KitBit: An AI Cognitive Model for Solving Intelligence Test and Numerical Series

Víctor Corsino, José Manuel Gilpérez, and Luis Herrera

Abstract—The resolution of intelligence tests and in particular the numerical sequences have been analyzed through different cognitive models and other approaches. In this article we present a new artificial intelligence computational model called KitBit, which aims to mimic some of the reasoning process that humans use to solve problems. KitBit uses a reduced set of actions and their combinations to build a predictive model that finds the underlying pattern of different numerical sequences, such as those included in intelligence tests and other much more complex. The system is capable of solving those problems in less than a second with standard computational power. The KitBit algorithms have been applied in the OEIS numerical sequence database, reaching the highest number of series solved to date. Likewise, it is capable of solving the intelligence tests reported by previous models, according to the literature. In this article we expose the fundamentals of the system and its application in finding the patterns of numerical series in general, and more specifically in cases where these series are part of intelligent tests, even though KitBit has demonstrated its ability to solve other types of Intelligence Quotient (IQ) problems, which will be addressed in later publications.

Index Terms—Pattern analysis, cognitive models, artificial intelligence.

1 Introduction

Intelligence is a broad concept that has been defined in many different and complementary ways [1], with human intelligence being the most evolved expression of natural intelligence. In general, it is accepted that an intelligent entity is one whose behavior reflects one or more of the qualities of intelligence. For example, for humans and even machines, a sequence of questions and answers can be a way to assess their intelligence, as well as other cognitive abilities, as suggested by the Turing test [2]. However, where the intelligence lies remains an open question, which is of special interest when we consider whether so-called artificial intelligence systems are really intelligent. In this sense, the most successful artificial intelligence systems to date have been designed to emulate human intelligence only in very specific tasks [3], [4]. However, the emulation of human behavior does not necessarily imply the emulation of the human processes of perception and cognition responsible for the production of intelligent knowledge.

Similarly, the production of intelligent knowledge does not necessarily imply the emulation of these cognitive processes. For example, AI is currently dominated by statistical models, which have unprecedented success in different domains, based on convolutional neural networks (CNN) [5], [6]. These systems make use of increasing computational power and a large amount of data. Then, we can ask if its mechanisms are similar to those that underlie the intelligent processes of the human brain. In this sense, artificial neurons lack the complexity of real neurons [7], as well as, in terms of efficiency, they require large infrastructures and high energy consumption, while the energy consumed by the human brain is equivalent to that of a small light bulb [8].

On the other hand, previous approaches of AI research, mainly from the 1980s, tried to create systems that simulate intelligent behavior or support the same capabilities as humans, being intimately related with cognitive science. The research in this area is focused on modeling the invariant aspects of human cognition and their implementation in cognitive architectures to create intelligent agents. Computational cognitive models are based on psychological observations that establish rules for how people solve problems [9]. The simulation attempts to emulate psychological processes through algorithms that, combined, complete the intelligent resolution of the problem [10]. Psychologists aim to discover the mechanisms underlying human reasoning and learning processes [11], while artificial intelligence researchers aim to emulate these processes to produce more efficient artificial reasoning and learning systems.

In this way, under the acronym KitBit, this work presents a new computational model that attempts to emulate the way of processing information by human intelligence, from a cognitive approach. The term KitBit implies the union of the concepts of Knowledge Imprint Tracer (Kit) and Behavior Imprint Tracer (Bit), which refer to the acquisition and production of intelligent knowledge and intelligent behavior, respectively. Thus, our model fits one of the four definitions of artificial intelligence proposed by S. Russell and P. Norvig [12], that is, a computational system that tries to think like an human (Kit) to obtain a behavior model derived from it (Bit). In this sense, in its current phase of development, KitBit tries to emulate the basic processes of acquisition and production of intelligent knowledge of the human brain, from which intelligent behavior would be derived. To test this ability, KitBit successfully solves the largest number of numerical series reported to date through a computer system, as well as other types of intelligence tests that will be reported in subsequent publications.

- *V. Corsino and J.M. Gilpérez are with School of Industrial and Aerospace Engineering - Universidad de Castilla La-Mancha, Avenida Carlos III s/n, Real Fábrica de Armas, 45071 Toledo, Spain.*
- *L.A. Herrera is CEO at ®Smart Technologies Investments.*

Manuscript received December 14, 2021; revised December 26, 2021.

2 KitBit model fundamentals

Intelligent knowledge production is closely related to pattern recognition as a fundamental faculty of human intelligence. Horn and Cattell [13] proposed two levels of human intelligence, fluid intelligence that faces new problems and which includes pattern recognition and abstract reasoning, and crystallized intelligence, based on experience and learning. In this way, the ability to identify and interpret patterns of variation in space and time, from which all experience is derived, is a fundamental quality of human intelligence [14]. Therefore, intelligence is determined by its predictive ability [15].

This predictive capacity arises from the intelligence ability to compare between before and after, establishing cause-effect or effect-cause relationships, especially the latter, filtering the world into categories and explicitly formulate ideas using nested or recursive symbol structures. Therefore, we could argue that, at a deep level, our brain is a differentiator. Faced with any event, represented by a set of data series in time, intelligence tries to find the order in which things happen. From the cardinality in the data, the ordinality in the facts is obtained, facilitating the anticipation in time. Nature adds and encrypts the information, while intelligence differentiates and deciphers that information. That would be the way we try to find out why things happen. The same comparison algorithm is also used to process data series in space. KitBit tries to emulate this type of basic algorithm that underlies all cognitive processes, assuming that the bearers of consciousness are patterns, not neurons [16].

In addition, our approach is based on a naturalistic conception of epistemology: human perception and the reasoning system obey a functional criterion of utility for the subject, the first utility for any individual being their ability to make precise predictions in the time or space. From this point of view, intelligent knowledge consists mainly in the identification of levels of description rich in patterns, categories, as well as patterns in time that allow anticipating events. On the other hand, intelligent behavior consists of ordering events, establishing priorities for the achievement of the selected objectives, modifying the order in which things happen seeking to maximize utility for the intelligent agent. Therefore, the cornerstone of intelligence would be its ability to make predictions about the future, being prediction, not behavior, the proof of intelligence [15]. From the perspective used by KitBit, prediction is a fundamental faculty of intelligence in which it is necessary to distinguish, handle and recursively relate two series of data. On the one hand, the one formed by the values of each variable in discrete and successive instants of time or in adjacent locations in space, and, on the other, those obtained by comparing each of these values with its immediate before or after, spatially or temporally.

These predictive processes take place in the neocortex, the area of the brain where we assume reasoning resides. It can be considered a complex self-associative biological memory, which stores patterns and sequences of patterns, and is capable of retrieving complete patterns from partial or distorted inputs [15], [17]. Anyone who has learned the pattern is able to extrapolate it indefinitely and therefore has learned the concept immersed in the series [18]. It is noteworthy that all regions of the cerebral cortex have a similar structure made up of millions of minicolumns. This observation led V.B. Mountcastle [19] to suggest that since these regions all look the same, *'maybe they are actually performing the same basic operation!'*. Likewise, Bach-y-Rita and Kercel [20], developed a sensory substitution method for blind people, showing visual patterns on the tongue. This capacity for sensory exchange suggests the existence of this basic interchangeable functional unit, used in all brain areas. In this way, the complexity of specific domains such as vision, hearing or tactile sensitivity, would be the emerging result of a general purpose circuit capable of making predictions in a unified way [21]. Therefore, the cerebral cortex would use a common function or algorithm, which is performed in all cortical regions. KitBit embraces this thesis. Starting from a basic algorithm, which we call *basic kita*, a collection of other different *kitas* is built. They are concatenated in a nested or iterated way to be applied to the variables that define a system or their combinations. Thus, this collection of *kitas* constitutes a dynamic *toolbox* of procedures to find patterns, which are assembled, selected and discarded according to their utility and energy consumption. This kind of *tool-plasticity* could be considered as a computational attempt to emulate the physical neuroplasticity of our brain. In a later stage, the analysis of the kinds of data, their patterns and the *kitas* used to identify them would allow to filter the experiences in categories rich in patterns. This patterns could also be found among the data, the different kinds of data or patterns and even the sequences of *kitas* themselves.

Among the psychological processes that KitBit tries to emulate are analogy and abstraction, as two fundamental aspects in the production of intelligent knowledge. The analogy has received the most attention from both cognitive psychologists and artificial intelligence researchers [22], [23]. It is an inductive mechanism that allows finding the common pattern between elements of different domains through a comparison mechanism. For example, events that occur in time are understood by analogy with objects that move in space. Abstraction, in turn, is an essential characteristic of any efficient knowledge acquisition system that allows determining characteristics common to many individuals. In this way, analogy and abstraction allow us to find common patterns between different situations or domains or to find similarities between their different patterns of variation.

This lead us to consider how, among the different possible patterns included in the incoming information, the cognitive process selects one of them. It begins by detecting some type of regularity in the perceived data, to then check if this first perception agrees with reality, discarding erroneous hypotheses. Our brain is continually discarding information that is irrelevant [24]. When the rule found is extremely complex, the brain considers it an irregularity and it is discarded [25]. Thus, the simplest is always selected as the best option. In the process, the brain tries different *ways of looking at the world*. To do this, it dynamically employs a focus procedure, regulating the resolution of the sequence, spatial or temporal, of the incoming data, using another procedure to frame, enlarge or reduce the focus in different frames of the data sequence. By varying these two parameters, different perspectives of the problem are obtained, in

order to select some of them from the pattern recognition algorithm. The selection criteria is based on determining the perspective richest in patterns, as well as the most useful according to the context and the purpose. Like the brain, our model approaches the problem from different perspectives to find patterns in the incoming data, evaluating if they are intelligible or not. Of these patterns, the one that needs the least energy costly combination of *kitas* to be obtained is selected. In addition, both the focus and the framing of the data are two of the main *kitas* of the model.

To find patterns we could define a level of description in time or space that provides a maximum local utility for the identification of regularities. That is, we choose levels of description fine enough to provide a meaning that allows us to understand and predict. One of the most important characteristic of intelligence is its ability to identify and select these categories. From this point of view, *kitas* are useful pattern-rich level-of-description seekers that try to identify these layers of emerging realities, somehow mimicking the processes by which complexity grows in nature.

