KitBit: A New AI Model for Solving Intelligence Tests and Numerical Series

Víctor Corsino, José Manuel Gilpérez, and Luis Herrera

Abstract—The resolution of intelligence tests, in particular numerical sequences, has been of great interest in the evaluation of AI systems. We present a new computational model called KitBit that uses a reduced set of algorithms and their combinations to build a predictive model that finds the underlying pattern in numerical sequences, such as those included in IQ tests and others of much greater complexity. We present the fundamentals of the model and its application in different cases. First, the system is tested on a set of number series used in IQ tests collected from various sources. Next, our model is successfully applied on the sequences used to evaluate the models reported in the literature. In both cases, the system is capable of solving these types of problems in less than a second using standard computing power. Finally, KitBit's algorithms have been applied for the first time to the complete set of entire sequences of the well-known OEIS database. We find a pattern in the form of a list of algorithms and predict the following terms in the largest number of series to date. These results demonstrate the potential of KitBit to solve complex problems that could be represented numerically.

Index Terms—Pattern recognition, number sequences, artificial intelligence.

1 Introduction

Pattern recognition is a main aspect of any intelligent system, and is therefore of great interest to AI [1]. Pattern recognition is defined as the search for regularities and structure in data [2], and is a fundamental part of machine learning. Computational models oriented to pattern recognition have been applied in many areas, such as image processing or voice recognition, and is currently applied with great success thanks to techniques such as Deep Learning. In particular, pattern recognition has traditionally been used to assess inductive reasoning skills in IQ tests. It is natural that this type of problem is of interest in AI, where different models have been developed to solve IQ tests [3], [4], [5] and the prediction of numerical sequences has been proposed as a method to evaluate the computational capabilities of machine learning models [6].

This work presents a new model for pattern recognition that is called KitBit, which successfully solves the largest number of numerical series reported to date through a computer system. To locate complex patterns and make predictions on series of numbers, KitBit foregoes the use of prior information or training, as well as great computational powers. Starting from a basic algorithm, which we call *kita basic*, a set of new different *kitas* is built. This collection of *kitas* constitutes a *toolbox* of procedures to find patterns, which are concatenated in an iterated way to be applied to the variables that define a system or their combinations.

Our model differs from symbolic regression where a recurrence function or relation is inferred from a sequence of numbers. To date, symbolic regression using neural networks has only been applied to simple cases with very limited results [7]. Using KitBit, instead of a mathematical function, the underlying pattern is represented by a sequence of *kitas*, which are equivalent to operations, most of them non-analytic, and which are performed on the elements of the series. In this way, KitBit analyzes the problem from different perspectives through a dynamic process of approximation and framing. This results in the underlying pattern in the form of a sequence of operations that can be easily identified and stored. Once the pattern is known, new terms can be predicted. KitBit algorithms have been used to find the implicit patterns in the integer sequences of the complete Online Encyclopedia of Integer Sequences (OEIS) database [8], [9], many of which are highly complex and reach the largest number of resolved series reported to date. Likewise, KitBit is capable of solving practically all of the numerical series that are present in IQ tests.

2 Resolution of IQ test and numerical series: previous models

Because IQ is considered to be a predominant benchmark for measuring human intelligence [10], [11], IQ tests are an interesting challenge for AI systems. They are also a useful tool to quantify how similar these systems are to human intelligence [12]. Among the many different types of tests, IQ tests based on numerical sequences are fundamental because the other tests can be transformed into a numerical problem. These problems can be defined mathematically by means of a function that assigns the natural numbers to the real numbers: $f : \mathbb{N} \rightarrow \mathbb{R}$ where each element is defined algorithmically. In addition, they are normally based on simple patterns or a combination of patterns, where the numbers are restricted to integer values that are generally small enough to allow easy mental calculations using the four basic arithmetic operations. The number of elements is also relatively small, usually no more than six to eight terms.

• *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 June 10, 2022; revised June 20, 2022.

Few publications are related to the computational resolution of the IQ test in the conventional literature on AI [13]. In 1963, Evans [14], and Simon and Kotovsky [15] designed AI programs that were capable of identifying regularities in patterns in analogy problems and letter series termination problems, respectively [16]. 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 [17]. In 1980, Hofstadter developed a series of computational models in the Copycat project with the main objective of understanding analogy [16]. Between 2006 and 2011, computational models aimed at solving IQ test problems started to become more popular, either trying to understand human cognition or as a method to evaluate AI techniques [16]. Meanwhile, between 2011 and 2014, several models were proposed to solve IQ tests based on numerical series, although some of them performed (in general) worse than human beings [3].

More recently, Liu et at. [3] conducted a comparison study of different state-of-the-art approaches to automatically solve an IQ test. The authors collected an extensive dataset of these tests, containing 10,000 questions of different types, where over 2,500 are numerical sequences. Meanwhile, other studies have focused more generally on numerical series datasets. For instance, Ragni and Klein [18] tested their model, which was based on an artificial neural network (ANN), on a series that was selected from OEIS. At that time, the OEIS database contained 187,440 number series, from which they selected 57,524 that consist of at least 20 numbers with values ± 1000. Using this approach, the authors solved 26,951 of the selected number series. Meanwhile, Siebers and Schmid [19] avoided the use of the OEIS database to test their semi-analytical cognitive model, considering that most of the OEIS series are too complex to be induced by humans. Instead, they used a random generator [20] to construct 25,000 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%.

Some approaches use a very limited number of series to test their models, which are summarized in the reviews provided by Schmid and Ragni [4], and Hernández-Orallo et al. [5]. The latter provides a complete account of 30 computer models that are 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 [21] evaluated the Seek-Whence model using only 12 numerical series (i.e., the Blackburn dozen) to solve the Bongard problem [22]. This approach uses pattern recognition and analogy according to the ideas of Hofstadter [23] without requiring typical mathematical operations. Using a similar approach, Mahabal [24] developed SeqSee (i.e., a computer program that emulates intelligent activities) to extend integer sequences, while avoiding the use of brute force or computing shortcuts that are implausible in humans. Mahabal analyzes 146 selected numerical series related using the group of intelligent activities emulated. Sanghi and Dowe [12] presented a simple computer program that was able to solve IQ tests focused on pattern recognition of certain types of numerical series, such as arithmetic progression, geometric progression, Fibonacci series or powers of a series. Their results are based on 12 IQ test, which are not presented. Burghardt [25] applied an algorithm that was called E-generalization to discover the pattern underlying numerical series. The results were presented in nine series, including alternating and Fibonacci series. Strannegård et al. [26] implemented ASolver, which is a rule-based system that includes memory restrictions that are similar to human capabilities. The system's performance was evaluated with 11 (non-published) IQ tests. Hofmann et al. [27] developed the IGOR2 analytical inductive programming system and compared its achievements with the approaches of Ragni and Klein [18], and Burghardt [25] based on 20 and eight selected series, respectively. Finally, the authors selected 100 series, some of which were included in OEIS, to test the model. Recently, Ryskina and Knight [28] used NPL models trained on OEIS to complete 57 integer series collected from online IQ test preparation websites.

