau(x)$ also to $N_\tau(x)$ to come up with corresponding functions.

7. For the sequences in L_2 we applied a semi-manual process. Basically, there were three different procedures that we applied so as to find solutions also in L_2 for the sequences $s_\tau(x), t_\tau(x)$ and $N_\tau(x)$.

   (a) In most of the cases, there was alternating behavior present. We could read off the periodicity from looking at graphs as for example in Figure 4. Sometimes, looking directly at the space-time diagrams was more informative. In all of these cases but one, we finally did find functions for the subsequences using our methodology as described

above. As splitting the sequences into 2, 3 or 6 alternating ones reduces the length of the input sequence of `FindSequenceFunction`, we run in some cases 40 or 60 more inputs with the C++ simulator as to end up with a sufficiently large data set.

One alternating TM did not succumb to this methodology. This was TM $582\,263$ whose treatment is included in Section 6.3. We run this TM for 35 inputs with the simulator and observed that $N_\tau(x)/(s_\tau(x)t_\tau(x))$ clearly converges to a constant –one for each subsequence– so we approximated $N_\tau(x)$ by $c \cdot s_\tau(x)t_\tau(x)$ which was enough for the log-limit without knowing the exact value of c.

(b) In some cases the regularity was not obvious to *Mathematica* but where evident when looking at space-time diagrams and/or the binary expansion of the output. In these cases we could manage by just feeding our insight into *Mathematica* in that we let it work, for example, on the binary expansion of the sequences.

(c) In some cases the recurrences were just too complicated for *Mathematica* v8. In these cases we carefully studied the space-time diagrams analyzing what kind of recurrences there were present. Then, the observed recurrences were fed into `FindSequenceFunction` where we left FindSequenceFunction find out the exact nature and coefficients of the corresponding recurrences. One such example concerns the TM that produces the largest possible outputs in (3,2) space: the so-called *Busy Beaver* as detailed in Section 6.2.

8. After having successfully (allegedly) found the functions $s_\tau(x)$, $t_\tau(x)$ and $N_\tau(x)$ we could compute the values for $d(x) = \liminf_{x\to\infty} \frac{N_\tau(x)}{t_\tau(x)}$ and $\liminf_{x\to\infty} \frac{s_\tau(x)}{t_\tau(x)}$. In most cases a simple limit sufficed. For alternating behavior we had to select most of the times the subsequences by hand so as to end up with the $\liminf$ value. For some alternating sequences the $\liminf$ value could just be obtained by combining on the one hand the $\liminf$ of $N_\tau(x)$ (as depicted in Figure 4) or $s_\tau(x)$ respectively, and on the other hand $t_\tau(x)$.

6 Most salient results of the experiment

In this section we shall present the main results of our investigations. The space of TMs which employ only 2 colors and 2 states is clearly contained in (3,2) space. However, we find it instructive to dedicate first a subsection to the findings in (2,2) space. Apart from the first subsection, all other results in this section refer to our findings in (3,2) space.

6.1 Findings in (2,2) space

In (2,2) space there was a total of 74 different functions. Of these functions, only 5 of them where computed by some super-linear time TMs. Note, this does

not mean that all TMs computing this function performed in super-linear time. For example, the tape identity has many constant time performing TMs that compute it, but also some exponential time performing TMs that compute it.

In total, in (2,2) space, there are only 7 TMs that run in super-polynomial time. Three of them run in exp-time, all computing the tape-identity. The other four TMs compute different functions. These functions do roughly compute a function that doubles the tape input, see Figure 5.
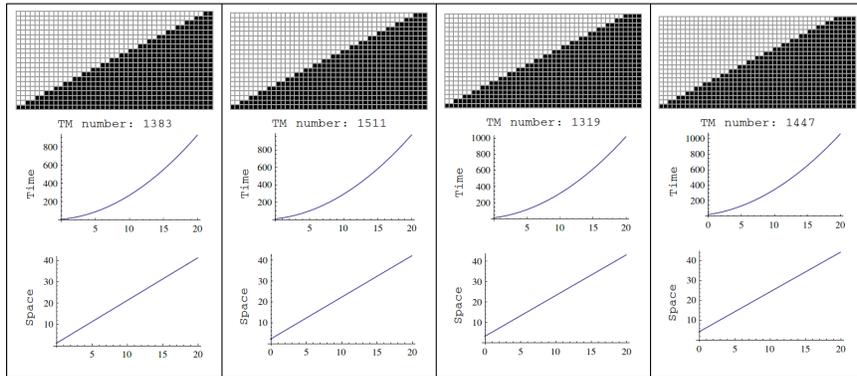

Figure 5: The figure shows the four different functions that are computed by the four TMs that have quadratic runtime in 2,2 space. The diagrams show the outputs on increasing inputs. So for example, in the left-most diagram we see that TM with number 1383 (recall this is the code in (2,2) space) outputs two black consecutive cells on input 1, and more in general it outputs $2n$ black consecutive cells on input n.

All these four TMs perform in quadratic time and linear space. We computed the dimension for these functions and all turned out to have dimension $\frac{3}{2}$. We observe that this is exactly the upper bound as predicted by the Space-Time Theorem. We saw this phenomenon in (3,2) space as well.

The only three exponential time performers used linear space so by Lemma 4.6 we already know that the dimension of those TMs should be one. This has been checked also in *Mathematica*. The check was not really performed to check our theoretical results, rather the check was used as a test-case for our analyzing software.

We saw that a TM in (2,2) space runs in super-polynomial time if and only if its dimension equals 1. This observation is no longer valid in (3,2) space though.

6.2 Exponential space and the Busy Beaver

In the remainder of this section we shall focus on the TMs in (3,2) space. That space contains 2 985 984 many different TMs which compute 3 886 different functions. Almost all TMs used at most linear space for their computations. The only exception to this was when the TM used exponential space. Curiously

enough, in (3,2) space there was no space usage in between linear and exponential space.

In [22] one can see an overview of the EXP-SPACE performing TMs. For most of these TMs it was not too hard to find an explicit formula for the space usage. An example is TM with number $683\,863$ whose corresponding space usage is:

$$s_{683.863}(x) = 2\left(\frac{x+1}{2} + 2^{\frac{x+1}{2}} - 1\right)$$

The space-time time diagrams for TM $683\,863$ contained sufficiently much regularity so that *Mathematica* could guess the corresponding functions. For various other EXP-SPACE performers we had to help *Mathematica* by suggesting it as to what kind of recursion it should look for. This occurred also with the so-called *Busy Beaver*.

