Bounded Kolmogorov Complexity Based on Cognitive Models

Claes Strannegård[1], Abdul Rahim Nizamani[2], Anders Sjöberg[3], and Fredrik Engström[4]

[1] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Sweden and Department of Applied Information Technology, Chalmers University of Technology, Sweden
claes.strannegard@gu.se
[2] Department of Applied Information Technology, University of Gothenburg, Sweden
abdulrahim.nizamani@chalmers.se
[3] Department of Psychology, University of Stockholm, Sweden
anders.sjoberg@psychology.su.se
[4] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Sweden
fredrik.engstrom@gu.se

Abstract. Computable versions of Kolmogorov complexity have been used in the context of pattern discovery [1]. However, these complexity measures do not take the psychological dimension of pattern discovery into account. We propose a method for pattern discovery based on a version of Kolmogorov complexity where computations are restricted to a cognitive model with limited computational resources. The potential of this method is illustrated by implementing it in a system used to solve number sequence problems. The system was tested on the number sequence problems of the IST IQ test [2], and it scored 28 out of 38 problems, above average human performance, whereas the mathematical software packages Maple, Mathematica, and WolframAlpha scored 9, 9, and 12, respectively. The results obtained and the generalizability of the method suggest that this version of Kolmogorov complexity is a useful tool for pattern discovery in the context of AGI.

Keywords: artificial general intelligence, cognitive model, Kolmogorov complexity, pattern discovery.

1 Pattern Discovery

Pattern discovery is a general cognitive ability that is critical to the survival of animals as a heuristic principle for finding food and shelter. It also plays a central role in human everyday life, aiding in decision making and predictions, and in application areas such as science and finance. Moreover, pattern discovery is crucial for performing on the human level and beyond on progressive matrix problems and number sequence problems appearing in IQ tests [3–6]. In fact,

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 130–139, 2013.

some IQ tests, *e.g.*, Raven's progressive matrices [5] and PA [4], consist exclusively of such problems.

For these reasons among others, automatic pattern discovery is a central challenge to AGI. Since both humans and machines have limited computational resources, a useful side-condition here is that all patterns need to be identifiable with limited computational resources. In this paper we focus on a special kind of pattern discovery. In fact we shall consider patterns in number sequences and aim for average human-level performance and beyond in this particular case. We shall use a method for pattern discovery which is relevant to AGI since it generalizes readily beyond the case of number sequences.

1.1 Number Sequence Problems

In the context of IQ tests, a *number sequence problem* is a finite sequence of integers [7]. A unique *answer* integer is associated with each problem. For instance, the (Fibonacci-inspired) number sequence problem $2, 2, 4, 8, 32$ has the answer 256. The answer is unique as it is the only number that yields points on the IQ test. Manuals accompanying IQ tests commonly state that the participants should be instructed to find the number that "fits the pattern." The criteria used to judge whether the answer fits the pattern are usually not explained further, but rather are illustrated by examples.

Many mathematical methods have been proposed to solve number sequence problems. The methods include neural network models, hidden Markov models, dynamic programming models, reinforcement learning models, graph theoretical models, evolutionary models, and symbolic planning models [8–10]. Several studies have explored human problem solving in relation to letter sequences and number sequences [11–13]. Mathematical software packages that include specialized functions for number sequence problems are considered in Section 4.

1.2 Kolmogorov Complexity

A common approach to the analysis of patterns in number sequences and other structures is Kolmogorov complexity and its variations, particularly those of Levin and Solomonoff [1]. The idea is that the shortest description of an object captures its essence. A version of Kolmogorov complexity $K(x)$ of an object x could be defined as the length of the shortest program p (in a given programming language) that outputs x and then halts. Because of the halting problem, Kolmogorov complexity is not computable and thus is not useful as a basis for automatic pattern discovery. A computable version of Kolmogorov complexity is Levin's Kt, defined as a combination of the program length and the number of steps that the program executes before halting [1]. This approach has been used in applications including the generation of IQ tests [14]. Other versions of Kolmogorov complexity have been defined in terms of the computational resources (time and space) used by universal Turing machines, including certain measures that are used in simplicity theory [15–17].

1.3 Structure of the Paper

Section 2 defines a version of Kolmogorov complexity in terms of a cognitive model of a human problem solver. In Section 3, a computational model for number sequence problems is constructed based on this complexity measure. In Section 4, this computational model is benchmarked on the IST IQ test. Sections 5 and 6 present the discussion and conclusions, respectively.