Although the previous models have tried to induce the next numbers in the series, they do not attempt to obtain a symbolic expression from the data. Recently, D'Ascoli et al. [7] trained a neural network to infer the symbolic regression of integer sequences. They evaluate their model on a subset of 10000 series from the OEIS database, which were labeled as 'easy' [29]. For this subset, the model reaches an accuracy of 53% at next term prediction but was only 27% accurate when predicting the next 10 terms. For the symbolic regression, the model achieves an accuracy of 19%.

In summary, the computational models that have been created to induce a pattern in a numerical series, either considering an intelligence test problem or a numerical series in general, use only a very small number of examples or select their examples with certain restrictions to test the model. In contrast, in this paper we will show how KitBit tries to solve the complete set of entire sequences of the OEIS database, as well as all the numerical series used to evaluate the different models found in the literature, to find a pattern in the largest number of numerical series of different types reported to date. This will lay a solid foundation to address other types of problems.

3 The KitBit model

The KitBit model is based on four fundamental components: a basic data storage unit or *edk*, a *kitas* or actions carried out on the *edk*, a pattern search system and a new elements prediction system. KitBit's algorithms are coded using the Python programming language and they are applied to solve numerical series problems by finding their underlying pattern and allowing new terms of the series to be inferred.

3.1 The basic unit or *edk*

For a series of numbers $X = \{x_1, x_2, ..., x_n\}$, where $x \in \mathbb{R}$ and $n \in \mathbb{N} \mid n \geq 2$, the *edk* is constructed as a difference table or a ratio table [30].

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

We will consider that an *edk* provides a solution to a certain problem if a constancy is found in its upper rows. We define 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 $k \in \{1, 2, ..., n-1\}$. In other words, 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 the second case, where $\epsilon = e^r$, $r \in \mathbb{R}_*^-$ is a preset and very small parameter.

3.2 The basic algorithms or *kitas*

The *kitas* are the set of algorithms that are applied in different configurations on a series or on an *edk*. A *kita* is the computational implementation, based on mathematical foundations, of a particular way of approaching the problem. Their combination allows us to find patterns of increasing complexity, while eventually reaching a solution.

Basic (*BAS*) and Divisions (*DIV*). These 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 us to focus on the series:

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

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

Multi-Level (*ML*). This focuses on the diagonals of the *edk*, selecting elements according to increments in the vertical and horizontal directions. This allows us to achieve patterns that may go unnoticed because they are not visible at the base of the *edk*. Let dy and dx be the increments on the vertical and horizontal axis, respectively, with $dy, dx \in \mathbb{N}$. Starting from edk_n, a set of $dy + dx$ child *edks* is generated:

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

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*. The bases of the new child *edks* are obtained by starting from the origin points $(p_{y_0}, p_{x_0}) = (0, 1)$, $(1, 1)$, $(2, 1)$, and so on, following the left-hand outer face of the edk_n. From the corresponding origin, the elements located on the slope formed by dy and dx are extracted. That is, in general, the base elements of the new *edks* are obtained by applying the equation:

$$(p_{y_\alpha}, p_{x_\alpha}) = (p_{y_0} + \alpha \cdot dy, p_{x_0} + \alpha \cdot dx) \tag{4}$$

with α taking values from 0, 1, 2, ... until the iteration corresponding to the limit of the *edk*.

Focusing (*FOC*). This is used to divide the original series into several subseries to find the implicit pattern in each. It uses two parameters, an initial offset s and a list of the size of the fragments to be extracted $D = \{d_1, d_2, ..., d_l\}$ iteratively. For an original sequence of length n, if we denote the sum of divisions as $s_d = \sum_{r=1}^{l} d_r$, then it must be satisfied that $s \in \mathbb{N} \mid 0 \leq s \leq s_d$ and $d, l \in \mathbb{N} \mid s + s_d \leq n \geq 4$. If this is fulfilled, then we start the *kita* by moving the first s elements of the original series, which are accumulated in what we will call *remaining series* R and which will be used later. Next, starting from the resulting *shifted series*, fragments are extracted according to the values contained in D, forming a total of l new series. Each new sequence S_m, with $m \in \{1, 2, ..., l\}$, is formed from successive iterations of fragments of size d_l of the remaining shifted series, which continues until this series is exhausted. The coordinates of the elements that form any series S_m can be calculated using this equation, pruning the result where $c \in C_{S_m} > n$:

$$C_{S_m} = \bigcup_{p=0}^{\lceil \frac{n-s}{s_d} \rceil - 1} \bigcup_{q=1}^{d_m} \{s + f(m) + q + p \cdot s_d\} \tag{5}$$

where C_{S_m} is the set of coordinates, $\cup$ is the orderly union with preserving of all the elements of the corresponding series, and $f(m) = \sum_{r=1}^{m-1} d_r$ if $m \neq 1$, or $f(m) = 0$ otherwise.

Next, new fragments will be added to each subseries S_m, this time obtained from the *remaining series*. These elements are inserted starting at the end of R and starting with the last series S_l. On each of the l iterations, we obtain the subseries of remainders R_m that would correspond to S_m extracting the last d_m elements of R. Note that some R_m series may have fewer elements than d_m or may even be empty sequences if R is exhausted. Therefore, focusing gives the series of the following type:

$$FOC(s, D)\,[edk_n] \rightarrow X_1, ..., X_l = R_1 \cup S_1, ..., R_l \cup S_l \tag{6}$$

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

Analogy (*ANA*). As in the case of *FOC*, this *kita* is used to break the original series into new sequences to more easily find the implicit pattern. It also uses two parameters: an initial offset $s \in \mathbb{N} \mid s \leq n-4$, which determines the number of elements ignored at the beginning of the series, and the number of elements per group or new child 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 the previous child series have been generated, for the first $t-1$ groups, a constancy is sought in the same row r of their *edk*, where $r \in \{1, 2, ..., n-1\}$. The prediction is made directly by placing the corresponding part of the *edks* of the previous groups on the group t.

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

$$EXP(e)\,[edk_n] \rightarrow \{x_1^e, x_2^e, ..., x_n^e\} \tag{7}$$

where $e \in \mathbb{R}_*$, except in cases like $0 \in X$ or $\exists\, x \in X \mid x < 0$. The *kita* ends by applying *BAS* to the result.

