Cyber-Genetic Neo-Plasticism
– An AI program Creating Mondrian-like Paintings
by using Interactive Bacterial Evolution Algorithm¹
Jian Yin Shen, Tom Gedeon

Abstract – This work investigates using an AI program to generate pictures simulating the style of Neo-Plastic artworks from Piet Mondrian. In this report we discuss how to generate fine Mondrian-like graphs by a process controlled by interactive Bacterial Evolution Algorithm.

Keyword – Genetic Algorithm, Bacterial Evolution Algorithm, Neo-Plasticism, Piet Mondrian

¹ Part of this work is also discussed in:
Tom Gedeon, Jian Yin Shen, Making art using evolutionary algorithms and artificial AI,
Proceedings of BOOM07, Taiwan – Australia New Media Workgroup, 7 pages
on another perspective of view
I. Introduction

Background And Brief Introduction of This Work

In this work we investigate the way an AI program can generate Neo-Plastic pictures by adopting an interactive bacterial evolution algorithm.

Within the domain of Artificial Intelligence, “creativity” problems (that require the computer to design something) are interesting and challenging topics because “creativity” is truly an essential part of “intelligence” and is believed to be owned only by human. However, there’s no Strong AI that truly owns “creativity”\(^2\) has been implemented yet, since even humans themselves have not realized the source or principle of their own creativity. Although having an AI which is as creative as humans to create artworks is not practical now, some weaker\(^3\) programs using “conventional” methods does output meaningful results. In short, these programs do not understand their creation, instead they just simply iteratively refine their works by a given mathematical model or evaluations from human, until the result is good enough. In these programs, evolutionary algorithms are often involved.

The evolutionary algorithm is a class of non-linear approximative algorithms that simulate the natural procedure of evolution\[^{14}\]. The core concept of evolution is natural selection that iteratively eliminates bad individuals and preserves good individuals. As an outcome all individuals within the system tend to be better after some generations. Adopting this concept into cyberspace we can solve a class of problems without the necessity of understanding the problem (for instance: what’s good art?).

The style we try to simulate is called “Neo-Plasticism”, which was created by Dutch artist Piet Mondrian. Illustrations 1-5 show some of Piet Mondrian’s famous composition series. In general, Neo-Plasticism proposed ultimate simplicity and abstraction, by using only straight (horizontal and vertical) lines and rectangular forms. The colour palette was reduced to the primary colors red, yellow and blue. Black, white and gray are also used.\(^9\)

Since Neo-Plastic artworks mostly contain geometric shapes they can be easily digitalized and analyzed. The simplicity of geometric shapes makes it easy to describe mathematically, thus we can implement an algorithm that randomly generates Neo-Plastic graphs strictly following the definition of De Stijl\(^4\) but does not guarantee the quality of graphs generated. Thus our target could be formalized as two steps:

---

2 Strong AI hypothesis: that AI are truly intelligent and self-aware. [3]
3 Weak AI hypothesis: that AI just behaves and always behaves intelligent. This allows intelligent AI being implemented by some less-amazing methods such as searching. However according to the rule of Occam's Razor, the definition seems redundant.
4 De Stijl is a Dutch artistic movement which purposed Neo-Plasticism. It is also the name of a journal.
1. Generate Neo-Plastic graphs.
2. Evaluate and refine these graphs to make better graphs.

To achieve step 1 we need a formalized definition of “Neo-Plasticism graph”, and an algorithm built on this definition to generate large number of graphs. Step 2 is where the interactive genetic algorithm gets involved. In step 2 we present one or more of the graphs generated in step 1 to the user to receive some “rating”. This “rating” information is used in refining the graphs. Step 2 is an interactive stage that uses human perception as an implementation of the fitness function in the evolutionary process.

**Piet Mondrian and His Neo-plasticism Fashion**

In 1919 Piet Mondrian returned to France and started producing grid-basic paintings, which latterly known as “Neo-Plasticism” or “De-Stijil” in Dutch.[15]

Basic rules of Neo-Plasticism paintings were found. The most significant characteristics for Mondrian’s Neo-Plasticism paintings is that they contain only simple vertical or horizontal lines with rectangles that formed by the lines filled with primary colors (red, blue, yellow)

During late 1920 and 1921 there was a style shift that the lines within Mondrian's painting tended to be fewer and bolder. [15] In the years followed Mondrian tended to use less color, much more preferred leaving most part of the paintings white. Mondrian also concluded that the spacing of lines, rather than the placement of color, was the major determinant of aesthetic appeal in his paintings [16].

