Pattern Evolver

An Evolutionary Algorithm that Solves the Nonintuitive Problem of Black and White Pixel Distribution to Produce Tiled Patterns that Appear Gray

by Keith Wiley

Contents

Abstract

Pattern Evolver is an evolutionary program that produces black and white bitmapped patterns, which, when tiled, create a smooth unbroken gray appearance to the human eye. Pattern recognition in the human eye and the human visual cortex is highly developed, to the point where humans will reconstruct lines and shapes that do not exist in the actual physical stimulus. As a result, creating smooth gray patterns is not intuitively easy. *Pattern Evolver* harnesses Darwinian selection to solve this task and thus falls into the category of evolutionary programs. Like biological systems, this program exemplifies evolution in an adaptive landscape with multiple local peaks.

Introduction

For a program I was recently hired to write, I discovered that I would need to produce degrees of contrast between light and dark areas on a black and white, one-bit deep monitor. Gray patterns had to be produced by varying ratios of black to white pixels. Since I was writing the program for an Apple Macintosh computer, the system was equipped to use 8 x 8 pixel patterns. I decided to use tiles of the same size so that I could easily encode the patterns in my program, and started making the patterns I would need. In a 64 pixel grid there are 65 possible ratios of black to white pixels 0:64 through 64:64. The smoothest possible gray for some ratios is intuitively easy to design. These are the ratios that divide evenly into 64: 0:64, 1:64, 2:64, 4:64, 8:64, 16:64, 32:64, and 64:64, for both black to white and white to black (Figure 1).

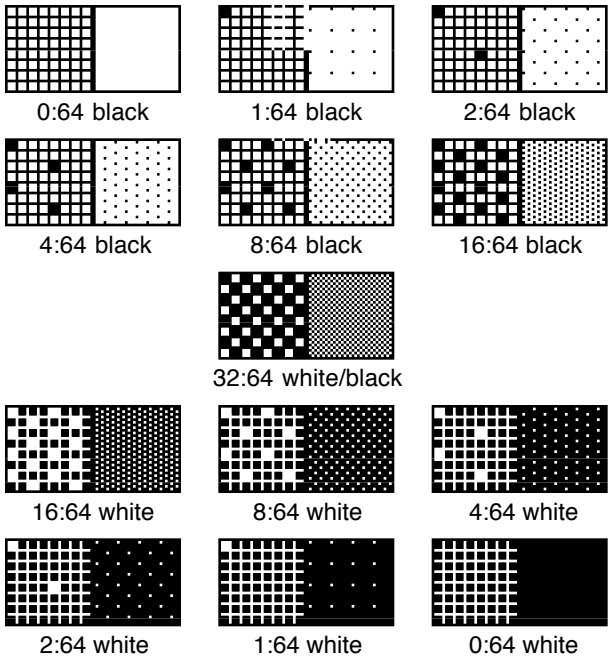

FIGURE 1 — The only thirteen perfectly smooth gray patterns on a grid of 64 pixels are those with black to white or white to black ratios that divide evenly into 64. For these ratios there is an exact and methodical pattern that produces an ideally smooth gray tiling effect. In all figures illustrating patterns, the genotype is enlarged on the left for easy viewing and an example of the phenotype is shown regular size and tiled four across and four down on the right.

A problem arises when one attempts to make all the other numerous patterns corresponding to the leftover ratios look like a smooth gray surface. In many cases, no symmetrical pattern of pixels is possible. Furthermore, when symmetrical patterns are possible they often do not appear smoothly gray. For example, to make a gray pattern of ratio 15:64, simply removing any one of the pixels in the predetermined 16:64 pattern from figure 1 will result in a symmetrical pattern, but not a smoothly gray pattern. It will have a big white dot on it and consequently will not look evenly gray. Instead it will look like a quilted pattern (Figure 2).

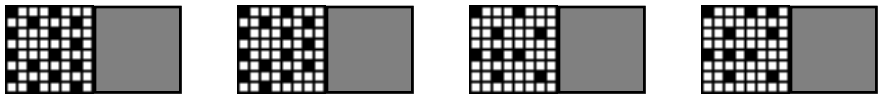

FIGURE 2. — Examples of symmetrical but unsmooth patterns. It is possible to make patterns that tile perfectly and yet do not produce the desired smooth gray appearance. Such patterns are abundant, nearly filling the set of remaining patterns that are left after removing the thirteen perfect smooth gray patterns in figure 1. Despite their frequency and ease of use, they defeat the

purpose of creating smooth gray patterns and must be discarded as relatively nice, but inadequate, patterns. A better solution can be evolved for the desired ratio.

In order to make all the other gray patterns, I had to work them out one by one, by moving individual pixels to spread the pixels out as evenly as possible, without producing lines and squiggles, or dark and light areas. It was tedious and time-consuming, and I had a feeling the results in several cases were not the smoothest, most nearly ideal patterns possible.

After finishing the program, I decided to try another approach to the patterns. The first idea I had was a standard biomorph program, the first of which was written by Richard Dawkins and described in his book *The Blind Watchmaker* (Dawkins, 1986). So, I wrote the first version of *Pattern Evolver* to produce a family of entirely random patterns of any desired ratio. The user clicked on the pattern that looked the most like what was wanted. In my case I was searching for a smooth gray appearance, but the program could also be used to evolve any pattern the user wanted. When a pattern was chosen by the user, it became the parent of a new family of sixteen patterns, each slightly mutated from the parent. By continuing this process the user could "push" the family from generation to generation up any fitness peak desired: gray, vertical lines, circles, whatever the user wanted. With this program I found that I could evolve smooth gray patterns of any desired ratio in a matter of minutes, while designing the patterns on my own took at least half an hour each.