2 Bounded Kolmogorov Complexity

Patterns are subjective in the sense that different people can observe different patterns in the same structure, as illustrated by the Rorschach tests [18]. As a further example, consider the number sequence 1,2. What might the next number be? Is it 1 (considering the repetitive sequence 1,2,1,2), 2 (considering the mirroring sequence 1,2,2,1), 3 (considering the incremental sequence 1,2,3,4), or 4 (considering the doubling sequence 1,2,4,8)? These observations indicate that (i) number sequence problems are not strictly mathematical problems, and that (ii) number sequence problems have subjective components, *i.e.*, components that depend on the subject. This subjective component led us to introduce a cognitive model of the subject and to focus exclusively on the patterns and computations that are available within that cognitive model.

Our main strategy for pattern discovery is as follows [19, 20]. Let Alice be a human problem solver. Introduce (i) a language modeling the terms that Alice might use to describe the patterns, (ii) a term rewriting system (TRS) modeling how Alice might compute the terms of this language, and (iii) a bounded version of this TRS, obtained by adding resource bounds reflecting Alice's cognitive resources. Finally, look for the smallest term that computes the object in question in the bounded TRS.

In the specific case of number sequence problems, the strategy is as follows:

1. Construct a language consisting of terms that describe number sequences. For example, the sequence $2, 2, 4, 8, 32, \ldots$ is described by the term $f(n-2) * f(n-1)$.
2. Define a TRS to compute terms $t(n)$ with numerical values inserted for n. For example, $8 * 32$ and $32 * 256$ may be computable in this TRS.
3. Define a bounded version of this TRS reflecting the bounded cognitive resources of Alice. For example, $8 * 32$ may be computable in this bounded TRS, while $32 * 256$ may not.
4. Given an input sequence $a_0, a_1, \ldots, a_n$, check whether a shortest term $t(n)$ exists such that (1) $t(i)$ reduces to a_i in the bounded TRS for $0 \leq i \leq n$, and (2) $t(n+1)$ reduces to some number a in the bounded TRS. If such a term exists, then use a preference relation to select one such $t(n)$ and output the corresponding a. Otherwise, report failure for that sequence.

3 Computational Model

This section will provide details on the computational model used to solve number sequence problems, as outlined above.

3.1 Terms

Terms are strings of symbols that are intended to describe number sequences.

Definition 1 (Term). *The set of terms is defined by the following clauses:*

- *Numerals:* $0, 1, 2, \ldots$, *are terms.*
- *The variable* n *is a term.*
- $f(n - c)$ *is a term, where* c *is a numeral greater than* 0.
- *If* t_1 *and* t_2 *are terms, then so is* $(t_1 \star t_2)$, *where* $\star$ *is one of* $+$, $-$, $\cdot$, *or* $\div$.

Parentheses are sometimes omitted in order to facilitate reading. Informally, a term describes a number sequence if all numbers in the sequence except for a limited number of base cases can be calculated using the term in which the functional symbols $+, -, \cdot$, and $\div$ are interpreted as addition, subtraction, multiplication, and integer division; n is interpreted as the position in the sequence and $f(n - c)$ as the number at position $n - c$.

In the formal definition below, it is helpful to think of b as the number of base cases. To exclude trivialities, we require this number b to be less than half of the sequence length.

Definition 2 (Description). *Let* t *be a term and let* b *be the largest* c *such that* $f(n - c)$ *occurs in* t. *Then* t *is a* description *of the number sequence* $a_0, a_1, \ldots, a_{l-1}$ *if the following holds:*

- $b < l/2$,
- *If* $b \leq i < l$, *then* $t(i)$ *evaluates to* a_i *when* $f(j)$ *is interpreted as* a_j, *for* $0 \leq j < i$.

Example 1. The term $f(n-2) \cdot f(n-1)$ is a description of the sequence $2, 2, 4, 8, 32$.

Example 2. The term $f(n - 1) + k$ is a description of the arithmetic sequence $a, a + k, \ldots, a + m \cdot k$ and $f(n - 1) \cdot k$ is a description of the geometric sequence $a, a \cdot k, \ldots, a \cdot k^m$.

Suppose t is a description of the number sequence $a_0, a_1, \ldots, a_{l-1}$. Then we say that t *predicts* the number $t(l)$ for this sequence.

Example 3. As already mentioned, the term $f(n - 2) \cdot f(n - 1)$ is a description of the sequence $2, 2, 4, 8, 32$. It predicts the number 256 for this sequence.