Logarithm (*LOG*). This calculates the first level of the *edk* as the logarithm of the lower level, where the absolute value of each element is the base of the logarithm and the absolute value of the next element is the exponent. By applying this tool to any edk_n, we have:

$$LOG\,[edk_n] \rightarrow \left\{\log_{|x_1|}|x_2|, \log_{|x_2|}|x_3|, ..., \log_{|x_{n-1}|}|x_n|\right\} \tag{8}$$

The general equation that implements this *kita* is:

$$y_i^1 = \log_{|x_i|}|x_{i+1}| \tag{9}$$

where $i \in \{1, 2, ..., n-1\}$. Finally, a new *edk* is created using differences with this new series as the basis.

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

$$DOP\left(\mathcal{O}_1, \mathcal{O}_2\right)\left[edk_n\right] \rightarrow \\ \left\{\mathcal{O}_1\left[x_2, x_1\right], \mathcal{O}_2\left[x_3, x_2\right], ..., \mathcal{O}_j\left[x_n, x_{n-1}\right]\right\} \tag{10}$$

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

$$y_i^1 = \mathcal{O}_{f(i)}\left[x_{i+1}, x_i\right], \quad f(i) = \begin{cases} 1, & i\%2 \neq 0 \\ 2, & i\%2 = 0 \end{cases} \tag{11}$$

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

Specular Symmetry (*SSYM*). This is used to predict the following items on the basis of the *edk* if a symmetric arrangement is detected. Generally, if from a position $j \in \{\mathrm{round}(n/2), ..., n-1\}$ of the series, then its elements begin to repeat in reverse order and the remaining elements of the series can be obtained by continuing this subsequence.

Repetition Symmetry (*RSYM*). This predicts the next elements in the basis of the *edk* if a repeating group is detected throughout it. 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. 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:

$$RSYM\left[edk_n\right] \rightarrow \{x_1, x_2, ..., x_j, x_1, x_2, ..., x_j, ...\} \tag{12}$$

Different Groups of Equal Elements (*DGE*). This groups consecutive and equal elements of the original series creating two new series: one of them formed by the element that is repeated in each group, and another formed with the sizes of the groups. That is, let $G = \{G_1, G_2, ..., G_t\}$ be the groups formed, with $t \in \mathbb{N} \mid 2 \leq t \leq n-1$, the two new series are formed as follows:

$$DGE\left[edk_n\right] \rightarrow \begin{cases} X_1 = \{G_1[1], G_2[1], ..., G_t[1]\} \\ X_2 = \{\mathcal{L}[G_1], \mathcal{L}[G_2], ..., \mathcal{L}[G_t]\} \end{cases} \tag{13}$$

where $G_j[1]$, $j \in \{1, 2, ..., t\}$ is the first element of the group j, and $\mathcal{L}$ is an operator which calculates its length. On these two series, X_1 and X_2, two *edks* are built by means of differences between the elements.

Different Groups of Different Elements (*DGD*). This *kita* is similar to DGE but in this case the elements within each formed group are not equal. It also creates two subseries from the basis of *edk*: the first consisting of the groups of elements whose progression repeats in the sequence, and the second consisting of the lengths of those groups. The original series is divided into:

$$DGD\left[edk_n\right] \rightarrow \begin{cases} X_1 = X_{\neq} \\ X_2 = \{\mathcal{L}[G_1], \mathcal{L}[G_2], ..., \mathcal{L}[G_t]\} \end{cases} \tag{14}$$

where $X_{\neq}$ is a subsequence of X with its different elements. On these two series, X_1 and X_2, two *edks* are built by means of differences between the elements.

Split Of Elements (*SOE*). This is used when in each element of the series all its digits have the same numerical value and there is a relationship between the number of digits of each element. In this way, from the original series, two new subseries are obtained: one with the digit that is repeated in each element, and another with the number of digits of each of them:

$$SOE\left[edk_n\right] \rightarrow \begin{cases} X_1 = \{\mathcal{R}[x_1], \mathcal{R}[x_2], ..., \mathcal{R}[x_n]\} \\ X_2 = \{\mathcal{N}[x_1], \mathcal{N}[x_2], ..., \mathcal{N}[x_n]\} \end{cases} \tag{15}$$

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

3.3 Search algorithms

The pattern search method used by KitBit is based on a state tree. A connected tree or graph without cycles [31] $G(S, K)$ 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 $m \in \mathbb{N} \mid 1 \leq m \leq 1 + \sum_{r=1}^{depth-1} n^r$ and $depth \in \mathbb{N} \mid depth \geq 1$ is the depth of the tree. Aside from the root state, the other nodes have a single parent and a number of descendants ranging from 0 to n that 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 (16).

$$s = \left\{ \begin{array}{c} \{edk_1, edk_2, ..., edk_l\} \\ k \in K \\ \{c_1, c_2, ..., c_l\} \ , \ c \in \{0, 1\} \\ \{f_1, f_2, ..., f_l\} \ , \ f \in \{1, 2, ..., l\} \end{array} \right\} \tag{16}$$

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 component is the action. The third component 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 not. The fourth component is another ordered series 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 all of the elements are ones in the third component of s, then this means that this state is a solution state to the problem.

This tree structure can be traversed in several ways by implementing different pattern search strategies. KitBit uses an uninformed search algorithm, with the popular Breadth First Search (BFS) and Depth First Search (DFS) strategies. In both algorithms, reaching a solution state also implies that we obtain the path or sequence of actions performed in each previous state. We denote this list as $K_s \subseteq K$, where $\subseteq$ is an orderly subset of K with preserving of all the elements. This sequence constitutes a recurrence formula, although not in the mathematical sense but as a set of algorithms that, when executed, produce the numerical series in question. Therefore, KitBit is capable of both predicting new items and providing the actions that generate them. The number of new items generated by K_s may be unlimited or not, depending on the structure of the series and the *kitas* used. Likewise, there may be more than one list of actions K_s solution to the problem. Therefore, KitBit implements the possibility of not stopping when a first solution is found but continuing to generate new states to search for all solutions. Finally, following Occam's principle, we will consider the optimal solution as the simplest—this being the one that requires the fewest number of *kitas*.

3.4 Prediction

Once the sequence of actions to solve the problem K_s has been determined, they are applied in reverse order to obtain new elements of the series. Starting from the final state reached, the operations to be carried out can be divided into two fundamental parts: the prediction of new elements and the insertion of new elements.