This program nevertheless seemed inadequate for two reasons. First, I wasn't sure that I obtained the best possible patterns, because searching all the possible patterns still took a long time. There were too many different kinds of nearly smooth grayness that had to be compared to each other. To look at all seventeen family members (the parent and sixteen offspring), decide on the best pattern, and click on it takes at least a few seconds and may take considerably longer if the decision is hard to make. This time adds up quickly. Second, smoothness, or grayness, just felt like such a clearly defined state for a pattern of pixels. Why did the user have to make the choices each generation? The computer could pick the grayest pattern in the family each generation much faster then a human could possibly do manually.

At this point the idea of using an evolutionary algorithm struck me. If I could simply define a mathematical way for the computer to determine how smooth a pattern was, it could pick the pattern with the highest calculated smoothness automatically. My next version of *Pattern Evolver* was capable of picking the smoothest pattern in the family and automatically making it the next parent. I left the user-controlled choices as an option in the program. In order for the computer to pick the grayest pattern, the program evaluates each pattern in a family and assigns it a grayness score. Devising this algorithm was difficult. I used the thirteen perfect patterns as my guides. I was looking for algorithms that would

assign a higher score to the perfect patterns than to any other distribution of pixels for those ratios. It took several attempts to find a satisfactory algorithm. The final result was capable of evolving a smooth-looking gray pattern of almost any ratio in less than a minute or two, and in several cases in less than thirty seconds. A couple of ratios still produce unsmooth gray patterns as the highest scoring, but only very few. Fixing this would be a task of optimizing the grayness calculating algorithm. Overall, the program works well.

Mutation

Each time one pattern from the family is chosen as the next parent, either because of its grayness score (see below) or because the user manually picks it, a new family is generated in the following manner:

- The chosen pattern is copied into the parent pattern, replacing the old parent.
- Each offspring is cloned from the new parent pattern and immediately mutated.
- The new family is redrawn on the screen (if the "Update Family" switch is on. It can be turned off to speed up the program. Of course, redrawing the screen is not necessary for the computer to pick the grayest pattern, only for the user to see what is happening).

This process is repeated each generation as one pattern from the family is chosen as the next parent.

Mutation consists of two steps. First micromutations occur. For every black pixel in the pattern, there is a probability that the pixel will drift one space in a randomly determined empty direction (Figure 3). Each pixel has eight possible directions to move in. Pixels cannot drift onto or over each other. So if a certain pixel has seven neighbors that are black and the program determines that the pixel will move, it will predictably go to the only empty space available. Pixels that are entirely surrounded by black pixels cannot by moved.

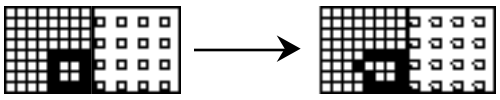

FIGURE 3. — An example of a micromutation. A single pixel is moved one space in a randomly assigned available direction.

Next macromutation occurs. There is a set probability that one of two possible macromutations will occur. Each consists of exchanging entire quarters of the pattern (so a four by four group of pixels is moved together). One consists of switching the two bottom quarters and one consists of switching the two right quarters (Figure 4). Note that macromutation occurs after micromutation.

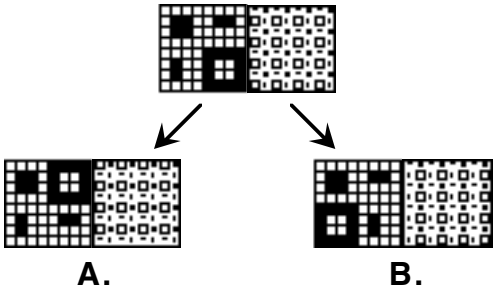

FIGURE 4.— Two examples of macromutations. The top pattern is the original pattern. The bottom patterns are the two possible macromutations. "A" shows that the lower right corner of the pattern has been switched with the top right corner. "B" shows that the lower right corner of the pattern has been switched with the lower left corner.

The probabilities of micro- and macromutations are selected by the user.

Measuring Grayness

The algorithm I devised for evaluating the grayness of a pattern is rather complicated. To begin, the grayness score is set to zero. Then the pattern accumulates points for having an even distribution of black pixels. For each of the sections of the pattern listed below, the number of black pixels is tallied. If the number is exactly the correct proportion for that section, then the maximum number of points is annexed to the total score. If either more or less than the precise proportion of black pixels are present, then fewer points are awarded down to as few as zero. By proportion I mean that in the sections with 32 possible cells (with half the area of the entire pattern), the number of black pixels must equal half the ratio in order to receive the most points.

The sections checked are as follows:

- Check the 8 horizontal halves (8 pixels horizontal x 4 pixels vertical);
  wrap vertically but not horizontally (Figure 5).