Classically speaking the *Busy Beaver* is defined as that TM that runs for the longest time when executed on a two-way infinite tape with empty input [34]. In the context of this paper, we shall call a TM β a Busy Beaver whenever for each TM τ, there is some value x_0 so that for all $x \geq x_0$ we have $t_\beta(x) \geq t_\tau(x)$. Equivalent machines $599\,063$ and $666\,364$ are the Busy Beavers in the (3,2) space. They compute the largest runtime, space and boxes sequences. They also produce the longest output strings. For the remainder of this subsection we shall denote the Busy Beaver TM by β. As mentioned, there are of course two actual TMs that compute the Busy Beaver but they have the exact same behavior and we shall not distinguish between them.
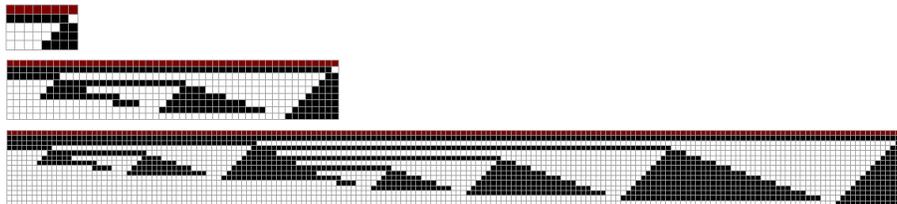

Figure 6: Execution of the Busy Beaver on the first three inputs

Figure 6 shows the execution of machine $666\,364$ for inputs 1 to 3. The diagrams have been rotated to save space. As one can see, the series of outputs is very regular and so is the sequence of cells used by the computation. Nonetheless, *Mathematica* did not find a recurrence between the consecutive values. This was due to a minor error term.

That is, if one looks at the amounts of used cells for consecutive inputs and their differences, then modulo a small error term, there is a clear tendency. Let x denote the number of consecutive black input cells. Looking at the ratio between consecutive values yielded us to isolate the disturbing difference term.

So, ignoring the exact nature of the error term, the recurrence equation for the space is given in (6).

27

$\mathbf{x}$	$\mathbf{s_\beta(x)}$	$\mathbf{s_\beta(x) - s_\beta(x-1)}$	$\mathbf{3/2(s_\beta(x-1) - s_\beta(x-2))}$	**Difference**
1	3	–	–	–
2	7	4	–	–
3	13	6	6	0
4	22	9	9	0
5	36	14	13+1/2	1/2
6	57	21	21	0
7	88	31	31+1/2	−1/2
8	135	47	46+1/2	1/2
9	205	70	70+1/2	−1/2
10	310	105	105	0

Table 2: The structure of the space sequence

$$
\begin{aligned}
s_\beta(1) &= 3 \\
s_\beta(2) &= 7 \\
s_\beta(p) &= \frac{1}{2}\left(5\ s_\beta(p-1) - 3\ s_\beta(p-2) + g(p)\right)
\end{aligned}
\tag{6}
$$

where $g(p)$ is a function[1] that takes values in $\{-1, 0, 1\}$.

The runtime depends on the space and we found the following recurrence relation for it:

$$
\begin{aligned}
t_\beta(1) &= 7 \\
t_\beta(i) &= \frac{3}{2}\ s_\beta(i-1)^2 - \frac{1}{2}\sin^4\left(\frac{1}{2}\pi\ s_\beta(i-1)\right) + \\
&\quad \frac{1}{2}\ s_\beta(i-1)(\cos(\pi\ s_\beta(i-1)) + 15) + \ t_\beta(i-1) + 8
\end{aligned}
\tag{7}
$$

Finally, by close inspection on the space-time diagrams we could guide *Mathematica* to look for specific kind of recurrences to finally come up with

---

[1]When we forced *Mathematica* to focus on the error term, it came up with the exact recurrence where $g(p) = \Big(-(3\ s_\beta(p-3) - 5\ s_\beta(p-2) + 2\ s_\beta(p-1))^3 - 2\big(\frac{3}{2}\ s_\beta(p-4) - \frac{5}{2}\ s_\beta(p-3) + s_\beta(p-2)\big)\big(1 - (3\ s_\beta(p-3) - 5\ s_\beta(p-2) + 2\ s_\beta(p-1))^2\big) + \big(1 - (3\ s_\beta(p-4) - 5\ s_\beta(p-3) + 2\ s_\beta(p-2))^2\big)\big(1 - (3\ s_\beta(p-3) - 5\ s_\beta(p-2) + 2\ s_\beta(p-1))^2\big)\Big)\sin^2\left(\pi(\frac{5}{2}\ s_\beta(p-1) - \frac{3}{2}\ s_\beta(p-2))\right)$ after defining $s_\beta(-1) = s_\beta(0) = 0$.

$$N_\beta(1) = 13$$

$$
\begin{aligned}
N_\beta(i) = \frac{1}{32} \big(&32\ N_\beta(i-1) + 32\ t_\beta(i) + 32\ s_\beta(i-1)^3 \\
&+ 152\ s_\beta(i-1)^2 + 140\ s_\beta(i-1) + 16\ s_\beta(i)^2 \\
&+ 16\ s_\beta(i) + 16\ s_\beta(i-1)^2 \cos(\pi\ s_\beta(i-1)) \\
&+ 24\ s_\beta(i-1) \cos(\pi\ s_\beta(i-1)) \\
&- 4\ s_\beta(i-1) \cos(2\pi\ s_\beta(i-1)) - 3\cos(\pi\ s_\beta(i-1)) \\
&- 9\cos(2\pi\ s_\beta(i-1)) - \cos(3\pi\ s_\beta(i-1)) + 32i - 19\big)
\end{aligned}
\tag{8}
$$

Using these recurrence equations we could finally compute the limits. We computed the limits both by standard methods on limits of recurrence relations and by employing *Mathematica* and both methods gave the same answers to the effect that all simultaneous EXP-TIME and EXP-SPACE TMs in (3,2) space all have fractal dimension $\frac{3}{2}$.

6.3   The space-time theorem revisited

One of our most important empirical findings is that the upper bound as given by the Space-Time Theorem is actually always attained in (3,2) space. Moreover we found two related empirical facts for (3,2) space. We mention them in this section.

Finding 0. For all TMs τ in (3,2) space we found that $d(\tau) \geq 1$. More in particular, we found that for each TM τ in (3,2) space which performed in super-linear time we have

$$\lim_{x \to \infty} \frac{s(\tau, x)}{t(\tau, x)} = 0.$$

and we conjecture that this holds in general for TMs with a larger number of states.

Finding 1. For all TMs τ in (3,2) space we found that

$$d(\tau) = 1 + \liminf_{x \to \infty} \frac{\log(s_\tau(x))}{\log(t_\tau(x))}$$

and we conjectured in the Upper Bound Conjecture (4.7) that this holds in general for TMs with a larger number of states.