3.2 Bounded Computations

In this section, we define a simple cognitive model in the form of a TRS together with a notion of bounded computation for that TRS. The terms of the TRS are those that were introduced above. The rewrite rules are given by a set of simple arithmetic facts, including 10×10 tables for addition, multiplication, subtraction, and integer division, together with a small number of algebraic rules. For instance, the TRS contains the rules $5 + 7 \to 12$, $2 \cdot 3 \to 6$, and $x \cdot (y + z) \to x \cdot y + x \cdot z$, where x, y, z represent arbitrary terms. This TRS is similar to the one used in [19]. Examples of computations (rewrite sequences) in this TRS are given in Examples 4 and 5.

Example 4. Here is a simple arithmetic computation.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{27 \cdot 3}{(30-3)\cdot 3}}{30\cdot 3 - 3\cdot 3}}{30\cdot 3 - 9}}{90-9}}{80+10-9}}{\dfrac{80+1}{81}}$$

Next we shall define a basic notion of bounded computation, by bounding the maximum length of the terms appearing in the computations.

Definition 3 (Term length). *The length $|t|$ of a term t is defined by:*

- *$|c|$ is the number of digits in c, disregarding trailing zeroes, when c is a numeral.*
- $|n| = 1$
- $|f(n-c)| = 1$
- *If t_1 and t_2 are terms, then $|(t_1 \star t_2)| = |t_1| + |t_2| + 1$.*

We put $|f(n-c)| = 1$, since in practice c will be a very small number (seldom larger than 2), telling you how far "to the left to look" in a given pattern. It seems natural to assume that this number is not kept in the working memory.

Definition 4 (Bounded computation). *A* bounded computation *is a computation in which the length of each term is no more than eight.*

The number eight was chosen to model a human problem solver with a working memory capacity above the average level. According to Miller [21], the working memory can typically hold about seven items. Later findings suggest that the working memory capacity is about four items in young adults and less than that in children and elderly [22].

The computations given in Examples 4 and 5 are both bounded. Complex terms such as $67 \cdot 89$, do not have bounded computations, however.

When searching for terms that describe a given number sequence, we shall restrict the search to bounded computations, in the sense that every number of the sequence must be computable via a bounded computation. This restriction also applies when computing the predicted number.

Example 5. Note that the term $f(n-1)+f(n-2)$ is a description of the sequence $2, 3, 5, 8, 13, 21$. One may verify that each element of this sequence, and also the predicted number 34, can be computed by means of bounded computations. For example, the following is a bounded computation of the element a_5:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{8+13}{8+10+3}}{10+8+3}}{10+11}}{10+10+1}}{\dfrac{20+1}{21}}$$

3.3 Subsequences

Some number sequences appearing in IQ tests are described by more than one term, with each term describing a subsequence of the full sequence.

Example 6. Consider the sequence $1, 2, 1, 3, 1, 4, 1, 5$, which contains the following subsequences:

- odd-positioned elements $1, \_, 1, \_, 1, \_, 1, \_$
- even-positioned elements $\_, 2, \_, 3, \_, 4, \_, 5$

In this example, to predict the next number it is enough to consider the odd positions. However, one subsequence could also work as the base case for the other subsequence, as in the following example.

Example 7. Every second position of the sequence $5, 10, 1, 2, 22, 44, 3, 6, 7, 14$ is described by $f(n-1) \cdot 2$.

Let us define what it means for a term t to describe every second number of a sequence by a slight variation of Definition 2.

Definition 5 (Description modulo 2). *Let t be a term and let b be the largest c such that $f(n-c)$ occurs in t. Then t is a* description modulo 2 *of the number sequence $a_0, a_1, \ldots, a_{l-1}$ if the following holds:*

- $b < l/2$,
- *If $b \leq i < l$ and $i \equiv l \pmod 2$, then $t(i)$ evaluates to a_i when $f(j)$ is interpreted as a_j, for $0 \leq j < i$.*

Descriptions modulo $m > 2$ are defined analogously. The notion of prediction extends in the expected way. Suppose t is a description modulo m of the number sequence $a_0, a_1, \ldots, a_{l-1}$. Then we shall say that t *predicts* the number $t(l)$ modulo m.

3.4 Index Shifts

The variable n in a term is interpreted as the position in a sequence, with the first position being 0. However, consider the following two sequences:

- $0, 1, 4, 9, 16$
- $1, 4, 9, 16, 25$

The first sequence is described using the term $n \cdot n$ of length 3, while the second is described by $(n+1) \cdot (n+1)$ of length 7. To avoid this pathology, we allow the starting value of n to vary from -9 to 9.