Another important aspect in the production of intelligent knowledge is the ability to understand how it has been produced, in other words, its traceability. When using CNN, it is difficult to know exactly how the model solves the problem [26], [27]. The system is like a black box where it is not possible to know the sequence of operations it performs. However, using KitBit, the sequence of operations performed to discover the underlying pattern is always known. So, traceability, knowing the kind of sequences of operations that solve each type of problem, allows establishing relationships between both, which may serve as a guide to improve KitBit predictive ability to solve problems of increasing complexity.

Finally, KitBit has its own internal logic, being able to renounce the use of previous information, as well as great computational powers to locate complex patterns and make predictions in systems whose components are the values in successive instants of time, in adjacent positions in space, or in any of the state variables that characterize a system. Using this basic process, intelligent knowledge acquisition is demonstrated by extracting the underlying pattern from number series, particularly those that are part of IQ tests.

3 Resolution of IQ test and numerical series: previous models

Since the IQ is considered a predominant benchmarks for measuring human intelligence [28] [29], IQ tests are an interesting challenge for AI systems, as well as a tool for quantify how similar these systems are to human intelligence [30]. Among the different types of tests, IQ tests based on numerical sequences are one of the fundamental ones, since other tests can be transformed into a numerical problem. They can be defined mathematically by means of a function that assigns the natural numbers to the real numbers: $f : \mathbb{N} \to \mathbb{R}$ where each element is defined algorithmically. They are normally based on simple patterns or a combination of them, where the numbers are restricted to integer values, generally small to allow easy mental calculations using the four basic arithmetic operations. The number of elements is also relatively small, usually no more than 6 to 8 elements. Number sequences based on more complex mathematical functions, such as polynomials, are not considered in IQ sequence questions.

As we can see, IQ tests are focused on discovering the underlying pattern in the problem. Therefore, its resolution through an artificial intelligence system would indicate its ability to recognize these patterns. However, there are few publications related to the computational resolution of the IQ test in the conventional literature on AI [31]. In 1963, Evans and Simon on the one hand and Kotovsky on the other, designed AI programs capable of identifying regularities in patterns in analogy problems and letter series termination problems, respectively [32]. In the 1960s and 1970s, Fredkin, Pivar and Finkelstein, and Pivar and Gord, in early research on inductive learning, developed LISP programs that operated on integers and automatically discovered interesting relationships in the data, but could only work with a limited class sequence [33]. Later, in 1980, Hofstadter developed a series of computational models in the Copycat project with the main objective of understanding the analogy [32]. But it was between 2006 and 2011 when computational models aimed at solving IQ test problems became more popular, either trying to understand human cognition or as a method of evaluation of AI techniques [32]. Subsequently, between 2011 and 2014 different models were proposed to solve intelligence tests based on numerical series, although some of them performed, in general, worse than human beings in their tests [34].

More recently, Liu et at. [34] have conducted a comparison study of different state-of-the-art approaches to solve IQ test automatically. They collected an extensive dataset of IQ test, containing 10000 questions of different type, where above 2500 are numerical sequences. Other studies are focused in more general numerical series datasets. For instance, Ragni and Klein [35] tested their model, based on an artificial neural network (ANN), in series selected from the Online Encyclopedia of Integer Sequences (OEIS) [36], [37]. At that time, the OEIS database contained 187440 number series, from where they selected 57524 which consist of at least 20 numbers with values ± 1000. Using their approach they solved 26951 of the selected number series. On the other hand, Siebers and Schmid [38] avoid to use the OEIS database to test their semi analytical cognitive model, considering that most of the OEIS series are too complex to be induced by human. Instead, they used a random generator [39] to construct 25000 number series using addition, subtraction, division, multiplication and exponentiation. Based on this dataset, the hit rate to induce the next three numbers in the series is 93.2%.

Other approaches use a very limited number of series to test their models. They are summarized in the excellent reviews provided by Schmid and Ragni [26] and Hernández-Orallo et al. [40]. The latter provides a complete account of some thirty computer models focused on solving intelligence tests, reviewing their purpose, their degree of specialization and the techniques used, as well as their achievements and the relationship between them. Meredith [41] evaluates the Seek-Whence model using only 12 numerical series, the blackburn dozen, to solve the Bongard problem [42]. It uses pattern recognition and analogy according the ideas of Hofstadter [43] without requiring

typical mathematical operations. In the same line of ideas, Mahabal [44] developed SeqSee, a computer program that emulates intelligent activities to extend integer sequences, avoiding the use of brute force or computing shortcuts implausible in humans. He analyzes 146 selected numerical series related using the group of intelligent activities emulated. Sanghi and Dowe [30] presented a simple computer program which was able to solve easily IQ tests focussed on pattern recognition of certain types of numerical sequences, as arithmetic progression, geometric progression, Fibonacci series or powers of a series. Their results are based in 12 IQ test which are not presented. Burghardt [45] applied the algorithm called E-generalization to discover the pattern underlying numerical series, presenting results in 9 series, including alternating and Fibonacci series. Strannegård et al. [46] implemented ASolver, a rule-based system which include memory restrictions arguing similarity with the human capabilities. The system performance was evaluated with 11 (non published) IQ test. Hofmann et al. [27] developed the analytical inductive programming system IGOR2 and compare its achievements with Ragni and Klein [35] and Burghardt [45] approaches based on 20 and 8 selected series, respectively. Finally, they selected 100 series, some of them included in OEIS to test the model.

As a conclusion, we can say that the computational models created to induce the pattern in numerical series, whether considered a problem in an IQ test or a numerical series in general, use, with few exceptions, a very small number of examples to test the model. On the other hand, even in those cases where the number of examples is large enough, they are selected within certain restrictions. In the case of KitBit, it tries to resolve the entire OEIS database - currently over 350000 series - and all the numerical series used as tests in the different models found in the literature, as well as new tests not tested so far. In this way, we will demonstrate that KitBit is capable of solving the largest number of numerical series of different types to date, laying a solid foundation to face other types of problems that will be presented in future publications.

4 The KitBit model

The KitBit model is based on four fundamental components:

- The basic data storage unit or *edk* (Experience Derivated Knowledge).
- The *kitas* (Knowledge Imprint Tracer Algorithm) or actions carried out on the *edk*.
- The pattern search system.
- The new element prediction system.

The model is applied to solve numerical series problems by finding their underlying pattern and allowing new terms of the series to be inferred.

4.1 The basic unit or *edk*

For a number series $X = \{x_1, x_2, ..., x_n\}$, where generally $x \in \mathbb{R}$ and $n \in \mathbb{N} \mid n \geq 2$, an *edk* can be built using it as the base level. Above it, the elements of a first row are calculated from the differences or divisions between the correlative terms of this fundamental level. The next higher levels are calculated in the same way, forming a triangle, as shown in (1), where $y_i \in \mathbb{R}$, $y_i^j = y_{i+1}^{j-1} - y_i^{j-1}$ in the case of an *edk* built by differences, or $y_i \in \mathbb{R}_*$, $y_i^j = y_{i+1}^{j-1} / y_i^{j-1}$ in the case of an *edk* built by divisions, with $j \in \{1, 2, ..., n-1\}$, $i \in \{1, 2, ..., n-j\}$ and $y_i^0 = x_i$.

$$edk_n = \begin{array}{ccccccc} y_1^{n-1} & & & & & & \\ y_1^{n-2} & y_2^{n-2} & & & & & \\ \vdots & \vdots & & & & & \\ y_1^1 & y_2^1 & y_3^1 & \dots & y_{n-1}^1 & & \\ x_1 & x_2 & x_3 & \dots & x_{n-1} & x_n \end{array} \quad (1)$$

We will consider that an *edk* provides a solution to a certain problem if a constancy is found in its upper rows, defining constancy as the existence of k levels of zeros, in the case of an *edk* built by differences, or k levels of ones if it is built by divisions, with $k \in \{1, 2, ..., n-1\}$. That is, an *edk* is a solution if $\forall j \geq k$ and $\forall i \in \{1, 2, ..., n-j\}$ it is true that $|y_i^j| < \epsilon$ in first case, or $|y_i^j - 1| < \epsilon$ in second case, where $\epsilon = e^r$, $r \in \mathbb{R}_*^-$ is a preset, very small parameter.

4.2 The basic algorithms or *kitas*

The *kitas* are the basic algorithms available for the model and are applied in different configurations on a series or on an *edk*. A *kita* supposes the computational implementation of a different perspective or way of seeing the problem, where each one of them tries to emulate an aspect of human reasoning based on mathematical foundations. Their combination allows finding patterns of increasing complexity, eventually reaching a solution to the problem. Since they try to emulate basic aspects of human intelligence, their character is general. That is, they are applicable both to the numerical series problems as well as to other IQ problems.

Basic (*BAS*) and Divisions (*DIV*). They are used to build an *edk* through differences or divisions, respectively.

Reduction (*RED*). This *kita* is used to focus on the upper levels of an *edk* rather than its base. Let edk_n be an *edk* whose base X is formed by n elements. A reduction of $r \in \{1, 2, ..., n-2\}$ levels allows focus on the series:

$$RED(r)\,[edk_n] \rightarrow \{y_1^r, y_2^r, ..., y_{n-r}^r\} \quad (2)$$

On this series, *BAS* is applied to build a new *edk*.

Multi-Level (*ML*). It focuses on the diagonals of the *edk*, selecting elements according to increments in the vertical and horizontal directions. This allows to achieve patterns that may go unnoticed, as they are not visible at the base of the *edk*. That is, let dy and dx be the increments on the vertical and horizontal axis, respectively, with $dy, dx \in \mathbb{N}$, assuming the edk_n, other $dy + dx$ *edks* are built by differences, with size less than or equal to the original,

$$ML(dy, dx)\,[edk_n] \rightarrow \{edk_{m_1}, edk_{m_2}, ..., edk_{m_{dy+dx}}\} \quad (3)$$

where $M = \{m_1, m_2, ..., m_{dy+dx}\}$, $m \in \mathbb{N} \mid 2 \leq m \leq n$ contains the sizes of each of the descendant *edks*. In each iteration, we start from an origin point, which is the $(p_y, p_x) = (0, 1)$ for the first child *edk*, which corresponds to the element x_1. For the second child *edk*, the origin will be $(1, 1)$ and corresponds to the element y_1^1. For the third child *edk* it will be $(2, 1)$, and so on until $(p_y, p_x) = (dy+dx-1, 1)$. From the corresponding origin, the elements located on the slope formed by dy and dx are extracted, until meeting the

limit of the *edk* (its right diagonal). That is, in general, the elements located in the *edk* indices that meet:

$$(p_{y_\alpha}, p_{x_\alpha}) = (p_y + \alpha \cdot dy, p_x + \alpha \cdot dx) \tag{4}$$

with α taking values from 1, 2, 3, ... until the iteration corresponding to the limit of the *edk*.

