Bongard Architecture: Towards Scalable and Transparent Machine Reasoning

Dmitry Grinberg, Tetiana Grinberg, Grigory Shcherbanskiy, Lev Cherbanski

Abstract

Bongard problems, a collection of 100 image puzzles first introduced by Mikhail Bongard (Bongard 1968), have posed a long-standing challenge for traditional machine learning architectures. Despite their simplicity, these puzzles demand explainability, reasoning and pattern recognition skills. Our work revives the approach to solving these problems proposed by Bongard's original research group, which has not been re-implemented or validated since. We demonstrate that this approach is able to successfully solve 100% of the problems in the Maksimov-Bongard problems (MBP) dataset (Maksimov 1975) which is explicitly constrained to contain only geometric puzzles. We extend the vocabulary of operators to demonstrate scalability of the architecture, yielding solutions to a considerable portion (44%) of the more widely known Bongard problems (BP) dataset (Bongard 1968). Finally, we argue for the value of Bongard-style architectures in AI applications that demand complex human-machine communication, transparency and compatibility with human cognitive processes.

Introduction

A Bongard problem is defined as two small collections of images belonging to different classes (let's call them class A and class B). The image collections typically have 4-6 image examples per class. Solution to a Bongard problem consists of a human-understandable classification rule that allows one to distinguish between classes A and B. An example of a Bongard problem can be seen in Figure 1.

Many are familiar with Bongard problems (BPs), the object of which is to find a characteristic shared by all shapes in one of two sets of geometric shapes, which is not shared by any of the shapes in the other set. However, few know that these problems were meant as a test dataset for an architecture that Bongard's team had outlined and partially implemented. Even less known is the fact that this architecture was only one of several that were implemented by Bongard's lab. These implementations were successfully used in applications such as X-ray analysis, earthquake prediction and oil discovery as far back as the 60s and 70s. Some of these architectures are still actively used to this day (Gvishiani, Soloviev, and Dzeboev 2020).

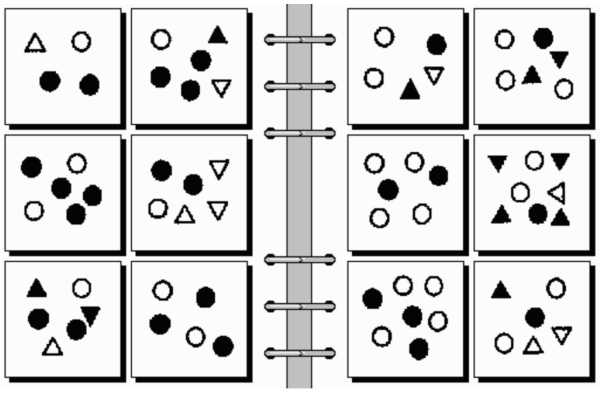

Figure 1: Bongard Problem no 28. Solution: Class A (left) has more black than white circles in each image. Class B (right) has more white than black circles in each image. Image source: (Foundalis 2022)

These facts are highly surprising, given that they should have placed Bongard's name in the same league as Marvin Minsky, Frank Rosenblatt, Norbert Wiener and the other great cyberneticists. Instead, he is known as the author of a peculiar and difficult-to-solve dataset, but not for his research philosophy and theoretical frameworks. This is in part due to Bongard's untimely demise (Bongard was a USSR climbing champion and died in a climbing accident in the Pamir mountains in the summer of 1971). Bongard's death happened shortly after the infamous Mansfield Amendment, and at the time when there was active tension between the Soviet Union and the Western world, making it hard for scientific developments to travel between the two domains. Bongard simply did not live long enough to be able to disseminate his ideas to the scientific community abroad.

The present paper aims to correct some of this historical injustice and reintroduce Bongard's AI philosophy into the modern scientific dialog. Since today he is mostly known for his visual reasoning dataset (aka Bongard Problems), we choose the architecture his team developed to solve this dataset as the natural basis of our exposition, since it allows us to compare Bongard's philosophical framework with its contemporaries that yielded the modern architectures. We hope that this is the first in a series of papers that will ex-

tend and evolve the Bongard architecture to make full use of modern hardware and scaling methods.

