Optimal Fractal Coding is NP-Hard[1]

(Extended Abstract)

Matthias Ruhl, Hannes Hartenstein

Institut für Informatik, Universität Freiburg
Am Flughafen 17, 79110 Freiburg, Germany
ruhl,hartenst@informatik.uni-freiburg.de

Abstract

In fractal compression a signal is encoded by the parameters of a contractive transformation whose fixed point (attractor) is an approximation of the original data. Thus fractal coding can be viewed as the optimization problem of finding in a set of admissible contractive transformations the transformation whose attractor is closest to a given signal. The standard fractal coding scheme based on the Collage Theorem produces only a suboptimal solution. We demonstrate by a reduction from MAXCUT that the problem of determining the optimal fractal code is NP-hard. To our knowledge, this is the first analysis of the intrinsic complexity of fractal coding. Additionally, we show that standard fractal coding is not an approximating algorithm for this problem.

1 Introduction

Data compression is inherently an optimization problem: the aim is to find the shortest description of a given data satisfying some quality constraint or, vice versa, to find the best quality representation for a given size. Usually one restricts oneself to a specific type of representation, e.g., vector quantization, transform coding or fractal coding. Since time is another cost factor in a compression scheme besides quality and size, it is of interest to analyze the time needed to find the optimal representation within a given scheme. In other words, it is useful to analyze the computational complexity of the involved optimization problem. It is of special importance to examine whether the problem may be generally intractable, i.e., NP-hard. In this case the result would provide an additional stimulus for the design of approximating or heuristic algorithms. Previous work in the field of data compression and complexity has shown that, e.g., optimal codebook design in vector quantization is NP-hard [1] as are versions of optimal pruning for tree structured vector quantizers [2]. For a comprehensive list of NP-hard optimization problems see [3].

This paper is concerned with the optimization problem involved with fractal compression. Fractal compression is a lossy data compression technique which attracted much attention in the last years mainly for its use in image compression. For a general introduction to this topic see [4, 5].

In this paper we analyze one-dimensional signal coding. In fractal coding, a target signal T is partitioned into disjoint *ranges* $r_i \in \mathbb{R}^m, i = 1, \ldots, n_r$, thus $T = (r_1, r_2, \ldots, r_{n_r}) \in \mathbb{R}^{m \cdot n_r}$. A *domain* is a signal block with twice the size of a range. The *domain pool*

[1]This paper appears in *Proceedings DCC'97 Data Compression Conference*, J. A. Storer, M. Cohn (eds.), IEEE Computer Society Press, March 1997.

contains the nonoverlapping domains that are unions of ranges, i.e., $d_1 = (r_1, r_2), d_2 = (r_3, r_4), ..., d_{n_d} = (r_{2\cdot n_d - 1}, r_{2\cdot n_d})$, where $n_d = \lfloor \frac{n_r}{2} \rfloor$. Let $\widehat{d_i}$ denote the m-dimensional vector obtained by down-filtering d_i – the k-th component of $\widehat{d_i}$ is the mean of the $(2k-1)$-th and $2k$-th component of d_i. In standard fractal coding for each range r_i the domain d_j is sought that minimizes the squared error distortion under an affine mapping, i.e., that minimizes

$$\|r_i - (s \cdot \widehat{d_j} + o \cdot \mathbf{1})\|^2,$$

where $\mathbf{1} = (1, 1, \ldots, 1) \in \mathbb{R}^m$,

$$(s, o) = \arg \min_{s,o \in \mathbb{R}} \|r_i - (s \cdot \widehat{d_j} + o \cdot \mathbf{1})\|^2,$$

and s is clamped to $[-s_{max}, s_{max}]$, $0 < s_{max} < 1$. s is called the scaling and o the offset parameter.

The sequence of triplets $(adr_i, s_i, o_i)_{i=1,\ldots,n_r}$, specifying the address adr_i of the domain that has been chosen for range r_i together with the scaling and offset parameters for the affine mapping, represents a contractive transformation f whose fixed point Ω_f, called attractor, is an approximation of the target signal as suggested by the Collage Theorem [6].