Focusing (*FOC*). It allows fragmenting the original string into several substrings where an implicit pattern can be found more easily. For this, two parameters are defined: the shift to the left s and the list of divisions D, with $s \in \mathbb{N} \mid 0 \leq s \leq \sum_{r=1}^{\mathcal{L}[D]} d_r$, where $\mathcal{L}$ is an operator which calculates the length of a sequence or ordered list, and $D = \{d_1, d_2, ..., d_l\}$ with $d, l \in \mathbb{N} \mid s + \sum_{r=1}^{l} d_r \leq \mathcal{L}[X] \geq 4$. Once these parameters are defined, the elements located to the left of the displacement are ignored, which will be used later. With this series shifted, the first d_1 elements of the series are extracted and stored in a new sequence S_1. The next d_2 elements are then extracted and stored in a sequence S_2. And so on until the elements of the series are exhausted. Therefore, each of the new series is initially formed as follows:

$$S_m = \begin{cases} \emptyset, & p > n \\ X' \subset X, & p \leq n \end{cases} \tag{5}$$

where $m \in \{1, 2, ..., l\}$, $\emptyset$ symbolizes an empty ordered series, and $X' \subset X$ is an ordered subseries of X without elimination of equal elements. The coordinates of the elements that make up this subseries can be calculated using the following equation, pruning the result where $c \in C_{X'} > n$:

$$C_{X'} = \bigcup_{p=0}^{\lceil \frac{n-s}{s_d} \rceil - 1} \bigcup_{q=1}^{d_m} \{s + f_1(m) + q + p \cdot s_d\} \tag{6}$$

where $C_{X'}$ is the set of coordinates, $s_d = \sum_{r=1}^{\mathcal{L}[D]} d_r$, and $\cup$ is the orderly union with preserving of all the elements of the corresponding series. $f_1(m)$ is different according to the value of m:

$$f_1(m) = \begin{cases} 0, & m = 1 \\ \sum_{r=1}^{m-1} d_r, & m \neq 1 \end{cases} \tag{7}$$

Next, it is necessary to insert in these series S_m the elements ignored due to the displacement, which we call as *series of remainders*. These elements are inserted starting at the end of the series of remnants and starting with the last series S. Let R be the series with the remaining elements, the subseries of remainders R_m that would correspond to S_m is:

$$R_m = \begin{cases} \emptyset, & m \neq l \wedge \bigcup_{r=0}^{l-m-1} R_{l-r} = R \\ R' \subset R, & m = l \vee \bigcup_{r=0}^{l-m-1} R_{l-r} \neq R \end{cases} \tag{8}$$

where $\emptyset$, $\subset$ and $\cup$ have the same meaning as before. The subsequence R' is formed as follows

$$R' = \{x_{t+f_2(t)}, x_{t+f_2(t)+1}, ..., x_{t+d_m-1}\} \tag{9}$$

where $t = s + 1 - f_3(m, l) - d_m$. $f_2(t)$ y $f_3(m, l)$ are calculated as follows

$$f_2(t) = \begin{cases} 0, & t > 0 \\ |t| + 1, & t \leq 0 \end{cases} \tag{10}$$

$$f_3(m, l) = \begin{cases} 0, & m = l \\ \sum_{r=0}^{l-(m+1)} d_{l-r}, & m \neq l \end{cases} \tag{11}$$

Therefore, focusing gives series of the type:

$$FOC(s, D)\,[edk_n] \rightarrow X_1, X_2, ..., X_l = R_1 \cup S_1, R_2 \cup S_2, ..., R_l \cup S_l \tag{12}$$

Finally, differences are made between the elements of each of the series, building new *edks* from them.

Analogy (*ANA*). As in the case of (*FOC*), this *kita* is used to fragment the original series into new series to find the implicit pattern more easily. To do this, first of all, two parameters are defined: the displacement $s \in \mathbb{N} \mid s < n - 4$, which determines how many elements are ignored from the first position of the series, and the number of elements that must be per group/series $e \in \mathbb{N} \mid 1 < e \leq n - s - 1$, $(n - s)\%e \neq 0$. Then, the original series is divided into $t = \lceil (n - s)/e \rceil$ groups G of size e, except for the last one (G_t) which must necessarily have a smaller size. Once this is done, the next step is to verify that all the groups must reach a constancy in the same row of their *edk* constructed by differences.

If all the complete groups are solved in the same way, then so will the last one and the prediction can be made directly. In other words, the inverse method is also implemented in the direct method.

Exponentiation (*EXP*). It is used to raise the elements of the base series of an *edk* to a certain power. This allows, in cases such as sequences formed by square numbers or any other power of x, to transform the series into a simpler one. Generally, applying a exponentiation with exponent e based on edk_n, we obtain:

$$EXP(e)\,[edk_n] \rightarrow \{x_1^e, x_2^e, ..., x_n^e\} \tag{13}$$

where $e \in \mathbb{R}_*$, except in cases like $0 \in X$, $\exists\, x \in X \mid x < 0$ or both, in which certain exponents of this range will not be applicable. The *kita* is finished by applying *BAS* to create a new *edk* from the result.

Logarithm (*LOG*). Construct the upper level to the base of an *edk* by applying the logarithm on its elements, in the following way: for each pair of elements, the absolute value of the one located to the left constitutes the base of the logarithm, and the absolute value of the located to the right the exponent. This allows, for example, to identify the pattern more easily in series in which each element is equal to the previous one raised to a certain exponent. Applying this tool to any edk_n, we have:

$$LOG\,[edk_n] \rightarrow \left\{\log_{|x_1|}|x_2|, \log_{|x_2|}|x_3|, ..., \log_{|x_{n-1}|}|x_n|\right\} \tag{14}$$

So the general equation that governs the behavior of this *kita* is:

$$y_i^1 = \log_{|x_i|}|x_{i+1}| \tag{15}$$

Where $i \in \{1, 2, ..., n-1\}$ is the index of the corresponding element, x_i and x_{i+1} are two consecutive elements of the base of the *edk*, and y_i^1 is the element that occupies the position i in the level following the base. Once all the

elements of the next level have been obtained, a new *edk* is created using differences using this next level as the basis.

Double Operation (*DOP*). It allows to build the next level to the base of an *edk* by alternating basic arithmetic operations (addition, subtraction, multiplication and division) on the elements of the base. This allows obtaining the pattern in the series in which there is a two-to-two arithmetic relationship between the elements of the series. Assuming again edk_n and two operators $\mathcal{O}_1, \mathcal{O}_2 \in \{+, -, \times, \div\}$, the result of applying the *kita* is as follows:

$$DOP\left(\mathcal{O}_1, \mathcal{O}_2\right)[edk_n] \rightarrow \\ \{\mathcal{O}_1\left[x_2, x_1\right], \mathcal{O}_2\left[x_3, x_2\right], ..., \mathcal{O}_j\left[x_n, x_{n-1}\right]\} \tag{16}$$

With $j = 1$ if n is even, and $j = 2$ otherwise. In this way, the equation that governs the process is the following:

$$y_i^1 = \mathcal{O}_{f(i)}\left[x_{i+1}, x_i\right], \quad f(i) = \begin{cases} 1, & i\%2 \neq 0 \\ 2, & i\%2 = 0 \end{cases} \tag{17}$$

As in other cases, the *kita* ends up building an *edk* by means of differences with the result series.

Specular Symmetry (*SSYM*). It is used to predict the following items directly on the basis of the *edk* if it has length greater than two and a symmetric arrangement is detected. Generally, if from a position $j \in \{\text{round}(n/2), ..., n-1\}$ of the sequence, its elements located in positions before ($Case\ 1$) or equal and before ($Case\ 2$) to j begin to repeat in reverse order, the remaining elements of the series can be obtained by continuing this subsequence. That is, if symmetry is detected, the result is calculated as:

$$SSYM\left[edk_n\right] \rightarrow \begin{cases} X_1, & Case\ 1 \\ X_2, & Case\ 2 \end{cases} \tag{18}$$

where $X_1 = \{x_1, x_2, ..., x_{j-1}, x_j, x_{j-1}, ..., x_2, x_1\}$ and $X_2 = \{x_1, x_2, ..., x_{j-1}, x_j, x_j, x_{j-1}, ..., x_2, x_1\}$.

Repetition Symmetry (*RSYM*). Predicts the next elements in the base series of the *edk* if it has more than two elements and a repeating group is detected throughout the original sequence. To do this, the series is divided into smaller groups, all of them of the same length except for the last one. Different sizes of groups are tested until a symmetry is found or until all the possible ways to create groups are completed. Once a repeating group is found, the elements to be predicted are obtained automatically. That is, let $G = \{x_1, x_2, ..., x_j\}$, $j \in \{2, 3, ..., n-1\}$ be the repeating group, the series can be extended indefinitely, according to the following equation

$$RSYM\left[edk_n\right] \rightarrow \\ \{x_1, x_2, ..., x_j, x_1, x_2, ..., x_j, x_1, x_2, ..., x_j, ...\} \tag{19}$$

Different Groups of Equal Elements (*DGE*). This *kita* divides the base series of length necessarily greater than two of the *edk* into two subseries. To do this, it makes groups between the elements of the series in which all the elements within the group are equal, but not between them. These equal elements must be located in consecutive positions to be considered as a group. The limits between groups, therefore, satisfy the relation $x_{i+1} \neq x_i$. Once the groups are found, new series are formed, each of them with their respective sizes. That is, let $G = \{G_1, G_2, ..., G_t\}$ be the groups formed, with $t \in \mathbb{N} \mid 2 \leq t \leq n-1$, each of the series into which the original is divided is constructed as:

$$DGE\left[edk_n\right] \rightarrow \begin{cases} X_1 = \{G_1\left[1\right], G_2\left[1\right], ..., G_t\left[1\right]\} \\ X_2 = \{\mathcal{L}\left[G_1\right], \mathcal{L}\left[G_2\right], ..., \mathcal{L}\left[G_t\right]\} \end{cases} \tag{20}$$

Where $G_j\left[1\right]$, $j \in \{1, 2, ..., t\}$ refers to the first element of the group j. On these two series X_1 and X_2 into which the original is divided, two *edks* are built by means of differences between the elements. Finally, it should be noted that this *kita* is useful for very specific series, in which there are several groups formed by the same element within the group, but different or not between the groups, and between whose group lengths there is also a relationship.

