

Cyber-Genetic Neo-Plasticism

– An AI program Creating Mondrian-like Paintings by using Interactive Bacterial Evolution Algorithm

Jian Yin Shen, Tom Gedeon

Abstract – This work investigates into the way for an AI program generating pictures simulating the style of Neo-Plastic artworks from Piet Mondrian. In the article 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

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”¹ 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 weaker² 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, often evolutionary algorithms are involved.

The evolutionary algorithm is a class of non-linear approximative algorithm that simulates 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 is 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 Stijl³ but does not guarantee the quality of graphs generated. Thus our target could be formalized as two steps:

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

1 Strong AI hypothesis: that AI are truly intelligent and self-aware. [3]

2 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.

3 De Stijl is a Dutch artistic movement which purposed Neo-Plasticism. It is also the name of a journal.

implementation of the fitness function in the GA process.

Related Work

Much research on computer generating pictures using genetic algorithm has been done.

Automatic Picture Generation: 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 deviation 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.

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.

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 are 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 far⁴.

4 “Misplaced attempts” as criticized. [8]

This work does not try to discover the rules of Mondrian's works. The AI program⁵ 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. A genetic algorithm – which gradually builds a solution and are good at finding satisfying solutions which are not necessarily globally optimal⁶ is quite suitable for our problem at hand.

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 IGA⁷ is suitable for the problems that need human creativity and intuition – which are not yet owned by artificial intelligence.

5 Darwindrian. The program that implements AI composing of Mondrian-like paintings.

6 There's no such graph that is “best above all” that everybody agrees on.

7 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.



MONDRIAN
OPPOSITION OF LINES: RED AND YELLOW
NEW YORK GRAPHIC SOCIETY

Illustration 1: *Opposition of Lines: Red and Yellow*

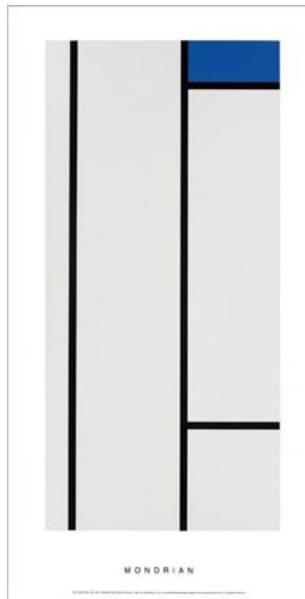
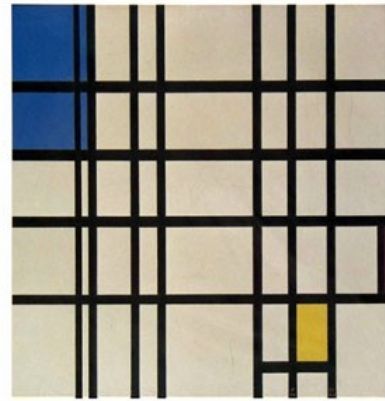


