On String Attractors

Nicola Prezza[1]

Department of Computer Science, University of Pisa, Italy
nicola.prezza@di.unipi.it

Abstract. String attractors are combinatorial objects recently introduced as an universal framework for studying dictionary compressors. In this brief communication we summarize possible future developments and open problems related to the subject.

Keywords: String attractors · compressed computation

1 Introduction

Let S be a string of length n. Informally, a string attractor [5] for S is a set $\Gamma \subseteq [1..n]$ with the following property: any substring of S has at least one occurrence in S crossing at least one position in Γ. The following definition formalizes a generalized version of this concept.

Definition 1 (String k-attractor [5]). *A* k-attractor *of a string $S \in \Sigma^n$ is a set of positions $\Gamma \subseteq [1..n]$ such that every substring $S[i..j]$ with $i \leq j < i + k$ has at least one occurrence $S[i'..j'] = S[i..j]$ with $j'' \in [i'..j']$ for some $j'' \in \Gamma$.*

When $k = n$, we simply call Γ an attractor *of S.*

String attractors were originally introduced as a unifying framework for known dictionary compressors: Straight-Line programs [7] (context-free grammars generating the string), Collage Systems [6], Macro schemes [10] (a set of substring equations having the string as unique solution; this includes Lempel-Ziv 77 [8]), the run-length Burrows-Wheeler transform [2] (a string permutation whose number of equal-letter runs decreases as the string's repetitiveness increases), and the compact directed acyclic word graph [1, 3] (the minimization of the suffix tree). As shown in [5], any of the above compressed representations induces a string attractor of the same asymptotic size [5]. The other way round also holds for a subset of the above compressors: given a string attractor of size γ, one can build a compressed representation of size $O(\gamma \log(n/\gamma))$. These reductions imply the following interesting facts:

(1) We can asymptotically relate the output sizes of different compressors.
(2) We can design universal compressed data structures (i.e. working on top of any of the above compressors).

With respect to point (2), it can be shown that one can perform optimal-time random access within $O(\gamma \text{ polylog } n)$ space [5]. Similarly, fast text indexing can be achieved within the same space [9].

Computing a minimum k-attractor is NP-complete for general alphabets and $k \geq 3$, and APX-complete for constant $k \geq 3$ [5]. If the alphabet and k are small, however, the problem admits polynomial-time solutions [4].

Verification and approximation, on the other hand, are much easier problems: we can decide whether Γ is a proper k-attractor in optimal $O(n)$ time and space [4] and, with a simple reduction to set-cover, we can compute a $O(\log k)$ approximation of the smallest k -attractor in polynomial time [5].

In the following sections we list some interesting open problems and promising future developments related to these new combinatorial objects. We divide the exposition in four sections: complexity and approximability problems, combinatorial problems (enumeration of attractors and relations with the string's complexity), generalizations (to trees, graphs, grids), and algorithmic problems (universal compressed data structures and quick algorithms on attractors).

2 Complexity and Approximability

The first open problem that we discuss is the complexity of 2-attractor:

Problem 1. Is the (decisional version of) 2-attractor solvable in polynomial-time?

The reason why Problem 1 is so interesting is that k-attractor is NP-complete for $k \geq 3$, and one can easily verify that 1-attractor can be solved in linear time with a naive algorithm. The only remaining case, $k = 2$, is still open. Interestingly, if we restrict our attention to just substrings of length 2 (not ≤ 2), then the corresponding problem — dubbed *sharp-2-attractor*, can be solved in polynomial time [4].

A related problem is that of improving the dependency on the alphabet size in the original NP-completeness proof of [5] (which used polynomial alphabets):

Problem 2. Is k-attractor still NP-complete for alphabets of size $\sigma = n^{o(1)}$?

Note that k-attractor can be solved in polynomial time when $\sigma^k \log \sigma^k \in O(\log n)$ [4], so the NP-completeness proof should somehow put some lower-bound restriction on σ (in particular, note that 3-attractor is not NP-complete in the binary case).