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 triangular shape. Next, the *edk* is traversed in an inverse way by applying (17).

$$\beta_i^j = \begin{cases} \beta_i^{j+1} + \beta_{i-1}^j, & \text{BAS} \\ \beta_i^{j+1} \cdot \beta_{i-1}^j, & \text{DIV} \end{cases} \tag{17}$$

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-hand of the *edk*. For *EXP*, once the prediction is made it is necessary 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*, *SSYM* and *RSYM* perform the inference of new elements directly in the pattern search process, so the prediction process in these cases is not necessary. Meanwhile, for *LOG*, once the elements that are 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 equation:

$$\beta_i^0 = \left(\beta_{i-1}^0\right)^{\beta_i^1} \tag{18}$$

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 $\mathcal{O}_1$ and $\mathcal{O}_2$ between the base elements and its top level:

$$\beta_i^0 = \mathcal{O}'_{f(g)}\left[\beta_i^1, \beta_{i-1}^0\right], \quad f(g) = \begin{cases} 1, & g(i,n) \\ 2, & \overline{g(i,n)} \end{cases} \tag{19}$$

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 new elements. The insertion of new elements in the original series depends on the sequence of *kitas* K_s solution of the problem. In general, terms predicted in the child *edks* are inserted directly into their parent *edk*. This is the case for *BAS*, *DIV*, *ANA*, *EXP*, *SSYM* and *RSYM*, where the elements obtained in the child *edk* are inserted after the last base element of the parent *edk*. Other *kitas* have their own peculiarities. In the case of *RED*, the elements predicted in the child *edk* are inserted into the index row r of the parent *edk*. For *LOG* and *DOP*, the predicted numbers are placed after the last element in the $r = 1$ row of the the parent *edk*. In the case of *ML*, the terms predicted in the descendant $dy + dx$ *edks* are placed at the following points on the corresponding slope of the parent *edk*, using (4) to calculate the coordinates. Similarly, for *FOC*, the predicted elements are inserted into the child l *edks* at the base of its parent, computing the new coordinates with (5), changing n to $n + n_\beta$, where n_β is the number of elements to predict. In *DGE*, *DGD* and *SOE*, two child *edks* X_1 and X_2 are calculated from the original. If we denote these secondary series under the indices a for the first, and b for the second, then the respective base element of the parent *edk* $\beta^0 \in B$ is obtained for each pair of predicted elements $\beta_a^0 \in B_a$, $\beta_a^0 \in B_b$ located at the same positions in both secondary series. In *DGE* the element β_a^0 is repeated β_b^0 times. In *DGD*, the next element is formed by extracting as many elements as β_b^0 indicates from X_1. In *SOE*, β^0 is constructed as $\beta^0 = \beta_a^0 \cdot \left(10^0 + 10^1 + ... + 10^{\beta_b^0}\right)$. This process is applied repeatedly and propagates until the last state is reached.

Table 1 shows some example applications of the *kitas* to predict new elements of the series. For clarity, very simple sequences that resolve to a single *kita* have been selected. In each row, the original sequence appears at the base of the *edk* in the 'Original series *edk* ' column. The '*kita edks*' and '*kita* prediction' columns contain the *edk* generated by *kita* and the prediction made, respectively. Finally, the base of the *edk* of the 'Result *edk* ' column contains the final prediction.

4 Tests and Results

An initial sample of 90 series, collected from real IQ tests, has been used to launch the model. Next, KitBit has been confronted with all the series used to test the computational models oriented to the resolution of numerical series found in the bibliography. Finally, the model faces the resolution of the series collected in the OEIS database, which is composed of more than 340,000 series of all kinds, without any type of limitation regarding its nature and characteristics.

4.1 Series from IQ test

Initially, the system was challenged with the set of 90 non-trivial series based on the IQ tests that are summarized in Table 2. A computer with an Intel Core i7 processor and 16 GB of RAM running a 64-bit Windows 10 operating system was used. The results are shown in Tables 3 and 4.

Table 3 shows the results of the four modes in which the algorithm was executed. The labels *S1Z* and *S2Z* correspond to the results using the BFS method stopping at the solution state, with constancy at one and two levels of the *edk*, respectively. The labels *N1Z* and *N2Z* also correspond to the results using the BFS method, in this case without stopping at the solution state, with constancy at one level and two levels, respectively. In all of the execution modes, the solution to the problem was obtained in 97.8% of the cases; that is, all

TABLE 1
Kita Application Examples.

kita	Original *edk*	*kita edks*	*kita* prediction	Result *edk*
RED(1)	32/27 3/4 8/9 2 3/2 4/3 1 2 3 4 3 3 6 18 72	0 0 0 1 1 1 1 2 3 4	0 0 0 0 0 0 0 1 1 1 1 1 1 2 3 4 5 6	32/27 3/4 8/9 2 3/2 4/3 1 2 3 4 5 6 3 3 6 18 72 360 2160
ML(1,1)	1 0 1 1 1 2 1 2 3 5 2 3 5 8 13 3 5 8 13 21 34	0 0 0 3 3 3 0 0 0 2 2 2	0 0 0 0 0 3 3 3 3 0 0 0 0 0 2 2 2 2	1 0 1 2 1 1 2 3 5 1 2 3 5 8 13 2 3 5 8 13 21 34 3 5 8 13 21 34 55 89
FOC(0,{1,1})	−16 8 −8 −4 4 −4 2 −2 2 −2 2 4 2 4 2 −6 −4 0 2 6 8	0 6 6 −6 0 6 0 6 6 −4 2 8	0 0 6 6 6 −6 0 6 12 0 0 6 6 6 −4 2 8 14	−16 8 −8 −4 4 −4 2 −2 2 −2 2 4 2 4 2 −6 −4 0 2 6 8 12 14
ANA(0,4)	28 −7 21 0 −7 14 0 0 −7 7 2 2 2 −5 2 1 3 5 7 2 4	0 0 0 2 2 2 1 3 5 7 2 2 4	0 0 0 2 2 2 1 3 5 7 0 0 2 2 2 2 4 6 8	28 −7 21 0 −7 14 0 0 −7 7 2 2 2 −5 2 1 3 5 7 2 4 6 8
EXP(1/4)	60 50 110 15 65 175 1 16 81 256	0 0 0 1 1 1 1 2 3 4	0 0 0 0 0 0 0 1 1 1 1 1 1 2 3 4 5 6	60 50 110 15 65 175 1 16 81 256 625 1296
LOG	−64812 65040 228 −65280 −240 −12 65536 256 16 4	0 0 0 0.5 0.5 0.5	0 0 0 0 0 0 0 0.5 0.5 0.5 0.5 0.5	−64812 65040 228 −65280 −240 −12 65536 256 16 4 2 $\sqrt{2}$
DOP(−,÷)	19 −6 13 3 −3 10 2 5 2 12 3 5 10 12 24	0 0 0 0 0 0 2 2 2 2	0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2	19 −6 13 3 −3 10 2 5 2 12 3 5 10 12 24 26 52
SSYM	{1, 2, 4, 8, 4}	-	{1, 2, 4, 8, 4, 2, 1}	-
RSYM	{1, 0, 2, 1, 0}	-	{1, 0, 2, 1, 0, 2, 1}	-
DGE	14 −8 6 4 −4 2 −2 2 −2 0 2 0 2 0 0 1 3 3 5 5 5	0 2 2 1 3 5 0 1 1 1 2 3	0 0 2 2 2 1 3 5 7 0 0 1 1 1 1 2 3 4	14 −8 6 4 −4 2 −2 2 −2 0 2 0 2 0 0 1 3 3 5 5 5 7 7
DGD	−26 14 −12 −6 8 −4 2 −4 4 0 0 2 −2 2 2 2 2 4 2 4 6	0 2 2 2 4 6 0 1 1 1 2 3	0 0 2 2 2 2 4 6 8 0 0 1 1 1 1 2 3 4	−26 14 −12 −6 8 −4 2 −4 4 0 0 2 −2 2 2 2 2 4 2 4 6 2 4
SOE	90 10 100 1 11 111	0 0 0 1 1 1 0 1 1 1 2 3	0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 2 3 4 5	90 10 100 1 11 111 1111 11111