The above mentioned scheme is usually called *collage coding* since it searches for the transformation f that minimizes the *collage error* $\|T - f(T)\|^2$. Clearly this problem can be solved in polynomial time. But, on the other hand, there may be an attractor Ω_{f^*}, which is closer to signal T than Ω_f, even though f^* has a larger collage error.

The set of possible fractal codes for a signal T is

$$\Pi = \{f \mathrel{\widehat{=}} ((adr_1, s_1, o_1), \ldots, (adr_{n_r}, s_{n_r}, o_{n_r})) \mid 1 \le adr_i \le n_d,\ s_i \in [-s_{max}, s_{max}],\ o_i \in \mathbb{R}\}.$$

f^* is an optimal fractal code for the signal T if the attractor Ω_{f^*} of f^* satisfies

$$\|T - \Omega_{f^*}\|^2 = \min_{f \in \Pi} \|T - \Omega_f\|^2.$$

The number of different domain-range assignments in Π alone is $(n_d)^{n_r} = \lfloor \frac{n_r}{2} \rfloor^{n_r}$. Thus, even with a finite set of admissible s and o values the number of feasible codes grows exponentially with the number of ranges or the signal length, respectively. We show that the problem of finding the optimal fractal code in Π is NP-hard. For our analysis we can assume that the signal contains only integer values. We also note that the computation of the attractor Ω_f given f can be computed in polynomial time.

Let us formally define the problem of optimal fractal coding as a decision problem, called FRACCODE:

INSTANCE: Signal $T = (r_1, ..., r_{n_r})$, each range r_i is an m-dimensional vector with integer components, positive number D.
QUESTION: Is there an element $f \in \Pi$ whose attractor Ω_f satisfies $\|T - \Omega_f\|^2 \le D$?

The rest of the paper is organized as follows. In sections 2 and 3, we give a polynomial reduction from MAXCUT to FRACCODE, thus showing the NP-hardness of this problem. The behavior of collage coding as a non-approximating algorithm is analyzed in section 4. In section 5, we conclude the paper summarizing and discussing the main results as well as raising some open questions.

2 The main theorem

In this and the next section we will show that finding an optimal fractal code is at least as hard as solving an instance of (unweighted) MAXCUT. MAXCUT is defined as follows:

INSTANCE: Undirected graph $\mathfrak{G} = (V,E)$ with n_v vertices and n_e edges, positive integer k.
QUESTION: Is there a partition of V into disjoint sets V_1 and V_2 such that the number of edges that have one endpoint in V_1 and one endpoint in V_2 is at least k?

Since MAXCUT is known to be NP-hard (cf. [7]), it then follows that optimal fractal coding is also NP-hard.

The reduction from MAXCUT will proceed as follows: given a graph $\mathfrak{G} = (V,E)$ we will construct in polynomial time a signal $T(\mathfrak{G})$ and a function $D(\mathfrak{G},k)$ monotonically decreasing in k, such that the following holds:

Theorem: $\mathfrak{G}$ has a cut of size $\geq k \Longleftrightarrow \exists f \in \Pi$ with attractor Ω_f, such that $\|\Omega_f - T(\mathfrak{G})\|^2 \leq D(\mathfrak{G},k)$.

Thus, the question whether there exists a cut of a given cardinality k is reduced to the question whether there is an attractor Ω_f that is closer to a given signal than $D(\mathfrak{G},k)$.

The construction of $T(\mathfrak{G})$ and $D(\mathfrak{G},k)$ will be given in this section. From the construction it will follow immediately that

Lemma 2.1: $\mathfrak{G}$ has a cut of size $\geq k \Longrightarrow \exists f \in \Pi$ with attractor Ω_f, such that $\|\Omega_f - T(\mathfrak{G})\|^2 = D(\mathfrak{G},k)$.

Combined with Lemma 3.1, shown in the next section, this proves our theorem.

2.1 Construction of $T(\mathfrak{G})$

The signal $T(\mathfrak{G})$ will consist of several segments that will be represented as staircase functions in the figures and in our discussion. In the following, we describe the design of these segments and give a rationale behind the construction.