Different Groups of Different Elements (*DGD*). This *kita* is useful for series formed by the same group that is changing both in content and in length. It creates two subseries from the base of the *edk*, which must have at least three elements. To do this, the series is divided into groups made up of one or more elements, the progression of which is repeated in the sequence. This operation is carried out by extracting the different elements of the base series and counting the number of their occurrences in the sequence. Next, it iterates over the series of different elements, selecting as many as possible, according to the number of appearances that remain, decreasing the number of appearances in each iteration. Once you have these groups, a series is formed with their size. That is, let $G = \{G_1, G_2, ..., G_t\}$ be the number of groups formed and $X_{\neq}$ a subsequence of X with its different elements, the original series is divided into:

$$DGD\left[edk_n\right] \rightarrow \begin{cases} X_1 = X_{\neq} \\ X_2 = \{\mathcal{L}\left[G_1\right], \mathcal{L}\left[G_2\right], ..., \mathcal{L}\left[G_t\right]\} \end{cases} \tag{21}$$

If the group that is repeated in the original sequence is not at the beginning of the sequence, the series X_2 must be ordered in reverse. In both series, X_1 and X_2, the last elements can be skipped if necessary, in case any group is incomplete. On these two sequences, their respective *edks* are constructed by means of differences. The former can be eliminated if the repeating group is at the beginning of the original series.

Split Of Elements (*SOE*). It is used when the series is made up of elements in which a certain numerical value is repeated among them, with a relationship between the digits of each element and the number of times each one appears. This *kita* can be used if it is true that $\forall x \in X : x \in \mathbb{N}$ and the length of the sequence is greater than 2. Once this is verified, *SOE* divides the series in two, one with the digits that make up each element, and another with the number of digits of each element. That is, the resulting series are:

$$SOE\left[edk_n\right] \rightarrow \begin{cases} X_1 = \{\mathcal{R}\left[x_1\right], \mathcal{R}\left[x_2\right], ..., \mathcal{R}\left[x_n\right]\} \\ X_2 = \{\mathcal{N}\left[x_1\right], \mathcal{N}\left[x_2\right], ..., \mathcal{N}\left[x_n\right]\} \end{cases} \tag{22}$$

Where $\mathcal{R}\left[x_i\right]$ is an operator that selects the number that is repeated in the x_i element of the original series, with $i \in \{1, 2, ..., n\}$ and $\mathcal{N}\left[x_i\right]$ is another operator that calculates the number of digits in the x_i element. Finally, the *BAS* is applied on the series X_1 and X_2 to form a new *edk*.

4.3 Search algorithms

The pattern search method used by KitBit is based on a state tree. A connected tree or graph without cycles [47] $G(S, K)$

is defined by a set of actions or *kitas* $K = \{k_1, k_2, ..., k_n\}$, $n \in \mathbb{N}$ that form the axes of the graph, and a set of states $S = \{s_1, s_2, ..., s_m\}$ that form the vertices of the same, where $m \in \mathbb{N} \mid 1 \leq m \leq 1 + \sum_{r=1}^{depth-1} n^r$ y $depth \in \mathbb{N} \mid depth \geq 2$ is the depth of the tree. Excepting to the root state, the rest of nodes have a single parent and a number of descendants ranging from 0 to n, and which are visually identified as vertices connected to the same parent node. Any state $s \in S$ is defined as a four-components structure, according to equation 23.

$$s = \left\{ \begin{array}{c} \{edk_1, edk_2, ..., edk_l\} \\ k \in K \\ \{c_1, c_2, ..., c_l\} \ , \ c \in \{0, 1\} \\ \{f_1, f_2, ..., f_l\} \ , \ f \in \{1, 2, ..., l\} \end{array} \right\} \tag{23}$$

Where the first component is an ordered sequence of $l \in \mathbb{N}$ basic units or *edks* on which a k action has been applied. The second is the action. The third is an ordered sequence of zeros and ones, in which a one at position $i \in \{1, 2, ..., l\}$ indicates that the corresponding *edk* is a solution and a zero that it is no solution. And the fourth component is another ordered sequence of natural numbers that indicate the position of the parent of the *edk* i in the list of base units of the previous state. When in the third component of s all the elements are ones, it means that this state is a solution state to the problem.

This tree structure can be traversed in different ways, implementing different pattern search strategies. KitBit uses an uninformed search algorithm, with the popular Breadth First Search (BFS) and Depth First Search (DFS) strategies. This means that the algorithm does not have information about which is the next best node to expand, but simply generates successors and checks if they are a solution. In both algorithms, if a solution state is reached, the path followed until reaching it is delivered. This information is key to predict new elements in the series by applying the corresponding actions inversely.

So far we have considered that the algorithm would stop at the moment of finding a solution. However, we are interested in finding not one, but all possible solutions to the problem. For this, the *goal_branching* parameter has been implemented, which allows searching all the possible solutions in the graph, without stopping the execution of the algorithm in case it has found a solution.

4.4 Prediction

As we have just seen, in the KitBit model, finding the solution to the problem involves going a certain path in the state tree. That is, to carry out a series of known operations that take us from one state to the next. Then, once the implicit pattern is known, the prediction of new elements of the series implies going through the state tree in the opposite direction, starting from the final state previously reached. To do this, a series of operations must be carried out, which can be divided into two fundamental parts and which are described below.

Prediction of new elements. The prediction process begins by adding as many zeros or ones at the top of the *edk* as elements to be predicted, depending on whether the constancy was achieved using *BAS* or *DIV*, respectively. These elements are denoted as $\beta \in B$ and are said to warp the *edk*, distorting its initial triangle shape. If the *edk* comes from the application of *ANA* or *SYM*, this step is skipped.

Next, the *edk* is traversed in an inverse way applying 24.

$$\beta_i^j = \begin{cases} \beta_i^{j+1} + \beta_{i-1}^j, & \text{BAS} \\ \beta_i^{j+1} \cdot \beta_{i-1}^j, & \text{DIV} \end{cases} \tag{24}$$

where $i \in \{1, 2, ..., n_{\beta_{j+1}}\}$, $j \in \{0, 1, ..., n_j - 1\}$, $\beta_0^0 = x_n$ and $\beta_0^j = y_{n-j}^j$, $\forall j \neq 0$. $n_{\beta_{j+1}}$ is the number of deformation elements that exist in row $j + 1$ and n_j is the number of rows involved. The prediction process would start with $\beta_i^{j+1} = \beta_{\min(i)}^{\max(j)}$ as the point of origin, that is, with the deformation element located at the highest level and furthest to the left of the *edk*. For *EXP*, it is necessary, once the prediction is made, to apply the inverse of the exponent that was passed as an argument (that is, $1/e$) to all the elements of the resulting series. In turn, *ANA* or *SYM* (*SSYM* and *RSYM*) perform the inference of new elements directly in the pattern search process, so the prediction process in this case is not necessary. On the other hand, for *LOG*, once the elements predicted in its child have been inserted in the parent *edk*, the prediction works on the basis of the *edk* and its upper row, using the following equation:

$$\beta_i^0 = \left(\beta_{i-1}^0\right)^{\beta_i^1} \tag{25}$$

Finally, regarding *DOP*, once the elements in its two child *edks* have been predicted and inserted in the parent *edk*, the prediction consists of applying the opposite operator to the one designated by $\mathcal{O}_1$ and $\mathcal{O}_2$ between the base elements and its top level. That is, the equation that dominates the prediction process in this case is:

$$\beta_i^0 = \mathcal{O}'_{f(g)}\left[\beta_i^1, \beta_{i-1}^0\right], \quad f(g) = \begin{cases} 1, & g(i,n) \\ 2, & \overline{g(i,n)} \end{cases} \tag{26}$$

where $\mathcal{O}'_1$ is the opposite operation of $\mathcal{O}_1$, and $g(i,n) = (i\%2 \neq 0 \wedge n\%2 \neq 0) \vee (i\%2 = 0 \wedge n\%2 = 0)$.

Inserting elements. The insertion is made from the predicted elements in the child *edks* to the parent *edk* from which they come. For this, the peculiarities of each *kita* must be taken into account. For *BAS*, *DIV*, *ANA*, *EXP* and *SYM*, simply insert the predicted elements in the child *edk* after the last element of the base series of its parent *edk*. In the case of *RED*, the predicted elements in the child *edk* are inserted in the index row r of the parent *edk*, after the elements that constitute it. Similarly, for *LOG* and *DOP*, the prediction results elements in the child *edks* are inserted after the last element of the index row 1 of the parent *edk*.

In the case of *ML*, the elements resulting from the prediction made in the $dy + dx$ descendant *edks* are inserted at the following points of each slope of the parent *edk*. The coordinates of these new elements can be calculated using (4). Similarly, for *FOC*, the predicted elements are inserted in the $\mathcal{L}[D]$ *edks* children in the base series of their parent, calculating the coordinates of each of the elements with 6, changing n to $n + n_\beta$ in it, where n_β is the number of elements we wish to predict.