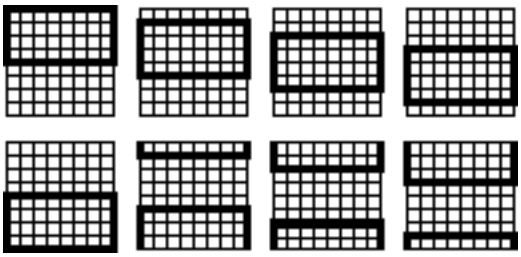

FIGURE 5. — The eight horizontal halves that are checked for pixel distriubution. For each of the eight sections, if the number of black pixels equals exactly half the ratio, then the maximum number of points is annexed to the grayness score. Fewer or more black pixels results in fewer points being annexed. The last three quadrants must be wrapped around to the opposite side of the pattern.

- Check the 8 vertical halves (4 x 8 pixels);
  wrap horizontally but not vertically
- Check the 8 horizontal quarters (8 x 2 pixels);
  wrap vertically but not horizontally
- Check the 8 vertical quarters (2 x 8 pixels);
  wrap horizontally but not vertically
- Check the 64 square quarters (4 x 4 pixels);
  wrap both horizontally and vertically
- Check the 8 horizontal eighths (8 x 1 pixels);
  one pixel thick, no wrapping is necessary
- Check the 8 vertical eighths (1 x 8 pixels);
  one pixel thick, no wrapping necessary
- Check the 64 square sixteenths (2 x 2 pixels);
  wrap both horizontally and vertically

Next points are deducted from the newly accumulated score for black pixels that are near each other. For every black pixel in the pattern, the overall score is decreased a number points based on that pixel's position in relation to the pixel being tested (Figure 6).

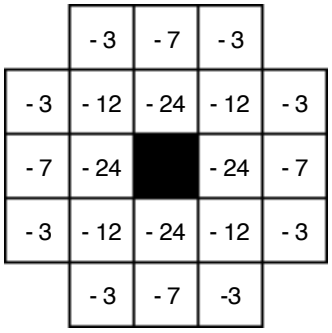

FIGURE 6. — A diagram for determining how many points are deducted for the presence of black pixels around the pixel in the center. This evaluation is carried out on all 64 pixels in the pattern, wrapping around the edges for the necessary pixels.

Based on Figure 6, the maximum number of points that can be lost due to black pixel neighbors of a single black pixel is 196 (the sum of the deductions in Figure 6) if all twenty cells surrounding it have black pixels in them. This does wrap around the edge of the pattern since the pattern will be tiled when it is displayed.

It is important to note that so many points are deducted from higher ratio patterns (31:64 for example) that even the best pattern possible will have a low grayness score simply because the pixels are close together. The scores are thus only useful in comparison to the scores of other patterns of the same ratio.

Adaptive Peaks as a Hindrance and Resilience as a Solution

Pattern Evolver displays the evolutionary phenomenon of adaptive peaks very well. Adaptive peaks in evolving systems were first emphasized by Sewall Wright in his theory of an adaptive landscape (Wright, 1932, 1982); I was amazed when I discovered this effect in *Pattern Evolver*.

If you set up a random family for *Pattern Evolver* and let it try to evolve the grayest pattern possible for that ratio, it will eventually reach a state in which it keeps picking the parent as the grayest pattern for endless generations because the parent is perennially awarded a higher grayness score than any of the offspring. It would appear that *Pattern Evolver* has found the smoothest gray possible. However, when I checked this conclusion, I got a surprise. Remember that I optimized the algorithm for calculating grayness so that the thirteen perfect patterns of the evenly divisible ratios would receive the highest scores for their ratios. By looking at the patterns that emerged for these evenly divisible ratios, I could figure out whether or not *Pattern Evolver* was indeed finding the global peak every time. I

discovered that sometimes *Pattern Evolver* was finding persistent parents for the perfect ratios that were not the perfect patterns shown in Figure 1, that indeed did *not* have higher scores than the perfect patterns from Figure 1, and yet that could not be beat by any of their offspring even if the program ran for an extended period of time.

I soon decided that local adaptive peaks must be my problem. *Pattern Evolver* picks the single pattern in a family that has the highest grayness score, and the next generation is based on that pattern. As a result *Pattern Evolver* will not move downward in fitness (grayness score) from a local maximum so that it can climb up a different, possibly higher peak.

My solution was to create what I called "resilience". I would take any parent that persisted a specified number of generations, add it to a list of likely best patterns, reseed a completely new, randomly chosen family, and try again. In this way a collection of good patterns could be compiled by taking one resilient parent after another and saving them all for the user to choose from.

The necessary resilience steadily increases throughout any single run of the program so that the present parent must be chosen as the next parent an increasingly higher number of generations in a row in order to be added to the list of best grays . In essence, it must have a higher grayness score than an increasingly larger number of offspring to be dubbed a successful gray pattern. Consequently the best grays list first fills up with likely, but not well tested candidates. These best grays are eventually replaced by newer, better grays that have had to remain parents for a longer time (withstanding the grayness scores of a higher number of varying offspring) and thus have better scores overall. Once the best grays list is full (the capacity is nine), any new parent that withstands the resilience test will only be added to the best grays list if its score is higher than the lowest scoring pattern in the best grays list. If this is the case, the previous lowest scoring pattern is removed and the new pattern is inserted into the best grays list in order of grayness score. If the parent withstands the resilience test but its score isn't good enough to get into the best grays list, it is discarded and a counter is annexed.

Every fifth time a parent is discarded , the resilience is increased by five generations, on the assumption that the resilience is too low to evolve good enough patterns to get into the best grays list. By raising the resilience, only parents with better scores will pass the resilience test and thus get into the best grays list.