First of all, each vertex of the graph $\mathfrak{G}$ will be represented by a distinct signal part, the *vertex ID*. IDs pertaining to different vertices will differ significantly from each other. This enforces that only signal parts with the same ID can be mapped to each other.

The signal $T(\mathfrak{G})$ is obtained by concatenating the following signal parts:

- Signal segment S_1 contains for each vertex $v \in V$ four ranges as shown in figure 1a). The first and third ranges contain the vertex ID for v, the second and fourth range contain complementary signals used as *flags*, called F_1 and F_2.

- Segment S_2 contains two ranges for each vertex v (cf. figure 1b)). The first half of the first range is again the vertex ID of v, shrunk to half its original width. The rest of the two ranges equals zero.

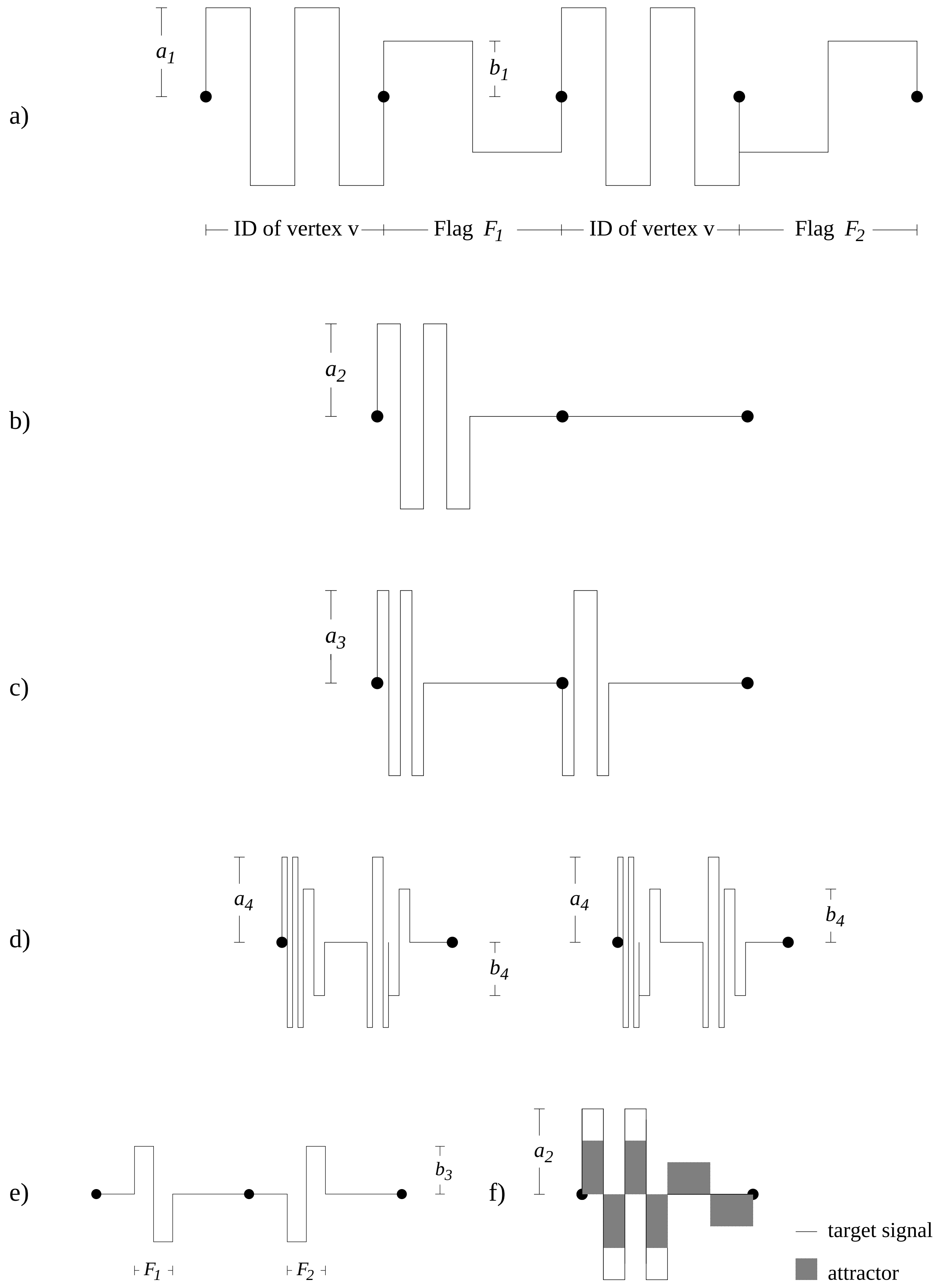

Figure 1: Building blocks of $T(\mathfrak{G})$

- In the third segment, S_3, for each edge $(v_i, v_j) \in E$ we have the following two ranges (cf. figure 1c)): The first quarter of the first range is the appropriately shrunk vertex ID of v_i, the first quarter of the second range contains the vertex ID of v_j. The rest of the ranges is zero.

- The fourth segment, S_4, contains two ranges for every edge $(v_i, v_j) \in E$ (cf. figure 1d)). Both contain the (shrunk) vertex IDs of v_i and v_j. Next to the vertex IDs are placed copies of the flags. In one range, these are the flags F_1 and F_2, in the other F_2 and F_1 (in this order).

The amplitudes $a_1, \ldots, a_4$ of the signal are related by $a_2 = s_1 \cdot a_1, a_3 = s_2 \cdot a_2, a_4 = s_3 \cdot a_3$. Furthermore, we set $b_i = \frac{a_i}{\sqrt{2}}$ for $i = 1, \ldots, 4$. Thus, all parameters are completely determined by a_4, s_1, s_2, s_3. We set a_4 to some arbitrary, but fixed, constant. The values of the s_i will be determined in section 3. Note that due to this definition the signal does not necessarily consist of integer values. The assumption is that the parameters can be scaled by some sufficiently large factor and then rounded.