Related Work

Interest in Bongard problems was revived by Douglas Hofstadter (Hofstader 1979) and Harry Foundalis. The latter designed Phaeaco, the first publicly available implementation of the Bongard problem solver (Foundalis 2006). Today, Bongard problems are becoming more relevant than ever, as it is becoming more and more evident that reasoning tasks pose a considerable challenge even to the most advanced image models (Thrush et al. 2022), and a renewed effort is made to design datasets and challenges intended specifically for testing reasoning capabilities of AI systems (Chollet 2019). Many of the newly introduced datasets either explicitly extend the original dataset (Yun, Bohn, and Ling 2020), or use it as inspiration in designing similar tasks for new domains (Weitnauer and Ritter 2012; Nie et al. 2020; Jiang et al. 2022). In the context of these developments, there have also been several recent attempts at solving the Bongard dataset with modern AI systems. It is worth noting that some of these attempts change the original problem definition, which demands explainability, and reformulate it as a simple classification task. An example of such an approach is the work of Kharagorgiev (Kharagorgiev 2020) which utilizes deep neural networks coupled with data augmentation to compensate for the small size of the training set, achieving an 18% success rate on the classification task. Other notable approaches have used program synthesis and inductive logic programming (Sonwane et al. 2021), reinforcement learning (Youssef et al. 2022) and Bayesian inference (Depeweg, Rothkopf, and Jäkel 2018).

Bongard's Approach to Pattern Recognition

In this section, we will describe Bongard's philosophical framework. Before we do this, however, we would like to introduce two considerations to place further discussion into a proper context.

The first consideration is that of purpose. The purpose of this exposition is similar to the goal of a computer scientist learning a new programming language, or a professional philosopher studying ancient Chinese — that is, to acquire the mental vocabulary and therefore an ability to pursue a direction of exploration which is not available from within one's familiar conceptual environment.

Without this type of exposition, the core concepts of a philosophical framework are often misunderstood or even simply forgotten for lack of translatability. This has been the case with the likes of Rene Descartes and Hermann Grassmann, who became mostly remembered for the "appendices" to their works (that is, addendums or extensions providing simplified applications of the theory proposed in the main volume), but whose fundamental insights were lost for a long time on the general scientific community, despite their undeniable value.

The second consideration is that of correct contrast. In order to understand Bongard's philosophy it is important to see it in contrast with the philosophy of neural networks and probabilistic classification models of his time. While the latter two enjoyed a long period of well-funded development, Bongard's architecture has lain mostly dormant with development only continued by a few of his friends and colleagues (Maksimov 1975; Weinzweig 2008); it is unknown where it could lead us after a similar amount of development. Due to space constraints, we will not introduce a detailed description of the Western philosophy of AI in the 1960s, however we expect that most readers are familiar with its major tenets.

Next we introduce some important definitions. Careful reading will reveal that some of the terms below differ from their standard textbook definitions. This is done to facilitate subsequent discussion of the Bongard architecture, which was developed using this conceptual vocabulary.

One of the key observations that Bongard makes is that problem solving requires finding a *degenerate transformation* of the input object, which removes all of the unnecessary information from the input object, reducing it to a single property value. For example, looking at an image of a green banana, a degenerate transformation corresponding to the color feature would convert the image to a single value corresponding to the color "green".

A feature is an algorithm that carries out a degenerate transformation of an input object. The value returned by the feature for a particular input object is this object's characteristic according to the feature.

Thus, the aim of a pattern recognition task is the following: given some examples of objects belonging to the same class, find the set of *features* (degenerate transformation algorithms) that would allow one to construct a sufficiently precise yet compact description (set of characteristics) of the class that would allow one to distinguish it from other classes.

More formally, given two sets of input objects $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, and a set of all available features $F = \{f_1, \ldots, f_m\}$, a classification rule distinguishing between A and B would consist of such a subset $F_c \subseteq F$ that would satisfy the following conditions:

- $f_i(a_l) = f_i(a_k)\ \forall a_l, a_k \in A, f_i \in F_c$
- $f_i(a_l) \neq f_i(b_k)\ \forall a_l \in A, b_k \in B, f_i \in F_c$

This is to say that all classifying features should assign the same value to objects within the same class, and different values to objects from different classes. In the simplest case, the set F_c could contain only a single feature f, albeit this feature could perform a very complex transformation of the input objects.

Now, obviously, constructing an exhaustive set of features F upfront can become an intractable task even for relatively simple input objects. Bongard proposes an elegant solution to this problem by introducing an intermediate set of functions (which he calls "operators") $V = \{v_1, \ldots, v_h\}$ that forms a kind of "factorization system" of the feature set F, such that $\forall f_i \in F, f_i = v_j \circ \ldots \circ v_{j+p}$ for some subset $\{v_j, \ldots, v_{j+p}\} \subseteq V$.