Finally, for *DGE*, *DGD* and *SOE*, the parent *edk* is divided into two *edks* that are denoted under the indices *a* (for X_1) and *b* (for X_2). For each pair of elements $\beta_a^0 \in B_a$, $\beta_a^0 \in$

B_b located in the same positions in both child series, the respective element of the base of the parent *edk* $\beta^0 \in B$ is formed as follows, depending on the case:

- In *DGE*, it is formed as $\mathcal{R}\left[\beta_a^0, \beta_b^0\right]$, where $\mathcal{R}$ is an operator that repeats the element β_a^0, β_b^0 times.
- In *DGD*, the following element is formed by extracting from X_1 as many elements as indicated by β_b^0, starting at the beginning of the sequence X_1 in 21 or by the end depending on the group that is repeated is at the beginning or not of the original series X, respectively.
- In *SOE*, β^0 is constructed using the equation $\beta^0 = \beta_a^0 \cdot 10^0 + \beta_a^0 \cdot 10^1 + ...\beta_a^0 \cdot 10^{\beta_b^0}$.

All this process is applied repeatedly and propagates until reaching the last state, which is the initial state in the search process, on which the definitive elements of the series are predicted.

5 Tests and results

The KitBit algorithms were coded using the Python programming language and tested in the largest set of number series problems to date. First, the results obtained on a set of 91 numerical series associated with the IQ test are presented, compiled from real evaluation tests from different sources. This set has been used to carry out the start-up and the first tests. Next, the model has been confronted with all those series that at the time were used to test the different computational models capable of solving numerical series found in the bibliography. Finally, the resolution of all the series collected in the well-known OEIS database is addressed, which currently has close to 350,000 numerical series of all kinds and represents the largest collection of existing numerical series.

5.1 Series from IQ test

Initially, the system was challenged with a set of 91 non-trivial series based on IQ tests, which are summarized in Table 1. For this, a computer with an Intel Core i7 processor and 16 GB of RAM with a Windows 10 64-bit operating system was used. The results obtained are shown in Tables 2, 3 and 4.

Table 2 shows the results for the four modes in which the algorithm was run. The columns labeled S1Z and S2Z present the results using the BFS method with stop in the solution state, with constancy at a single level and two levels for S1Z and S2Z, respectively, of all *edks* in that state. The columns labeled N1Z and N2Z are the results without stopping in the solution state, with constancy at a single level and two levels for N1Z and N2Z, respectively, then looking for new solutions, also using the BFS method.

In all the execution modes, the number of problems for which a solution was obtained was 97.8% of the total, that is, all the series evaluated except two of them. From these solutions, the expected solution was obtained between 91.2% and 93.4% of the cases, while a different solution than expected between 4.4% and 6.6% was obtained. This result is consistent from the mathematical point of view, since there is no single correct solution in an extrapolation problem in a series [18], but there can be several functions that reproduce the same sequence of numbers [48]. From a psychological point of view, however, the correct solution would be the simplest [18] and which in our case we adopt as the expected solution, according to Ockham's Razor Principle [49]. In our case, we say that we find an unexpected solution when it appears as the first option, for the original number of elements in the series. The expected pattern could be found in a later position of the state tree. By increasing the number of elements provided to solve the problem, it is possible to find the expected pattern as the first solution in a certain number of these series. For example, in the case of series 75, 76, 80 and 84, the expected solution is found as the first option if one element is added to the original series, in the execution cases where this solution is not found with said original series, while a second element needs to be added in the case of series 72. For series 4, the correct solution can be reached by starting the series from the element in the second position. Finally, only in two cases no solution was found. This is because the algorithms that govern these series cannot be reproduced by any combination of the *kitas* that the system currently has. Eventually, these series could be solved by adding new algorithms emulating new ways of perceiving reality.

To solve this group of series, a set of 60 *kitas* was used, made up of the types described in section 4.2, applied with different parameters. Table 3 shows the percentage of use of each type of *kita* in the resolution of the series group. The *kita BAS* is used in all cases because the root of the state tree is always built using this *kita*. This information provides us with a heuristic to improve the efficiency of the model and reduce resolution times by ordering the application of the algorithms according to their probability of use depending of the kind of problem. Those that have not been used in the resolution of this set of series are not discarded, since they can be useful in other types of problems. On the other hand, the depth of the graph used to solve the problems is always lower when constancy in a single level of the *edk* is imposed as a condition, since simpler patterns result, while a greater number of *kitas* when constancy is imposed on two levels.

Regarding the resolution times in S1Z and S2Z cases, in most of the series a pattern is found in a time less than a millisecond, except for some exceptional cases that require a longer resolution time with the configuration used, which deviates from the mean.

Finally, Table 4 shows the percentage of the minimum number of elements n_e needed to find a solution, which depends on the resolution mode used. In most series, only between 3 and 9 elements are necessary, although in some problems they require up to 12 elements to be solved. This indicates that, in general, the number of items that KitBit needs to find the underlying pattern is very small.

In conclusion, KitBit successfully solves practically all of this first set of non-trivial series based on IQ tests using standard computational equipment in times less than a second in most cases. This result provides enough confidence in the model to face more complex problems, starting with those raised in the previous models that appear in the literature.

TABLE 1
Numerical Series Problems Based on IQ Test.

i	sequence	i	sequence	i	sequence
0	0, 1, 1.7071, 2.3660, 3, 3.6180, 4.2247, 4.8229	31	2, 2, 4, 4, 8, 8, 16, 16, 32, 32, 64, 64, 128, 128, 256, 256	62	0, 2, 9, 28, 75, 186, 441, 1016, 2295, 5110, 11253, 24564
1	2, 16, 4, 256, 16, 65536, 256, 4294967296, 65536	32	8, 6, 4, 3, 1, -1, -2, -4, -6, -7, -9, -11, -12, -14, -16	63	3, 6, 18, 36, 108, 216, 648, 1296, 3888, 7776, 23328, 46656, 139968
2	3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584	33	362880, 40320, 5040, 720, 120, 24, 6, 2, 1, 1	64	2, 3, 5, 9, 17, 33, 65, 129, 257, 513, 1025, 2049, 4097, 8193
3	1, 3, 7, 12, 18, 26, 35, 45, 57, 70, 84, 100, 117	34	14, 1, -5.5, -8.75, -10.375, -11.1875, -11.59375	65	2, 10, 26, 50, 82, 122, 170, 226, 290, 362, 442, 530, 626
4	-1, 0, 0, 3, 12, 17, 102, 109, 872, 881	35	-1, -1, 0, 2, 5, 9, 14, 20	66	4, 1, 0, 1, 4, 9, 16, 25, 36
5	0, 3, 8, 24, 63, 168, 440, 1155, 3024, 7920	36	-2, 3, 1, 4, 5, 9, 14, 23	67	9, 3, 6, 6, 2, 5, 3, 1, 4, 0, 0, 3, -3
6	1, 3, 6, 10, 15, 21, 28, 36	37	-3, -1, -4, 0, -5, 1, -6, 2, -7	68	-9, 1, -5, 3, -4, 2, -6, -2, -11, -9
7	1, 0, -1, 0, 1, 0, -1, 0	38	-4, -1, -3, 0, -2, 1, -1, 2, 0	69	97.5, 57, 30, 12, 0, -8, -13.333
8	1, 3, 5, 7, 9, 11, 13, 15, 17	39	1, 2, 0, 2, -1, 2, -2, 2, -3	70	6, 4, 3, 3, 2, 2, 3, 1, 6, 0
9	1, 2, 4, 7, 11, 16, 22, 29	40	1, -1, 0, -3, -1, -5, -2, -7, -3, -9	71	7, 16, 52, 196, 772, 3076, 12292
10	1, 4, 9, 16, 25, 36, 49, 64, 81	41	0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1	72	2, 3, 5, 4, 2, 4, 7, 5, 2, 5, 9, 6, 2, 6, 11
11	3, 5, 10, 12, 24, 26, 52, 54, 108	42	-2, 0, -3, 1, -4, 2, -5, 3, -6	73	8, 0, 2, 0, 0, 0, 2, 0, 8, 0, 18, 0
12	3, 4, 8, 17, 33, 58, 94, 143	43	23, 34, 45, 56, 67, 78	74	3, 0, 1, -1, -2, -5, -9, -16
13	3, 6, 18, 72, 360, 2160, 15120, 120960	44	-4, -3, 0, 5, 12, 21, 32	75	1, 0, -1, -1, -2, -1, -3, -4, -1
14	4, 5, 8, 13, 20, 29, 40, 53	45	-4, -2, 2, 8, 16, 26, 38	76	3, 9, 22.5, 45, 67.5, 67.5, 33.75
15	5, 11, 17, 23, 29, 35, 41	46	6, 4, 0, -6, -14, -24, -36	77	0, 2, 9, 24, 50, 90, 147, 224
16	11, 9, 7, 5, 3, 1, -1, -3	47	-2, 0, 4, 10, 18, 28, 40	78	3, 8, 16, 28, 45, 68, 98, 136
17	30, 29, 27, 26, 24, 23, 21, 20, 18, 17, 15	48	-6, -1, 5, 12, 20, 29, 39	79	0, 0, 4, 5, 14, 16, 30, 33, 52, 56
18	144, 121, 100, 81, 64, 49, 36, 25	49	6400, 1600, 400, 100, 25, 6.25, 1.5625	80	0, 8, 15, 35, 48, 80, 99, 143, 168
19	2, 2, 4, 6, 10, 16, 26, 42, 68	50	0, 7, 24, 51, 88, 135, 192	81	4, 32, 108, 256, 500, 864, 1372, 2048
20	81, 27, 9, 3, 1, 1/3, 1/9	51	2, 4, 12, 48, 240, 1440, 10080	82	0, 17, 74, 195, 404, 725, 1182, 1799
21	1, 3, 6, 10, 15, 21, 28	52	-10, 12, 44, 86, 138, 200, 272	83	6, 24, 60, 120, 210, 336, 504, 720
22	1, 1, 2, 3, 5, 8, 13, 21	53	-1, 1, 0, 1, 1, 1, 2, 1	84	3, 6, 15, 42, 123, 366, 1095
23	21, 20, 18, 15, 11, 6, 0	54	-1, 2, -2, -4, 8, -32, -256, 8192	85	23, 31, 53, 83, 135, 217, 351, 567
24	8, 6, 7, 5, 6, 4, 5, 3, 4	55	-10, 0, 15, 35, 60, 90, 125, 165	86	2, 6, 19, 53, 126, 262, 491, 849, 1378
25	4294967296, 65536, 256, 16, 4, 2	56	-2, -1, 1, 5, 13, 29, 61, 125	87	1, 2, 5, 9, 16, 27, 45, 74
26	3, 7, 14, 24, 37, 53, 72	57	-3, -2, 0, 1, 3, 4, 6, 7, 9	88	0, 2, 6, 12, 20, 30, 42, 56, 72, 90, 110, 132
27	-3, -1, 2, 6, 11, 17, 24	58	1, 2, 6, 14, 29, 56, 102, 176, 289	89	3, 5, 8, 12, 17, 23, 30, 38
28	-1, 0, 3, 8, 15, 24, 35	59	0, 1, 2, 8, 29, 80, 181, 357, 638	90	5, 8, 20, 68, 260, 1028, 4100
29	-9, 2, 12, 21, 29, 36, 42	60	8, 1, 0, -1, -8, -27, -64, -125, -216		
30	1, 0, 0, 1, 3, 6, 10, 15	61	-6, -4, 0, 2, 6, 8, 12, 14, 18		

TABLE 2
Results Obtained for the Problems in the Table 1.