We now sketch how the optimal attractor for $T(\mathfrak{G})$ is related to the size of the cut of $\mathfrak{G}$. Let us assume that the ranges of S_i have to be coded by domains from S_{i-1} for $i = 2, 3, 4$, and S_1 is given as side information. As said before, the ID design leads to the property that an ID mismatch will be *very* costly. Thus, for each range in S_2 the only possible domains are the two with corresponding ID in S_1. Both contribute the same distortion in the attractor. Selecting one of them for each range corresponds to the partitioning of V into V_1 and V_2. The flag (F_1 resp. F_2) associated with a vertex v therefore indicates to which set of the partition v belongs (V_1 resp. V_2). Again, each range of S_3 has to be coded by the domain of S_2 with the same vertex ID. In the attractor this third segment contains the information which edges of the graph $\mathfrak{G}$ belong to the cut. The segment S_4 will be used to *count* the number of these edges. An edge in the cut consists of a pair of vertices to which different flags (F_1 and F_2, or vice versa) have been assigned. In that and only in that case, we can find an exact match for one of the ranges in S_4 belonging to that edge. Thus, the error of the attractor is coupled with the size of the cut.

2.2 ID design

To make things explicit, we now give the remaining details of our construction. The IDs are built using the following lemma:

Lemma 2.2: For each $n \in \mathbb{N}$ there exists a binary code with n codewords $c_1, \ldots, c_n$, each of length $\ell = O(n)$, such that for $i \neq j$ the Hamming distances $d_H(c_i, c_j)$ and $d_H(c_i, \overline{c_j})$ equal $\ell/2$. $\overline{c_i}$ denotes the binary complement of c_i.
Proof: We will show by induction that the lemma holds for $n = \ell = 2^m$ for all $m \in \mathbb{N}$. For all other n simply choose n of the codewords constructed for size $2^{\lceil \log n \rceil}$. To begin the induction, $c_1 = 0$ is such a code for $n = 2^0$. For $n = 2^{m+1}$ take the set $\{c_i c_i, c_i \overline{c_i} | 1 \leq i \leq 2^m\}$, where the $(c_i)_{i=1,\ldots,2^m}$ form a code of the desired type of length 2^m. This gives a new binary code of size 2^{m+1} that is easily shown to have the desired property. $\square$

Let $(c_i)_{i=1,\dots,n_v}$ be a binary code of n_v codewords constructed as in lemma 2.2. The binary code $C := \{c_i\overline{c_i} | 1 \leq i \leq n_v\}$ not only has the property that two different codewords differ in half their bits, but also has the following features which we will use in our calculations:

- Every codeword consists of ℓ 0s and ℓ 1s.
- For two different codewords $p, q \in C$ the following holds: for all $i, j \in \{0,1\}$ there are exactly $\frac{\ell}{2}$ positions where p has a i-bit and q has a j-bit.