**Analysis Over Mondrian's Work**

Mondrian's Neo-Plasticism paintings feature visually pleasing images with simple geometrical shapes and compositions. Attempts were made to discover the “rules of aesthetics” that lies within. An experiment[6] confirmed the existence of the aesthetics of Mondrian's painting that make them more than vertical-horizontal lines and colour bricks: “...that the pictures are not random configurations of lines, but instead are optimal aesthetic configurations”[6].

Participators were asked to make a preference judgment upon two groups of Mondrian-like paintings: one group being the duplication from original pieces and another group being “pseudo Mondrian” which were slightly modified on limited spots from the originals. The fact that statistically the participators shown more preference over original ones indicates that the composition of elements of Mondrian's painting was carefully considered by Piet Mondrian, although the artist himself claimed not to use an explicit rule[7].

Although the work [6] did not give any description or conclusion upon the model that lies within, it did propose a methodology, that the rules of “Mondrian's painting” -- or the “nature of composition” can be empirically discovered by experimenting with multiple different compositions originating from
Mondrian’s work.

Besides the empirical technique that requires human evaluation, there were multiple attempts to retrieve mathematical models by performing quantitative analysis over the geometrical structure of Mondrian’s paintings. It is sure that Piet Mondrian himself did not consider math while composing, so the problem is whether he did it subconsciously. Although there is no strict evidence proving that simple “number math”[4] (measuring grids and lines of the graph, finding a “golden ratio” or something similar) is a dead end, such attempts did not give convincing results so far[5].

This work does not try to discover the rules of Mondrian’s works. The AI program[6] that generates Mondrian-like pictures does not have a “predefined” model of aesthetics for automatic graph quality evaluation. Instead the program collects and refines possible parameters of the model according to humans reaction. The method is more empirical than mathematical. An evolutionary algorithm – which gradually builds a solution and are good at finding satisfying solutions which are not necessarily globally optimal[7] is quite suitable for our problem at hand.

Automatic Picture Generation

Much research on computer generating pictures using genetic algorithm has been done. An attempt to automate generating textures by using a statistical model has been successful[1]. Although it is very hard, or impossible to give an explicit and precise mathematical model for the concept “aesthetic” (such an attempt would probably lead to determinism), a minor quantitative model focusing on one single aspect of “aesthetic” is still capable of functioning as a rating module in the evolution procedure. The model used in the evolutionary image synthesis focused solely on the colour variation of the painting.

The research[13] reveals that colour gradients of many famous artworks conform a normal distribution. It is believed that the human response over an artwork is (partially) determined by the visual stimuli that it gives on colour. Humans are more attracted to the part where colour shifts and are more comfortable at a certain “shifting rate” of colors. Varieties of famous masterpieces are analyzed and a mathematical model about the colour gradient is abstracted. The model consists a group of statistical parameters collected from the samples, such as mean and standard divination of the pixels' RGB value. Since masterpieces are considered aesthetic by a large population of humans, the model could be used in the fitness function in the evolutionary process to generate nice pictures.

5 “Misplaced attempts” as criticized. [8]
6 Darwinian. The program that implements AI composition of Mondrian-like paintings.
7 There’s no such graph that is “best above all” that everybody agrees on.
The model is quantitative (all its values are real numbers) and simple enough to be programmed in a fully automated evolutionary process without intervention of humans, thus the number of elements and generations can be massive.
as long as the computer can cope ([1] uses 50 generations during an experiment). Large number of solutions and generations are not practical in an interactive scenario since human is easily get exhausted.

However, the outputted images are somehow bald, since the model focused solely on colour gradient supports no higher level of concepts that has bigger granularity than pixels, such as architecture or proportion. The generated pictures are pretty abstract without showing any identifiable figures. However, this approach is very successful when the palette is sampled from impressionistic paintings (for instance, artworks from Claude Monet), which are heavily focused on the usage of colour to express their themes.