	Results (%)				Depth of the graph				Time (ms)		
	Solved	Expected	Unexpected	Not solved	1 (%)	2 (%)	3 (%)	4 (%)	t_{min}	t_{med}	t_{max}
S1Z	97.802	91.209	6.593	2.198	47.191	33.708	16.854	2.247	$21.1 \cdot 10^{-3}$	18.878	831.119
S2Z	97.802	92.308	5.494	2.198	43.820	30.337	22.472	3.371	$22.3 \cdot 10^{-3}$	20.653	854.817
N1Z	97.802	93.406	4.396	2.198	47.191	20.225	23.595	8.989	-	-	-
N2Z	97.802	93.406	4.396	2.198	43.8202	16.854	29.2135	10.1123	-	-	-

TABLE 3
Use of *Kitas* in Solving the Problems in the Table 1.

KITA:	ANA	BASIC	DGD	DGE	DIV	DOP	EXP	FOC	LOG	ML	RED	RSYM	SOE	SSYM
S1Z (%)	3.371	100	0	0	14.607	2.247	1,124	22.472	2.247	11.236	15.730	0	0	1.124
S2Z (%)	3.371	100	0	0	15.730	3.371	1.124	20.225	2.247	10.112	22.472	4.494	0	2.247
N1Z (%)	1.124	100	0	0	11.236	3.371	5.618	22.472	1.124	20.225	28.090	0	0	1.124
N2Z (%)	1.124	100	0	0	10.112	4.494	7.865	11.236	2.247	20.225	33.708	11.236	0	3.371

TABLE 4
Minimum Number of Elements (n_e) Necessary to Find the Expected Solution in the Problems of Table 1.

n_e:	3	4	5	6	7	8	9	10	11	12	13	14	15
S1Z (%)	9.638	45.783	15.663	18.072	1.205	3.614	1.205	1.205	-	1.205	1.205	-	1.205
S2Z (%)	-	9.524	45.240	13.095	9.524	11.905	5.952	-	1.19	1.19	1.19	-	1.19
N1Z (%)	16.471	43.530	20	10.588	4.706	1.176	-	2.353	-	1.176	-	-	-
N2Z (%)	1.176	15.294	40	22.353	8.235	7.060	5.882	-	-	-	-	-	-

5.2 Series from previous models

The system was then put to the test in a new set of 68 series compiled from different articles dedicated to proposing procedures for solving numerical series [33], [50], [51], [52], [53], [54] and [48]. The list of the problems is summarized in Table 5. Column *i* indicates the index in this dataset and *a* the reference article. KitBit approaches the complete list using the same computer system and the same set of *kitas* as in the previous subsection. The results are shown in Tables 6, 7 and 8.

Table 6 shows the results for this set of series in the four previous execution modes. The number of problems for which a solution was obtained was between the 95.6% and the 97.1% of the total. That is, the model solves all the series evaluated except two or three of them, depending on the execution mode. From these solutions, the expected solution was obtained between 72.1% and 89.7% of the cases, while a different solution than expected between 7.3% and 23.5% was obtained. The results are better when constancy is required in a single level of the *edk*, since the higher the constancy condition, the greater is the number of elements required to find the expected pattern. In this way, increasing the elements of the series between 1 and 3, the expected solution is achieved in 12 more series approximating the success rate to 90% above in all execution modes. Finally, in all execution modes, the percentage of unsolved problems is between 2.9% and 4.4%. It should be noted that the procedures presented in the articles from which these series were previously analyzed did not resolve all the cases. In fact, the KitBit success rate is higher. In other words, KitBit is capable of solving a greater number of the proposed series than the reference publications.

Table 7 shows the percentage of use of each type of *kita* in the resolution of this group of series. It is observed that they are different than for the previous group, using *kitas* that had not been used previously. However, their ordering is practically the same, so the priority in their application does not need to be modified. Also, as in the previous group, the resolution times were less than one millisecond most of the times in S1Z and S2Z execution modes.

Finally, Table 8 shows that, in most cases, between 3 and 7 elements are needed to find the solution to the problem. This result is consistent with the previous group, for which we found similar results.

If we compare the previous results with those presented in the publications from which the sequences were extracted, the following observations can be made:

- KitBit gets the correct pattern for the series resolved in [33].
- KitBit is able to obtain the correct solution from all the series evaluated in [50], while the authors fail some of them using two different methodologies.
- KitBit manages to correctly solve the 20 problems raised in [51], while their authors solve 17 of them and the IGOR2 model [52] manages to solve 14.
- Likewise, KitBit also solves all the series proposed to test the IGOR2 [53] model, even the one that this model fails.
- Regarding the series proposed to test SeqSolver [54], KitBit can solve them all correctly, except number 48. In this series there does not seem to be a relationship between all the elements, but only between each element located in an even position with the which is in an immediately preceding odd position. This behavior has not been contemplated in KitBit, but could be implemented in the future.
- Finally, with respect to the series proposed to validate the ASolver method [48], KitBit finds a pattern for all of them except one, which is not explicitly mentioned in the article that ASolver method can solve it. However, in four of them, KitBit finds a pattern that is not the expected one. They are sequences where a pattern is sandwiched between values that constitute noise, so that by removing the noise, KitBit could also find the pattern.

As a conclusion of the tests carried out on this battery of series, we can affirm that KitBit is capable of solving the numerical series of IQ tests reported in the bibliography, as well as other numerical series not related to the IQ tests, performing better in most of the cases than any of the methods previously proposed, with great potential to solve pending series by introducing new *kitas*.

5.3 Series from OEIS database

OEIS is the largest database of integer sequences, which compiles all the information available on such sequences and is widely cited in the literature. Their number is continually growing, and at the time of our tests (September 2021) it contained 347736 series. From this initial set, those repeated sequences, and those with less than 3 elements or less than 4 if they were formed by the repetition of the same element, were eliminated. In total, 341553 remained in which the KitBit algorithms were executed. In the first place, the BFS technique was used without stopping in solution state, requiring constancy in a single level of the *edks* and an epsilon of $\epsilon = e^{-18}$, with a list of 55 *kitas* and a depth in the state tree equal to 2. Next, on the unsolved problems, the same technique with a depth equal to 3 was used, and again, on the unsolved, a depth equal to 4 was applied, in this case

TABLE 5
Numerical Series Problems Collected from Articles [33], [50], [51], [52], [53], [54] and [48].

i	a	sequence	i	a	sequence	i	a	sequence
0	[33]	5, 9, 35, 125, 345, 785, 1559, 2805, 4685	23	[51]	54, 48, 42, 36, 30, 24, 18	46	[54]	2, 3, 5, 8, 13, 21, 34
1	[33]	2, 5, 11, 21, 37, 63, 107	24	[51]	6, 8, 5, 7, 4, 6, 3, 5	47	[54]	1, 2, 1, 3, 1, 4, 1, 5
2	[33]	6, 288, 884736, 173946175488, 2188749418902061056	25	[51]	6, 9, 18, 21, 42, 45, 90, 93	48	[54]	5, 10, 1, 2, 22, 44, 3, 6, 7, 14
3	[33]	1, 2, 8, 48, 384, 3840	26	[51]	7, 10, 9, 12, 11, 14, 13, 16	49	[54]	0, 1, 4, 9, 16
4	[33]	5, 13, 35, 97, 275, 793, 2315, 6817	27	[51]	8, 10, 14, 18, 26, 34, 50, 66	50	[54]	1, 4, 9, 16, 25
5	[33]	12, 44, 144, 432, 1216, 3264, 8448, 21248	28	[51]	8, 12, 10, 16, 12, 20, 14, 24	51	[54]	4, 7, 12, 20, 32
6	[50]	1, 2, 3, 4, 5	29	[51]	8, 12, 16, 20, 24, 28, 32, 36	52	[54]	4, 7, 12, 20, 33
7	[50]	1, 4, 9, 16	30	[51]	9, 20, 6, 17, 3, 14, 0, 11	53	[48]	7, 11, 15, 19, 23
8	[50]	1, 3, 9, 27	31	[52]	0, 1, 4, 9	54	[48]	0, 2, 4, 6, 8, 10
9	[50]	0, 1, 1, 2, 3, 5, 8, 13	32	[52]	0, 2, 4, 6	55	[48]	3, 6, 17, 66, 327
10	[50]	1, 2, 4, 8, 16	33	[52]	1, 1, 2, 3, 5	56	[48]	1, 1, 2, 6, 24, 120, 720, 5040, 40320
11	[51]	12, 15, 8, 11, 4, 7, 0, 3	34	[52]	0, 1, 2, 1, 4, 1	57	[48]	2, 5, 8, 11, 14, 17
12	[51]	148, 84, 52, 36, 28, 24, 22	35	[52]	0, 0, 1, 1, 0, 0, 1, 1	58	[48]	3, 6, 12, 24, 48
13	[51]	2, 12, 21, 29, 36, 42, 47, 51	36	[52]	0, 1, 3, 7	59	[48]	1, 2, 3, 5, 8, 13, 21, 34, 55
14	[51]	2, 3, 5, 9, 17, 33, 65, 129	37	[52]	1, 2, 2, 3, 3, 3, 4, 4, 4, 4	60	[48]	1, 1, 2, 6, 24, 120
15	[51]	2, 5, 8, 11, 14, 17, 20, 23	38	[53]	1, 4, 7, 10, 13, 16, 19, 22	61	[48]	1, 2, 3, 4
16	[51]	2, 5, 9, 19, 37, 75, 149, 299	39	[53]	2, 4, 3, 5, 4, 6, 5, 7	62	[48]	3, 2, 1, 0
17	[51]	25, 22, 19, 16, 13, 10, 7, 4	40	[53]	4, 11, 15, 26, 41, 67, 108, 175	63	[48]	1, 11, 111, 1111
18	[51]	28, 33, 31, 36, 34, 39, 37	41	[53]	5, 6, 12, 19, 32, 52, 85, 138	64	[48]	1, 44, 1, 27, 1, 92
19	[51]	3, 6, 12, 24, 48, 96, 192	42	[53]	8, 10, 14, 18, 26, 34, 50, 66	65	[48]	1, 39, 1, 35, 1, 28
20	[51]	3, 7, 15, 31, 63, 127, 255	43	[53]	8, 10, 14, 18, 26, 34, 42, 58	66	[48]	1, 1, 7, 1
21	[51]	4, 11, 15, 26, 41, 67, 108	44	[53]	1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5	67	[48]	46, 147, 9, 1, 1, 1
22	[51]	5, 6, 7, 8, 10, 11, 14, 15	45	[54]	2, 2, 4, 8, 32, 256			