of the series except for two of them. The solution found, which we will call type A solution, consists of an algorithmic pattern that is constituted by a list of *kitas*, which allows us to reproduce all of the known terms of the series. Table 3 also shows the minimum number of elements needed to find the solution; that is, the other known elements are obtained by taking n_e terms of the original series. In most series, this number varies between three and nine. This shows that, in general, the number of terms needed to find the underlying pattern is very small. The table also presents the depth of the graph used; that is, the percentage of cases in which it has been necessary to use from one to four *kitas* to solve the problem. We can see that about half of the series need only one *kita*. This indicates that, in general, they are relatively simple series. This depth is always greater when constancy is imposed on two levels of the *edk* because more complex patterns result. Resolution times are always less than a second, with an average of 15 ms.

TABLE 2
Number Series from IQ Test.

i	sequence	i	sequence	i	sequence
0	0, 1, 1.7071, 2.3660, 3, 3.6180, 4.2247, 4.8229	30	2, 2, 4, 4, 8, 8, 16, 16, 32, 32, 64, 64, 128, 128, 256, 256	60	0, 2, 9, 28, 75, 186, 441, 1016, 2295, 5110, 11253, 24564
1	2, 16, 4, 256, 16, 65536, 256, 4294967296, 65536	31	8, 6, 4, 3, 1, -1, -2, -4, -6, -7, -9, -11, -12, -14, -16	61	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	32	362880, 40320, 5040, 720, 120, 24, 6, 2, 1, 1	62	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, 135	33	14, 1, -5.5, -8.75, -10.375, -11.1875, -11.59375	63	2, 10, 26, 50, 82, 122, 170, 226, 290, 362, 442, 530, 626
4	0, 3, 12, 17, 102, 109, 872, 881	34	-1, -1, 0, 2, 5, 9, 14, 20	64	4, 1, 0, 1, 4, 9, 16, 25, 36
5	0, 3, 8, 24, 63, 168, 440, 1155, 3024, 7920	35	-2, 3, 1, 4, 5, 9, 14, 23	65	9, 3, 6, 6, 2, 5, 3, 1, 4, 0, 0, 3, -3
6	1, 3, 6, 10, 15, 21, 28, 36	36	-3, -1, -4, 0, -5, 1, -6, 2, -7	66	-9, 1, -5, 3, -4, 2, -6, -2, -11, -9
7	1, 0, -1, 0, 1, 0, -1, 0, 1	37	-4, -1, -3, 0, -2, 1, -1, 2, 0	67	97.5, 57, 30, 12, 0, -8, -13.333
8	1, 3, 5, 7, 9, 11, 13, 15, 17	38	1, 2, 0, 2, -1, 2, -2, 2, -3	68	6, 4, 3, 3, 2, 2, 3, 1, 6, 0, 11, -1
9	1, 2, 4, 7, 11, 16, 22, 29	39	1, -1, 0, -3, -1, -5, -2, -7, -3, -9	69	7, 16, 52, 196, 772, 3076, 12292
10	1, 4, 9, 16, 25, 36, 49, 64, 81	40	0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1	70	2, 3, 5, 4, 2, 4, 7, 5, 2, 5, 9, 6, 2, 6, 11, 7, 2
11	3, 5, 10, 12, 24, 26, 52, 54, 108	41	-2, 0, -3, 1, -4, 2, -5, 3, -6	71	8, 0, 2, 0, 0, 0, 2, 0, 8, 0, 18, 0
12	3, 4, 8, 17, 33, 58, 94, 143	42	23, 34, 45, 56, 67, 78	72	3, 0, 1, -1, -2, -5, -9, -16
13	3, 6, 18, 72, 360, 2160, 15120, 120960	43	-4, -3, 0, 5, 12, 21, 32	73	1, 0, -1, -1, -2, -1, -3, -4, -1, -5
14	4, 5, 8, 13, 20, 29, 40, 53	44	-4, -2, 2, 8, 16, 26, 38	74	3, 9, 22.5, 45, 67.5, 67.5, 33.75, 0
15	5, 11, 17, 23, 29, 35, 41	45	6, 4, 0, -6, -14, -24, -36	75	0, 2, 9, 24, 50, 90, 147, 224
16	11, 9, 7, 5, 3, 1, -1, -3	46	-2, 0, 4, 10, 18, 28, 40	76	3, 8, 16, 28, 45, 68, 98, 136
17	30, 29, 27, 26, 24, 23, 21, 20, 18, 17, 15	47	-6, -1, 5, 12, 20, 29, 39	77	0, 0, 4, 5, 14, 16, 30, 33, 52, 56
18	144, 121, 100, 81, 64, 49, 36, 25	48	6400, 1600, 400, 100, 25, 6.25, 1.5625	78	0, 8, 15, 35, 48, 80, 99, 143, 168, 224
19	2, 2, 4, 6, 10, 16, 26, 42, 68	49	0, 7, 24, 51, 88, 135, 192	79	4, 32, 108, 256, 500, 864, 1372, 2048
20	81, 27, 9, 3, 1, 1/3, 1/9	50	2, 4, 12, 48, 240, 1440, 10080	80	0, 17, 74, 195, 404, 725, 1182, 1799
21	1, 1, 2, 3, 5, 8, 13, 21	51	-10, 12, 44, 86, 138, 200, 272	81	6, 24, 60, 120, 210, 336, 504, 720
22	21, 20, 18, 15, 11, 6, 0	52	1, 0, 1, 1, 1, 2, 1, 3, 1, 4	82	3, 6, 15, 42, 123, 366, 1095, 3282
23	8, 6, 7, 5, 6, 4, 5, 3, 4	53	-1, 2, -2, -4, 8, -32, -256, 8192	83	23, 31, 53, 83, 135, 217, 351, 567
24	4294967296, 65536, 256, 16, 4, 2	54	-10, 0, 15, 35, 60, 90, 125, 165	84	2, 6, 19, 53, 126, 262, 491, 849, 1378
25	3, 7, 14, 24, 37, 53, 72	55	-2, -1, 1, 5, 13, 29, 61, 125	85	1, 2, 5, 9, 16, 27, 45, 74
26	-3, -1, 2, 6, 11, 17, 24	56	-3, -2, 0, 1, 3, 4, 6, 7, 9	86	0, 2, 6, 12, 20, 30, 42, 56, 72, 90, 110, 132
27	-1, 0, 3, 8, 15, 24, 35	57	1, 2, 6, 14, 29, 56, 102, 176, 289	87	3, 5, 8, 12, 17, 23, 30, 38
28	-9, 2, 12, 21, 29, 36, 42	58	0, 1, 2, 8, 29, 80, 181, 357, 638	88	5, 8, 20, 68, 260, 1028, 4100
29	1, 0, 0, 1, 3, 6, 10, 15	59	8, 1, 0, -1, -8, -27, -64, -125, -216	89	-6, -4, 0, 2, 6, 8, 12, 14, 18