The Interactive Genetic Algorithm is widely adopted to solve problems that involve subjective judgment from humans. Applications include: music composing that generates melody[10], industry design[11], facial image generation by combining partial images of facial photos[12]. It is reasonable to believe that IGA\(^8\) is suitable for the problems that need human creativity and intuition – which are not yet owned by artificial intelligence.

---

\(^8\) About (Interactive) Genetic Algorithm and Bacteria Evolution Algorithm: precisely in this work we use BEA to solve the problem. BEA is treated as a variation of Genetic Algorithm since both of them share the same procedure with implementation difference on several spots, and “interactive” is a word describing the implementation of fitness function for both. See Part II – a genetic algorithm scaffold.
II. Definitions, Issues and Technical Alternatives

Goals

The interesting characteristics of the Neo-Plasticism painting make it very easy to be analyzed. Unlike other “traditional” artworks that contain many emotional and subjective elements, the base role of Mondrian’s painting is very well defined, explicit and unambiguous, thus makes it highly mathematically describable. Computer algorithm that can be used to analyze even generate Mondrian-like paintings is much easier to implement without touching those abstract and subjective concepts such as aesthetics or figures.

In this part we will define concepts and problems so that they can be used to create a program that creates and manipulates Mondrian-like paintings, to resurrect Piet Mondrian in a cyber-genetic way.

Defining rules of Mondrian-like Graphs

As stated by the declaration of De Stijil, only vertical lines and horizontal lines are allowed in the graph. The rectangle is also stated as the basic element of Neo-Plasticism, but from a programmatic point of view, rectangles are nothing more than a “byproduct” formed by horizontal and vertical lines. Thus a Mondrian-like graph could be deemed as a collection of horizontal and vertical lines, in another word, “line-based”. However, certain rules must apply on the lines or the produced graph will be totally chaotic. To clearly define these rules following node types are defined:

- **Nodal point** – from which 3 or more lines emanate. Since only vertical and horizontal lines are allowed in the graph, a nodal point could emanate 3 or 4 lines.
- **Arbitrary point** – from which exactly 2 lines emanate.
- **Terminal point** – from which one line emanate.

And we further define several special cases, just for convenience:

- **Cross point** – from which 4 lines emanate, which makes the point the center of a cross. This is a special case of nodal.
- **Online point** – from which 2 lines emanate, which form a 180 degree angle. This is a special case of arbitrary.
- **Right angle point** – from which 2 lines emanate, which form a right angle. This is a special case of arbitrary. Such point is not allowed.
- **Initial point** – no line connected to this point as yet.

Examining Mondrian’s related artworks it could be revealed that each line in

9 We call that graphs generated simulating the style of Piet Mondrian’s as “Mondrian-like”, to distinguish from Mondrian’s original works.
10 Concepts borrowed from [5], which is a topology analysis over Mondrian’s painting.
the graph must have both of its ends being nodal or online, or in some very rare cases, have one end being terminal. If we treat the edge of canvas being a special set of invisible lines, rule 1 could be given:

- Each line in a Mondrian-like graph, must have one of its ends being nodal or online, another end being terminal, nodal or online.

In part III we introduce an algorithm that utilizes rule 1 to generate random Mondrian-like graphs.

**Defining The Problem**

Although the word “create” or “design” hints at shaping something from nothing, we can still classify a “creativity problem” as simple searching problems (in which the word “search” hints at finding something that already exists) – it sounds counterintuitive and humiliates intelligence, but from a point of view of AI the following terms are quite equivalent:

- Draw a nice picture on a given canvas (for instance: 100 x 100 pixels) or
- Lookup a nice picture among all possible pictures of a given canvas

Principally all possible pictures can be defined: if using 256bit RGB coding for a canvas containing 100 x 100 pixels, there would be \((256^3)^{10000}\) pictures, which is quite a big number but still finite. We can imagine that an AI painter composing a painting starting from an empty white canvas. Each time it puts a stroke on the canvas, it produces a semi-production of the final masterpiece, and this semi-production can always be found among all possible pictures. If we record the whole composing procedure, from the first stroke on the empty canvas to the last stroke that finishes the masterpiece, we can have a sorted queue of semi-productions, with the first element being the empty canvas and the last element being the finished masterpiece. This queue could be deemed as a search path in a solution space with every semi-production being a node of the search path: that the AI is not actually composing a painting, it is looking for an existing good solution by going through a search path. From this point of view, our target “create good Mondrian-like pictures” is quite equivalent to “searching good Mondrian-like pictures”. Of course linear algorithms are not practical for problems with such big solution spaces, that is why a genetic algorithm is introduced. We define the following terms for the problem:

**Solution space:** a set containing all Mondrian-like graphs of a given specification

**Solution:** a Mondrian-like graph that does not violate rule 1

**Target:** find a solution in the solution space which is good enough.

And the following genetic-algorithm-specific issue are involved:
Chromosome Encoding: a chromosome contains all needed information to construct a solution.
The simplest way to encode the chromosome is not to encode at all – the data structure used to store the solution is taken directly as the chromosome. However, chromosomes must support genetic actions such as crossover and mutation. It must guarantee that after crossover or mutation, the encoding is still valid – that is, the derived chromosome must still be able to construct a solution.

Fitness Value: fitness value is a rating of a chromosome measuring “how good it is”. The simplest form of fitness value can be a single numeric value, but it is far from enough to rate complicated solutions. A fitness value could be a data structure containing multiple assessments for different objectives[2].

A Genetic Algorithm Scaffold

Most problems utilizing genetic algorithm has the same structure that merely expresses the procedure of evolution:

```pseudocode
#Pseudo-code of evolution
genetic_algorithm:
  this_generation = Nil
  next_generation = Nil
  while true:
    if this_generation == Nil:
      this_generation = generate_first_gen()
    end if
    evaluate_all(this_generation)
    while next_generation NOT FILLED:
      parents = select(this_generation)
      offspring = multiply(parents)
      mutate(offspring)
      #add offspring to the next generation
      next_generation <- offspring
      if target_reached?(next_generation):
        return result(next_generation)
      else:
        #next iteration
        this_generation = next_generation
        next_generation = Nil
      end if
```

However, the functions marked bold are problem dependent. Their meaning and functionality defines as follows:

- **generate_first_gen()**: At the very beginning the first generations of chromosomes must be generated.
- **evaluate_all()**: This is a rating phase during which each chromosome is assigned a fitness value. The fitness value is calculated by the fitness function
which could be a mathematical model, or human, or a combination of both.  

**select()**: Choose a pair of parents from the given generation. The most famous algorithm of doing is *roulette wheel*: each chromosome occupies a sector area of a roulette wheel, the size of the area is decided by it's fitness value. A chromosome is selected if the ball falls into it's sector area. The chromosome with higher the fitness rating has the bigger the sector area, and with better chance to be selected.

**multiply()**: a pair of chromosomes exchange part of them and produce their successor chromosome.

This is the main difference between bacterial evolution algorithm and conventional genetic algorithm. Biologically the multiplication of bacteria is very different from vegetables or mammals. Instead of crossover (\(X_1X_2 + X_3Y_4 = X_1X_3 + X_2Y_4\)), bacteria simply duplicate themselves and occasionally absorb DNA from others.

Part IV discusses the implementation of bacterial behavior of multiplication of our problem.

**target_reached()?**: This function evaluates whether a *good enough* Mondrian-like graph exists in the given generation thus returns it and terminates the algorithm.
III. Digitalization of Mondrian-like painting

Pixel Graph vs Vector Graph

A digital representation of the artwork needs to be strictly defined for a computer program if it intends to produce such artwork. The most commonly used method digitalizing a painting is to put it into a scanner which outputs a pixel-based bitmap. In the bitmap the painting is represented by a n*m matrix in which each element is a pixel represented by RGB value. However, since the pixel-based solution is quite of small granularity, complicated statistical methods are required to performed analysis over bitmap or generating them in a reversed manner.

However, although a pixel-based solution seems to be the only choose of digitalizing paintings that are very complicated in shapes and colors (Illustration 9), it is not efficient at describing Neo-Plastic artworks since most pixels are wasted on representing the same attribute (Illustration 10).