TABLE 6
Results Obtained for the Problems in the Table 5.

	Results (%)				Depth of the graph				Time (ms)		
	Solved	Expected	Unexpected	Not solved	1 (%)	2 (%)	3 (%)	4 (%)	t_{min}	t_{med}	t_{max}
S1Z	97.059	86.765	10.294	2.941	24.242	50	19.697	6.061	$12.4 \cdot 10^{-3}$	32.953	1354.094
S2Z	95.588	72.059	23.529	4.412	16.923	33.846	33.846	15.385	$19.2 \cdot 10^{-3}$	45.253	1463.843
N1Z	97.059	89.706	7.353	2.941	24.242	37.879	28.788	9.091	-	-	-
N2Z	95.586	82.353	13.235	4.412	16.923	24.615	29.231	29.231	-	-	-

TABLE 7
Use of *Kitas* in Solving the Problems in the Table 5.

KITA:	ANA	BASIC	DGD	DGE	DIV	DOP	EXP	FOC	LOG	ML	RED	RSYM	SOE	SSYM
S1Z (%)	1.515	100	0	1.515	24.242	3.030	3.030	21.212	0	22.727	21.212	4.545	1.515	3.030
S2Z (%)	3.077	100	0	1.538	21.538	6.154	9.231	12.308	0	30.769	40	13.846	1.538	7.692
N1Z (%)	0	100	0	3.030	15.152	4.545	4.545	21.212	0	36.364	30.303	4.545	1.515	1.515
N2Z (%)	0	100	0	1.538	24.615	10.769	10.769	6.154	0	43.077	47.692	21.538	1.538	3.077

TABLE 8
Minimum Number of Elements (n_e) Necessary to Find the Expected Solution in the Problems of Table 5.

n_e:	3	4	5	6	7	8	9	10
S1Z (%)	35.593	28.814	6.780	18.644	8.475	-	-	1.694
S2Z (%)	10.204	30.612	20.408	16.327	20.408	-	-	2.041
N1Z (%)	40.984	22.951	22.951	4.918	6.557	1.639	-	-
N2Z (%)	17.857	30.357	17.857	23.214	8.929	1.786	-	-

with two list of *kitas* of 34 and 32 elements. Obviously, a smaller number of kitas reduces the chances of finding a solution. However, we have chosen this option to limit the total resolution time for this large set of problems. As in the previous datasets, for each of the series, its elements were eliminated one by one, determining the minimum number necessary to find a pattern. In this case, we identify the series as solved if the expected solution is obtained and, given the large number of series to be solved, the algorithms were executed using Google's free cloud resources through Google Colab.

We begin by addressing the resolution of the entirety of the series mentioned above without imposing any restrictions as is shown in Table 9. KitBit finds a pattern that correctly reproduces at least the last element in the sequence in 87514 sequences, 25.62% of the total. This number is by far the largest number of series solved to date for the OEIS database. We must remember that the maximum number of series reported so far had been 26951 [51] (7.89% of the database) over a selection of 57524 series with at least 20 elements with values lower than 1000.

Beyond the objective criterion indicated to eliminate 1,78% of the OEIS series as the basis of our analysis, due to its very small number of elements, we have refrained from selecting any smaller subset trying to avoid introducing any bias in the results. Anyway, analyzing the unsolved series, we find that a high percentage of them fall into the following categories: a) series based on complex mathematical functions that include trigonometric, hyperbolic and logarithmic functions; b) series that grow very rapidly, such as exponential or n^{th} powers, where the order of magnitude increases between contiguous elements of the series; c) series based on prime numbers, which would require a specific treatment; d) and series based on patterns of multiple complexity or combinations of other series. These categories are highly artificial and complex: 10.42% of the series contain very large and very small numbers in the same series, which although difficult for our algorithms, it is capable of solving some of them; the series combinations of other series are 17.21% of the total and suppose a degree of artificial difficulty that, at this stage, our system does not propose to address; others, such as those based on prime numbers, which represent another 13.48%, require the addition of a new 'way of looking at' the system, oriented to these kind of numbers that we will address in the future. Even with this degree of difficulty, KitBit is capable of solving 43914 of 196625 this kind of series, 22.33% of them. Outside of these categories, OEIS database contains 144928 series, for which KitBit currently finds a solution in 30.08% of them as is shown in Table 9.

It is clear that if these criteria for selecting subsets of the original database were further restricted, the percentage results of solved series could improve. However, the objective of our search is to progressively improve the results on the complete OEIS database. It should be considered that the results presented correspond to a limited set of kitas and to a certain depth of the state tree, used to solve a large number of series in a reasonable time. Thanks to the modularity of KitBit, it is to be expected that the system will be able to solve a greater number of series by expanding the kitas configuration or by implementing new ones that are more specific to certain problems, such as prime numbers, for example.

Regarding the most used *kitas*, in Table 10 it is observed that the group of *kitas* formed by FOC, RED, DOP and SYM is used in most of the cases, being the appearance of the rest of actions less frecuent. This result contrasts with that obtained for the two groups of previous series based on the IQ test. This is due to the complexity of the series in the OEIS database, which is considerably higher, most of which are based on mathematical functions that cannot be solved by a human being. Lastly, in Table 11 we can see the depths of the state tree used in the different cases.

In short, KitBit is the computational system capable of solving the largest number of series in the OEIS database to date, well above the maximum value reported in the literature. Unlike previous results, KitBit faces the entire database, without excluding any series a priori, to obtain conclusions about the capacity of the system and the possible improvements that could be implemented to solve a greater number of problems. Within the series solved by KitBit, many of them are of a high difficulty, well above those usually found in the IQ tests. With all this, KitBit has great potential to address different types of problems that can be summarized as complex numerical series, such as different time series or prediction of movement in space.

6 Conclusions and Future Work

This article explains the fundamentals and initial development of KitBit, a new AI cognitive model based in a naturalistic approach to epistemology. This conception of the model starts from the premises that 1) the emulation of human behavior in specific tasks does not necessarily imply the emulation of the processes of perception and cognition responsible for the production of intelligent knowledge and 2) the production of intelligent knowledge does not necessarily imply the emulation of these cognitive processes. Assuming that intelligence does not arise from neurons, but rather that the carriers of consciousness are patterns, the main characteristic of intelligence would reside in the ability to find patterns of variation in space and time. Thus, intelligent knowledge would consist of discovering the order in which things happen, and intelligent behavior in modifying that order, establishing priorities, seeking to maximize utility for the intelligent agent. Trying to emulate some of this human perception and cognition processes, KitBit uses the combination of a set of basic computational algorithms, *kitas*, that are able to achieve this goal either alone, or in a nested or concatenated way. In this way, KitBit meets certain conditions that we believe necessary to achieve this goal in such a human-like way :

- Natural intuition. The KitBit internal logic allows to obtain the results presented in this paper without using any prior information.
- Learning skills. The order at which *kitas* are applied is dynamic, allowing the addition of new ones.
- Energy efficiency. KitBit is very sensitive to improvements due to the introduction of new *kitas* and not very dependent on the available computing power.

TABLE 9
Results Obtained for the OEIS Database.

	Total series	Solved
Entire database	341553	87514 (25.62%)
Selection	144928	43600 (30.08%)

TABLE 10
Use of *Kitas* in Solving the Problems in OEIS database (%).

ANA	BASIC	DGD	DGE	DIV	DOP	EXP	FOC	LOG	ML	RED	RSYM	SOE	SSYM
2.115	100	0.203	3.589	3.266	8.219	0.377	37.164	0.037	9.609	47.447	24.807	0.042	51.278

TABLE 11
Depth of the Graph in Solving the Problems in OEIS database (%).

1	2	3	4
7.568	23.290	42.562	26.580

- Traceability. The sequence of *kitas* used to find each solution are easily identified, managed, stored and reproduced.
- Scalability. KitBit establish relations between solutions and sequences of *kitas* identifying levels of description rich in patterns, that serve as a guide to improve its predictive ability to solve problems of increasing complexity.

So, assuming these preconditions and using a standard processing capacity, our model has managed to solve a large set of problems related to series of numbers commonly used to evaluate human IQ. KitBit solves, in very short time per series, more than 97.8% of a database of 91 standard series, and 97% of a set of 68 series collected from previous publications. Therefore, KitBit shows a capacity that exceeds that of the human being, both in time and results obtained. These results, as far as we know, are in all cases superior to those obtained by any other model to date.