Illustration 3: *Composition in White and Blue*



Piet Mondrian
COLLECTION OF EUROPEAN MASTERS

Illustration 2: *Rhythmus*

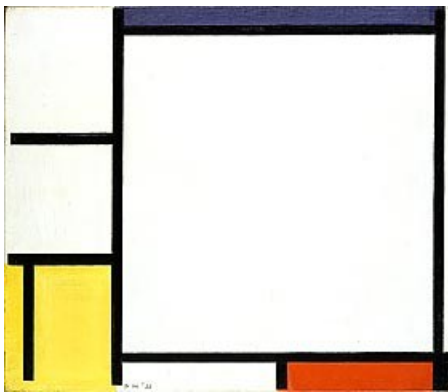


Illustration 4: *Composition of Red, Blue and Yellow*

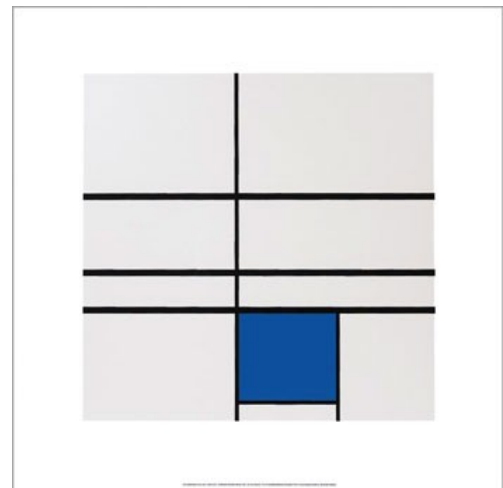


Illustration 5: *Composition with Blue*

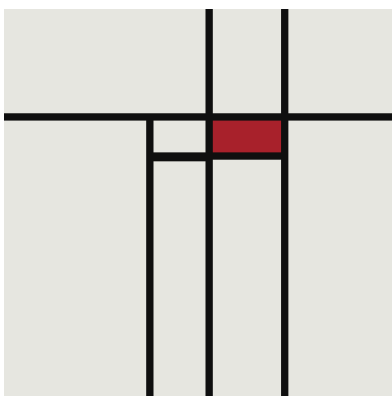


Illustration 6: *Early Selected production of Darwindrian prototype, 480 x 480*

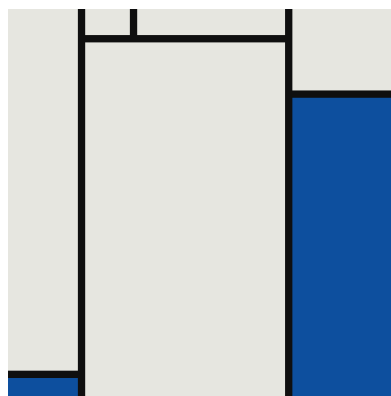


Illustration 7: *Early Selected production of Darwindrian prototype, 480 x 480*

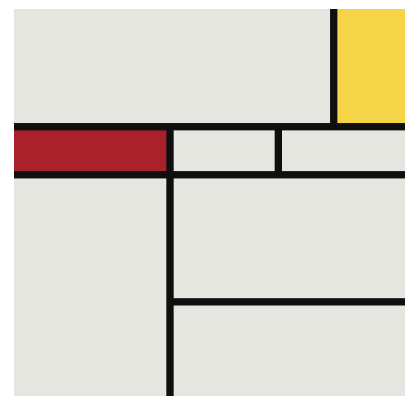


Illustration 8: *Early Selected production of Darwindrian prototype, 480 x 480*

II. Definitions, Issues and Technical Alternatives

Defining rules of Mondrian-like⁸ Graphs

As stated by the declaration of De Stijl, 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 defined⁹:

- *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 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.

⁸ We call that graphs generated simulating the style of Piet Mondrian's as “Mondrian-like”, to distinguish from Mondrian's original works.

⁹ Concepts borrowed from [5], which is a topology analysis over Mondrian's painting.

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 issues 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:

#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 its fitness value. A chromosome is selected if the ball falls into its 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).



Illustration 9: Academic art - Flaming June - Fridrick Lord Leighton. The complexity of colour gradient and curves of this Academic-art style masterpiece makes it impossible, or very low efficient, to be described as a collection of vectors. Describing pixel by pixel is the only choice.

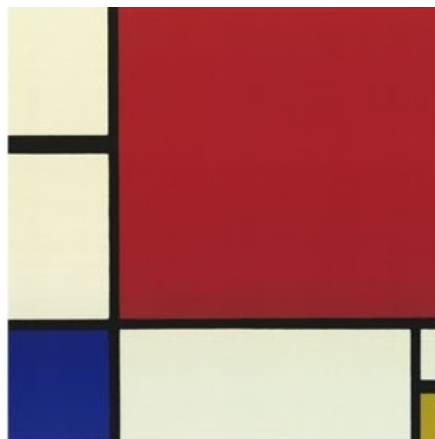


Illustration 10: Neo-Plasticism - Composition of Red, Blue And Yellow - Piet Mondrian.. If using a pixel-based approach, there would be a lot of redundant data. For instance, all pixels in the big red rectangle has the same RGB value.