The next problems are concerned with (in-)approximability. To start with, all polynomial-time approximation algorithms we know of achieve rate $O(\log k)$ (including reductions from dictionary compressors).

Problem 3. Can we compute a $o(\log k)$-approximation of the smallest k-attractor in polynomial time?

The n-attractor problem is deeply related to set-cover [5], for which it is known that no $o(\log n)$-approximation can be computed in polynomial time. A natural question is therefore whether this inapproximability result transfers to the minimum attractor problem. Note that the linear-time algorithm Lempel-Ziv 77 achieves $O(\log n)$ approximation; such an inapproximability result would

therefore imply that LZ77 is optimal among polynomial-time dictionary compressors. The best inapproximability result we know so far is that k-attractor cannot be approximated within factor $< 11809/11808$ in polynomial time, so we get the additional open problem:

Problem 4. Can we approximate k-attractor within factor $11809/11808 + \epsilon$ in polynomial time, for any $\epsilon \geq 0$?

3 Combinatorics

String attractors raise (quite naturally) several questions of combinatorial nature. One of the most intriguing open problems in this case is that of representing the string in attractor-space. All random access data structures provided in [5], as well as the reductions to compressed formats, achieve $O(\gamma \log(n/\gamma))$ space (in words of $\Theta(\log n)$ bits). In what follows, let γ be the size of a smallest attractor for the string. We are left with the question:

Problem 5. Can we encode a string within $o(\gamma \log(n/\gamma) \log n)$ bits of space? is this space below the Kolmogorov complexity for some infinite family of strings?

A related question, important to understand the power of these objects (as well as designing smaller data structures), is:

Problem 6. Does there exist an infinite family of strings such that $\gamma = o(b)$, where b is the size of the smallest bidirectional macro scheme? [10]

Macro schemes [10] are the most general and powerful dictionary compressors known to date. Since fast random access can be supported within $O(b \log(n/b))$ space, a positive answer to the above question would imply smaller random access data structures (of size $o(b \log(n/b))$ for some strings).

A similar problem is that of the relation between the number r of equal-letter runs in the Burrows-Wheeler transform and γ:

Problem 7. Does it hold that $r \in O(\gamma \text{ polylog } n)$?

Note that the above relation holds for all other known dictionary compressors [5] (except CDAWG). A positive answer to Problem 7 would immediately prove new relations between dictionary compressors (e.g. is $r \in O(z \text{ polylog } n)$, z being the size of the LZ77 parse?).

A natural combinatorial question, related to the previous ones, is:

Problem 8. How many strings in Σ^n have a smallest attractor of size γ?

Let $f(\gamma, n, \sigma)$ be the size of this class of strings (i.e. strings of length n, on alphabet of size σ, and with smallest attractor of size γ). Note that the class with $\gamma = 1$ is precisely that of unary strings, therefore $f(1, n, \sigma) = \sigma$. Now, a simple information-theoretic argument shows that any encoding for this class will use, in the worst case, at least $\log_2(f(\gamma, n, \sigma))$ bits. This implies that no string can be represented within $o(\log_2(f(\gamma, n, \sigma)))$ space (see problem 5).

4 Generalizations

It is natural to try generalizing string attractors to more complex objects such as trees, graphs, and grids. Here we give a tentative formulation for (labeled) graphs:

Definition 2 (Graph attractor). *A graph attractor is a set Γ of edges such that every connected subgraph has an isomorphic occurrence crossing at least one of the edges in Γ.*

Note that this formulation coincides with Definition 1 in the case of undirected labeled path graphs. Note also that the verification problem on graphs (is Γ a "graph attractor"?) now becomes much harder due to the NP-completeness of subgraph isomorphism. Indeed, it is easy to see that the verification problem itself becomes NP-complete: given an (unlabeled, undirected) graph $G = \langle V, E\rangle$, then G contains a subgraph isomorphic to a given (disjoint) graph H if and only if $\Gamma = E$ is a graph attractor for the graph $G \cup H$. Nevertheless, Definition 2 could be useful to approach the hard problem of graph compression, for which only few techniques exist:

Problem 9. Can we use Definition 2 to compress a graph G? (say, within space $O(|\Gamma|$ polylog $|\mathrm{G}|)$).