TABLE 3
Results Obtained for the Series in Table 2.

	results (%)	depth of the graph				time (ms)	n_e (%)							
	solved	1 (%)	2 (%)	3 (%)	4 (%)	t_{med}	3	4	5	6	7	8	9	10-16
S1Z	97.8	43.2	36.4	19.3	1.1	16.0	9.1	45.5	13.6	17.1	1.1	5.7	1.1	6.8
S2Z	97.8	43.2	31.8	23.9	1.1	15.6	-	9.1	43.2	13.6	8.0	12.5	8.0	5.6
N1Z	97.8	43.2	25.0	23.9	7.9	-	16.0	42.0	18.2	10.2	4.5	3.4	-	5.7
N2Z	97.8	43.2	20.5	28.4	7.9	-	1.1	14.8	37.5	21.5	8.0	8.0	8.0	1.1

TABLE 4
Use of *Kitas* to Solve the Series of Table 2 (%).

kita:	*ANA*	*BASIC*	*DGD*	*DGE*	*DIV*	*DOP*	*EXP*	*FOC*	*LOG*	*ML*	*RED*	*RSYM*	*SOE*	*SSYM*
S1Z	2.3	100.0	0.0	0.0	17.0	2.3	1.1	26.1	2.3	10.2	17.0	0.0	0.0	0.0
S2Z	2.3	100.0	0.0	0.0	17.0	2.3	1.1	23.9	2.3	10.2	21.6	1.1	0.0	1.1
N1Z	0.0	100.0	0.0	0.0	11.4	3.4	5.7	26.1	1.1	20.5	27.3	0.0	0.0	1.1
N2Z	0.0	100.0	0.0	0.0	10.2	3.4	8.0	15.9	2.3	20.5	30.7	8.0	0.0	2.3

To solve this group of series, a set of 60 *kitas* was used, which are formed by the types described in Section 3.2, applied with different parameters. Table 4 shows the percentage of use of each type of *kita*, which provides a heuristic to improve the efficiency of the model.

In conclusion, KitBit successfully solves almost all of this first set of non-trivial series based on IQ tests, using standard computer equipment in sub-second times. This result provides enough confidence in the model to deal with more complex problems, starting with those raised in previous models that appear in the literature.

4.2 Series from previous models

KitBit was then tested with a new set of 67 series compiled from different articles dedicated to proposing procedures for solving numerical series [30], [33], [18], [35], [36], [32] and [34]. The list of the problems is summarized in Table 5. Column *i* indicates the index in this dataset and *a* indicates the reference article. KitBit approaches this list using the same set of *kitas* as in the previous subsection. The results are shown in Tables 6 and 7.

Table 6 shows the results for this set of number sequences in the four execution modes that were given earlier. The number of solved problems is between 73.1% and 89.6%, and is able to improve up to 91% in all cases if we increase the number of terms of some series. In other words, the model solves the entire set of series evaluated with a higher success rate than previous models, except for six of them. In this case, the solutions are also of type A; that is, in all cases an algorithmic pattern is found that includes all the terms of the series. The depth of the graph shifts towards the use of a greater number of *kitas*, which indicates the greater complexity of the series when compared with the previous set. The number of elements necessary to solve the problem is shown in Table 6. Between three and 10 elements are necessary to find the solution, as we found with the previous group. Table 7 shows the percentage of use of each type of *kita*. This shows that some that had not been used with the previous group of series are relevant in this case. Resolution times were also similar to the previous group, always below one second, with an average time of 34 ms.

We can make several observations if we compare these results with those presented in the publications from which the sequences were extracted. Our system obtains the pattern for the series resolved in [30]. KitBit is able to obtain the solution of all the series evaluated in [33], while the authors fail to obtain some of them using two different methodologies. KitBit manages to correctly solve the 20 problems posed in [18], while its authors solve 17 of them and the IGOR2 model [35] manages to solve 14. Likewise, KitBit algorithms also solve all of the series proposed to test the IGOR2 model [36], even the one that this model fails to obtain. For the series proposed to test SeqSolver [32], KitBit can solve all of them, except one. Finally, regarding the series proposed to validate the ASolver [34] method, KitBit finds a pattern for all but five of them. In conclusion, KitBit is capable of solving the numerical series of IQ tests reported in the bibliography, as well as other series not related to the IQ tests, and performs better than all of the methods that have previously been proposed.

4.3 Series from OEIS database

OEIS is the largest database of the entire series, which compiles all available information about them and is widely cited in the literature. Its size is continuously growing and, at the time of our tests, it contained a total of 347,736 sequences. The repeated sequences were eliminated from this initial set, as well as those that have less than three elements or less than four if they are formed by the repetition of the same element. The sample on which the KitBit algorithms were finally run was 341,553 entire series. The BFS technique was used without stopping at the solution state, with single-level constancy of the *edks* and an epsilon of $\epsilon = e^{-18}$. Initially, 55 *kitas* and a state tree depth equal to two were used. Next, a depth equal to three was used on unsolved problems. Again, on the unsolved cases, a depth equal to four was applied. In this case we reduced the number of *kitas* to 34 and 32 to limit the total resolution time of this large set of problems. As in the previous data sets, the resolution of each sequence was carried out using the minimum number of elements necessary to find the pattern capable of reproducing the complete sequence as it appears in the OEIS database. Given the large number of series to be solved, the algorithms were run using Google's free cloud resources through Google Colab.