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 a 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 it's colour.

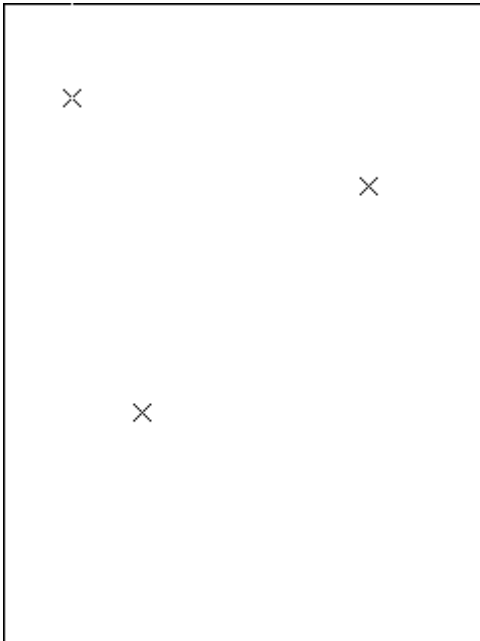
Random Generation of Mondrian-like graph

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:

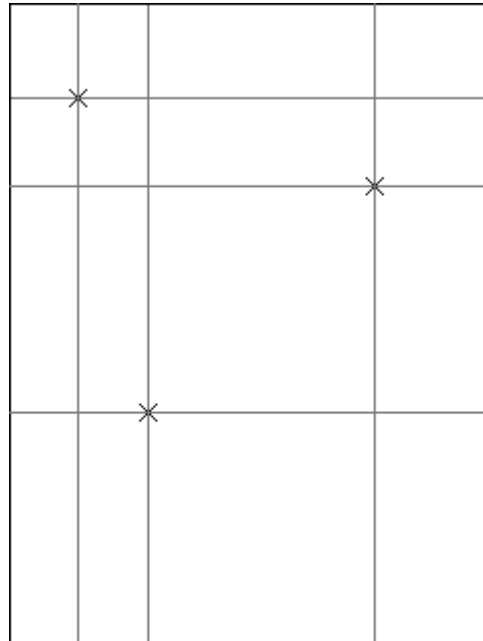
Step 1 (pic step 1): several *original* points are sampled from a certain distribution (the number of points and the distribution are controlled by chromosome, see IV). Original point is a special kind of point from which the whole graph is derived.

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's no other lines drawn on the canvas where it could terminate on; line 1 also can not terminate on the middle of nowhere, because that would violate rule 1 for both ends being terminal. Line 2 must also end at 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, latter option is selected. It should be noticed that in 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 3 original points, and we back to the leftmost original point ready to draw the 7th line for it.



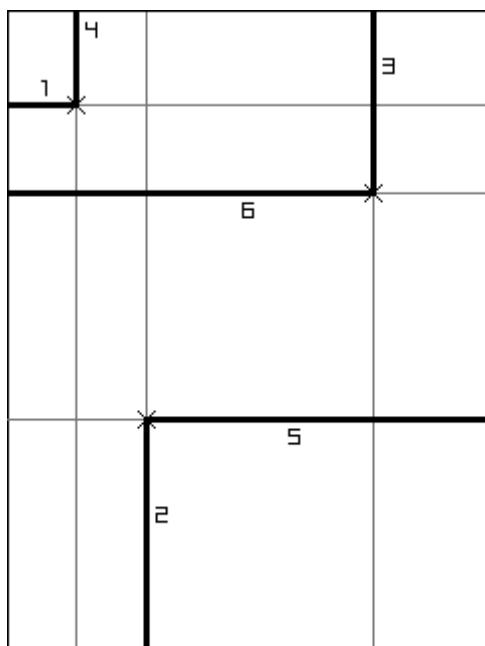
step 1: Random original points generated on a given canvas