Similarly, a natural generalization of string attractors could be useful to compress repetitive sets of multi-dimensional points (e.g. two-dimensional grids). In this case, the generalization is more straightforward:

Definition 3 (Grid attractor). *An attractor for a grid on $[1..n] \times [1..n]$ is a set Γ of points such that every sub-grid has an occurrence crossing at least one of the points in Γ.*

It is actually not hard to generalize the random access data structure of [5, Thm 5.3] to support access on grids within $O(|\Gamma|$ polylog n) space, where Γ is a grid attractor as defined in Definition 3. The same idea can be generalized to arbitrary dimension $d \geq 2$. We leave this extension as future work.

5 Algorithms and Data Structures

To conclude this communication, we consider problems related to algorithms and data structures on string attractors. A natural research direction is to explore which queries can be supported on these combinatorial objects. As seen above, random access [5], as well as text indexing [9], can be supported in $O(\gamma$ polylog n) space. While in the former case the provided upper-bound matches the lower bound, the problem of optimality remains open for text indexing:

Problem 10. Can we locate patterns in $O(m + occ)$ time within $O(\gamma$ polylog n) space, where m is the pattern length and occ is the number of occurrences?

Similarly, no structures have been proposed yet for queries such as RMQ (range minimum), rank, and select. An interesting research direction could also be that of devising dynamic data structures based on string attractors (e.g. strings with insert, delete, and access operations).

As far as verification and optimization are concerned, it is still not known whether we can check that a given set Γ is a valid string attractor in optimal $O(n/\log_\sigma n + \gamma)$ time and space (this problem was addressed in [4]). The same holds for exact algorithms for computing a minimum attractor: the Fixed Parameter Tractable algorithm described in [4, Thm 24] runs in $O(n) + exp(O(\sigma^k \log \sigma^k))$ time. Is it possible to reduce the dependency on k and σ?

We conclude the section with a problem of practical importance. The greedy approximation algorithm for k-attractor discussed in [5] (i.e. iteratively pick the text position that covers the largest number of uncovered substrings) achieves a $O(\log k)$ approximation factor, but a naive implementation runs in $O(n^3)$ time. Implementing efficiently this approximation (say, in $O(n \log n)$ time) would be extremely valuable to build more space-efficient data structures in practice.

References

1. Anselm Blumer, Janet Blumer, David Haussler, Ross McConnell, and Andrzej Ehrenfeucht. Complete inverted files for efficient text retrieval and analysis. *Journal of the ACM*, 34(3):578–595, 1987.
2. Michael Burrows and David J. Wheeler. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.
3. Maxime Crochemore and Renaud Vérin. Direct construction of compact directed acyclic word graphs. In *Combinatorial Pattern Matching (CPM)*, pages 116–129. Springer, 1997.
4. Dominik Kempa, Alberto Policriti, Nicola Prezza, and Eva Rotenberg. String Attractors: Verification and Optimization. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
5. Dominik Kempa and Nicola Prezza. At the Roots of Dictionary Compression: String Attractors. In *Annual Symposium on Theory of Computing (STOC)*, pages 827–840. ACM, 2018.
6. T. Kida, T. Matsumoto, Y. Shibata, M. Takeda, A. Shinohara, and S. Arikawa. Collage system: A unifying framework for compressed pattern matching. *Theor. Comput. Sci.*, 298(1):253–272, 2003.
7. John C. Kieffer and En-Hui Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.
8. A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Trans. Information Theory*, 22(1):75–81, 1976.
9. Gonzalo Navarro and Nicola Prezza. Universal Compressed Text Indexing. *Theoretical Computer Science*, 2018.
10. James A. Storer and Thomas G. Szymanski. Data compression via textual substitution. *Journal of the ACM*, 29(4):928–951, 1982.