Thus, in Bongard's framework, a classification problem boils down to constructing a classifying feature as a composition of some subset of functions in V. The set V thus

forms a "vocabulary" through which classification rules can be expressed.

On the face of it, this might look very similar to a standard neural network, since the latter also learns a composition of functions that yields some useful output. However, this is the point at which Bongard's philosophy most obviously diverges from that produced by Rosenblatt's school of thought, which was content with using any selection of operators, however inscrutable, which could be reasonably composed into a classifying rule. On the other hand, Bongard's aim was to emulate the human method of solving classification problems, which necessitated a selection of a vocabulary V that is reasonable to a human. This objective therefore necessitated a different approach to finding the classification rule, leading to a generative and search-like process rather than arbitrary activation functions and weights with the familiar backpropagation and gradient descent.

Maksimov-Bongard dataset

The first implementation of Bongard's visual reasoning algorithm (dubbed "Geometry" in his book) was completed after his death by his colleague Vadim Maksimov (Maksimov 1975). Rather than using the full dataset presented in "Pattern Recognition" (Bongard 1968), the implementation used a simplified dataset, for which the basic operators could be implemented within the constraints of computer hardware the group had to work with. We will refer to this simplified dataset as Maksimov-Bongard Problems (MBPs).

The Bongard Algorithm

Vocabulary of Basic Operators for MBPs

We implement the logical architecture and all of the operators described in Maksimov's paper (Maksimov 1975). Since Maksimov and team did not provide source code for their implementation, many doubted that the described algorithm did in fact work and exhibit the set of properties outlined in Bongard's book. This motivated us to replicate the algorithm exactly as described in the paper, down to the lower-level pixel manipulations.

Inputs and Outputs

The program can handle three types of inputs — ordered collections of images (denoted by the letter P), boolean values (B), and numbers (N). Initial input into the algorithm consists of a single collection P of images p and a collection B of boolean values b (each boolean value records the class assignment of the corresponding input image). Images are represented as binary arrays. The program attempts to find a sequence of basic operators from its vocabulary that would yield either B_{input} or $\neg B_{input}$ as its final output.

Sequence of Execution

In Maksimov's implementation the sequence of execution of the basic operators was rigid due to the hardware limitations of that time. However, since each operator acted as a transformation of the input data, the exact shape of the resulting search tree could vary widely between different input collections.

Vocabulary of basic operators for MBPs

Operator name	Inputs	Outputs
Group A - drawing (D) and measuring (M)		
Area	P_m	N_m
Line length	P_m	N_m
Center of mass coordinates	P_m	$2N_m$
Angle	P_m	N_m
Major axis length	P_m	N_m
Minor axis length	P_m	N_m
Contour (C)	P_m	P_m, $2B_m$
Fill (F)	P_m	P_m, $2B_m$
Convex hull (T)	P_m	P_m, $2B_m$
Group B - boolean inputs		
Solving operator	B_1	-
Logical operator (L)	B_m	$4B_1$,$4B_2$,...,$4B_{m-1}$
Clustering operator (H)	N_m, B_m	kB_m
Number of parts (Q)	B_m	$N_1, N_2, \ldots, N_{m-1}$
Comparison operator (R)	N_l, N_m, B_m	kB_m
Union operator (U)	P_m, B_m	$P_1, P_2, \ldots, P_{m-1}$
Group C - separation		
Separation by connectivity (S)	P_m	P_{m+1}, B_{m+1}
Separation by borders (J)	P_m	P_{m+1}, B_{m+1}
Separation by branching points (K)	P_m	P_{m+1}, B_{m+1}

Figure 2: Vocabulary of basic operators for MBPs. P_m — image collection on level m. N_m — number collection on level m. B_m — boolean collection on level m.

The operators are divided into three groups based on the order of their execution by the algorithm (see Figure 2). Group A is applied first (until it exhausts all possible inputs), followed by Group B and Group C. When separation operators (group C) are applied, they produce a collection of images of a *potentially different cardinality*, thereby creating a new "level" in the search tree. The "level" of a given collection of objects is denoted by the subscript index. An example of the level creation process can be seen in Figure 3.