We start by addressing the resolution of the 341,553 series, finding the solution in 87,514 cases, or 25.6% of the total; as shown in Table 8. This is the largest number of series resolved to date for the OEIS database. Previously, 26,951 sequences (7.89% of the database) were reported from a selection of 57,524 series with at least 20 elements and values less than 1,000 [18]. The authors of this study considered a series solved when they are 'able to correctly predict the last number of the series'. In our case, we are always able to predict a larger number of elements, even the entire series, depending on the *kitas* used to obtain the solution. In this way, we find two types of solutions: type A solutions (as mentioned previously), and a new type of solutions that we will call type B. Type A solutions are those in which an algorithmic pattern is obtained that allows us to reproduce the sequence starting from a given term, using all previous terms up to the first to get that pattern. For this type, KitBit has managed to solve 28,293 series, or 8.3% of the total. Type B solutions are those in which we are capable of reproducing the sequence starting from a certain term. However, the pattern obtained has not taken into account all of the previous terms but rather a finite number of them. To this last type fundamentally belong the solutions that use *SSYM* that, by not using all the initial elements of the series, allow forward prediction but do not include these in the pattern found. KitBit has managed to find a type B solution for 59,221 cases, or 17.3% of the total, where the number of elements n that it is able to predict depends on the complexity of the series and the number of known elements. In type B solutions, the next term is predicted in 44.4% of the cases and from two to nine terms in 45.7% of them. In 9.9% of the remaining solutions, more than 10 terms are obtained. For type A solutions, more than 10 terms are predicted in 65% of them. The extrapolation of the series beyond the known elements, for both type A and type B solutions, depends on the *kitas* used and the pattern found.

By analyzing the unresolved 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;

TABLE 5
Numerical Series Collected from the Literature.

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

TABLE 6
Results Obtained for the Series in Table 5.

	results (%)		depth of the graph				time (ms)	n_e (%)							
	solved	solved+	1 (%)	2 (%)	3 (%)	4 (%)	t_{med}	3	4	5	6	7	8	9	10
S1Z	88.1	91.0	26.2	47.6	21.3	4.9	30.6	32.8	29.5	8.2	18.0	6.6	1.6	-	3.3
S2Z	73.1	91.0	21.3	32.8	36.1	9.8	34.0	8.2	29.5	19.7	14.8	19.7	3.3	1.6	3.2
N1Z	89.6	91.0	26.2	32.8	34.4	6.6	-	41.0	23.0	23.0	4.9	4.9	3.2	-	-
N2Z	83.6	91.0	21.3	26.2	31.2	21.3	-	13.1	31.1	16.4	21.3	11.5	6.6	-	-

TABLE 7
Use of *Kitas* to Solve the Series of Table 5 (%).

kita:	*ANA*	*BASIC*	*DGD*	*DGE*	*DIV*	*DOP*	*EXP*	*FOC*	*LOG*	*ML*	*RED*	*RSYM*	*SOE*	*SSYM*
S1Z	0.0	100.0	0.0	0.0	29.5	1.6	3.3	24.6	0.0	21.3	21.3	1.6	1.6	0.0
S2Z	0.0	100.0	0.0	3.3	26.2	1.6	6.6	19.7	0.0	32.8	39.3	3.3	1.6	0.0
N1Z	0.0	100.0	0.0	0.0	14.8	4.9	4.9	24.6	0.0	36.1	32.8	1.6	1.6	0.0
N2Z	0.0	100.0	0.0	0.0	23.0	8.2	8.2	11.5	0.0	42.6	44.3	13.1	1.6	0.0

b) series that grow very quickly, such as exponential functions or n^{th} powers, where the value between contiguous elements of the series increases by one order of magnitude; c) series based on prime numbers; and d) series based on combinations of other series. Among these categories, we find that 4.6% of series contain very large and very small numbers. Although these are very difficult for our algorithms to solve, they are capable of solving some of them (0.3%). The series that result from the combination of others are 17.4% of the total and suppose an artificial degree of difficulty that, at this time, our system does not intend to address. Series based on prime numbers represent 16%, which require new algorithms to be incorporated. Even with this degree of difficulty, KitBit is able to solve 51,168 out of 240,617 series of this type, or 21.3% of them. Out of these categories, the OEIS database contains 100,936 series, for which KitBit finds solutions in 36% of them; as shown in Table 8. By further restricting the selection criteria, as in other models, the results would improve. Likewise, it is expected that a greater number of series would be resolved by expanding the set of *kitas* and the depth of the graph.

The variety of series collected in OEIS, many of which are highly complex, is reflected in the depth of the state tree. We observe that for depths of 1, 2, 3 and 4 levels, the percentage of resolved sequences is 23.4%, 21.4%, 28.2% and 27.0%, respectively. As we can see, a quarter of the resulting

TABLE 8
Results for the Series in the OEIS Database.

	Total series	Type A	Type B	Type A + Type B
Entire database	341553	28293 (8.3%)	59221 (17.3%)	87514 (25.6%)
Selection	100936	11938 (11.8%)	24408 (24.2%)	36346 (36.0%)

TABLE 9
Most Frequent Patterns in the Type A Series. n_s is the Number of Series Solved Using this Pattern.

pattern	n_s	pattern	n_s	Pattern	n_s
BASIC (only)	6623	*ML*(2,1), *ML*(0,1), *FOC*(0,{1,1})	310	*RED*(6), *DGD*	232
FOC(0,{1,1})	1992	*RED*(1), *DIV*	309	*RED*(4), *DGD*	213
ML(1,1), *ML*(0,1), *FOC*(0,{1,1})	1289	*ANA*(0,5)	303	*RED*(7), *DGD*	212
FOC(0,{1,1,1,1})	873	*ANA*(0,3)	285	*FOC*(0,{1,1,1}), *RED*(1), *DIV*	206
FOC(0,{1,1,1})	628	*RED*(1), *DGE*, *FOC*(0,{1,1})	278	*RSYM*	195
RED(1), *RSYM*	564	*RED*(1), *ML*(0,1)	275	*FOC*(0,{1,1}), *RED*(1), *DIV*	179
ML(0,1), *FOC*(0,{1,1})	488	*ML*(1,1)	272	*RED*(2), *DOP*(×,+)	178
RED(1), *ML*(0,1), *FOC*(0,{1,1})	429	*ANA*(0,4)	269	*FOC*(0,{1,1,1}), *FOC*(0,{1,1,1,1})	177
ANA(0,2)	370	*FOC*(0,{1,1,1}), *FOC*(0,{1,1})	236	others	10908

TABLE 10
Examples of Solved Sequences.

OEIS index	sequence	pattern	prediction
A071420	7, 8, 5, 5, 3, 4, 4, 6, 9, 7, 8, 8, 7, 8, 5, 5, 3, 4, 4, 6, 9, 7, 8, 8	*FOC*(0,{1,1,1}), *FOC*(0,{1,1,1,1})	7, 8, 5, 5, 3, 4, 4, 6, 9
A082310	0, 1, 7, 57, 455, 3641, 29127, 233017	*FOC*(0,{1,1}), *RED*(1), *DIV*	1864135, 14913081, 119304647, 954437177
A121294	1, 2, 5, 8, 10, 12, 17, 22, 27, 30, 33, 36, 43, 50, 57, 64, 68	*RED*(1), *DGE*, *FOC*(0,{1,1})	72, 76, 80, 89, 98, 107, 116, 125
A194385	6, 12, 18, 24, 30, 36, 228, 234, 240, 246, 252, 258, 264, 456, 462, 468, 474, 480, 486, 492, 684	*RED*(6), *DGD*	690, 696, 702, 708, 714, 720
A227589	1, 4, 7, 12, 16, 23, 29, 38, 46, 57, 67, 80, 92, 107, 121	*ML*(1,1), *ML*(0,1), *FOC*(0,{1,1})	138, 154, 173, 191, 212, 232

sequences have required up to a fourth level of depth, this being the maximum allowed in the execution of the model. In addition, the most used combinations of *kitas* are shown in Table 9. The most used are *FOC* (56.9%), *ML* (32.7%), *RED* (28.7%) and *ANA* (9 .8%), plus *BAS*. This result contrasts with those obtained for the two groups of the series of IQ tests, due to the greater complexity of the series in OEIS.