In Proposition 4.10 we saw that a sufficient condition for the Upper Bound Conjecture to hold is that $\liminf_{x \to \infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x) t_\tau(x))} = 1$ but it is not known if this is also a necessary condition. The following finding is related to this.

Finding 2. For all TMs τ in (3,2) space we found that

$$\liminf_{x\to\infty} \frac{\log(N_\tau(x))}{\log(s_\tau(x)t_\tau(x))} = 1.$$

In Lemma 4.9 it was shown that $\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x)\cdot t_\tau(x)} \neq 0$ is a sufficient condition for the Upper Bound Conjecture to hold but it is not known if it is also necessary. The following finding is related to this.

Finding 3. For all TMs τ in (3,2) space we found that

$$\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x)\cdot t_\tau(x)} \in (0,1]$$

if this limit was well defined. Thus, in particular, we found that $\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x)\cdot t_\tau(x)} \neq 0$. Moreover, we found that only a limited amount of numbers were attained as limits of this quotient. The values found in $(3,2)$ for $\lim_{x\to\infty} \frac{N_\tau(x)}{s_\tau(x)t_\tau(x)}$ are:

$$\frac{1}{9},\frac{1}{6},\frac{7}{30},\frac{1}{4},\frac{5}{18},\frac{5}{16},\frac{1}{3},\frac{3}{8},\frac{8}{21},\frac{7}{18},\frac{5}{12},\frac{3}{7},\frac{4}{9},\frac{7}{15},\frac{1}{2},\frac{5}{9},\frac{9}{16},\frac{2}{3},\frac{3}{4},\frac{7}{9},1$$

It is possible that a few other limit values exist but were not found by the way we computed the functions generating $N_\tau(x)$.

For two of the exp-space performers we couldn't find the boxes function. These TMs were $582\,263$ (and its twin machine), whose execution for inputs 1 to 6 is shown in Figure 7. This TM possesses alternating behavior with periodicity two. For these two machines we used Lemma (4.8) to settle the computation of $d(\tau)$.

For the sequences of even number of consecutive black input cells we found that the fraction $\frac{N(x)}{s_\tau(x)\cdot t_\tau(x)}$ tended to 0.31 whereas for the odd number of consecutive black input cells we saw it tended to 0.11. The exact value of the fraction is of course irrelevant in the computation of the limit that determines $d(\tau)$.

Finding 4. For all TMs τ in (3,2) space we found that $d(\tau) = 1$ if and only if the TM ran in super-polynomial time using polynomial space. We suspect that this equivalence holds no longer true in higher spaces, i.e., spaces $(n,2)$ for $n > 3$.

Finding 5. For all TMs τ in (3,2) space we found that $d(\tau) = 2$ if and only if the TM ran in at most linear time. It is unknown if this equivalence holds true in higher spaces. Note that the if part holds in general and is proven in Lemma 3.4.
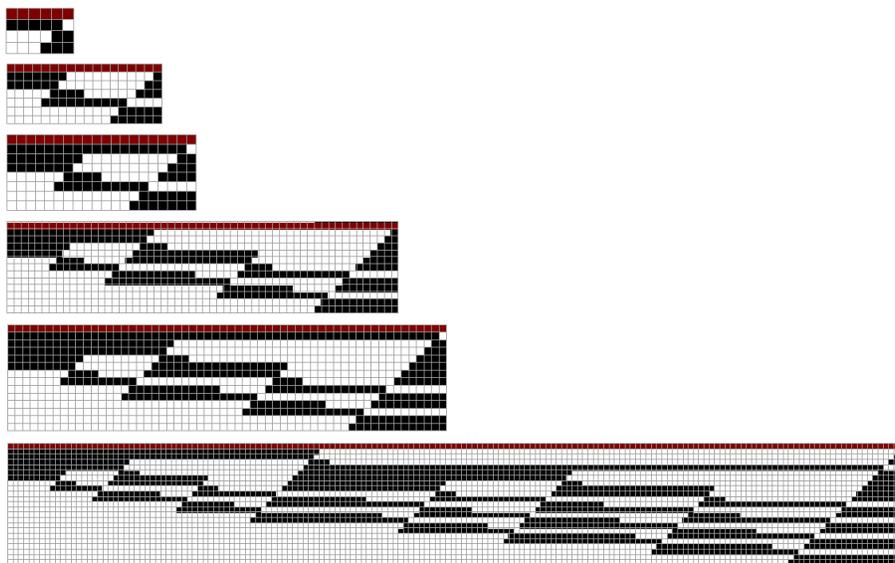
Figure 7: Execution of machine $582\,263$ on the first six inputs and their corresponding space-time diagrams

6.4 Richness in the microcosmos of small Turing machines

The authors have explored the space of small Turing machines before. On occasion they have been so much impressed by the rich structures present there that they came to speak of *the microcosmos of small Turing machines*. For this paper we had to mine (3,2) space even further and at some point were surprised to be surprised once more.

In particular, Figure 8 shows a very curious phenomenon that we call *symmetric performers*. There turns out to be a pair of different TMs so that the space-time diagram on every even input of the one machine is the exact symmetric image of the space-time diagram of the other TM on the same input.

Of course, this can only happen in case the TM computes the tape-identity since the input must equal the output in order to yield a symmetric image. At first, one might be tempted to think that this phenomenon is bound to occur since we can define for each TM τ its reversed machine $\hat{\tau}$: replace each instruction $\langle \mathsf{color}, \mathsf{state} \rangle \mapsto \langle \mathsf{color}', \mathsf{state}', \mathsf{direction} \rangle$ by its canonical reversal

$$\langle \mathsf{color}', \mathsf{state}' \rangle \mapsto \langle \mathsf{color}, \mathsf{state}, \overline{\mathsf{direction}} \rangle$$

where $\overline{\mathsf{direction}}$ changes right to left and vice-versa. However, note that both machines start in State 1 so that this imposes already a strong condition on possible solutions of symmetric performers.

Let us denote by τ and $\tilde{\tau}$ a pair of symmetric performers. It is clear that if a TM τ terminates on input x it does so in an even number of steps: for each computation where the head moves one to the left (the end of the tape is on the
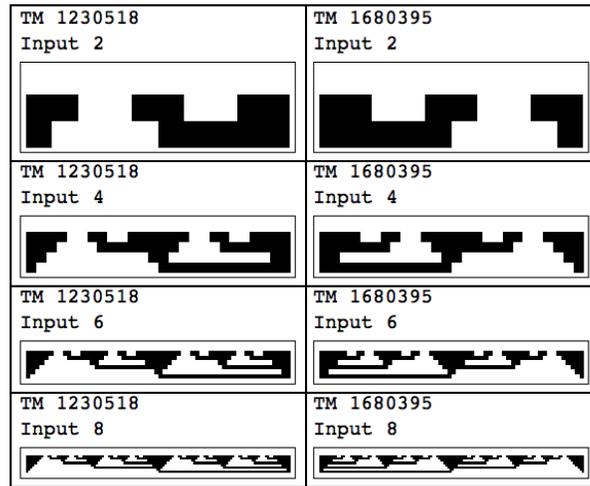
Figure 8: Symmetric performers

right by our convention), there must be a step where the machine moves one step to the right. In particular, for symmetric performers that terminate in $2n$ many steps on input x, we have that if the tape configuration at step m differs from the tape configuration at step $m+1$ in $\tau(x)$, then the head position in step m on $\tau(x)$ is the same as the head position in step $2n - (m+1)$ on $\tilde{\tau}(x)$.