In this fashion, *Pattern Evolver* both gives the user access to fairly good grays quickly and keeps getting tougher on the patterns as time goes on so that the user receives better and better patterns.

Eventually, the best gray pattern possible for the desired ratio should emerge and not be beaten out of the best grays list, no matter how high the resilience is.

A Sample Run

The following is an example of what happens in a sample run. Some figures show the entire family for that generation, but most just show the pattern from the previous generation chosen as parent of the present generation. Where full families are shown, the pattern at the top of the family is the parent. The other sixteen patterns are the offspring, all based on the parent (with the exception of generation zero which is randomly seeded). Micromutation rate is 20%, and macromutation rate is 20%. For this example, I have chosen the ratio of 11:64, because designing a smooth gray pattern of this ratio manually is exceedingly difficult. Watch as *Pattern Evolver* effortlessly produces a beautiful smooth gray pattern. First, a random family of the desired ratio is generated. Only at this point is there no relation between the pattern in the parent position and the rest of the patterns. From generation one onwards, the sixteen offspring will always be related to the parent.

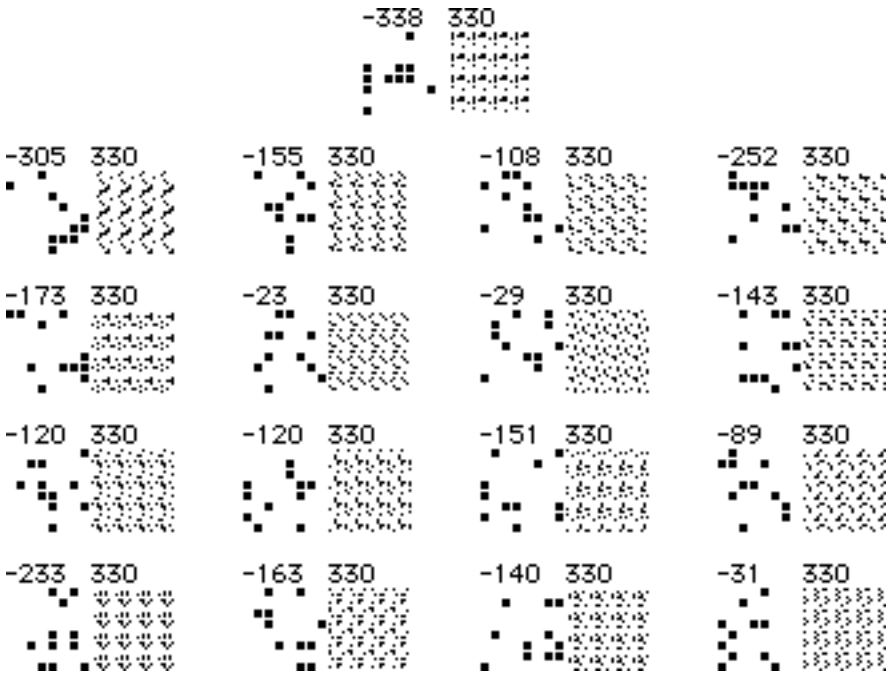

FIGURE 7. — Randomly seeded generation 0. The number to the right above each pattern is the maximum number of points that could be tallied in the first half of the evaluation process, before any points are deducted for black pixels that are close to each other, in other words, the highest score for this ratio. This score is generally not realistically attainable since points will certainly be deducted in even the smoothest pattern at ratios higher than about 4:64 to 6:64. The number to the

left above each pattern is the score that *Pattern Evolver* has assigned to that particular pattern. The pattern chosen from this generation was the one with the highest score: second column, second row. Therefore it is the parent shown for generation 1 in Figure 8.

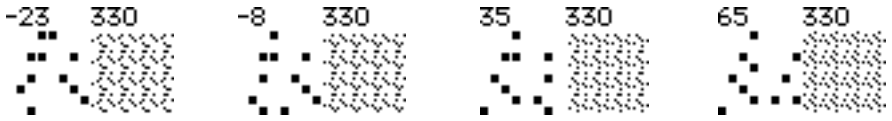

FIGURE 8. — Parents for generations 1, 2, 3, and 4. In only 4 generations (generation 0 is not counted because it is randomly seeded) some improvement in the smoothness is already apparant.