Finally, Table 10 shows several examples showing the reference in OEIS, the terms used to find the underlying pattern, the pattern itself as a list of *kitas*, and the prediction. These examples, like many of the series within OEIS, are non-trivial and serve to demonstrate the model's ability to find the underlying pattern and predict the next terms in sequences with this degree of difficulty. Consequently, KitBit is a computational system that is capable of solving the largest number of sequences contained in OEIS to date, well above the maximum value reported in the literature.

5 Conclusions

We have presented the fundamentals of KitBit, which is a new computational model capable of obtaining the underlying pattern in series of numbers, as well as its efficiency in solving IQ tests and a series of greater complexity, such as those found in the OEIS database. KitBit uses a set of algorithms or *kitas* that perform different operations on the numerical series, most of them non-analytical, to find the underlying pattern without the need for training or prior knowledge, in very short times, always less than a second, using standard computing power. This pattern is represented by a sequence of *kitas* instead of a mathematical function, which allows us to reproduce the known elements of the numerical sequence and extrapolate to new ones.

KitBit has managed to solve 97.8% of a first set of 90 series associated with IQ tests collected from different sources. In addition, it solves 91% of a second set of 67 series compiled from previous publications aimed at solving numerical series and intelligence tests, surpassing the results of these methods. Finally, KitBit deals with the resolution of 341,553 entire series included in OEIS database and is the only model to date that considers the resolution of its entirety. KitBit is capable of solving 87,514 sequences, or 25.6% of the total. This is the largest number of solved series to date. Within this result we find two types of solutions: a first group in which the sequence is specified from a certain term, using the pattern found in all the previous terms up to the first; and a second group, in which the sequence is also specified from a certain term but the pattern found does not include all the previous terms up to the first.

These results are promising to face other types of problems using the same methodology, such as the resolution of other types of IQ test or more specific complex problems through the development of new *kitas*, which will be presented in future publications.

REFERENCES

[1] H. Liu, J. Yin, X. Luo, and S. Zhang, "Foreword to the special issue on recent advances on pattern recognition and artificial intelligence," *Neural Computing and Applications*, vol. 29, no. 1, pp. 1–2, Jan. 2018.

[2] J. C. Bezdek, "On the relationship between neural networks, pattern recognition and intelligence," *International Journal of Approximate Reasoning*, vol. 6, pp. 85–107, 1992.

[3] 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.

[4] 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.

[5] 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.

[6] H. Nam, S. Kim, and K. Jung, "Number sequence prediction problems for evaluating computational powers of neural networks," in *AAAI Conference on Artificial Intelligence*. AAAI Press, Jan. 2019, pp. 4626–4633.

[7] S. d'Ascoli, P.-A. Kamienny, G. Lample, and F. Charton, "Deep Symbolic Regression for Recurrent Sequences," *arXiv 2201.04600*, 2022.

[8] N. Sloane, "The on-line encyclopedia of integer sequences, 130," *Towards Mechanized Mathematical Assistants. Springer, New York*, vol. 33485, 2007.

[9] N. J. A. Sloane, "The on-line encyclopedia of integer sequences," in *Annales Mathematicae et Informaticae*, vol. 41, 2013, pp. 219–234.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] T. G. Evans, "A heuristic program to solve geometric-analogy problems," in *Proceedings of AFIPS '64 (Spring)*, vol. 25, 1965, p. 327–339.

[15] K. Simon, Herbert A. Kotovsky, "Human acquisition of concepts for sequential patterns," *Psychological Review*, vol. 70(6), p. 534–546, 1963.

[16] 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.

[17] N. Dean and G. E. Shannon, *Computational Support for Discrete Mathematics*. American Mathematical Soc., 1994, vol. 15.

[18] 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.

[19] M. Siebers and U. Schmid, "Semi-analytic natural number series induction," in *Annual Conference on Artificial Intelligence*. Springer, Sep. 2012, pp. 249–252.

[20] S. Colton, A. Bundy, and T. Walsh, "Automatic invention of integer sequences," in *AAAI/IAAI*, 2000, pp. 558–563.

[21] M. J. Meredith, "Seek-Whence: A Model of Pattern Perception," Ph.D. dissertation, Indiana University, 1986.

[22] M. M. Bongard, *Pattern Recognition*. Rochelle Park, N.J.: Hayden Book Co., Spartan Books, 1970.

[23] D. Hofstadter, *Fluid Concepts and Creative Analogies*. Basic Books, 1995.

[24] A. A. Mahabal, "Seqsee: A Concept-centered Architecture for Sequence Perception," Ph.D. dissertation, Indiana University, 2009.

[25] J. Burghardt, "E-generalization using grammars," *Artificial Intelligence*, vol. 165, no. 1, pp. 1–35, Mar. 2005.

[26] C. Strannegård, M. Amirghasemi, and S. Ulfsbäcker, "An anthropomorphic method for number sequence problems," *Cognitive Systems Research*, vol. 22, pp. 27–34, Jun. 2013.

[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] M. Ryskina and K. Knight, "Learning Mathematical Properties of Integers," *arXiv 2109.07230*, 2021.

[29] C. W. Wu, "Can machine learning identify interesting mathematics? An exploration using empirically observed laws," *arXiv 1805.07431*, 2018.

[30] A. Bhansali and S. S. Skiena, "Analyzing Integer Sequences," vol. 15, 1992, pp. 1–16.

[31] 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.

[32] 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.

[33] F. Faber, R. Kidder, and T. Lang, "Making the machine intelligent Algebraic methods to make a computer programme pass an IQ test," Mar. 2003.

[34] C. Strannegård, M. Amirghasemi, and S. Ulfsbäcker, "An anthropomorphic method for number sequence problems," *Cognitive Systems Research*, vol. 22, pp. 27–34, Jun. 2013.

[35] 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.

[36] 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.

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 this 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 on applied artificial intelligence in different fields, such pattern recognition, machine learning, neurobiology and genetics in partnership with several universities and research centers.

Luis Herrera has a BS in Aerospace Engineering from Universidad Politécnica de Madrid (UPM). He holds a Certificate in Exponential Technologies from Singularity University. He is CEO of Smart Technologies Investments, which is the company that is developing the KitBit project.