An assessment of the Building Block Hypothesis as Applied to the Automatic Generation of a LOGO Command Sequence for the Turtle Based Reproduction of a Line Drawing

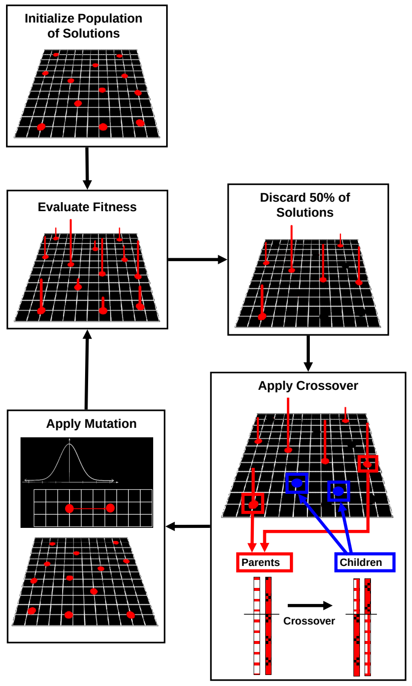
Benjamin J. Bush[1] **and Todd Ebert, Ph.D.**[2]

[1]Department of Mathematics and Statistics, California State University, Long Beach, Long Beach, CA 90840

[2]Department of Computer Engineering and Computer Science, California State University, Long Beach, Long Beach, CA 90840

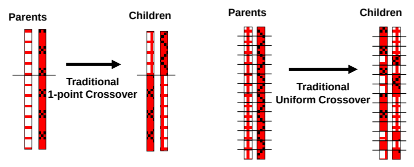
1) EVOLUTIONARY ALGORITHMS:

Recently, computer-based techniques have revolutionized the field of function optimization; the most famous being the biologically inspired evolutionary algorithms. All evolutionary algorithms have a **starting population** of chromosomes (elements of the search space of possible solutions), a **fitness function** which assigns a real value to each chromosome, one or more **selection operators** that eliminate chromosomes based on fitness, one or more **variation operators** that perturb chromosomes, and a **transition function** for computing the next generation from the current population. The basic outline of the evolutionary algorithm used in this study is illustrated below.



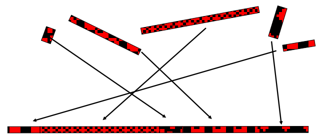
2) THE CROSSOVER VARATION OPERATER:

Inspired by sexual reproduction, the crossover operator is very controversial. Crossover is used to recombine parent chromosomes to form child chromosomes. **Traditionally,** 2 highly fit parent chromosomes are selected for recombination. In **single point crossover**, a crossover index is randomly chosen and the subsequences following the crossover point are exchanged. In **uniform crossover,** each gene pair is exchanged with probability ½.

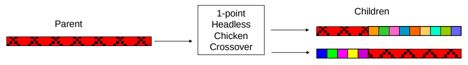


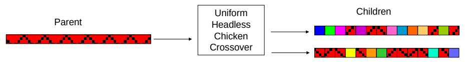
3) THE BUILDING BLOCK HYPOTHESIS:

There is much speculation on the source of crossover's effectiveness; the most famous being the **Building Block Hypothesis,** which proposes that crossover is effective because it finds and assembles highly fit blocks of code.



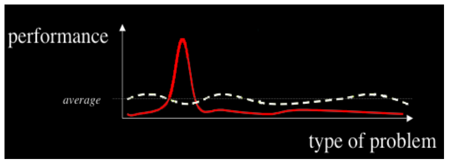
The Hypothesis suggests that single point crossover is more desirable than uniform crossover because the former is more likely to combine blocks without disrupting their linkage (Fogel p77). **Traditionally**, highly fit parent chromosomes are chosen to undergo crossover precisely because they may contain good building blocks. However, several studies argue that traditional crossover performs no better than **Headless Chicken Crossover** (Banzhaf p153, Fogel p156). Headless Chicken Crossover recombines a single parent with a completely randomly generated chromosome; the result is nothing more than a large mutation (fig 5). These studies suggest that crossover may work not by assembling good blocks, but instead functions as a macromutation operator.





4) THE NO FREE LUNCH THEOREM:

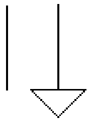
In 1997, Wolpert and Macready shook the evolutionary algorithm community by proving the "no free lunch" theorem. The theorem states that no best algorithm for all optimization problems exists. "Gains in performance on one problem [are] offset by [loss of] performance on [other] problems." (Fogel p134).



Consequently, we must assume that the Building Block Hypothesis and traditional single point crossover are not always applicable. Consequently, empirical studies are now being used to determine the applicability of the Building Block Hypothesis to particular optimization problems, with the hope that in the future, generalizations will be made (Fogel p167). This is one such study.

5) THE LOGO TURTLE:

The LOGO Turtle is a small robot (represented by a triangle) used to create line drawings with simple LOGO commands (Ferrari 2002, p442). For example, the command sequence **forward 50, right 90, penUp, forward 25, right 90, penDown, forward 50** results in the following:



Since line drawings may be stored as bitmaps (Boolean matrices), a turtle can be represented as a function *t*:

$$t : C \to B$$

Where *C* is the set of all LOGO command sequences and *B* is the set of all bitmaps.

6) THE ELTRUT PROBLEM:

Given a master bitmap M, find a command sequence *c* such that *t*(*c*) and M are as visually similar as possible. In other words, find a member of the preimage of M. We call this problem **Eltrut** (turtle spelled backwards). We represent commands as ordered triples: pen status (up or down), mode (turn or move), and magnitude (in units or degrees), where units are integers that vary over
[-20, 20].