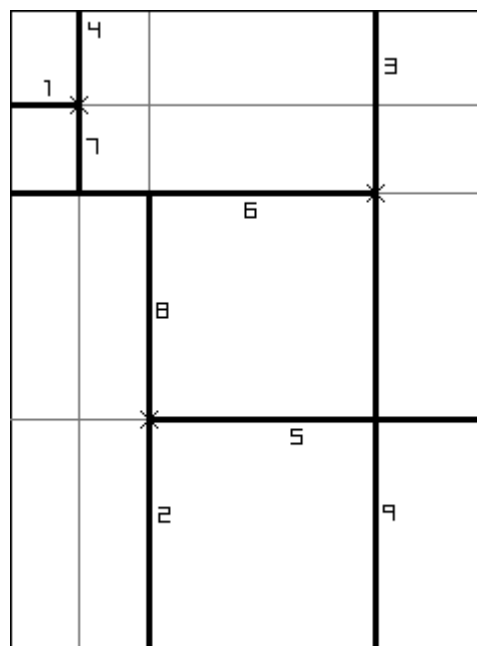
step 2: Imaginary lines drawn crossing original points

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



step 3a: Draw lines emit from the original point (sequence marked by number)



step 3b: Skeleton complete

A random 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 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 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 explicit 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 had enough
- nodal point which has 3 lines* – 20%

Step 5: the mission of stage 5 is to fill the 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 filled with colour¹⁰.

¹⁰ 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 :

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

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

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

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

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

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:

$$M_n = \begin{bmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \\ \dots & \dots \\ m_{n0} & m_{n1} \end{bmatrix} \quad m_{n0}, m_{n1} \text{ is the end points of line } n$$

$$\forall 0 \leq v \leq n, (m_{v0} \cdot x = m_{v1} \cdot x) \vee (m_{v0} \cdot y = m_{v1} \cdot y)$$

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 :

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

Then there's no guarantee of holding $(a_{u0} \cdot x = b_{u1} \cdot x) \vee (a_{u0} \cdot y = b_{u1} \cdot y)$ 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 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 vertexes 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 structure would be written with each symbol of it's members enumerated within square brackets:

$$structure = [member_0, member_1, \dots, member_N]$$

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

$$structure.member_n$$

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:

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

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

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

Chromosome Rating

Each chromosome is assigned a chromosome value by the fitness function, which identifies it's 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 $c_x = [g_0, g_1, \dots, g_N]$, and a fitness function $Fitness(c)$, there is a fitness value f_x :

$$f_x = Fitness(c_x) = [r_0, r_1, \dots, r_N], r_n = R_n(g_n)$$

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:

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

Implementation of Fitness Function $Fitness(c)$

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

Bacterial Multiplication

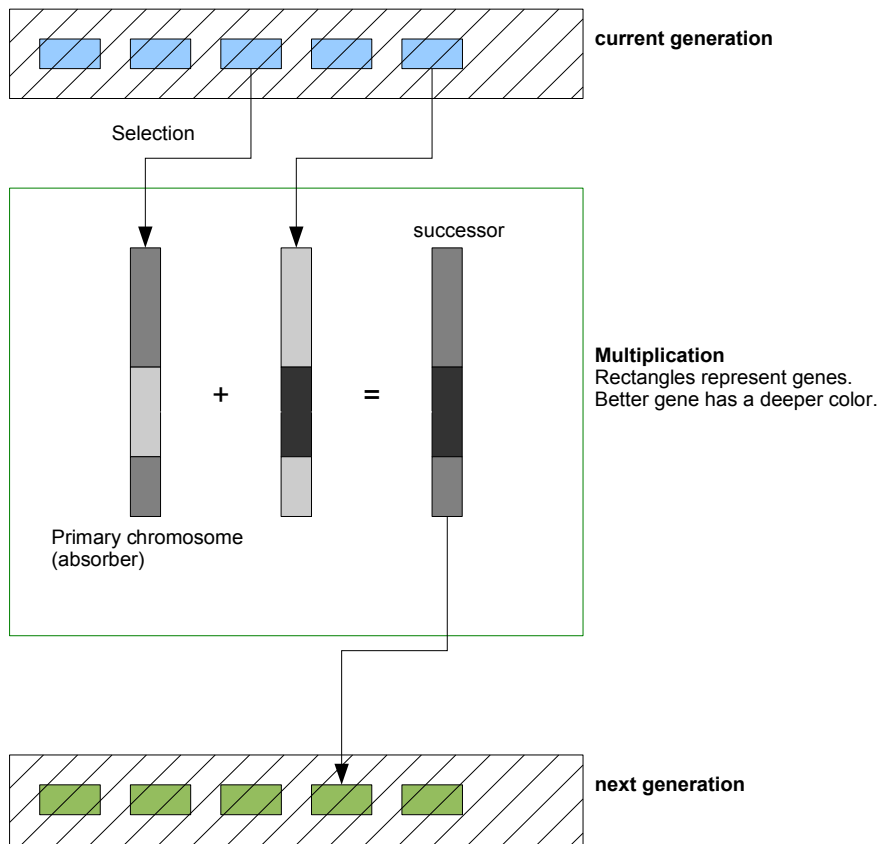


Illustration 11: bacterial behavior: spawn next generation by combining good genes from others

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 it's own. Then we can implement the multiplication function introduced in **part II - a genetic evolution scaffold:**

$multiply(c_0, c_1):$

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

Illustration 11 shows the stages of the bacterial way of spawning: a chromosome absorbs good gene pieces from another one and replace it's own.

V. Human evaluation as part of fitness function using a GUI interface

GUI design and consideration

The GUI is a essential part of implementation of the fitness function. Since there is no convincing mathematical model that successfully describes the aesthetics of Piet Mondrian's Painting, human intelligence must be introduced 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 Stijl but also visually aesthetic, the program requires evaluation from a human user as input to refine it's 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. However the rating of previous graphs can be viewed but not changed. (To be implemented) The “...” button can be used to replace the current graph with another graph from the same generation.

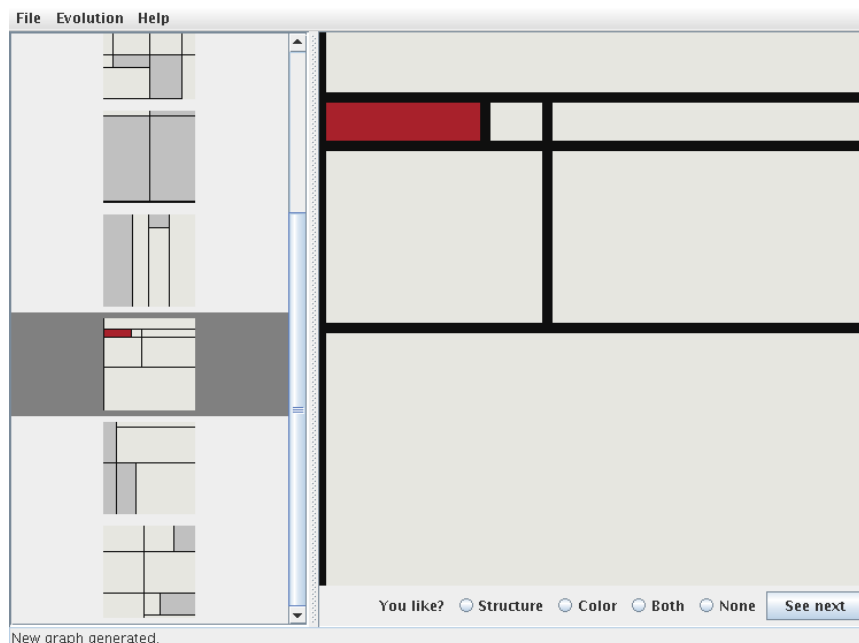


Illustration 12: Darwindrian: main window

Cumulated fitness model (CFM)