When one interprets the 0s and 1s of the binary code C as $-a_1$ and $+a_1$ respectively, this provides us with n_v vertex IDs that have a linear length representation and the property that an ID mismatch will introduce a large distortion. We add construction segments $S_{0,1}, \dots, S_{0,L}$, with $L = O(\log n_v)$, to the signal in order to have all ingredients for coding segment S_1 without any distortion. We set $S_0 := S_{0,1} \dots S_{0,L}$.

For the edge counting we also need an extra block in S_3 of the shape as sketched in figure 1e). Of course, this also leads to the addition of some construction blocks in segments S_0, S_1, S_2. Details are omitted due to space constraints.

2.3 Constructing an attractor

For the signal $T(\mathfrak{G}) = S_0S_1S_2S_3S_4$ as described above we now give a transformation f that will be used to define $D(\mathfrak{G}, k)$.

First of all, the segment S_0S_1 can be coded without any distortion. By hypothesis, $\mathfrak{G} = (V, E)$ has a cut of cardinality k by partitioning V into V_1 and V_2. For a range in S_2 we choose the domain in S_1 with the same ID and the flag set in accordance with the graph partition. The scaling and offset are set to $s = \frac{2}{3}s_1$ and $o = 0$. In this way the height of the attractor is $\frac{2}{3}$ the height of the original signal in S_2 (see figure 1f). Thus, the total distortion of Ω_f in segment S_2 is $n_v \cdot \frac{1}{6}a_2^2$. (See case 2 in the proof of lemma 3.1 for how to derive this value.)

For each range in S_3 we choose the corresponding domain of S_2 and scale it using $s = s_2, o = 0$. The error introduced in segment S_3 is $n_e \cdot \frac{1}{12}a_3^2$.

The distortion in segment S_4 depends on the number k. For each edge there are two ranges in S_4 differing only in the flags. Depending on whether or not an edge belongs to the cut, we do the following:

- *Edge belongs to cut.* In this case, one of the two ranges can be coded without any distortion by the corresponding domain in S_3. The second range will be coded by the extra block mentioned above ($s = s_3, o = 0$) yielding a distortion of $\frac{1}{4}a_4^2$.
- *Edge does not belong to cut.* In this case, we code both of the ranges in S_4 with the corresponding domain in S_3 ($s = s_3, o = 0$) yielding a total error of $\frac{5}{12}a_4^2$.

Thus, the total error introduced in segment S_4 is $(\frac{1}{4}n_e + \frac{1}{6}(n_e - k))a_4^2$. We define $D(\mathfrak{G}, k)$ as the distortion made by the attractor in all segments of the signal:

$$D(\mathfrak{G}, k) := \frac{1}{6}n_v a_2^2 + \frac{1}{12}n_e a_3^2 + (\frac{1}{4}n_e + \frac{1}{6}(n_e - k))a_4^2$$

This finishes our proof of Lemma 2.1. $\square$

3 The main theorem, continuation of proof

Lemma 3.1: $\mathfrak{G}$ has a maximal cut of size $k \Longrightarrow \nexists f \in \Pi$ with attractor Ω_f such that $\|\Omega_f - T(\mathfrak{G})\|^2 \leq D(\mathfrak{G}, k+1)$.

Proof: Assume that, on the contrary, such an attractor Ω_f does exist. From Lemma 2.1 we know there is an attractor Ω with $\|\Omega - T(\mathfrak{G})\|^2 = D(\mathfrak{G}, k)$. Obviously, Ω_f is closer to the original signal $T(\mathfrak{G})$ than Ω. Consequently it must approximate $T(\mathfrak{G})$ better on at least one of the segments of the signal, S_0, S_1, S_2, S_3, S_4. By setting s_1, s_2, s_3, depending only on the input graph $\mathfrak{G}$, we will enforce that the error $||\Omega_f - T(\mathfrak{G})||^2$ is at least $||\Omega - T(\mathfrak{G})||^2 - \frac{1}{2}(D(\mathfrak{G}, k) - D(\mathfrak{G}, k+1)) > D(\mathfrak{G}, k+1)$. Thus, our hypothesis is false and the lemma is proven.