Indeed it comes as a surprise that all these constraints can be met in (3,2) space, if only just for the even inputs.

Part III: A literature study

In the third and final part of the paper, we shall try to locate our results within the landscape of known theoretical results that link fractal dimensions to other notions of complexity.

7 Relations between fractal dimensions and other notions of complexity: An incomplete survey of the literature

In this paper we have worked with a variant of box-counting dimension and with computational space and time complexity. These are just some out of a myriad of different complexity measures in the literature. Since eventually the notion of being complex or not is relative to a framework and the ultimate framework in which all these complexity notions can be embedded in is our own cognitive system, on philosophical grounds one can expect relations between the various *a priori* unrelated complexity notions (see [21, 20]). And, indeed, in the literature we find various relations between different notions of complexity.

In this final section we wish to place our results in the context of other results in the literature that link different complexity notions. Our point of departure will be fractal dimensions and possible relations to complexity notions of a computational nature.

The current section is neither self-contained nor do we pretend to give an exhaustive overview of the literature. We shall try to provide sufficient pointers so that this section at least can serve as a point of departure for a more exhaustive and self-contained study.

7.1 Box-counting dimension within the landscape of topological and fractal dimensions

In this paper we decided to work with a variant of box-counting dimension since this has many desirable computational properties and applications. Let us first see where box-counting dimension fits into the landscape of various versions of fractal and other dimensions.

Edgar divides geometrical dimensions in two main groups, *topological* and *fractal* dimensions (see [14]). Topological dimensions are invariants of topological spaces in that they are invariant under homeomorphisms. Moreover, topological dimensions have integer values although some versions allow transfinite (ordinal) values too.

The most basic of all topological dimensions is the so-called *cover dimension* also called *Lebesgue dimension*. In order to describe this dimension we need some additional notions.

The *order* of a family $\mathcal{A}$ of sets is $\leq n$ by definition when any $n+2$ of the sets have empty intersection. We denote this by $o(\mathcal{A}) \leq n$. We say that $o(\mathcal{A}) = n$

when $o(\mathcal{A}) \leq n$ but not $o(\mathcal{A}) \leq n - 1$. Thus, for example, if any two sets in $\mathcal{A}$ have empty intersection the order of $\mathcal{A}$ is 0.

The *cover dimension* of a set S is n –we write $\mathsf{Cov}(\mathsf{S}) = \mathsf{n}$– whenever each open covering of S has a refinement of order n. Thus, for example, a collection of two separate points in $\mathbb{R}^n$ has cover dimension 0 since we can separate the points by two disjoint opens. Likewise, any line-like space admits an open cover of order 1, that is, any intersection of three different opens is empty. Similarly we can cover a planar set by open tiles where each row of tiles is shifted to the right, say, w.r.t. the adjacent rows of tiles. This collection of tiles has order two since any collection of four different of such open tiles will be empty.

Fractal dimensions on the other hand can have non-integer values. In a sense, the fractal dimension of some object S is an indication of how close S is to some integer-valued dimensional space. Dimension in integer-valued dimensional spaces in a sense express degrees of freedom and as such this provides us an information theoretical focus on dimension. More common is the geometrical focus on (fractal) dimension as for example expressed by Falconer ([15]): "Roughly, dimension indicates how much space a set occupies near to each of its points."

The most fundamental, and most common notion of fractal dimension is that of *Hausdorff dimension* ([16]) which was introduced already in 1919 building forth upon ideas of Carathéodory from 1914 ([10]).

In order to relate our box-counting dimension to the more common Hausdorff dimension we shall outline the definition and some basic properties of Hausdorff dimension.

For S a subset of some metric space we can consider countable open coverings $\mathcal{A}$ of S and define

$$\mathcal{H}_\varepsilon^s(S) := \inf \sum_{A \in \mathcal{A}} (\mathsf{diam}\ A)^s.$$

Here, $\mathsf{diam}\ A$ denotes to the usual diameter of A as the supremum of distances between any two points in A. The infimum is taken over all $\mathcal{A}$ that are countable open ε-covers of S. This means that the diameters of the open sets in our cover do not exceed ε. It is essential that we may take the diameters of the open sets in our cover to vary and in particular we can choose them as small as convenient. Next, we define

$$\mathcal{H}^s(S) := \lim_{\varepsilon \to 0} \mathcal{H}_\varepsilon^s(S).$$

The main theorem about these $\mathcal{H}^s(S)$ is that there is a unique s so that

- $\mathcal{H}_\varepsilon^t(S) = \infty$ for $t < s$;

- $\mathcal{H}_\varepsilon^t(S) = 0$ for $t > s$.

This unique s is called the Hausdorff dimension of S: $\mathsf{dim}_\mathsf{H}(F)$. As mentioned, this dimension was introduced in 1919 by Hausdorff ([16]) and the main theory was later developed mainly by Besicovitch and his students [3, 6, 7, 4, 5] so that ([16]) Mandelbrot often speaks of *Hausdorff-Besicovitch dimension*.

The Hausdorff dimension comes with a natural dual dimension called *packing dimension*. Although the notion of packing dimension is natural and related to the Hausdorff dimension it was only introduced about sixty years later by Tricot ([43]) and Sullivan ([39]).

The main idea behind packing dimension of some spatial object F is to somehow measure the volume of disjoint balls one can find so that the center of these balls lie within F. As with the case of Hausdorff dimension one parametrizes this concept with the target dimension s:

$$\mathcal{P}_\delta^s(F) := \{\sup \sum_i |B_i| \mid \{B_i\}_i \text{ are disjoint balls at radii} \leq \delta \text{ and center in } F\}$$

Since $\lim_{\delta \to 0} \mathcal{P}_\delta^s(F)$ is not a measure (this is easy to see by considering countable dense sets of some F with positive dimension) one applies a standard trick which transforms this into a measure by defining

$$\mathcal{P}^s(F) := \inf_{\{F_i\}_i} \{\sum_i \lim_{\delta \to 0} \mathcal{P}_\delta^s(F_i) \mid F \subseteq \bigcup_{i=1}^\infty F_i\}.$$

Here the infimum is taken over countable collections of sets F_i so that $F \subseteq \bigcup_{i=1}^\infty F_i$. The main theorem of this notion $\mathcal{P}^s(F)$ shows that Packing dimension is in a sense dual to Hausdorff dimension: There is a unique s so that

- $\mathcal{P}^s(F) = 0$ for $t < s$;

- $\mathcal{P}^s(F) = \infty$ for $t > s$.