For detailed descriptions of each basic operator, see Section 3 in (Maksimov 1975). We first implemented all operators exactly as described in Maksimov's paper. Having ascertained that the original descriptions were indeed valid, we re-implemented the operators using modern libraries such as OpenCV, Scikit-image, and some others for speed and readability (Bradski 2000; Van der Walt et al. 2014; Miniak-Górecka, Podlaski, and Gwizdałła 2022; Virtanen et al. 2020; Viry 2021).

Figure 4 shows the architectural diagram of the algorithm.

The algorithm runs until one of the collections of booleans on the top level matches either B_{input} or $\neg B_{input}$. This check is performed by the solving operator as a simple if statement. Once a matching boolean collection has been found, the program returns the sequence of operators that has led to its discovery. This sequence constitutes the clas-

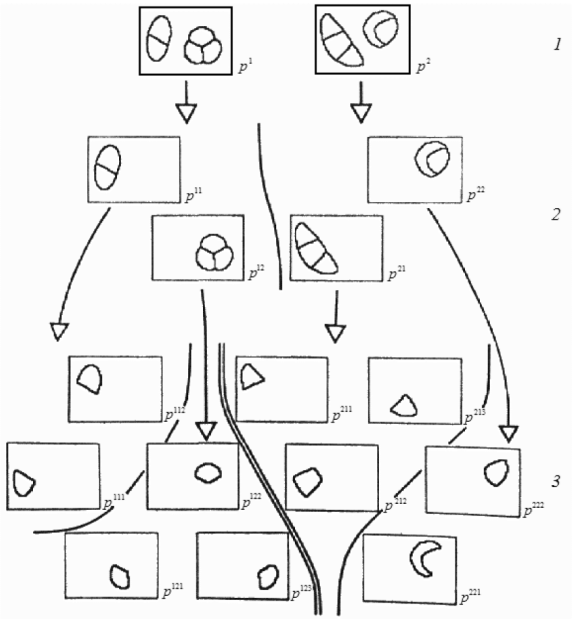

Figure 3: An example of transformations undergone by input image upon application of operator S (separation by connectivity) followed by J (separation by borders). Note how each transformation creates a new "level" (numbered from 1 to 3). Image source: (Maksimov 1975)

sification rule. It is worth noting that all of the basic operators used in the vocabulary are indeed "basic", in the sense that they use off-the-shelf functions from popular image processing and machine learning libraries — just as one might use an off-the-shelf activation function in a neural network. Most of the engineering effort required to implement the algorithm went into aligning the output formats of the various operators, and creating a data structure that would be able to track which inputs have or have not been processed by a given operator.

Experimental Results on MBPs

When applied to MBPs, the algorithm is able to successfully solve all of them. We ran three types of experiments:

1. An exhaustive search with human-defined sequence and vocabulary of operators (used to ascertain solvability and best-case run-time for each problem);
2. An exhaustive brute-force search with partially constrained operator vocabulary (the vocabulary was constrained to only include those operators which would be considered "reasonable" by a human);
3. A greedy search with the full vocabulary of operators (run to the first discovered solution).

Figure 5 contains results of the first two types of experiments. Experiment type 3 was only performed on a subset of problems, since it generated extremely large search trees.

As we can see, both experiment type 1 and experiment type 2 led to a successful discovery of classification rules.

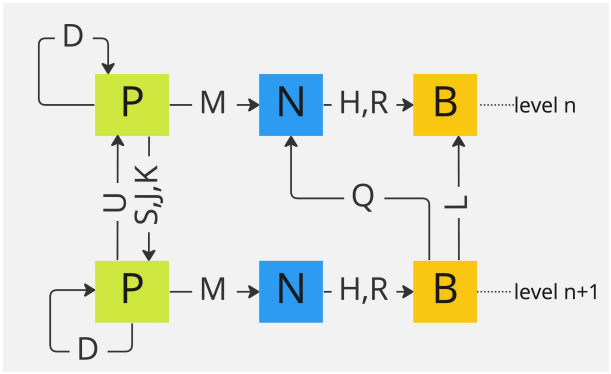

Figure 4: Architectural flow of the algorithm. For simplicity, D denotes drawing operators, M — measuring operators.

Figure 6 contains an example solution graph (rendered using the Graphviz library). A classification rule can be expressed as the sequence of operators whose outputs were required to arrive at the solution boolean (B_{input} or $\neg B_{input}$).