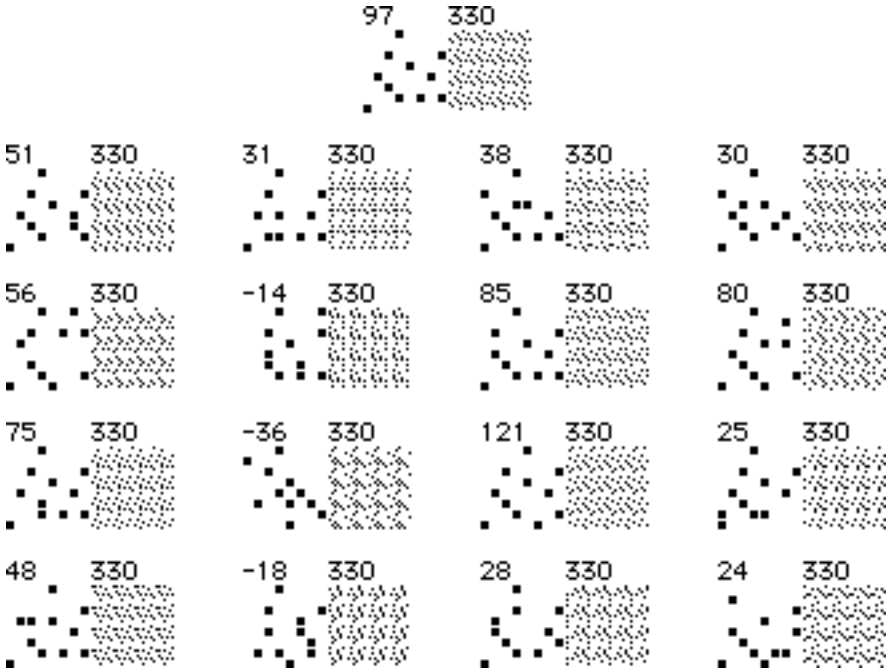

FIGURE 9. — Generation 5

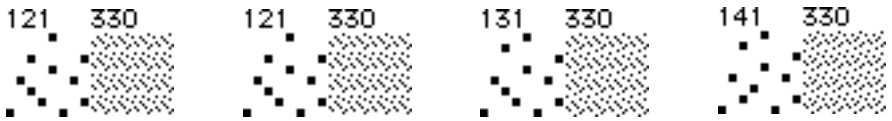

FIGURE 10. — Parents for generations 6, 7, 8, and 9. The parent pattern is beginning to look fairly smooth at this point. Notice that one parent was repeated for a second generation. This pattern would have become a best gray pattern if it had lasted a couple of more generations.

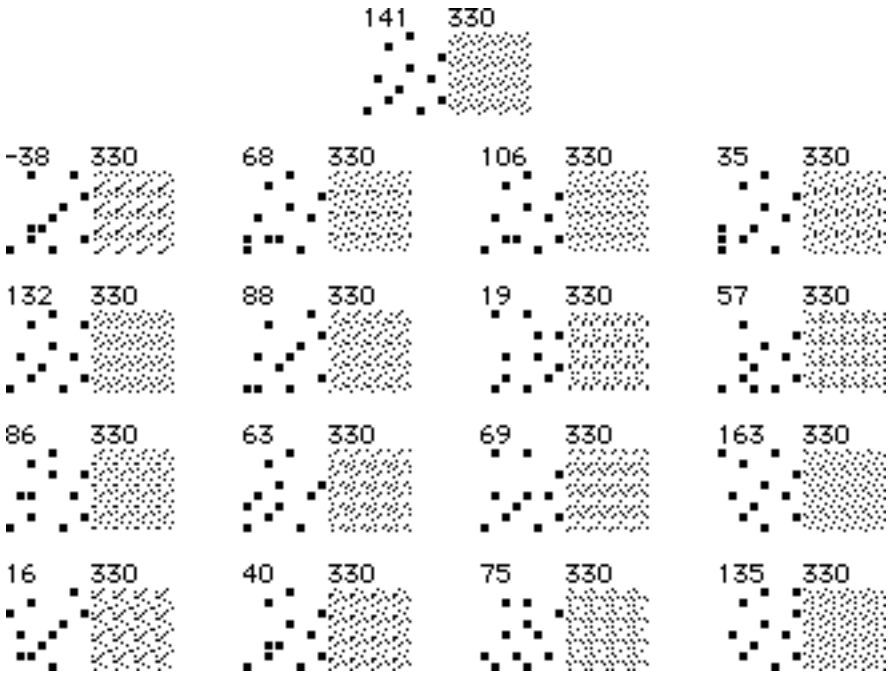

FIGURE 11. — Generation 10

FIGURE 12. — Parents for generations 11, 12,,13, and 14. These parents are all identical because the parent of each family had the highest score and was consequently chosen as parent for the next generation.

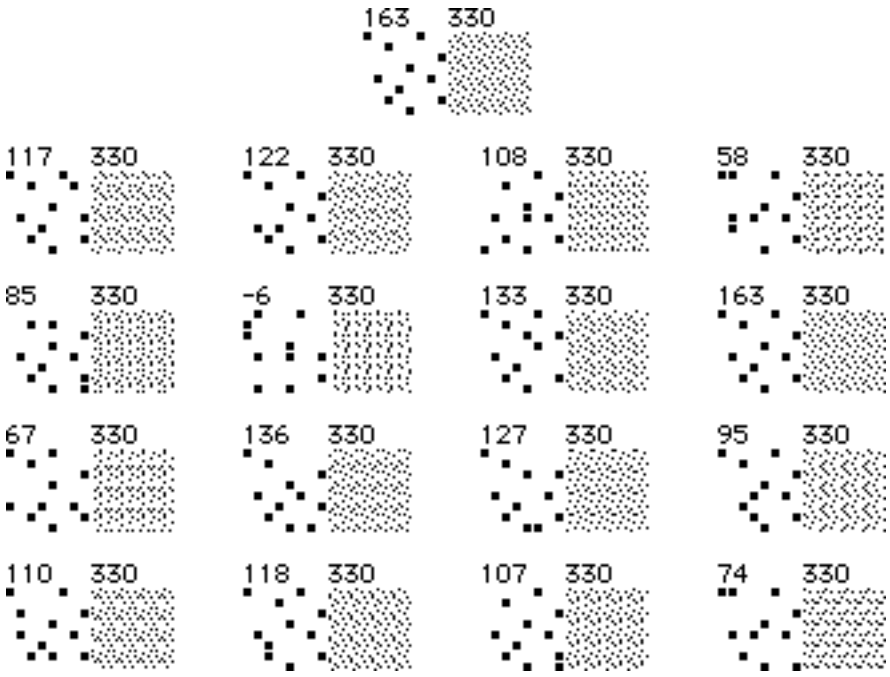

FIGURE 13. — Generation 15

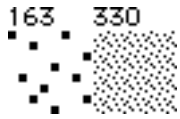

FIGURE 14. — Parent for generation 16. This pattern has passed the first resilience test by being chosen for five sequential generations after becoming parent. It was added to the best grays list, a new family was randomized, and the process was repeeated.