This unique s is called the packing dimension of F and we write $\mathsf{dim}_\mathsf{P}(F)$. It is not hard to see that Packing dimension is an upper bound to Hausdorff dimension, that is, $\mathsf{dim}_\mathsf{H}(F) \leq \mathsf{dim}_\mathsf{P}(F)$.

A fundamental property that is not hard to prove of the dimensions we have seen so far is that: $\mathsf{Cov}(F) \leq \mathsf{dim}_\mathsf{H}(F)$. Mandelbrot defines a *fractal* to be any set F with $\mathsf{Cov}(F) < \mathsf{dim}_\mathsf{H}(F)$. However, this notion of fractal is often considered (also by Mandelbrot himself) a notion of fractal that is too broad, since it admits "true geometric chaos". J. Taylor proposes (see [41]) to denote by fractals only Borel sets F for which $\mathsf{dim}_\mathsf{H}(F) = \mathsf{dim}_\mathsf{P}(F)$.

We can now see how box-counting dimensions (or box dimensions for short) naturally fit the scheme of fractal dimensions we have seen above. In particular, the box dimension is like Hausdorff dimension only that we now cover by balls/boxes of *fixed size* rather than by ball of flexible size not exceeding some maximum value ε.

Alternatively and equivalently, in order to define the box dimension, we can divide space into a regular mesh with mesh-size δ and count how many cells $N_\delta(F)$ are hit by a set F. Then, we define $\mathcal{B}_\delta^s(F) := N_\delta(F)\delta^s$ and $\mathcal{B}^s(F) := \liminf_{\delta \to 0} N_\delta(F)\delta^s$.

Again, there is a cut-off value s_0 so that $\mathcal{B}^s(F) = \infty$ for $s < s_0$ and $\mathcal{B}^s(F) = 0$ for $s > s_0$. This cut-off value is given by

$$\liminf_{\delta \to 0} \frac{\log(N_\delta(F))}{\log(1/\delta)}.$$

which is close to the notion we started out with in this paper in Definition 3.1. Inspired by this cut-off value, we define

$$\underline{\dim}_\mathsf{B} := \liminf_{\delta \to 0} \frac{\log(N_\delta(F))}{\log(1/\delta)}$$

and

$$\overline{\dim}_\mathsf{B} := \limsup_{\delta \to 0} \frac{\log(N_\delta(F))}{\log(1/\delta)}.$$

In case $\underline{\dim}_\mathsf{B}(F) = \overline{\dim}_\mathsf{B}(F)$ we call this the box-counting dimension: $\dim_\mathsf{B}(F)$ which now exactly coincides with Definition 3.1. One can easily show that box dimension always provides an upper bound to Hausdorff dimension. Moreover, box dimension has many desirable computational properties thereby being amenable for computer applications.

Notwithstanding the good computational behavior, box dimension has various undesirable mathematical properties: in particular, a countable union of measure zero sets can have positive box dimension. For example, one can show that in $\mathbb{R}$ with the standard topology we have $\dim_\mathsf{B}\{0, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots\} = \frac{1}{2}$ which is of course highly undesirable.

Mathematically, this undesirable properties can be impaired with the same trick that was applied to the packing dimension by defining *modified box dimension* as

$$\underline{\dim}_\mathsf{MB}(F) := \inf_{\{F_i\}} \{\sup_i \underline{\dim}_\mathsf{B}(\mathsf{F_i}) \mid F \subseteq \bigcup_{i=1}^{\infty} F_i\} \quad \text{and}$$

$$\overline{\dim}_\mathsf{MB}(F) := \inf_{\{F_i\}} \{\sup_i \overline{\dim}_\mathsf{B}(\mathsf{F_i}) \mid F \subseteq \bigcup_{i=1}^{\infty} F_i\}$$

But, of course, by doing so, we would loose all the good computational properties. In general, we have that

$$\dim_\mathsf{H}(F) \le \underline{\dim}_\mathsf{MB}(F) \le \overline{\dim}_\mathsf{MB}(F) = \dim_\mathsf{P}(F) \le \overline{\dim}_\mathsf{B}(F)$$

and it is known that none of the inequalities can be replaced by equalities. However, we note that under Taylor's definition of fractal, the first four dimensions collapse and modified box dimension is an equivalent of Hausdorff dimension and indeed the modified box-counting dimension is a natural quantity to consider.

Moreover, if F has a lot of self-similarity, then modified box-counting dimension is actually equal to the plane box counting dimension:

Proposition 7.1. *Let $F \subseteq \mathbb{R}$ be compact so that for any open set V we have* $\overline{\dim}_\mathsf{B}(F) = \overline{\dim}_\mathsf{B}(F \cap V)$, *then* $\overline{\dim}_\mathsf{B}(F) = \overline{\dim}_\mathsf{MB}(F)$.

So in various situations, box counting coincides with Hausdorff dimension. The most famous example is probably that this equality holds for the Mandelbrot set. In addition, there are various other situations where box-counting and Hausdorff dimension coincide ([37, 38]).

7.2   Computability properties of fractals

As a first link between fractals and computability properties we want to mention that of various fractal objects one has studied the computational complexity.

Probably the most famous examples of fractals are Julia sets and the corresponding 'roadmap Mandelbrot set'. Let us briefly recall some basic definitions. By $\mathsf{FJ}(f)$ we denote the *filled Julia set* of a function f defined on the complex numbers. This set $\mathsf{FJ}(f)$ is defined as the set of values z in the domain of f on which iterating f on z does not diverge. That is,

$$\mathsf{FJ}(f) := \{z \mid \limsup_{n \to \infty} |f^n(z)| < \infty\}.$$

By $J(f)$ –the *Julia set of f*– we denote the boundary of $\mathsf{FJ}(f)$. Following C.T. Chong ([12]), we can consider $f_\theta(z) = z^2 + \lambda z$ with $\lambda = e^{2\pi i \theta}$ and $\theta \notin \mathbb{Q}$. Using this notation, the corresponding Julia sets are denoted by J_θ.

One can express that J_θ is well-behaved by saying that it has a *Siegel disk* at $z = 0$. Basically, this says that f is locally linearizable at $z = 0$ by a rotation and we refer the reader to e.g. [32] for further details.