3.5 Term Preference

As the following example illustrates, different terms can describe the same sequence, which sometimes results in different predictions.

Example 8. The sequence $4, 7, 12, 20$ is described by both

- $f(n-1) + f(n-2) + 1$ and
- $(f(n-1) - f(n-2)) \cdot 4$.

Both these terms are of length 5, but they make different predictions, 33 and 32, respectively.

Because of such ambiguities, we need to define a preference order on the terms that describe a given sequence:

- Shorter terms are preferred over longer terms. This is motivated by Occam's razor [23].
- Descriptions are preferred over descriptions modulo 2, which in turn are preferred over descriptions modulo 3, and so on. This ensures that complete descriptions are preferred over partial descriptions.
- Terms not including n (except as $f(n-c)$) are preferred over terms containing n. This is motivated by the view that previous values are more basic than positions [24].
- Terms predicting smaller values are preferred over others.

These criteria, listed according to their priority, ensure that a unique prediction will always be made. Performance decreased when these criteria were applied in a different order than the one listed above.

3.6 Implementation

This model was implemented in a computer program written in Haskell. The program searches for a term that describes a given number sequence, using the preference order of terms listed above. Since the length of terms is the primary preference, the program iterates over each length beginning with length one. For each length, it enumerates all terms and tests them against the sequence. Terms that describe the sequence are chosen, and one of them is selected using the preference order listed previously. This term is then used to extrapolate the sequence.

4 Results

To assess the performance of our computational model, we tested it against some widely used mathematical software tools that have special functions for solving number sequence problems. The following tools were used in the test:

- Seqsolver, as described above
- Mathematica [25] with `FindSequenceFunction` [26]
- Maple [27] with the `listtorec` function [28]
- Online Encyclopedia of Integer Sequences [29]
- WolframAlpha [30] with the online search function (which is connected to the Online Encyclopedia of Integer Sequences).

Table 1: Performance on the number sequence problems of the IQ-tests PJP (11 items), PA (29 items), and IST (38 items) of the above-mentioned tools. SeqSolver was developed with the help of the tests PJP and PA (which explains the good performance on those tests). Then it was tested using the (previously unseen) test IST.

Program	**PJP**	**PA**	**IST**
SeqSolver	11	29	28
Mathematica	6	9	9
Maple	6	6	9
OEIS	4	11	11
WolframAlpha	6	9	12

We tested the above tools on the number sequence problems of the IQ tests PJP [3], PA [4], and IST [2]. There was a time-limit of 30 seconds per item, motivated by the time limit of 30 minutes for the IST test. Table 1 summarizes the results. The IQ score of SeqSolver on IST was at least 130 (a precise IQ score could not be given in this case because of the limited size of the norm group) and for WolframAlpha it was 99 [31]. Decreasing the working memory capacity from 8 to 7 resulted in a decreased score for SeqSolver and increasing it to 9 did not change the score, although it increased the runtime considerably.

5 Discussion

In general, introducing a cognitive model into a pattern discovery system serves several purposes. First, it provides a notion of which patterns are amenable to human-level problem solving. Second, the cognitive model may expedite the problem solving process. In fact, a limited amount of time can be spent on each term, as all computations take place inside a finite cognitive model. Third, this approach may increase the chance of solving the problem "correctly," as any terms that are too computationally demanding can be excluded. Finally, this model provides a notion of cognitive workload, which can serve as a tie-breaker between terms of the same length.

This study used a particularly simple notion of patterns. This notion could be broadened by adding more powerful operators. The same strategy would work in that case, as all computations (terminating or not) will be aborted as soon as they exceed any of the resource limits.

6 Conclusion

We defined a version of Kolmogorov complexity by constructing a cognitive model of a human problem solver. This measure of complexity involves only those computations that can be performed inside the cognitive model, with its limited cognitive resources. This approach resulted in a set of computations that

is small enough to be computable, yet large enough to contain a substantial part of the computations that are necessary for human level performance. This complexity measure differs from other proposed computable versions of Kolmogorov complexity as it makes explicit reference to models of human cognition.

The notion of bounded Kolmogorov complexity and the method presented here are by no means restricted to number sequence problems. In fact, the method applies analogously to other types of objects than numbers and other types of operators than arithmetical. More precisely, the method applies to arbitrary term rewriting systems or production systems with finite computational resources.

In summary, the results obtained and the generalizability of the method suggest that resource-bounded Kolmogorov complexity is a useful tool for pattern discovery in the context of AGI.