For the following discussion let us assume that the ranges of S_2 have to be coded by domains from S_1, ranges from S_3 by domains from S_2 and ranges from S_4 by domains from S_3. At the end of the proof we will indicate how to remove this restriction.

Case 1: Ω_f *is better than* Ω *on* S_0 *or* S_1
Since Ω and $T(\mathfrak{G})$ do not differ on S_0 and S_1, no improvement is possible.

Case 2: Ω_f *is better than* Ω *on* S_2
For simplicity first assume that Ω_f is identical to $T(\mathfrak{G})$ on part S_1. For a range there are two possibilities for choosing a domain:

1. When mapping a domain with a fitting ID segment, the incurred error is (depending on scaling factor s and offset o)

   $$E(s,o) = \frac{1}{4}\Big((a_2 - (a_1 s + o))^2 + (-a_2 - (-a_1 s + o))^2 + (b_1 s + o)^2 + (-b_1 s + o)^2\Big)$$

   Solving this equation for the optimal values of s and o yields $s = \frac{2}{3}s_1, o = 0$. This leads to an error of $E(\frac{2}{3}s_1, 0) = \frac{1}{6}a_2^2$ for the range.

2. When mapping a domain with an incorrect ID segment on a range, the error will be

   $$E(s,o) = \frac{1}{8}\Big((a_2 - (a_1 s + o))^2 + (-a_2 - (-a_1 s + o))^2 + (a_2 - (-a_1 s + o))^2 + (-a_2 - (a_1 s + o))^2 + 2\cdot(b_1 s + o)^2 + 2\cdot(-b_1 s + o)^2\Big)$$

   Again, solving for optimal s, o yields $s = o = 0$ with an error of $E(0,0) = \frac{1}{2}a_2^2$, three times the error incurred when matching correct IDs.

Thus, the error of Ω_f on S_2 is at least $(n + 2l)\cdot\frac{1}{6}a_2^2$, where l is the number of incorrect ID assignments. We choose s_2 so small that the error made by one ID mismatch is larger than the error made by Ω on segments S_3 and S_4:

$$2\cdot\frac{1}{6}a_2^2 > \frac{1}{12}n_e a_3^2 + (\frac{1}{4}n_e + \frac{1}{6}(n_e - 0))a_4^2 \iff s_2 < \frac{2}{\sqrt{(1+5s_3^2)n_e}}$$

Therefore, l must equal zero, since otherwise the error incurred in segment S_2 alone would be larger than $D(\mathfrak{G},k)$.

Let us now deal with the assumption that Ω_f equals $T(\mathfrak{G})$ on segment S_1. Note that the difference between Ω_f and $T(\mathfrak{G})$ on S_1 has to be less than $D(\mathfrak{G},k)$, and this value does not depend on s_1. By choosing s_1 sufficiently small, we can assure that the error of $D(\mathfrak{G},k)$ is very small relative to a_1. This relative error will then change our calculations slightly. But by scaling s_1 we can make these differences arbitrarily small, in fact, significantly smaller than $\frac{1}{2}(D(\mathfrak{G},k)-D(\mathfrak{G},k+1))$.

Case 3: Ω_f *is better than* Ω *on* S_3
We use a similar argument as in case 2.

First, we can assume that Ω_f looks essentially like Ω on S_2. This is because by choosing s_2 small enough, any difference that is noticeable after scaling down a domain from S_2 would mean a large additional error in the domain, larger than any potential savings in S_3 and S_4. Details of this argument will be given in the full version of this paper.

Secondly, by matching correct IDs on our ranges, we incur exactly the same error as in Ω. On the other hand, by choosing s_3 small enough, we can assure that if we use incorrect IDs, the error will be larger than the error made by Ω in S_4 (which does not depend on s_3). Thus, the total error would be larger than the error of Ω.