We use a vector-based approach for describing Neo-Plastic paintings. Instead of recording data based on pixels, the graph is parsed into lines and curves which are represented by a set of vectors. A vector based solution is very efficient describing graphs containing mostly geometric shapes and curves – which is a most significant feature of Neo-Plasticism. Although quite abstract and simple, the seemingly-random layout and colour of rectangles and lines within Neo-plastic paintings shows aesthetics:

*Even the most perfect, the most general geometrical form expresses something specific. To destroy this limitation (or individuality) of expression as far as possible is the task of art, and constitutes the essential of all style -- Piet Mondrian*
The basic element of Mondrian's Neo-Plastic painting is the line. The advantage of using a vector-based approach rather than pixel-based approach is significant: instead of describing every pixel on a line, we only need to describe the line's end points and the coloured rectangle areas.

**Random Generation of Mondrian-like graph**

By utilizing nodes types defined in part II defining the problem, we can develop an algorithm that generates Mondrian-like graphs obeying rule 1. Following steps describes the procedure:

**Step 1 (pic step 1):** several original points are sampled from a certain distribution (the number of points and the distribution are controlled by chromosome, see IV). An original point is a special kind of point from which the whole graph is derived.

**Step 2 (pic step 2):** draw imaginary vertical and horizontal lines across all the original points.

**Step 3:** A set containing all imaginary lines is a super set of Mondrian-like graphs that originate from these original points. Drawing some of these imaginary lines obeying rule 1 makes a Mondrian-like graph. Each original point could emit lines at 4 directions: East, West, North, South. We start to construct the graph by emitting one line each time for each point, from left to right. In **step 3a**, line 1 is the first line drawn on the canvas. According to rule 1, it must end at one of the four edge lines since there are no other lines drawn on the canvas where it could terminate; line 1 also can not terminate in the middle of nowhere, because that would violate rule 1 for both ends being terminal. Line 2 must also end at an edge because it could not intersect with line 1. So is line 3. Line 4 has the option to end at one of the edge, or end at line 3, the latter option is selected. It should be noticed that here line 1 and line 4 forms a right angle which is not acceptable, but soon we will see that this angle will be fixed either in step 3 or step 4. This procedure repeats until we have added 2 lines for each after 3 original points, and we continue the leftmost original point ready to draw the 7th line for it.
In **step 3b**, line 7 has the following options:

- end at line 6
- end at bottom edge
- end at line 3
- end at right edge
A procedure made the decision for line 7 to end on line 6, this fixes the right angle formed by line 1 and 4. But even if line 7 does not fix it, the right angle will still be eliminated at step 4. Also line 8 ends on line 6, line 9 ends on the bottom edge.

**Step 4** is a remediation stage that makes sure no right angle points exist in the graph at the end.

It is not hard to see that if we execute **step 3** for 3 times, every original point would have 3 lines connected, thus all points are guaranteed to be nodal and step 4 not needed. But this approach would reduce some variation that the graph can possibly have since it eliminates the presence of online and terminal (which is rare but does happen) points. If we execute step 3 for 4 times, all points will be cross, all imaginary lines would be drawn and the graph will always become a boring net.

To give more variation to the graph, the execution of **step 3** is probability based. Instead of definitely giving a line to an original point each time, we assign a probability value of “giving a line”, according to the current type of the original point. These probability values are actually controlled by the chromosome, but we give an example with arbitrary values here:

- **initial point** – 100%: a point without any line connected definitely deserve a line to be given
- **right angle point** – 100%: such a point is not allowed and must be eliminated by giving a line
- **online point** – 30%
- **terminal point** – 90%: a terminal point is rare, but does exist in Mondrian's works. So we give a high probability that terminal point will be eliminated by giving a line
- **cross point** – 0%: already has enough lines
- **nodal point which has 3 lines** – 20%

**Step 5:** the mission of stage 5 is to fill some rectangles with primary colors. Since the skeleton of the graph is line-based and no rectangle is recorded previously, the first target is to search for all the rectangles with the graph with some algorithm. It should be noticed that many rectangles are nested. In Mondrian's work (illustration 1 and 2) it is allowed that a rectangle contained other rectangles is filled with colour¹¹.

---

¹¹ The early prototype program sometimes filled the whole frame with colour since the frame itself is considered a (biggest) rectangle.
IV. Evolution of Mondrian-like Graph

Issues of Chromosome Encoding

The first problem to encounter when implementing an evolutionary algorithm is the encoding of the chromosome for the solution.