Since there is no previously known mathematical model, we wish to construct a fitness model based on user's input. The CFM is a list consisting a series of *standard genes* which acts as criteria rating other genes:

$$cfm = [gs_0, gs_1, \dots, gs_N]$$

The standard genes gs_n firstly come from some chromosome of the first generation, then are continuously refined by the gene-refine functions which cumulatively change the standard genes towards the current genes given:

$$cfm.gs_n \leftarrow gene - refine_n(g_n, gs_n)$$

Since now we have a standard, it is possible to evaluate other chromosomes (stage 4) in the current generation according to the *standard genes* in CFM. Each gene rating function $R_n(g_n)$ can be implemented by comparing the difference between the to-be-rated gene g_n and standard gene gs_n in CFM.

Expected Effect

In the current version of Darwindrian a graph is rated on two aspects: structure or colour. Assume that some certain genes g_1, g_2 of the chromosome controls the structure of the graph and g_3, g_4 control colour. Assume that chromosome c_0 is randomly selected at stage 1 and visualized (stage 2), and user selected that they liked “colour” from the GUI on stage 3. Then genes $c_0.g_3$ and $c_0.g_4$ are selected to refine the *standard genes* (see Symbol Table):

$$cfm.gs_3 \leftarrow gene - refine_3(g_3, gs_3)$$

$$cfm.gs_4 \leftarrow gene - refine_4(g_4, gs_4)$$

The users selection steers the CFM towards his own preference. In stage 4 (fitness testing), chromosomes that are close to user's preference will gain higher fitness values and then have a better chance to pass on their characteristics to the next generation (stage 5).

Symbol Table

The following table shows a summary of symbols and functions mentioned. The row marked yellow is a possible example of a gene being represented and evaluated.

gene	Gene rating value	Standard gene	Gene refinement
g_n	$r_n = R_n(g_n)$	gs_n	$gs_n \leftarrow gene - refine_n(g_n, gs_n)$
$g_x = x$	$r_x = R_x(x) = x - x' $	$gs_x = x'$	$gs_x \leftarrow gene - refine_0(x, x') \leftarrow x' + (x' - x) / 2$
Chromosome			Cumulated Fitness Model
$c = [g_0, g_1, \dots, g_N]$			$cfm = [gs_0, gs_1, \dots, gs_N]$
Fitness value			
$f = fitness(c) = [r_0, r_1 \dots r_N]$			