Experimental Results on BPs

Next, we performed the same experiments on Bongard's full dataset. In order to do this, Maksimov's original "vocabulary" was enhanced with three additional basic operators. This provided it with sufficient expressivity to find classification rules for **44% of BPs**. The additional operators include:

1. **Contour hierarchy.** This operator was added as an additional output to separation by connectivity. It returned the hierarchy level of the external contour of each connected element in the image.
2. **Polygon detection.** This operator evaluated the number of vertices present in a shape.
3. **Endpoint detection.** This operator detected and counted endpoints in a contoured image.

These additional operators were once again implemented using functions from standard image processing libraries.

The three additional operators do not attempt to cover the entire vocabulary proposed by Bongard for the solving of his dataset; his vocabulary included operators such as detectors of identical objects and axes of symmetry. Rather, the three operators were used to demonstrate generalizability of the architecture.

Discussion

We have demonstrated that Bongard's architecture is indeed viable for classification problems that require reasoning. However, in the current AI industry environment of fluent language models that can outperform most humans on many academic and professional exams, an inherently non-verbal architecture of this type may appear uninteresting.

The raw reasoning capacity of the Bongard architecture is not its most valuable property. Rather, we believe that the

MBP #	Human solution operator sequence	Total # of unique solution booleans	Runime in seconds	Limited operator vocabulary used for brute-force search	Total # of unique solution booleans	Runtime in seconds
1	[C]	2	0.01	[C]	2	0.01
2	[F, C]	2	0.03	(C, F)()()	4	0.04
3	[S, M, H, L]	1	0.16	(M,)(L, H)(S,)	1	0.2
4	[F, S, C, L]	2	0.04	(C, F)(L,)(S,)	4	0.24
5	[M, H]	2	0.02	(M,)(H,)()	2	0.02
6	[S, M, H, L]	2	0.06	(M,)(L, H)(S,)	2	0.05
7	[S, M, H, L]	2	0.04	(M,)(L, H)(S,)	2	0.04
8	[S, F, U, M, H]	2	0.22	(M, F)(L, H, U)(S,)	2	0.35
9	[S, F, U, M, H]	1	0.48	(M, F)(L, H, U)(S,)	2	13.22
10	[T, M, H]	46	0.07	(M, T)(H,)()	46	0.13
11	[S, M, H, L]	6	0.04	(M,)(L, H)(S,)	26	0.2
12	[S, Q, H]	1	0.01	()(H, Q)(S,)	1	0.02
13	[S, J, Q, H]	2	0.02	()(H, Q)(S, J)	2	0.06
14	[S, J, F, Q, H]	2	0.18	(F,)(H, Q)(S, J)	2	1
15	[S, J, F, Q, H, L]	1	0.2	(F,)(L, H, Q)(S, J)	1	1.1
16	[S, C, S, Q, H, L]	2	0.02	(C,)(L, H, Q)(S,)	2	0.04
17	[S, M, H, U, F, T]	2	1.07	(M, F, T)(H, U)(S,)	4	9.75
18	[S, M, H, U, M, H]	7	0.27	(M,)(H, U)(S,)	7	0.39
19	[F, S, M, H, L]	1	0.12	(M, F)(L, H)(S,)	1	0.64
20	[F, S, M, H, L]	2	0.2	(M, F)(L, H)(S,)	2	0.63
21	[F, S, M, H, L]	2	0.38	(M, F)(L, H)(S,)	2	1.1
22	[C, M, H]	2	0.04	(M, C)(H,)()	2	0.01
23	[C, M, H]	2	0.04	(M, C)(H,)()	2	0.05
24	[S, M, H, U, T, C, M, H]	1	1.44	(M, C, T)(H, U)(S,)	14	48.75
25	[C, M, H, H, H]	1	0.07	(M, C)(H,)()	1	0.03
26	[F, M, H]	2	0.02	(M, F)(H,)()	2	0.04
27	[S, M, H, L]	2	0.01	(M,)(L, H)(S,)	2	0.05
28	[S, C, U, M, H]	2	0.17	(M, C)(H, U)(S,)	6	14.84
29	[F, S, C, U, M, H, L]	4	0.34	(M, C, F)(L, H, U)(S,)	24	36.11
30	[C, S, F, M, H, L]	2	0.16	(M, C, F)(L, H)(S,)	2	0.58
31	[S, C, S, M, H, H, U, T, M, H, L]	5	3.22	(M, C, T)(L, H, U)(S,)	54	1175.39
32	[S, M, H, U, M, H, L]	7	3.51	(M,)(L, H, U)(S,)	7	34.22
33	[S, M, H, U, M, H]	2	0.12	(M,)(H, U)(S,)	2	0.43
34	[F, M, T, M, R]	6	0.06	(M, F, T)(R,)()	6	0.1
35	[M, T, M, R]	3	0.02	(M, T)(R,)()	3	0.03
36	[S, C, M, H, U, T, M, R]	8	0.7	(M, C, T)(H, R, U)(S,)	262	233.1
37	[S, M, H, U, T, M, R]	12	0.8	(M, T)(H, R, U)(S,)	17	39.74
38	[S, M, R, L]	2	0.03	(M,)(L, R)(S,)	2	0.04
39	[S, F, C, M, R, L]	6	0.26	(M, C, F)(L, R)(S,)	4	0.37
40	[C]	2	0.01	(C,)()()	2	0.01
41	[S, F, T, M, R, L]	2	0.07	(M, F, T)(L, R)(S,)	4	0.4
42	[S, C, U, T, C, M, R]	2	0.49	[S, C, U, T, C, M, R]	2	0.44
43	[S, K, C, M, H, H, L]	2	0.79	(M, C)(L, H)(S, K)	2	2.1
44	[S, K, C, M, H, U, C, M, H, L]	4	6.41	(M, C)(L, H, U)(S, K)	18	53.92
45	[K, T, M, R, U, F]	2	0.44	(M, F, T)(R, U)(K,)	5	2.03
46	[S, T, M, R, U, T, M, H]	2	0.2	(M, T)(H, R, U)(S,)	10	13.09
47	[C, S, F, M, H, U, M, H]	4	0.28	(M, C, F)(H, U)(S,)	14	146.11
48	[C, S, F, M, H, U, M, H]	4	0.35	(M, C, F)(H, U)(S,)	8	317.99