The simplest way of doing this is to use the data-structure that represents a solution directly as the chromosome. However this approach is applicable only if the solution is “structurally smooth”, because the chromosome need to support genetic operations such as mutation or crossover and it must guarantee that after executing genetic operations the derived chromosomes must still remain valid. For a chromosome space C:

$$\forall c_1, c_2, \ldots, c_n \in C, \text{ genetic operation}(c_1, c_2, \ldots, c_n) \in C$$

For example, a 2D array $P_{xy}$ representing a pixel graph containing $(x+1)*(y+1)$ pixels can be used directly as a chromosome:

$$P_{xy} = \begin{bmatrix}
p_{00} & p_{01} & \cdots & p_{0y} \\
p_{10} & p_{11} & \cdots & p_{1y} \\
\vdots & \vdots & & \vdots \\
p_{x0} & p_{x1} & \cdots & p_{xy}
\end{bmatrix}$$

For a crossover operation on solutions $A_{xy}, b_{xy}$ at some crossover point $uv$:

$$\text{crossover}(A_{xy}, B_{xy}, uv) = \begin{bmatrix}
a_{00} & a_{01} & \cdots & a_{0y} \\
\cdots & \cdots & \cdots & \cdots \\
a_{u0} & a_{uv} & \cdots & b_{u(y+1)} \\
\vdots & \vdots & \vdots & \vdots \\
b_{[u+1]0} & b_{[u+1]1} & \cdots & b_{[u+1]y} \\
\vdots & \vdots & \vdots & \vdots \\
b_{x0} & b_{x1} & \cdots & b_{xy}
\end{bmatrix}$$

The derived chromosome (or solution) remains valid (still a pixel graph) because each element in the matrix has the same meaning. However, this approach is not applicable for scenarios with rules, in which elements in the structure have different meanings. Let’s consider a simplified solution space of our Mondrian problem, in which every solution is a vector graph that:

1. Contains $n+1$ lines
2. All lines must be either horizontal or vertical

The possible data-structure representing a Mondrian-like graph $M_n$ and the rule can be formally defined as following:
\[ M_n = \begin{bmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \\ \vdots & \vdots \\ m_{n0} & m_{n1} \end{bmatrix} \]

\( m_{n0}, m_{n1} \) is the end points of line \( n \)

\[ \forall 0 \leq v \leq n, \quad (m_{v0}.x = m_{v1}.x) \lor (m_{v0}.y = m_{v1}.y) \]

Assume that we have solutions (or chromosomes, in this context they are equivalent) \( A_n, B_n \). Performing a crossover operation on some crossover point \( u_0 \):

\[
crossover(A_n, B_n, u_0) = \begin{bmatrix} a_{00} & a_{01} \\ \vdots & \vdots \\ a_{u0} & b_{u1} \\ b_{(u+1)0} & b_{(u+1)1} \\ \vdots & \vdots \\ b_{n0} & b_{n1} \end{bmatrix}
\]

Then there's no guarantee of holding \( (a_{u0}.x = b_{u1}.x) \lor (a_{u0}.y = b_{u1}.y) \) and the rule is breached. We can see it is required that the recombination of genes between/among two or more chromosomes must still form a valid chromosome. To avoid derived chromosomes going “out of scope”, each gene within the chromosome must be “perpendicular” to others so that replacing one would not affect the rest. Encoding a Mondrian-like graph by specifying every vertex in the chromosome is trivial and futile since the successor is highly likely to violate rule 1 defined in Part II. Since the bacterial multiplication behavior is even more complex than crossing over, the encoding of the chromosome in the Mondrian-like problem must fulfill more strict requirement.

**Structure of The Chromosome of Mondrian-like Painting**

**Convention**

Some structures described in this work are represented as array. Such a structure would be written with each symbol of it's members enumerated within square brackets:

\[
\text{structure} = [\text{member}_0, \text{member}_1, \ldots, \text{member}_n]
\]

and it's member could be accessed using dot and index value:

\[
\text{structure.member}_n
\]

assignment operation can be carried out by using “<-”:

\[
s_a.member_n \leftarrow s_b.member_n
\]
which replaces the value of left hand side with right hand side.