The score of the first parent chosen from the randomly seeded family was -23. The score went up almost every generation and ended up at 163 for the first pattern added to the best grays list, a much better score, and sure enough, the resulting pattern does look much smoother than any from the preceeding generations.

Nevertheless, the resulting pattern (Figure 14) is not perfectly smooth. This is because the necessary resilience starts very low, only five generations. It slowly increases to as high as 1000 if necessary. From the starting point, it took *Pattern Evolver* only seven seconds to come up with the best pattern possible for a ratio of 11:64 (Figure 15). At this point the resilience had increased to twenty.

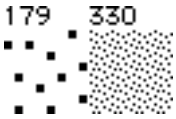

FIGURE 15. — The smoothest gray pattern for a ratio of 11:64 that *Pattern Evolver* could discover. Others with the same score also exist and it may require a little human discretion to choose between the maximally equal scoring patterns. Nevertheless, it is impossible for a human to design a pattern this smooth in the time that *Pattern Evolver* is capable of.

A couple of other patterns with scores of 179 appeared soon afterwards but they all looked similar in grayness (thus the identical scores), and no pattern ever emerged that beat the score of 179, even after the necessary resilience had risen to 45 generations as parent. It is reasonable to conclude that *Pattern Evolver* has found the smoothest possible gray patterns for a ratio of 11:64 pixels. There is most likely no pattern that would yield a score higher that 179 for this ratio using *Pattern Evolver's* grayness calculating algorithm, and there may very well be no pattern that would appear smoother to the human eye.

In seven seconds, *Pattern Evolver* discovered a smooth gray pattern that a human would be lucky to find in thirty minutes or an hour. If you consider that the 65 possible ratios minus the thirteen easy-to-do perfect ratios leaves 52 such patterns, a human could spend a full 24 hours or more completing the entire range of gray patterns and would probably not even do a very good job. *Pattern Evolver* could produce the entire collection in a few minutes.

Local Maxima

That *Pattern Evolver* discovered local adaptive peaks makes it a remarkable program. Local maxima also occur in biological evolution, as first argued by Sewall Wright (Wright, 1932, 1982). In biological systems, the numerous possible combinations of genes presumably have complex patterns of adaptiveness. The combinations of genes that correspond to well adapted individuals are called local maxima. Evolutionary biologists call these local maxima "adaptive peaks" and the topography of adaptive values an "adaptive landscape". It is probable that such a landscape will have several local peaks which may be of comparable adaptiveness. If a population has combinations of genes that cluster around one peak, then successful migration to another peak requires some strategy for traversing the valleys between the peaks. The easiest such strategy is increased mutation, which. In*Pattern Evolver*, mutation is increased either by a higher rate of micromutations or by the presence of macromutations. These two possibilities correspond to two methods of moving to a different peak. In the first possibility,

the population spreads out around the first peak, to the point where certain lineages of the population reach a valley and then advance up another peak. In the second possibility is that a lineage jumps, as a result of a rather large mutation, onto a different peak. This second method is what macromutations in *Pattern Evolver* attempted to replicate.

It is interesting to speculate about when genetic patterns will have a complex adaptive landscape with numerous local peaks. Perhaps if the phenotypes of the genes, whether a tile of pixels or a biological organism, are evaluated by complex and overlapping criteria, then the combinations of individual genes that have high fitness will be nearly unpredictable. The interactions of combinations of genes with a particular environment might be so complex that the maxima cannot be easily specified. In this case, Darwinian evolution, coupled with a process for exploring multiple peaks, can provide a good method for discovering the maxima in the landscape.

Pattern Evolver differs from biological evolution in one important way. The latter has no mechanism for systematically comparing many peaks in the landscape. *Pattern Evolver* picks a new starting location on the landscape every time it randomizes the family and consequently searches as many local peaks across the landscape as possible. The result of this process is that *Pattern Evolver* has a good chance of discovering the global peak simply by trial and error. Biological evolution has no such good fortune. If a population climbs up a local peak, it stands a good chance of being stranded on that peak indefinitely without ever exploring other areas of the landscape. In order to try another peak, a biological population would have to evolve slowly, generation by generation, down the present peak and back up another. Because descending a peak means evolving toward lower adaptiveness, it cannot be realistically accomplished. To jump from peak to peak in a single generation would require both a significant change in the genome and the slight chance that the resulting genome would land, not only on another local slope, but actually higher on the new slope than it currently is on its current local peak. If local peaks are few and far between in a biological landscape, such a change in the genome in a single generation would essentially never happen.

The way Darwinian selection moves patterns in such a landscape will depend both on the shape of the landscape and on the nature of mutation. A flatter landscape, consisting of shallow divides and closely spaced peaks, will make transitions between peaks by chance more likely. A mountainous landscape, with wide, deep valleys, increases the chance that a population will be stranded on a local peak without being able to migrate or jump to another peak.