Case 4: Ω_f *is better than* Ω *on* S_4
Again, we can assume that Ω and Ω_f look 'the same' on segment S_3.

We will now examine the error that possible domain-range pairings will incur. To this end, we distinguish two cases: first, if an edge belongs to the cut, i.e., the flags of the two vertices are different, and secondly, if the flags are the same.

1. *Edge belongs to cut.* In this case, one of the two edge copies in S_4 can be mapped with error zero. As for the other copy, by computing the optimal transformation parameters for all possible domains in S_3, we see that mapping the extra block on the range yields the minimum error of $\frac{1}{4}a_4^2$.

2. *Edge does not belong to cut.* In this case, there exists no exact matching domain in S_3 for the two edge copies in S_4. Thus, these ranges can only be coded with a mismatched ID, wrong flag or the extra block. By computing the optimal transformation parameters for all possible domains, we see that the error for each of the two edge copies is at least $\frac{5}{24}a_4^2$ for a total of $\frac{5}{12}a_4^2$.

Thus, the error of Ω_f in S_4 is at least $n_e\frac{1}{4}a_4^2+(n_e-k)\frac{1}{6}a_4^2$ which is exactly the error of Ω.

It remains to be shown how to assure that ranges from segment S_i are only coded by ranges from segment S_{i-1} for $i=2,3,4$. This can be achieved by a slight modification of the signal $T(\mathfrak{G})$. All considerations made so far will still remain valid. All we have to do is to change the IDs slightly. We add, for example, at the left end of the ID a peak of height h and width δ (cf. figure 2). The rest of the ID is shrunk accordingly. Observe that at the left end of the ranges in S_2,S_3,S_4 we now have a peak of width $\frac{\delta}{2},\frac{\delta}{4},\frac{\delta}{8}$, respectively. By choosing h sufficiently large, we can achieve our goal: If, for example, f maps a domain from S_3 to S_2, it will map a peak of width $\frac{\delta}{8}$ (the width is halved) on a peak of width $\frac{\delta}{2}$. If h is chosen large enough, this will lead to an arbitrarily large error no matter what the rest of the domain and range look like. This consideration concludes our proof of lemma 3.1. □

4 Collage coding

In this section we demonstrate that collage coding is not an approximating algorithm for the optimization problem of fractal coding.

Lemma 4.1: For every $\Delta \in \mathbb{R}^+$ there exists a signal T and a transformation $g \in \Pi$ such that

$$\|\Omega_f - T\|^2 > \Delta \cdot \|\Omega_g - T\|^2,$$

where f is the transformation that gives the best collage error $\|T - f(T)\|^2$.

Proof *(Sketch)*: For each range the standard algorithm looks for the best matching domain in a 'greedy' manner without taking into account the dependencies between different domain range assignments. This can be seen by coding the signal depicted in figure 3, where ranges are numbered $1,2,\ldots$. When the a_i are appropriately chosen, we can force the collage coder to encode the ranges in S_i by domains in S_{i-1} for $i = 2,3,4$. The only 'choice' for an attractor is to decide which domain of S_2 is to be mapped on the ranges in S_3.

In one case, if domain (3,4) is mapped on ranges 7 and 8, the whole of S_4 can be coded without any distortion. This is mapping g.

The other case – (5,6) is mapped on 7 and 8 –, will lead to a large error in S_4. The collage coder can be forced to use this choice in f by making the transformation from (5,6)