**A formal definition of the chromosome**

Assume that $ML$ is the solution space (a set of all Mondrian-like painting); $C$ is the chromosome space:

$$C = \{ c | c = [g_0, g_1, \ldots, g_N], solution - of (c) \in ML, N \text{ is constant } \}$$

$$\forall c_0, c_1, \ldots, c_N \in C, c = [c_0 \cdot g_0, c_1 \cdot g_1, \ldots, c_N \cdot g_N] \Rightarrow c \in C$$

Literally, a chromosome assembled from parts taken from other chromosomes is always valid.

**Probability-based Gene-encoding**

On this Mondrian painting problem, one approach is to encode a gene specifying arbitrary characteristics of the painting, such as the directions of the lines or the color and position of the rectangles. This idea is straightforward but is not a good solution. A consideration is that we don’t want to be too deterministic over art. We can not ignore the fact that artworks are attracting (at least partially) because they are subjective and emotional products from human minds, no precise “formula” can be used to massively generate artworks, and every piece of art is unique.

Most of the gene-encoding for a Mondrian chromosome is probability-based. For example, we need to decide lines that emit from one specific original point.

Arbitrary encodings could be:

- $[\text{East, West, North, None}]$
- $[\text{West, North, None, East}]$

Then we have to inevitably deal with the chromosome invalidation problem described in *Issues of Chromosome Encoding*. A result of crossover of the above two could be:

- $[\text{East, West} \mid \text{None, East}]$

Which is illegal because it forms a right angle, and two lines are overlapping with each other.

To avoid this problem we simply specify “which direction the line is most likely to go” instead of “which direction the line will go”. For one specific original point we encode:

- $[\text{East} = 0.9, \text{West} = 0.2, \text{North} = 0.3, \text{South} = 0.5, \text{None} = 0.3]$  

During picture generation a selector will choose a direction according to the probability distribution.

This will also create an interesting effect that even given one specific chromo-
some, the generation algorithm will be able to give different but visually similar pictures. By using probability-based gene-coding, we avoided the chromosome invalidation problem as well as being deterministic.

**Chromosome Rating**

Each chromosome is assigned a chromosome value by the fitness function, which identifies its quality in the evolution. The simplest form of a “fitness value” would be a single real value. However, the thing we trying to rate here is “art” which is too complicated to be rated by a single value.

We use a multi-objective fitness value to rate the chromosome. Each gene of a chromosome has a value associated. Formally, for any chromosome \( c \times = [g_0, g_1, \ldots, g_N] \), and a fitness function \( \text{Fitness}(c) \), there is a fitness value \( f_x \):

\[
f_x = \text{Fitness}(c_x) = [r_0, r_1, \ldots, r_N], r_n = R_n(g_n)
\]

Where \( R_n(g_n) \) are a group of rating functions, each of them assigns a rating value \( r_n \) on gene \( g_n \).

With a multi-objective fitness value structure, varieties of evaluations based on different standards could be summarized. For example if a Mondrian-like graph is going to be evaluated on “structure”, then three specific genes \( g_a, g_b, g_c \) are involved in the evaluation because they're the genes that directly control the structure of the graph. Then a evaluation function focused on a certain aspect could be given:

\[
\text{Evaluate - structure}(f) = (f.r_a + f.r_b + f.r_c)/3
\]

**Implementation of Fitness Function** \( \text{Fitness}(c) \)

As a utilization of interactive genetic algorithm, human users work partially as the fitness function. **Part V** discusses this issue.
Bacterial Multiplication

Assume that there are chromosomes $c_0, c_1$ and let $c_0$ be the primary chromosome that absorbs some genes from $c_1$ and preserves most of its own. Then we can implement the multiplication function introduced in part II - a genetic evolution scaffold:

\[
\forall 0 \leq n \leq N, R_n(c_0 \cdot g_n) < R_n(c_1 \cdot g_n) \Rightarrow c_0 \cdot g_n \leftarrow c_1 \cdot g_n
\]

Illustration 11 shows the stages of the bacterial way of spawning: a chromosome absorbs good gene pieces from another one and replace its own.
V. Human evaluation as part of fitness function using a GUI interface

Programming purpose

The name “Darwindrian” is a composition of two names “Darwin” and “Mondrian”. As the name hints, “Darwindrian” is an AI program that creates painting of Neo-Plasticism style which was originated by Piet-Mondrian by utilizing BEA algorithm - a type of computer algorithm simulating the procedure of evolution, which is a theory proposed by Darwin.