Figure 5: Experimental results for MBPs. The operators are described in Figure 2.

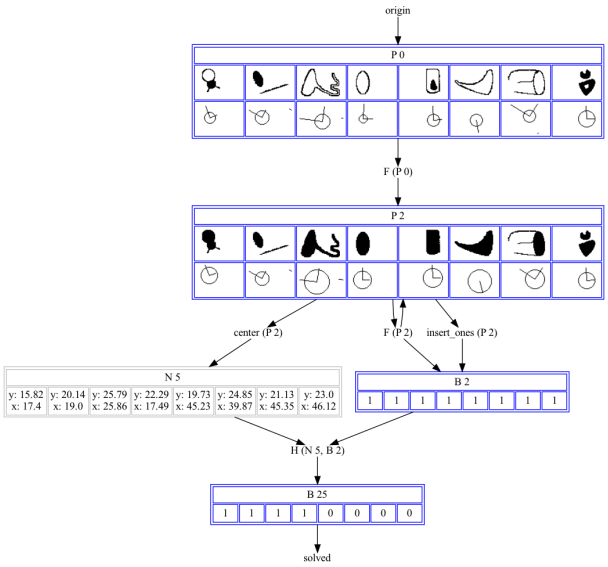

Figure 6: Example solution graph for an MBP. The algorithm applies the Fill drawing operator, then measures centers of mass of resulting figures and discovers that those can be grouped into two clusters, which yields the solution.

true value of Bongard's algorithm lies in its intrinsic interpretability. As can be seen in Figure 6, a simple visualization is quite sufficient to surmise how the algorithm has arrived at the classifying rule. Such a visualization can also be used to convey a classification rule that the human user did not devise themselves — that is, an AI agent based on the Bongard architecture can successfully teach a human. This is possible because, despite using a similar set of basic operators, the program is able to execute a much wider search, which is beyond reach of a human's limited working memory. Of course, with the increase in vocabulary size, or when one cannot choose a constrained vocabulary to perform the search, one could end up with a much more complex graph (e.g. see Figure 7), which evokes renderings of a neural network-generated decision tree. Still, this is where being built with human-compatible basic operators makes the Bongard architecture stand apart — since the individual operators have intuitive meaning to the human user, they can also be validated and then chunked to form more complex conceptual structures (or "complex operators", as Maksimov and Bongard called them).