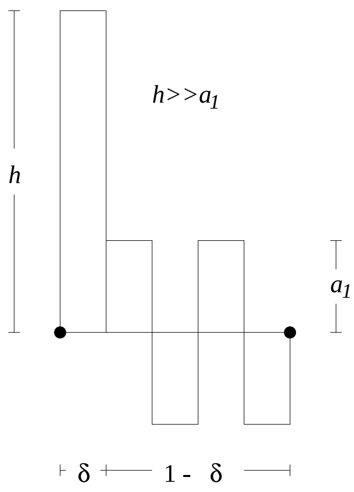

Figure 2: Adding a peak to the ID

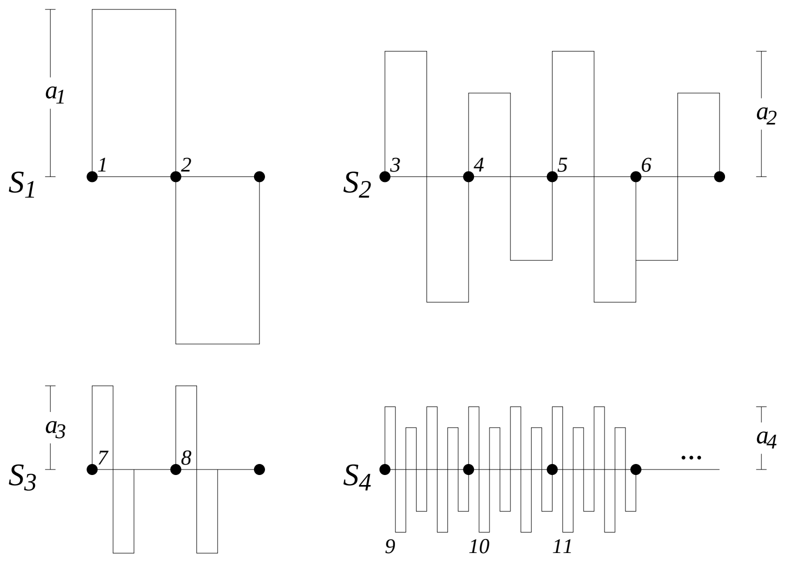

Figure 3: A hard signal for collage coding, consisting of segments $S_1,\ldots,S_4$

to 7 and 8 just a little bit cheaper than the transformation from (3,4) to 7 and 8. By adding many copies of range 9 to S_4, we see that the distance between the attractor found with the collage based scheme and the optimal attractor can be made arbitrarily large. $\square$

5 Conclusion

In this paper we have examined for the first time the intrinsic complexity of fractal coding. We have shown that finding an optimal fractal code for a given signal – the *inverse problem* of fractal coding – is NP-hard. Consequently, there exists no algorithm that solves this problem *fast*, i.e., in polynomial time, unless P = NP. Additionally, we have shown that collage coding is not an approximating algorithm for this problem.

What does this result imply for current research in fractal coding? First note that our proof readily carries over to higher dimensional signals, in particular to (two-dimensional) images. We are aware that our transformation parameters s and o are continuous, whereas in an implementation they will have to be discrete. But, in fact, the transformations constructed in section 2 use only a finite number of parameter values. Thus, the problem remains NP-hard when a finite set of admissible parameter values are given as additional input. Our result shows that an algorithm supposedly solving the problem in polynomial time cannot be a simple modification of existing ones. To build such an algorithm, one therefore has to deviate from our model in some significant way.

We close with some ideas for further research. First, it should be examined whether the bounds in our proof can be strengthened. Secondly, one should consider restrictions of the problem, e.g., with fixed scaling factor, or input signals with a certain regularity. And, finally, the next step after proving a problem NP-hard is to determine whether it admits an approximating algorithm.

References

[1] Lin, J., *Vector quantization for image compression: algorithms and performance*, Doctoral Dissertation, Brandeis University, 1992.

[2] Lin, J., Storer, J., Cohn, M., *Optimal pruning for tree-structured vector quantization*, Information Processing & Management vol. 28, no. 6, 1992.

[3] Crescenzi, P., Kann, V., *A compendium of NP optimization problems*, Technical report SI/RR-95/02, Universita di Roma ”La Sapienza”, 1995.

[4] Fisher, Y., *Fractal Image Compression — Theory and Application*, Springer-Verlag, New York, 1994.

[5] Saupe, D., Hamzaoui, R., Hartenstein, H., *Fractal image compression: an introductory overview*, in: Saupe, D., Hart, J. (eds.), *Fractal Models for Image Synthesis, Encoding and Analysis*, SIGGRAPH ’96 Course Notes XX, New Orleans, 1996.

[6] Barnsley, M., *Fractals Everywhere*, Academic Press, 1988.

[7] Garey, M. R., Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.