General Description of Darwindrian

Darwindrian is an AI program which generates aesthetic pictures that mimic the artwork from Piet Mondrian. This is done in an interactive way: one or a few generated pictures as well as some questions regarding them are displayed on the screen, and the user view these pictures then give answers to the question according to his/her own feelings. The program then generate a new batch of pictures which were refined by the user's answers.

Structure of Darwindrian

Darwindrian consists 3 main modules:

1. Random Graph Generator
   Darwindrian generates Mondrian-like pictures in a random way. However, the procedure is controlled by Chromosome, which specifies certain argument of the generation (for example: forms of distribution from which the random function samples)
   Part III discusses the implementation of Graph-Generator in detail.

2. Evolution Manager
   This component manages Chromosomes that contains variables which affects the graph-generation procedure. Chromosomes are iteratively refined by receiving responses from the GUI Layer, so that pictures generated by Graph-Generator could become more and more corresponding to the user's taste.
   Part IV discusses the implementation of Chromosome and Evolution Manager in detail.

3. GUI Layer
   The GUI Layer is responsible for displaying generated pictures and receiving responses from the human user.
   Part V discusses GUI implementation in detail.

GUI design and consideration

The GUI is an essential part of implementation of the fitness function. Since there is no convincing mathematical model that successfully describes the aesthetics of Piet Mondrian's compositions, human intelligence must be intro-
duced in the evaluation stage. The process is shown in Illustration 13.

“Darwindrian” (composition of “Darwin” and “Mondrian”) is a program that composes Neo-plastic paintings simulating the works of Piet Mondrian. To generate pictures that not only structurally obeys the definition of De Stiji but also visually aesthetic, the program requires evaluation from a human user as input to refine its future work.

The basic layout of the GUI is quite simple. The right side of the the main window shows a generated graph either from the current generation, or from a past generation if user selects previous icon from left. Beneath the graph there are radio buttons that ask the user about the opinion about the graph. A “See next” button will collect user's selections and show a graph from the next generation. The left side shows a list of icons, each of them representing one generation. The user can click on the icon to review the graph of previous generations as well as the rating given. The rating of previous graphs can be changed.

![Illustration 12: Darwindrian: main window](image)

**Darwindrian running effects**

The evolving process of Mondrian-like paintings was started with “Mutation” options off. That means there are no new random samples added during the evolution. The only chromosomes shown (except for the first generations) are derived from their parents.
Samples in first generation:

It can be observed that the first generation of Mondrian-like paintings are quite random, for the program receives no preference data from the user. Several pictures can be observed that have very big coloured rectangles covering most of the canvas, which are obviously not aesthetic. The user rated these pictures by preferring “None” for them.
In second generation, the pictures with over-sized colour rectangles are eliminated:

Some significantly bad samples were eliminated according to the user's preference

The second generation shows some level of similarity among the samples, which is reasonable because they're “siblings” from previous generation. For example the pair (p9, p10) and (p2, p7) are somehow structurally similar\textsuperscript{12}.

\textsuperscript{12} Although can not eliminate the possibility that they are similar just by pure coincidence.
An interesting result is noticeable after several generations has passed as shown in the following images. The structure of the remaining 2 Mondrian-like pictures became very similar to each other. This is very reasonable because after several generations of interchanging genes closely with each other, their chromosomes became similar. Such phenomenon is very close to the biologic concept of “close breeding” which is harmful, because it brings less variation of the genes of the successors thus makes them disadvantaged in the natural competition.

Only 3 left after several generations. The remaining samples shows similarity with each other

The only two samples left after more even more generations. Evolving process requires at least 2 samples. (because the mutation option is turned off, there's no fresh random samples added in)
VI. Acknowledgment

Thanks to Dr. Alistair Rendell and Dr. Peter Strazdin for directions on the draft of this work. Thanks to Kerryn Boorman and I.C Manus finding and providing a reference for this work.
VII. References


[17] Tom Gedeon, Jian Yin Shen, *Making art using evolutionary algorithms and artificial AI*, Proceedings of BOOMo7, Taiwan – Australia New Media Workgroup, 7 pages