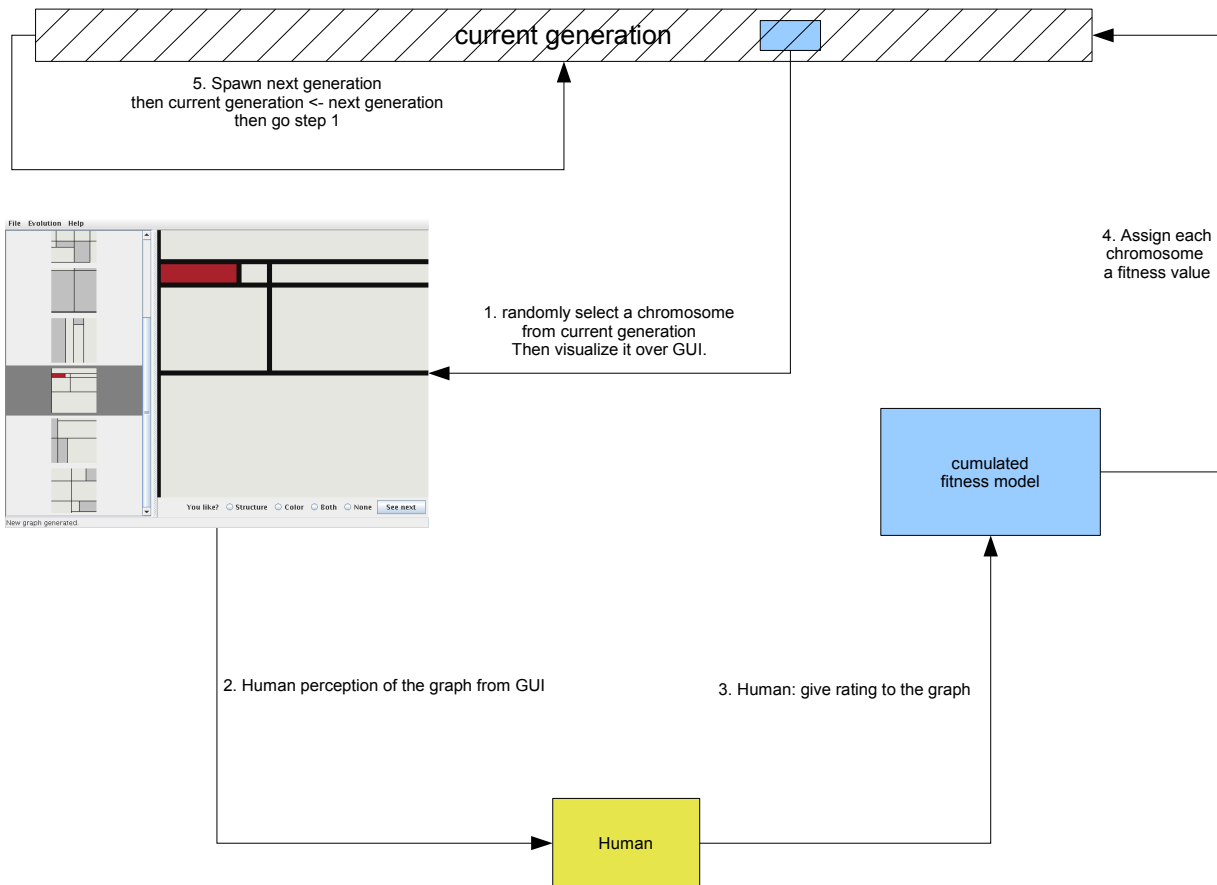


Illustration 13: Darwindrian: evolution procedure

VI. Evaluation

An experiment[6] shows that Piet Mondrian's originals are more than random compositions of lines and rectangles by revealing the fact that even minor modifications lead to less subjective satisfaction of audiences. In this part we discuss alternatives of evaluating "Mondrian-like graphs" which are generated by Darwindrian. The method taken in experiment[6] is not suitable for our problems, since this method only shows optimization of Mondrian's original works but we need evaluations over pseudo Mondrian's that are generated by AI. Further, we also need to prove the effectiveness of the BEA.

The simplest way of evaluation would be having participators giving a rating scale, for example from 1~ 10, over final productions. However, the conclusion is not convincing enough even if we obtain a production that satisfies most of the audiences. Sometimes the prototype Darwindrian¹¹ does occasionally give visually pleasing results from a totally random process without BEA involved, but it will not prove the effectiveness or the (weak) intelligence of Darwindrian since we can (principally) obtain Shakespeare's poem through another totally random process: having millions of monkeys hopping over millions of typewriters.

If the evolutionary procedure run by Darwindrian is statistically proved to be functional, it is reasonable to believe that Darwindrian truly works and provides or will provide visually pleasing Mondrian-like paintings with essentials of Neo-Plasticism lying within. Instead of asking preferences over final productions from participators, we ask a degree of satisfaction for samples taken from each generation of the evolutionary procedure. If the value generally increases it identifies the functionality of Darwindrian.