References

1. Li, M., Vitányi, P.: An introduction to Kolmogorov complexity and its applications, 3rd edn. Springer (2008)
2. Amthauer, R., Brocke, B., Liepmann, D., Beauducel, A.: Intelligenz-Struktur-Test 2000 R, IST 2000R (2001)
3. Sjöberg, A., Sjöberg, S., Forssén, K.: Predicting Job Performance. Assessio International, Stockholm (2006)
4. Personaladministrativa Rådet: PA Number series, Stockholm, Sweden (1969)
5. Raven, J.C.: Standard Progressive Matrices Sets A, B, C, D & E. Oxford Psychologists Press Ltd. (1990)
6. Raven, J.C.: Advanced Progressive Matrices Set I. Oxford Psychologists Press Ltd. (1990)
7. Jensen, A.R.: Bias in mental testing. Free Press, New York (1980)
8. Sun, R., Giles, C.L. (eds.): Sequence Learning - Paradigms, Algorithms, and Applications. Springer, London (2001)
9. Bhansali, A., Skiena, S.S.: Analyzing integer sequences. In: Computational support for Discrete Mathematics: DIMACS Workshop, March 12-14, pp. 1–16. American Mathematical Society (1994)
10. Tetruashvili, S.: Inductive inference of integer sequences. Senior thesis, Computer Science Department - CarnegieMellon University (2010)
11. Holzman, T.G., Pellegrino, J.W., Glaser, R.: Cognitive variables in series completion. Journal of Educational Psychology 75(4), 603 (1983)
12. LeFevre, J.A., Bisanz, J.: A cognitive analysis of number-series problems: Sources of individual differences in performance. Memory & Cognition 14, 287–298 (1986)
13. Haverty, L.A., Koedinger, K.R., Klahr, D., Alibali, M.W.: Solving inductive reasoning problems in mathematics: not-so-trivial pursuit. Cognitive Science 24(2), 249–298 (2000)
14. Hernandez-Orallo, J.: Beyond the Turing test. Journal of Logic, Language and Information 9(4), 447–466 (2000)
15. Chater, N.: The search for simplicity: A fundamental cognitive principle? The Quarterly Journal of Experimental Psychology: Section A 52(2), 273–302 (1999)
16. Chater, N., Vitányi, P.: Simplicity: a unifying principle in cognitive science? Trends in Cognitive Sciences 7(1), 19–22 (2003)
17. Dessalles, J.L.: Algorithmic simplicity and relevance. arXiv preprint arXiv:1208.1921 (2012)

18. Exner Jr, J.E.: The Rorschach: A comprehensive system. John Wiley & Sons Inc. (2003)
19. Strannegård, C., Amirghasemi, M., Ulfsbäcker, S.: An anthropomorphic method for number sequence problems. Cognitive Systems Research 22, 27–34 (2013)
20. Strannegård, C., Cirillo, S., Ström, V.: An anthropomorphic method for progressive matrix problems. Cognitive Systems Research 22, 35–46 (2013)
21. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review 63(2), 81 (1956)
22. Cowan, N.: Working memory capacity. Psychology Press, New York (2005)
23. Li, M., Vitányi, P.M.B.: An introduction to Kolmogorov complexity and its applications. Springer (2008)
24. Siebers, M., Schmid, U.: Semi-analytic natural number series induction. In: Glimm, B., Krüger, A. (eds.) KI 2012. LNCS, vol. 7526, pp. 249–252. Springer, Heidelberg (2012)
25. Mathematica: Version 8.0. Wolfram Research Inc., Champaign, Illinois (2012)
26. Wolfram Research, Inc.: Integer Sequences (2013), Published electronically at `http://reference.wolfram.com/mathematica/ref/FindSequenceFunction.html`
27. Maple: Version 16.0. Maplesoft, Waterloo ON, Canada (2012)
28. Salvy, B., Zimmermann, P.: GFUN: a Maple package for the manipulation of generating and holonomic functions in one variable. ACM Transactions on Mathematical Software (TOMS) 20, 163–177 (1994)
29. Sloane, N.J.A.: Online Encyclopedia of Integer Sequences (2005), Published electronically at `http://oeis.org`
30. Wolfram Research, Inc.: Wolfram Alpha (2012), Published electronically at `http://wolframalpha.com`
31. Liepmann, D., Beauducel, A., Brocke, B., Amthauer, R.: Intelligenz-struktur-test 2000 r. manual. Hogrefe, Göttingen (2007)