Finally, KitBit addresses the resolution of the nearly 350000 numerical series of all kinds, included in the well-known OEIS database. Our system found a pattern in 87514 sequences, 25.62% of the total, being the highest number of series solved to date for the OEIS database. Analyzing the unsolved series, we find that a high percentage of them are highly artificial and complex, impossible for a human to solve. Outside of these categories, OEIS contains 144928 series, for which KitBit currently finds a solution in 30.08% of them. Thanks to the modularity of KitBit, it is to be expected that the system will be able to improve this result by expanding the *kitas* configuration or by implementing more specific ones.

Once the theoretical foundations and basic algorithms of KitBit have been presented, together with these first results, the next steps to further develop the model will consist of a) application to other types of IQ tests; b) development of new kitas to solve specific complex problems; c) study of multivariate problems, d) and study of time series associated with industrial, medical, economic and social systems.

REFERENCES

[1] S. Legg and M. Hutter, "A collection of definitions of intelligence," *Frontiers in Artificial Intelligence and applications*, vol. 157, pp. 17–24, Jun. 2007.
[2] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, Oct. 1950.
[3] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep Blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.
[4] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager *et al.*, "Building Watson: An overview of the DeepQA project," *AI magazine*, vol. 31, no. 3, pp. 59–79, Jul. 2010.
[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May. 2015.
[7] G. Marcus and E. Davis, *Rebooting AI: Building artificial intelligence we can trust*. Vintage, 2019.
[8] A. C. M. de Lara, "Interpreting the High Energy Consumption of the Brain at Rest," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 46, no. 1, Nov. 2019, p. 30.
[9] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychological review*, vol. 111, no. 4, p. 1036, 2004.
[10] A. Lovett, K. Forbus, and J. Usher, "A structure-mapping model of Raven's Progressive Matrices," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 32, no. 32, 2010, p. 2761–2766.
[11] P. A. Carpenter, M. A. Just, and P. Shell, "What one intelligence test measures: a theoretical account of the processing in the Raven Progressive Matrices Test," *Psychological review*, vol. 97, no. 3, p. 404, 1990.
[12] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.
[13] J. L. Horn and R. B. Cattell, "Refinement and test of the theory of fluid and crystallized general intelligences." *Journal of educational psychology*, vol. 57, no. 5, pp. 253–270, 1966.
[14] S. Zeki, "Abstraction and idealism," *Nature*, vol. 404, no. 6778, pp. 547–547, Apr. 2000.
[15] J. Hawkins and S. Blakeslee, *On Intelligence, How a New Understanding of the Brain Will Lead to the Creation of Truly Intelligent Machines*. Times Books, 2004.
[16] D. R. Hofstadter, *I am a strange loop*. Basic books, 2007.
[17] D. Buonomano, *Your brain is a time machine: The neuroscience and physics of time*. WW Norton & Company, 2017.
[18] H. A. Simon and K. Kotovsky, "Human acquisition of concepts for sequential patterns," *Psychological review*, vol. 70, no. 6, p. 534, 1963.
[19] V. Mountcastle, *An organizing principle for cerebral function: the unit module and the distributed system*. The Mindful Brain (Gerald M. Edelman and Vernon B. Mountcastle, eds.) Cambridge, MA: MIT Press, 1978.

[20] P. Bach-y Rita and S. W. Kercel, "Sensory substitution and the human–machine interface," *Trends in cognitive sciences*, vol. 7, no. 12, pp. 541–546, Dec. 2003.
[21] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Networks*, vol. 22, no. 3, pp. 200–212, Apr. 2009.
[22] D. Gentner, K. J. Holyoak, and N. K. Boicho, *The analogical mind: Perspectives from cognitive science*. MIT press, 2001.
[23] H. Prade and G. Richard, *Computational approaches to analogical reasoning: Current trends*. Springer, 2014, vol. 548.
[24] S. Dehaene, *Consciousness and the brain: Deciphering how the brain codes our thoughts*. Penguin, 2014.
[25] G. W. Leibniz, *Philosophical Essays*, 1690.
[26] U. Schmid and M. Ragni, "Comparing computer models solving number series problems," in *International Conference on Artificial General Intelligence*. Springer, Jul. 2015, pp. 352–361.
[27] J. Hofmann, E. Kitzelmann, and U. Schmid, "Applying inductive program synthesis to induction of number series a case study with IGOR2," in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, 2014, pp. 25–36.
[28] E. W. Rowe, C. Miller, L. A. Ebenstein, and D. F. Thompson, "Cognitive predictors of reading and math achievement among gifted referrals," *School Psychology Quarterly*, vol. 27, no. 3, pp. 144–153, Sep. 2012.
[29] D. L. Dowe and J. Hernández-Orallo, "How universal can an intelligence test be?" *Adaptive Behavior*, vol. 22, no. 1, pp. 51–69, Feb. 2014.
[30] P. Sanghi and D. L. Dowe, "A computer program capable of passing IQ tests," in *4th Intl. Conf. on Cognitive Science (ICCS'03), Sydney*, 2003, pp. 570–575.
[31] J. Mańdziuk and A. Żychowski, "DeepIQ: A Human-Inspired AI System for Solving IQ Test Problems," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2019, pp. 1–8.
[32] F. Martınez-Plumed, J. Hernández-Orallo, U. Schmid, M. Siebers, and D. L. Dowe, "Historical account of computer models solving iq test problems," in *Evaluating General-Purpose AI (EGPAI 2016), 22nd European Conference on Artificial Intelligence (ECAI 2016)*, 2016, pp. 20–21.
[33] N. Dean and G. E. Shannon, *Computational Support for Discrete Mathematics*. American Mathematical Soc., 1994, vol. 15.
[34] Y. Liu, F. He, H. Zhang, G. Rao, Z. Feng, and Y. Zhou, "How Well Do Machines Perform on IQ tests: a Comparison Study on a Large-Scale Dataset," in *IJCAI*, 2019, pp. 6110–6116.
[35] M. Ragni and A. Klein, "Predicting numbers: an AI approach to solving number series," in *Annual Conference on Artificial Intelligence*. Springer, 2011, pp. 255–259.
[36] N. Sloane, "The on-line encyclopedia of integer sequences, 130," *Towards Mechanized Mathematical Assistants. Springer, New York*, vol. 33485, 2007.
[37] N. J. A. Sloane, "The on-line encyclopedia of integer sequences," in *Annales Mathematicae et Informaticae*, vol. 41, 2013, pp. 219–234.
[38] M. Siebers and U. Schmid, "Semi-analytic natural number series induction," in *Annual Conference on Artificial Intelligence*. Springer, Sep. 2012, pp. 249–252.
[39] S. Colton, A. Bundy, and T. Walsh, "Automatic invention of integer sequences," in *AAAI/IAAI*, 2000, pp. 558–563.
[40] J. Hernández-Orallo, F. Martínez-Plumed, U. Schmid, M. Siebers, and D. L. Dowe, "Computer models solving intelligence test problems: Progress and implications," *Artificial Intelligence*, vol. 230, pp. 74–107, Jan. 2016.
[41] M. J. Meredith, "Seek-Whence: A model of pattern perception," Ph.D. dissertation, Indiana University, 1986.
[42] M. M. Bongard, *Pattern Recognition*. Rochelle Park, N.J.: Hayden Book Co., Spartan Books, 1970.
[43] D. Hofstadter, *Fluid Concepts and Creative Analogies*. Basic Books, 1995.
[44] A. A. Mahabal, "Seqsee: A concept-centered architecture for sequence perception," Ph.D. dissertation, Indiana University, 2009.
[45] J. Burghardt, "E-generalization using grammars," *Artificial intelligence*, vol. 165, no. 1, pp. 1–35, Mar. 2005.
[46] C. Strannegård, M. Amirghasemi, and S. Ulfsbäcker, "An anthropomorphic method for number sequence problems," *Cognitive Systems Research*, vol. 22, pp. 27–34, Jun. 2013.
[47] M. Pelillo, K. Siddiqi, and S. W. Zucker, "Continuous-based heuristics for graph and tree isomorphisms, with application to computer vision," in *Approximation and Complexity in Numerical Optimization*. Springer, 2000, pp. 422–445.
[48] C. Strannegård, M. Amirghasemi, and S. Ulfsbäcker, "An anthropomorphic method for number sequence problems," *Cognitive Systems Research*, vol. 22, pp. 27–34, Jun. 2013.
[49] S. Legg and M. Hutter, "A formal measure of machine intelligence," *arXiv preprint cs/0605024*, 2006.
[50] F. Faber, R. Kidder, and T. Lang, "Making the machine intelligent Algebraic methods to make a computer programme pass an IQ test," Mar. 2003.
[51] M. Ragni and A. Klein, "Predicting numbers: an AI approach to solving number series," in *Annual Conference on Artificial Intelligence*. Springer, Oct. 2011, pp. 255–259.
[52] J. Hofmann, E. Kitzelmann, and U. Schmid, "Applying inductive program synthesis to induction of number series a case study with IGOR2," in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, Sep. 2014, pp. 25–36.
[53] U. Schmid and M. Ragni, "Comparing computer models solving number series problems," in *International Conference on Artificial General Intelligence*. Springer, Jul. 2015, pp. 352–361.
[54] C. Strannegård, A. R. Nizamani, A. Sjöberg, and F. Engström, "Bounded Kolmogorov complexity based on cognitive models," in *International Conference on Artificial General Intelligence*. Springer, Jul. 2013, pp. 130–139.

Víctor Corsino is a PhD student in Science and Technology Applied to Industrial Engineering, Universidad de Castilla La-Mancha (UCLM), Spain. He has a degree in electronic engineering from said university, with a mention in advanced electronic technologies, and a master's degree in artificial intelligence from Universidad Politécnica de Madrid (UPM). His current research interests include pattern recognition, machine learning, neuroanatomy and microsensors.

José Manuel Gilpérez is Full Time Professor at the School of Industrial and Aerospace Engineering (EIIA) in Toledo, Universidad de Castilla La-Mancha (UCLM), Spain. His current research interests are focused in applied artificial intelligence in different fields such pattern recognition, machine learning, neurobiology and genetics in partnership with several universities and research centers.

Luis Herrera is a BS in Aerospace Engineering from Universidad Politécnica de Madrid (UPM). He holds a Certificate in Exponential Technologies from Singularity University. He is a Blue Angel Investor in business analytics and energy projects. He is CEO of Smart Technologies Investments, the company developing KitBit project.