The *Turing degree* of a set is an indication of how complicated that set is. A set of natural numbers A is of Turing degree at most that of B –we write $A \leq_T B$– if the question about $x \in A$ can be decided on an idealized computer using various queries of the form $y \in B$. We say that two sets A and B have the same Turing degree –we write $A \equiv_T B$– whenever both $A \leq_T B$ and $B \leq_T A$.

Likewise, we say that a set B is *computably enumerable* –or c.e. for short– in A if we can, using an idealized computer, enumerate all the elements of B using queries about A. Note that enumerability of B does not give a procedure to decide membership. It only guarantees you that if some element belongs to B, then at some stage it will be enumerated in the enumeration.

We call a set A *recursive*, *computable* or simply *decidable* if we can decide with an idealized computer without any oracles whether $x \in A$ or not for any $x \in \mathbb{N}$. Likewise, we call a set A simply c.e. when it is c.e. in the empty set $\varnothing$.

The Turing degree of a set –the equivalence class under $\equiv_T$ so to say– is a robust notion in various ways. For example, it makes sense to speak of 'being c.e. in the degree of A' whence we shall often refrain from distinguishing A from its corresponding degree.

We can conceive a real number as a set of natural numbers. Let us restrict ourselves to the real interval $[0, 1]$. Then we can conceive any real number a in this interval as a set by looking at the binary expansion of a and using this string $0, a_0 a_1 a_2 \ldots$ to define a set A where $i \in A$ iff $a_i = 1$. So by this identification it makes sense to speak of the Turing degree of a real number.

We shall shortly discuss that one can set up real analysis in such a way that it also makes sense to speak about the Turing degree of non-discrete objects like J_θ. Braverman and Yampolsky follow in [9] an approach of what is called *Constructive Analysis* as initiated by Banach and Mazur ([1]), with influences of Markov ([29]). The main idea behind this constructive analysis is that we can conceive continuous objects as entities that we can computably approximate to the precision that we require (see e.g. [44] for an overview).

Braverman and Yampolsky have studied (see [9]) the relations between the Turing degree of θ and that of J_θ. In particular they prove that $\mathbf{b}$ is a c.e. Turing degree if and only if it is the degree of J_θ with θ recursive so that J_θ has a Siegel disk.

C.T. Chong has generalized this result ([12]): Let $\mathbf{c}$ be a Turing degree. For every $\mathbf{d} \geq \mathbf{c}$ we have that $\mathbf{d}$ is c.e. in $\mathbf{c}$ if and only if it is the degree of a Julia set J_θ with Siegel disk and $\deg(\theta) = \mathbf{c}$.

It is good to stress that all these results are sensitive to the underlying model of computation and real analysis and the results would change drastically if one were to switch to other models like the so-called Blum-Schub-Smale model (see [8]).

The results presented in this subsection relate the Turing complexity of the fractal to the complexity of the parameter generating it. However there are no links from the Turing degrees of the Julia sets to the corresponding dimensions. In the next subsection we shall discuss various results of this sort.

7.3 Effective dimension and computations

In this subsection we shall present certain results that relate Hausdorff dimension to other notions of complexity. In order to do so we shall first rephrase the notion of Hausdorff dimension in the setting of binary strings. Next, we shall shall define so-called effectivizations of Hausdorff dimension. It is these effectivizations which can be related to other notions of complexity. Again, this subsection will be far from self-contained. We refer the reader to [13] for further details. And actually the presentation here is largely based on this treatise (mainly Chapter 13) and we shall closely follow it in structure.

Thus, let us reformulate the definition of Hausdorff dimension in the realm of binary sequences, i.e., in the realm of Cantor space which we shall denote by 2^ω. We shall interchangeably speak of sequences or of reals when we refer to elements of Cantor space. The collection of finite binary strings we shall denote by $2^{<\omega}$.

For $\sigma \in 2^{<\omega}$ we denote the length of σ as $|\sigma|$. For $\sigma \in 2^{<\omega}$ we define $[\![\sigma]\!] := \{\sigma\tau \mid \tau \in 2^\omega\}$ where $\sigma\tau$ denotes just string concatenation. Whenever we shall consider Cantor space as a topological space, we shall consider the topology generated by the (sub-) basic open sets of the form $[\![\sigma]\!]$. For $\Sigma \subseteq 2^{<\omega}$ we define $[\![\Sigma]\!] := \bigcup_{\sigma \in \Sigma} [\![\sigma]\!]$.

Thus, for any $R \subseteq 2^\omega$ we define an *n-cover* of R to be a set $\Sigma \subseteq 2^{\geq n}$ so that $R \subseteq [\![\Sigma]\!]$. Cantor space can be endowed with a measure in the standard way by

defining $\mu(\llbracket\sigma\rrbracket) = 2^{-|\sigma|}$. Thus, in analogy to Section 7.1 we now define:

$$\mathcal{H}_n^s(R) := \inf\{\sum_{\sigma\in\Sigma} 2^{-s|\sigma|} \mid \Sigma \text{ an } n\text{-cover of } R\}$$

and $\mathcal{H}^s(R) := \lim_{n\to\infty} \mathcal{H}_n^s(R)$. So, as before, we define $\mathsf{dim}_\mathsf{H}(R) := \inf\{s \mid \mathcal{H}^s(R) = 0\}$. It is easy to see that for every $r \in [0,1]$ there is $R \subseteq 2^\omega$ with $\mathsf{dim}_\mathsf{H}(R) = r$.

Within the context of Cantor space we shall now give a definition of what is called *Effective Hausdorff dimension*. The effective pendant is defined via

$$\mathcal{EH}_n^s(R) := \inf\{\sum_{\sigma\in\Sigma} 2^{-s|\sigma|} \mid \Sigma \text{ a } c.e. \; n\text{-cover of } R\}$$

and $\mathcal{EH}^s(R) := \lim_{n\to\infty} \mathcal{EH}_n^s(R)$, so that the *effective Hausdorff dimension* is defined as $\mathsf{dim}_\mathsf{EH}(R) := \inf\{s \mid \mathcal{EH}^s(R) = 0\}$.

One can now show ([27]) that for every computable real $r \in [0,1]$, there is a set $R \subseteq 2^\omega$ with $\mathsf{dim}_\mathsf{EH}(R) = r$. By a theorem of Hitchcock's we have that for important subsets F of Cantor space it holds that $\mathsf{dim}_\mathsf{H}(\mathsf{F}) = \mathsf{dim}_\mathsf{EH}(\mathsf{F})$:

Theorem 7.2 (Hitchcock [17]). *Let F be a countable union of Π_1^0 classes[2] of Cantor space, then $\mathsf{dim}_\mathsf{H}(\mathsf{F}) = \mathsf{dim}_\mathsf{EH}(\mathsf{F})$.*

In the same paper Hitchcock also proves an equality for Σ_2^0 classes and *computable Hausdorff dimension* (covers are required to be *computable* rather than c.e.). So, for some objects Hausdorff dimension and effective Hausdorff dimension coincide.