The nature of mutation will also affect how a population evolves in a landscape. If only minor mutations are possible between generations, then the step size across the landscape is smaller.

Consequently, it is easier to get stranded on a peak, unable to step over a valley to another peak. If rather significant mutations are possible then the possible step size between generations across the landscape is greater, and it is thus easier to cross valleys to other peaks. Macromutations, in this case, might enable a population to explore the landscape more adequately in search of the global peak.

Macromutations, however, are a double-edged sword in evolution. By increasing the chance of jumping from peak to peak, they also increase the chance that a population will jump off the global peak before reaching the summit. In other words, if macromutations were too frequent, a population may not spend the time necessary to perfect its score on one peak before giving up and jumping to another peak.

In *Pattern Evolver* I included two degrees of mutation, micro- and macromutation. Macromutation was added once I realized that I was dealing with local peaks in an adaptive landscape. I recognized that local peaks would hinder the performance of *Pattern Evolver* and thought that if I could induce more dramatic change between a parent and its offspring, then some macromutated offspring might launch the lineage in a new direction, in essence, jumping from one local peak onto the slope of another, higher peak.

Testing the Influence of Macromutations

To see whether or not the method of macrocmutation I devised, by shifting quarters of the pattern, actually increased the chances of finding the global maximum, I ran an experiment. Choosing the perfect ratio of 8:64 black:white pixels, I ran combinations of micro- and macromutations at different resiliences, 1000 times each, and recorded the number of global peaks found for each combination of settings.

The results did not confirm my expectation of significantly better performance with macromutations (Figure 16). Nevertheless, macromutations often did improve the performance to some degree, but not in a simple, linear fashion. My particular method of macromutation seemed to hurt *Pattern Evolver's* performance for certain combinations of micro- and macromutations. Especially when the rate of micromutation was low, zero macromutation generally outperformed low macromutation. Furthermore, the most successful combinations involved both high rates of micro- *and* macromutation. In these cases, macromutations did seem to play some role in discovering the global peak.

It is interesting that medium rates of macromutation produced poorer results than zero and high rates, especially for low rates of micromutation (exhibited most effectively in Figure 16D, as well as 16B, 16C, 16E and 16G). As micromutation increases, the span of the U-shaped pattern becomes smaller,

perhaps because increased micromutation levels the field for macromutation, negating differences that would otherwise be exhibited across various rates of macromutations. Imagine a population randomly walking around on an adaptive landscape. With no macromutations, it can only take small steps and gradually climb up one peak. With high rates of macromutation the population would hop all over the landscape from one slope to another, always slowly edging its way higher but not very quickly or methodically, certainly not up any one peak consistently. If a population jumped off the global peak, it would have a good chance of jumping back later. With an intermediate rate of macromutation, there might be enough jumping between peaks to coax a lineage into jumping off a slope, perhaps even off the global peak, but macromutations might be so few that the lineage never jumps back onto the global slope in a finite period of time. In this way, a medium rate of macromutation would perform worse than no macromutation at all and likewise would perform worse than a high rate of macromutation. I had not foreseen this possibility prior to analyzing the data from the experiment.