The evaluation can be hosted on a web site with a web-embedded version of Darwindrian presented. User will be asked for a survey automatically after he or she has played with Darwindrian and has gone through a certain number of generations. Data is collected and stored in the database for further analysis.

The source code of prototype Darwindrian can be downloaded at:

<http://code.google.com/p/darwindrian/>

VII. Further Development

Implementation of the Chromosome Structure and Functions

As described in part IV - V, a framework that utilizes the BEA has already been implemented, and certain rules for the genes that are yet to be defined have been summarized. In the next stage, concrete data structures for each gene and corresponding rating functions (as listed in part V Symbol Table) will be implemented. Multiple possible combinations of gene structures will be experimented to see the overall effect.

Software Development

The program (Darwindrian prototype) needs to be completed in the next stage. The basic GUI form of Darwindrian will not change greatly from the current prototype version, but details may vary. Considerations that will shorten the work-load of the user (for instance, mouse clicks needed before getting a good result) will be taken.

For experiment considerations, a web-page-embedded version of the program should also be constructed so that the experiment that requires many participators could be held over Internet. More programming issues are expected to arise and to be solved correspondingly.

Experiment

An experiment that verifies the functionality of Darwindrian will be designed (see part VI), including the following aspects:

¹¹ Prototype Darwindrian: early demo version of Darwindrian that generates Mondrian like pictures, with only very small amount of genes defined and no BEA involved.

GUI design: part of the data that is required to form the evaluation of Darwindrian may request direct input from the user. For this situation corresponding GUI components need to be developed.

The form of experiment: the most possible way would be publishing Darwindrian over Internet and making it runnable within the browser, the data is collected automatically in the background. This is the cheapest way to collect more samples.

Data processing: an statistical summarization model for the data collected should be advised.

1. VIII. Acknowledgment

Thanks to Alistair Rendell for directions on the early draft of this work. Thanks to Kerryn Boorman and I.C Manus finding and providing a reference for this work.

IX. References

- [1] Brian J.Ross, William Ralph, HaiZong, "Evolutionary Image Synthesis Using a Model of Aesthetics", *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, page 1087- 1094
- [2] Hideyuki Takagi, "Interactive Evolutionary Computation: Fusion of The Capacities of EC Optimization And Human Evaluation", *Proceedings of the IEEE*, vol.89, no.9, pp.1275-1296 (2001)
- [3] Stuart Russell, Peter Norvig, *Artificial Intelligence – A Modern Approach*, page 947
- [4] Charles Bouleau, "The Painter's Secret Geometry" (London: Thames Hudson)
- [5] A. Hill, "Art and Mathesis: Mondrian's Structures", *Leonardo*, I(1968), pp.233-4
- [6] I. C. McManus, B. Cheema, J.Stoker, "The Aesthetics of Composition: A Study of Mondrian", *Empirical Studies of The Arts*, Vol 11(2) 83-95, 1993
- [7] Reynolds, *Symbolist Aesthetics And Early Abstract Art*, page 173
- [8] Reynolds, *Symbolist Aesthetics And Early Abstract Art*, page 260
- [9] Wikipedia contributors, "De Stijl," *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=De_Stijl&oldid=82482430
- [10] J.A Biles, "Genjam: A genetic algorithm for generating jazz solos," in *Int. Computer Music Conf*, (ICMC94), (Aarhus, Denmark), pp. 131-137, 1994
- [11] J. Graf, "Interactive evolutionary algorithms in design", *Int. Conf. on Artificial Neural Nets and Genetic Algorithms*, pp. 227-230, Apr. 1995
- [12] S. -B. Cho, J.-Y. Lee, "Emotional image retrieval with interactive evolutionary computation", *Advances in Soft Computing-Engineering Design and Manufacturing*, pp. 57-66, SpringerVerlag, London, 1999
- [13] W. Ralph, "Painting the Bell Curve: The Occurrence of the Normal Distribution in Fine art," *In preparation*, 2006
- [14] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992