However, for other important classes they differ. In particular we have that the Hausdorff dimension of any sequence in Cantor space equals zero. However, there may be no simple effective covers around so that a single sequence can have positive effective Hausdorff dimension.

There is a link between Turing degrees and effective Hausdorff dimension albeit this link is not very straight-forward. Recall that for $A \in 2^\omega$ we have $\mathsf{dim}_\mathsf{H}(A) = 0$ but that we can have $\mathsf{dim}_\mathsf{EH}(A) > 0$ when no simple effective covers are around.

Thus, in a sense, having non-zero effective Hausdorff dimension is an indication of containing complexity. And in fact it can be shown that if $A \in 2^\omega$ with $\mathsf{dim}_\mathsf{EH}(A) > 0$, then A can compute a non-recursive function. To be more precise, A can compute a fix-point free function f (that is, a function f so that $W_{f(e)} \neq W_e$ for all numbers e) by results of Terwijn, Jockush, Lerman, Soare and Solovay ([42, 18]).

This result establishes a relation between effective Hausdorff dimension and computational complexity. However, the relation between effective dimension and computable content is not monotone nor simple. In particular, one can show that if $\mathsf{dim}_\mathsf{EH}(A) = \alpha$, then there exist sets B of arbitrary high Turing

---

[2] A subset A of Cantor space is a Π_1^0 class if it is the collection of paths for some computable tree. An alternative definition requires that for some computable relation R we have $A := \{\sigma \in 2^\omega \mid \forall n\ R(\sigma{\restriction}n)\}$.

degree with $\mathsf{dim}_{\mathsf{EH}}(B) = \alpha$. However locally, Hausdorff dimension can provide an upper bound to Turing degrees:

Theorem 7.3 (Miller [31]). *Let r be a left-c.e. real[3]. There is a Δ_2^0-definable set $R \in 2^\omega$ with $\mathsf{dim}_{\mathsf{EH}}(R) = r$ so that moreover*

$$A \leq_T R \quad \Rightarrow \quad \mathsf{dim}_{\mathsf{EH}}(A) \leq \alpha.$$

It is exactly this kind of results that we are interested here in this section: theorems that relate different notions of complexity. Another classical result links Kolmogorov complexity to effective Hausdorff dimension. Let us briefly and loosely define Kolmogorov complexity referring to e.g. [25] for further details.

For a string $s \in 2^{<\omega}$ the Kolmogorov complexity $K(s)$ is roughly the length of the shortest program that outputs s when computed on a particular universal Turing machine. Of course this is dependent on a particular choice of a universal Turing machine, but different choices of a universal Turing machine only manifest itself in an additive constant in K. The relation between Kolmogorov complexity and effective Hausdorff dimension is given by a theorem of Mayordomo:

Theorem 7.4 (Mayordomo [30]). *Let A be a sequence in Cantor space and let $A{\upharpoonright}n$ denote the first n bits of this sequence.*

$$\mathsf{dim}_{\mathsf{EH}}(A) = \liminf_{n \to \infty} \frac{K(A{\upharpoonright}n)}{n}.$$

Moreover, there is a link from effective Hausdorff dimension to a notion that is central to probability theory: *Martingales*. Martingales indicate expected outcomes of betting strategies. Lutz introduced in [26] an adaptation of this notion that can be linked to effective Hausdorff dimension (or constitute an alternative definition for that matter).

Definition 7.5. *An s-gale is a function $d : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ such that $d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2^s}$.*

This is a generalization of 'gales' (as introduced/simplified by [24]) where $d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}$ expresses a certain fairness condition of the betting strategy. In particular, one can see d as a pay-off function where the equality expresses that your expectation is to not loose money.

We say that a certain gale d *succeeds* on A whenever $\limsup_{n \to \infty} d(A \upharpoonright n) = \infty$.

The *Success set* of d is the collection of all A on which d succeeds and is denoted by $S[d]$. The link from Hausdorff dimension to these gales is given by a theorem by Lutz:

---

[3]We omit the technical details here and refer to [13] for them. However, one can think of a left-c.e. real as a c.e. real that does converge but for which we cannot computably estimate the rate of convergence.

Theorem 7.6 (Lutz [26])**.**

$$\mathsf{dim}_{\mathsf{EH}}(X) = \inf\{q \in \mathbb{Q} \mid X \subseteq S[d] \text{ for some } q\text{-gale } d\}.$$

In the context of this paper it is good to mention that other notions of dimension also have their effective counterparts. In particular, Reimann studied an effectivization of box counting dimension in [35] and the corresponding relations to the other complexity notions are similar to the ones mentioned here.

Also, for various dimensions the *computable* versions have been studied, where the open covers are no longer required to be c.e. but rather computable. We refer the reader to [13] for further details.

7.4   Our results

In this section we have tried to present a selection of readily accessible results in the literature that relate fractal dimension to other notions of complexity.

We mentioned results of Braverman and Yampolsky and the generalization thereof by Chong in 7.2. These results related computational (Turing degrees) properties of Julia sets to the same computational properties (Turing degrees) of the parameter that generates the Julia set. In a sense, this is not a result that links Hausdorff dimension to a different notion of complexity. However, since it is one of the few results on computational properties of fractals, we have decided to include it in the overview.

Next, in 7.3 we only worked in the realm of Cantor space. There we considered an effectivization of Hausdorff dimension and this notion was linked to Turing degrees as in Theorem 7.3, to Kolmogorov complexity as in Theorem 7.4 and to martingales as in Theorem 7.6.

Effective Hausdorff dimension however works with highly idealized notions relatively high up in the computational hierarchy. Our results involve complexity classes which are more down-to-earth like PTIME and PSPACE. Moreover, in our theorems the two complexity notions that are related –computational complexity versus fractal dimension– are not applied to exactly the same object as is the case in the earlier mentioned results. Rather they link two attributes of a small Turing machine: the computational (runtime) complexity on the one hand and the fractal dimension of the corresponding space-time diagrams on the other hand. It is in these respects that, to the best of our knowledge, our results are new in their kind.

Acknowledgements

References

[1] S. Banach and S. Mazur. Sur les fonctions calculables. *Ann. Polon. Math.*, 16, 1937.

[2] Charles H. Bennett. Logical depth and physical complexity. In *The universal Turing machine: A half-century survey*, pages 227–257. Oxford University Press, 1988.

[3] A. S. Besicovitch. Sets of fractional dimensions. Part I. *Mathematical Annals*, 101:161–193, 1929.

[4] A. S. Besicovitch. Sets of fractional dimensions. Part IV. *London Mathematical Society*, s1-9:126–131, 1934.