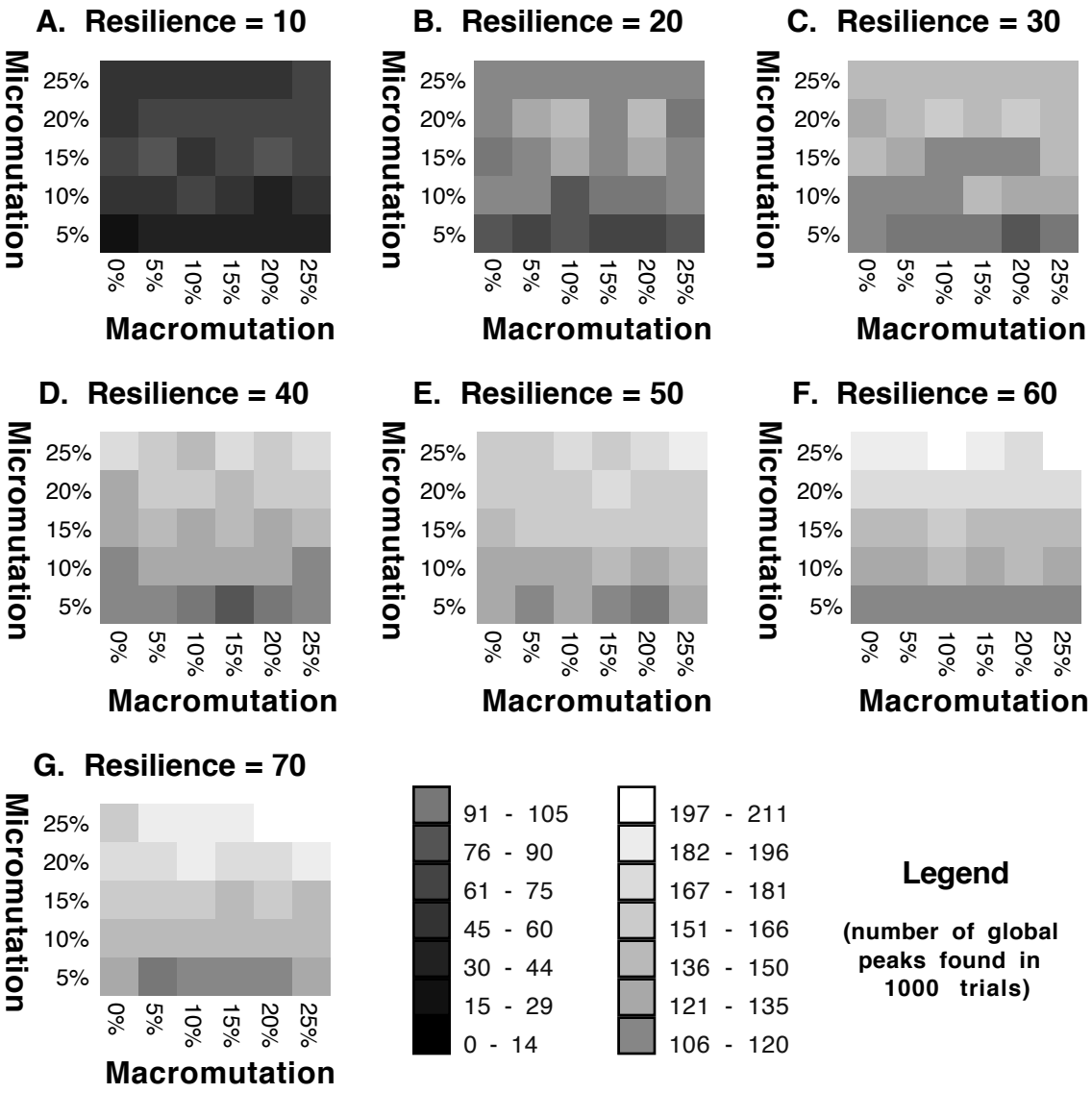

FIGURE 16. — Number of runs per 1000 which discovered the global peak, as a function of rates of micro- and macromutation. It appears that there is little overall improvement in performance associated with increased macromutation. However, there are some interesting patterns present. For low rates of micromutation, it appears that medium rates of macromutation perform worse than both lower and higher rates of macromutation. My explanation is that medium rates of macromutation are just high enough to make lineages jump off of good peaks, but are not high enough to find the global peak. Since such lineages may very well jump off of the global slope but may never jump back, this rate performs worse than both lower and higher rates of macromutation. Lower rates would prevent the lineage from jumping off the global slope, and higher rates would allow the lineage to correct itself if it jumps off the global slope by jumping back.

Another interesting observation is that the number of trials that found global peaks is rather low. The highest number was 211 in 1000 trials (resilience = 70, micromutation = 25% and macromutation = 20% (Figure 16G)). It is possible that with a higher resilience, global peaks would be found more often as each lineage has more time to search the landscape (although performance seems to level off as resilience increases). This low overall performance might be interpreted in several ways. My hunch is that the adaptive landscape associated with *Pattern Evolver* consists of numerous thin-based, spike-shaped peaks with wide valleys between them, as opposed to few slowly inclining peaks with shallow valleys. Such a scenario would make wandering about on the landscape much more difficult. Once a lineage starts up a local peak, the odds of any small or even a large mutation landing on another slope or peak instead of in a valley are very low. If the landscape has numerous inferior local peaks and only one global peak, or a few inferior peaks with very wide bases that consequently funnel lineages then it would be hard to find global peaks, pretty much regardless of mutation rates, both micro- and macro.

Closing Remarks

I set out to solve a nonintuitive problem by using evolutionary means, to recursively take a sample of possible solutions, pick the best solution, and breed that solution with mutations. I later introduced an automatic grayness measuring algorithm into the program so that the computer would solve the problem set forth entirely without human guidance.

The distribution of pixels in gray patterns similar to those discussed in this paper is truly nonintuitive in nature. There is no other way for a human to design such patterns except to move pixels about one by one in a rather unmethodical fashion, almost entirely by trial and error. Such an approach may yield an acceptable solution, yet when nothing less than the ideal solution will suffice and a limited amount of time is available in which to find that solution, the human design approach falls desperately short of acceptable performance. As the number of pixels, N, increases, the number of possible patterns increases extremely fast. For a ratio of just 5:64 black to white pixels, there are over 900 billion possible distributions in an 8 x 8 grid, 64 x 63 x 62 x 61 x 60. The human design approach is thus highly inefficient beyond a rather small number of pixels. Coupling a computer's computational speed with Darwinian evolution trivializes the task to the point where I cannot imagine why people have ever had to design such patterns in the first place. *Pattern Evolver* quickly produces patterns that rival any pattern ever designed by a human, and, in the case of many of the most obscure ratios, they probably surpass those designed by all humans in the past.

How to Obtain Pattern Evolver

Pattern Evolver is available at several sites on the World Wide Web as well as being registered on the major search engines. My *Pattern Evolver* webpage is located at http://www.wam.umd.edu/~keithw/patternEvolver.html. Please e-mail me if you have any questions.

References

DAWKINS, RICHARD. 1986. *The Blind Watchmaker*. W. W. Norton & Company.

WRIGHT, SEWALL. 1932. "The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in Evolution". *Proceedings of the Sixth International Congress of Genetics*. 1 The University of Chicago Press.

WRIGHT, SEWALL. 1982. "Character Change, Speciation, and the Higher Taxa". *Evolution*. 36 The University of Chicago Press.