Such explicit modeling of reasoning and concept formation can become a highly fruitful tool for verifying theories about both human-to-human and human-to-machine communication. After all, humans are black-box systems with respect to each other — a gap that is bridged to some extent by shared language and culture. This gap becomes increasingly more daunting when the knowledge that needs to be transferred is highly sophisticated, and with the advent of tools such as Chat-GPT, that allow humans to easily fake understanding (Cotton, Cotton, and Shipway 2023; OpenAI 2023). Similarly, if we are to ever build reliable communication protocols between humans and AI systems, such protocols would have to be able to handle the transfer of content that is highly nuanced and complex — be it scientific knowledge or ethical and cultural values. Human civilization now stands in dire need of establishing avenues for such communication that do not rely solely on normative or policy-based interventions — after all, if a language model can pass the Bar Exam better than 90% of human students (Chalkidis 2023), how hard would it be to make it find an exploit loopholes in policies that rely heavily on the external presentation of model outputs (as opposed to a lucid representation of its internal decision-making process)? We need tools that would allow us to provably ascertain successful transmission of complex knowledge in both human-to-human and human-to-machine communication. Similar to how zero-trust proofs are necessary in cryptography, a kind of zero-trust "proof-of-understanding" protocol is needed to counteract the onslaught of mindless but very convincing "stochastic parrots".

We believe that the key to provability lies in limiting the extent to which black-box model architectures are used to replicate conscious human thought. AI models must provide transparency in all the parts of their operation that correspond to cognitive functions to which humans have introspective access. Moreover, the model should be able to transfer its expertise to a human — either through direct expla-

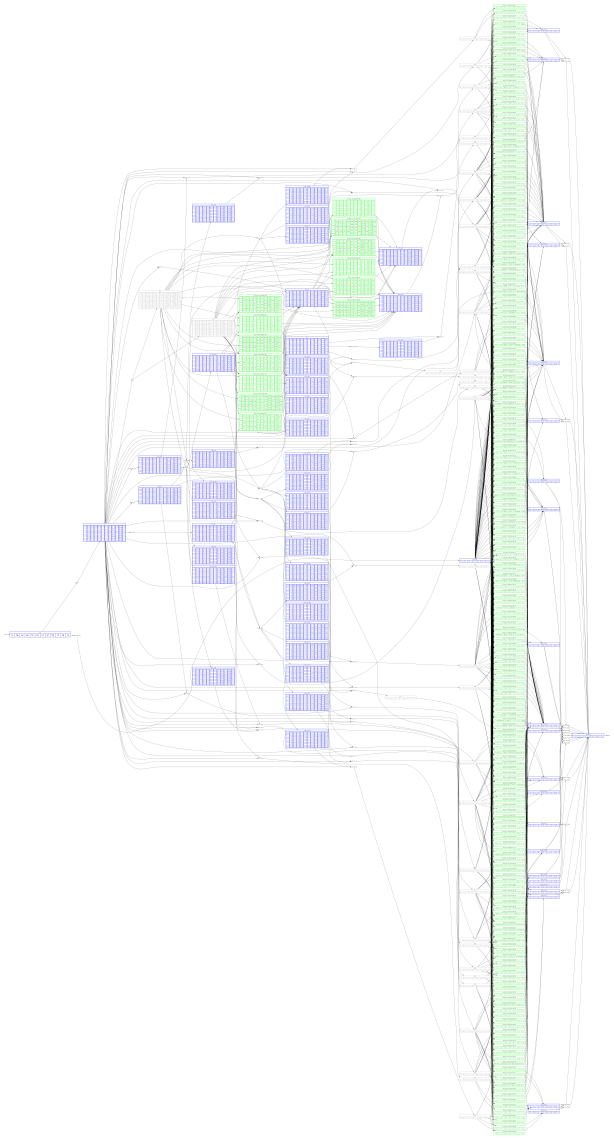

Figure 7: Solution graph for BP 28.

nation or through explanation coupled with validated tool use (by "validated" we mean that the human must demonstrate proper understanding of the tool). Human-to-machine and machine-to-human interactions are already emerging in many domains that have been disrupted by AI technologies — consider, for instance, Go players who successfully win against AI-based bots by using an instance of the same AI system they're playing against to augment their decision making. What is currently missing is true "mind sharing" between AI and its human users, which is impossible without a common conceptual and cognitive framework.