[5] A. S. Besicovitch. Sets of fractional dimensions. Part V. *London Mathematical Society*, 12:18–25, 1934.

[6] A. S. Besicovitch. Sets of fractional dimensions. Part II. *Math. Ann.*, 110:321–330, 1935.

[7] A. S. Besicovitch. Sets of fractional dimensions. Part III. *London Mathematical Society*, s2-47:436–454, 1942.

[8] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation.* Springer-Verlag, New York, 1998.

[9] M. Braverman and M. Yampolsky. *Computability of Julia Sets, Algorithms and Computation in Mathematics.* Springer, 2009.

[10] C. Carathéodory. über das lineare mass von punktmengen, eine veralgemeinerung des längenbegriffs. *Nachrichten von der Wissenschaften zu Götingen, Mathematisch-Physikalische Klass*, pages 404–426, 1914.

[11] G.J. Chaitin. Computing the busy beaver function. In T. M. Cover and B. Gopinath, editors, *Open Problems in Communication and Computation*, pages 108–112. Springer Verlag, NY, 1987.

[12] C.T. Chong. Complex dynamics and turing degrees. Slides published online, 2009.

[13] R.G. Downey and D.R. Hirschfeldt. *Algorithmic Randomness and Complexity.* Springer, 2010.

[14] E. G. Edgar. *Measure, Topology, and Fractal Geometry.* Springer-Verlag. New York, 1990.

[15] K. J. Falconer. *Fractal Geometry, Mathematical Foundations and Applications.* Wiley, Chichester, 2003.

[16] F. Hausdorff. Dimension und äusseres mass. *Mathematische Annalen*, 79:157–179, 1919.

[17] J.M. Hitchcock. Correspondence principles for effective dimensions. *Theory of Computing Systems*, 38:559–571, 2005.

[18] C.G. Jockush Jr., M. Lerman, R. I.BanachMazur:1937 Soare, and R.M. Solovay. Recursively enumerable sets modulo iterated jumps and arslanov's completeness criterion. *Journal of Symbolic Logic*, 54:1288–1323, 1989.

[19] J. J. Joosten. Turing machine enumeration: NKS versus lexicographical. Wolfram Demonstrations Project.

[20] J. J. Joosten. Complexity fits the fittest. In I. Zelinka, A. Sanayei, H. Zenil, and O. E. Rossler, editors, *How Nature Works*, pages 51–63. Springer Verlag, 2013.

[21] J. J. Joosten. On the necessity of complexity. In H. Zenil, editor, *Irreducibility and Computational Equivalence: 10 Years After the Publication of Wolfram's A New Kind of Science*, pages 11–24. Springer, 2013.

[22] J. J. Joosten, F. Soler-Toscano, and H. Zenil. Fractal dimension versus Time complexity in Turing machines. Wolfram Demonstrations Project.

[23] J. J. Joosten, F. Soler-Toscano, and H. Zenil. Program-size versus time complexity slowdown and speed-up phenomena in the micro-cosmos of small turing machines. *IJUC*, 7(5):353–387, 2011.

[24] P. Lévy. Théory de l'addition des variables aléatoires. *Gauthier-Villars, Paris*, 1937.

[25] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer; 3rd. Ed., 2008.

[26] J. H. Lutz. Gales and the constructive dimension of individual sequences. *ICALP 2000*, 1853:902–913, 2000.

[27] J. H. Lutz. The dimension of individual strings and sequences. *Information and Computation*, 187:49–79, 2003.

[28] M. Margenstern. Frontier between decidability and undecidability: a survey. *Theor. Comput. Sci.*, 231(2):217–251, 2000.

[29] A. A. Markov. On constructive mathematics. *(Russian) Tr. Mat. Inst. Steklov translated in Amer. Math. Soc., Trans., II Ser. 98, 1-9*, 67:8–14, 1962.

[30] E. Mayordomo. A kolmogorov complexity characterization of constructive hausdorff dimension. *Information Processing Letters*, 84:1–3, 2002.

[31] J. S. Miller. Extracting information is hard: a turing degree of non-integral effective hausdorff dimension. *Advances in Mathematics (to appear??)*, 2014.

[32] J. Milnor. *Dynamics in one complex variable. Introductory lectures*. Princeton University Press, 2006.

[33] T. Neary and D. Woods. Small fast universal Turing machines. *Theoretical Computer Science*, 362(1-3):171–195, 2006.

[34] T. Rado. On non-computable functions. *The Bell System Technical Journal*, 41(3):877–884, 1962.

[35] J. Reimann. *Computability and Fractal Dimension, Doctoral Dissertation*. Universitä Heidelberg, 2004.

[36] R. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[37] L. Staiger. A tight upper bound on kolmogorov complexity and uniformly optimal prediction. *Theory of Computing Systems*, 31:215–229, 1998.

[38] L. Staiger. Constructive dimension equals kolmogorov complexity. *Information Processing Letters*, 93:149–153, 2005.

[39] D. Sullivan. Entropy, hausdorff measures old and new, and limit sets of geometrically finite kleinian groups. *Acta Mathematica*, 153:259–277, 1984.

[40] K. Tadaki. The hausdorff dimension of the halting self-similar sets of t-universal prefix-free machines. In *ISIT*, pages 1287–1291. IEEE, 2010.

[41] S. J. Taylor. The measure theory of random fractals. *Mathematical Proceedings of the Cambridge Philosophical Society*, 100:383–406, 1986.

[42] S.A. Terwijn. Complexity and randomness. *Rendiconti del Seminario Matematico di Torino*, 62:57–74, 2004.

[43] C. Jr. Tricot. Two definitions of fractional dimension. *Mathematical Proceedings of the Cambridge Philosophical Society*, 91:57–74, 1982.

[44] Weihrauch. *Computable Analysis*. Springer, 2000.

[45] S. Wolfram. *A New Kind of Science*. Wolfram Media, January 2002.

[46] H. Zenil. Compression-based investigation of the dynamical properties of cellular automata and other systems. *Complex Systems*, 19-1:1–28, 2010.

[47] H. Zenil. On the dynamic qualitative behaviour of universal computation. *Complex Systems*, 20-3:265–277, 2012.

[48] H. Zenil. Two-dimensional kolmogorov complexity and validation of the coding theorem method by compressibility. *arXiv:1212.6745 [cs.CC]*, 2012.

[49] H. Zenil, F. Soler-Toscano, and J. J. Joosten. Empirical encounters with computational irreducibility and unpredictability. *Minds and Machines*, 22(3):149–165, 2012.

[50] H. Zenil and E. Villarreal. Asymptotic behaviour and ratios of complexity in cellular automata rule spaces. *International Journal of Bifurcation and Chaos*, 13-9, 2013.

[51] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, 1970.