Limitations and Future Work

While the current implementation of the algorithm has successfully proven the basic feasibility and scalability of Bongard's approach, it does suffer from some serious limitations:

- The algorithm cannot dynamically select a "good" set of operators that should be applied to a given problem at a given stage of the search. This becomes especially problematic when additional basic operators are introduced into the vocabulary.
- The current vocabulary of basic operators is only applicable to simple binary images.

To extend the algorithm to a wider set of use cases, one would need to augment it with a suitable heuristic function, as well as replace the current basic operators with something that can be applied to more complex images. Pre-trained neural network-based models would be a natural candidate for this role.

Finally, neither Bongard's nor Maksimov's datasets provided validation images for all of the problems (Maksimov's dataset did provide some test images, which were correctly classified by the trained model). Further work would benefit from expansions of the Bongard dataset to include validation data.

References

Bongard, M. M. 1968. The recognition problem. Technical report, FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OHIO.

Bradski, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chalkidis, I. 2023. ChatGPT May Pass the Bar Exam Soon, but Has a Long Way to Go for the LexGLUE Benchmark. *Available at SSRN 4385460*.

Chollet, F. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

Cotton, D. R.; Cotton, P. A.; and Shipway, J. R. 2023. Chatting and Cheating: Ensuring academic integrity in the era of ChatGPT. *Innovations in Education and Teaching International*, 1–12.

Depeweg, S.; Rothkopf, C. A.; and Jäkel, F. 2018. Solving bongard problems with a visual language and pragmatic reasoning. *arXiv preprint arXiv:1804.04452*.

Foundalis, H. 2022. Bongard Problem 28.

Foundalis, H. E. 2006. Phaeaco: A cognitive architecture inspired by Bongard's problems.

Gvishiani; Soloviev; and Dzeboev. 2020. Problem of recognition of strong-earthquake-prone areas: a state-of-the-art review. *Izvestiya, Physics of the Solid Earth*, 56: 1–23.

Hofstader, D. 1979. Gödel, Escher, Bach: an eternal golden braid.

Jiang, H.; Ma, X.; Nie, W.; Yu, Z.; Zhu, Y.; and Anandkumar, A. 2022. Bongard-hoi: Benchmarking few-shot visual reasoning for human-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19056–19065.

Kharagorgiev, S. 2020. Solving bongard problems with deep learning. *k10v. github. io*.

Maksimov, V. 1975. A system capable of learning to classify geometric images. *Modeling of Learning and Behavior, Nauka, Moskva*.

Miniak-Górecka, A.; Podlaski, K.; and Gwizdałła, T. 2022. Using k-means clustering in python with periodic boundary conditions. *Symmetry*, 14(6): 1237.

Nie, W.; Yu, Z.; Mao, L.; Patel, A. B.; Zhu, Y.; and Anandkumar, A. 2020. Bongard-logo: A new benchmark for human-level concept learning and reasoning. *Advances in Neural Information Processing Systems*, 33: 16468–16480.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Sonwane, A.; Chitlangia, S.; Dash, T.; Vig, L.; Shroff, G.; and Srinivasan, A. 2021. Using Program Synthesis and Inductive Logic Programming to solve Bongard Problems. *arXiv preprint arXiv:2110.09947*.

Thrush, T.; Jiang, R.; Bartolo, M.; Singh, A.; Williams, A.; Kiela, D.; and Ross, C. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5238–5248.

Van der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J. D.; Yager, N.; Gouillart, E.; and Yu, T. 2014. scikit-image: image processing in Python. *PeerJ*, 2: e453.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Viry, M. 2021. jenkspy: Jenks Natural Breaks optimization in Python. Accessed: 2023-03-14.

Weinzweig. 2008. On the works of M.M. Bongard and their continuation. Accessed: 2023-03-15.

Weitnauer, E.; and Ritter, H. 2012. Physical bongard problems. In *Artificial Intelligence Applications and Innovations: 8th IFIP WG 12.5 International Conference, AIAI 2012, Halkidiki, Greece, September 27-30, 2012, Proceedings, Part I 8*, 157–163. Springer.

Youssef, S.; Zečević, M.; Dhami, D. S.; and Kersting, K. 2022. Towards a Solution to Bongard Problems: A Causal Approach. *arXiv preprint arXiv:2206.07196*.

Yun, X.; Bohn, T.; and Ling, C. 2020. A deeper look at bongard problems. In *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33*, 528–539. Springer.