7) FITNESS FUNCTION:

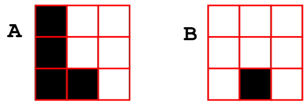
At the heart of every evolutionary algorithm is a **fitness function,** which evaluates the quality of a chromosome. For Eltrut, a "good" chromosome *c* must meet 2 requirements: First, *t*(*c*) must be visually similar to the master bitmap. Second, a turtle running c should not spend time outside the drawing region. To address the first requirement, we define a comparison function between two bitmaps (Boolean matrices):

$$\partial(A,B) = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, h(i,j,B)$$

where *h(i,j,B)* is the smallest Euclidean distance between (*i* , *j*) and the any black pixel in B:

$$h(i,j,B) = MIN\{\sqrt{(x-i)^2+(y-j)^2} \mid B_{xy} = 1\}$$

For example, consider the following 3×3 bitmaps *A* and *B*:



$$\partial(A,B) = \sqrt{(2-1)^2+(1-3)^2} + \sqrt{(2-1)^2+(1-2)^2}$$
$$+\sqrt{(2-1)^2+(1-1)^2} + \sqrt{(2-2)^2+(1-1)^2} \approx 4.65$$
$$\partial(B,A) = \sqrt{(2-2)\ \ +(1-1)^2} = 0$$

A small value of ∂ implies that wherever a black pixel can be found in A, the corresponding pixel in B has a black pixel nearby. However, ∂ is not symmetric. A symmetric function is needed to approximate the notion of visual similarity, so we define the distance d between two bitmaps A and B as:

$$d(A,B) = MAX\{\partial(A,B), \partial(B,A)\}$$

To address the second requirement, a penalty must be applied if a turtle running the chromosome spends any time outside the drawing region. For any *n*×*n* drawing region and any chromosome *c* of length *λ*, let $p(\,(c_1, c_2, \dots, c_\lambda)\,) = (\,(p_{1x}, p_{1y}), (p_{2x}, p_{2y}), \dots, (p_{\lambda x}, p_{\lambda y})\,)$ such that (p_{ix}, p_{iy}) is the position of the turtle after executing commands c_1 through c_i. Then the penalty function is defined as:

$$\Omega(c) = \sum_{i=1}^{l} o(p_i(c))^2$$

where

$$o(p_i) = \begin{cases} \sqrt{|p_{ix}-n|^2 + |p_{iy}-n|^2} & if \quad |p_{ix}| > n \quad or \quad |p_{iy}| > n \\ 0 & otherwise \end{cases}$$

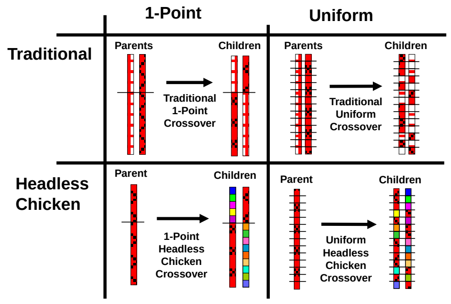
Higher values of Ω and *d* both correspond with lower fitness. For our purposes, actual fitness values are unimportant; relative fitness provides sufficient information for our selection operator. We conveniently define our fitness function such that all fitness values fall within (0,1] as follows:

$$F(c) = \frac{1}{d(t(c),M) + \Omega(c) + 1}$$

where *M* is the master bitmap and *t* is the turtle function described in section 5.

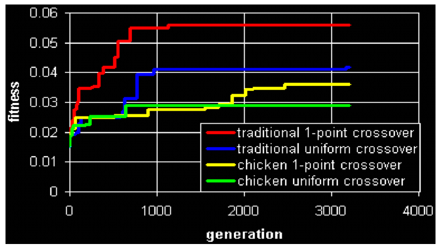
8) METHODS:

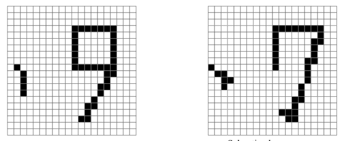
Does the Building Block Hypothesis hold for the Eltrut problem? To investigate, we compared the performance of 4 different crossover techniques: traditional (2 fit parents) single point crossover, traditional uniform crossover, headless chicken (1 fit parent, 1 randomly generated parent) single point crossover, and headless chicken uniform crossover.


1-Point
Uniform
Traditional
Headless Chicken
Parents
Children
Traditional 1-Point Crossover
Traditional Uniform Crossover
Parent
1-Point Headless Chicken Crossover
Uniform Headless Chicken Crossover

9) RESULTS:

Across all comparisons, traditional crossover performed better than headless chicken crossover, and single point crossover performed better than uniform crossover. The algorithm using traditional single point crossover performed significantly better than any other implementation, but unfortunately stagnated at a suboptimal point.





10) INTERPRETATION:

The data supports the conclusion that there are indeed building blocks within Eltrut. The superiority of traditional crossover over headless chicken crossover suggests that crossover is assembling building blocks, not simply acting as macromutation operator. The superiority of single point crossover over uniform crossover suggests that an advantage is gained when blocks are preserved. Failing to use either the traditional or single point techniques results in a performance drop. This suggests that building blocks are very important in the Eltrut problem.

11) REFERENCES:

Fogel, David B. (2000) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, New York, IEEE Press
Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D., (1998) *Genetic Programming—An Introduction: On the Automatic Evolution of Computer Programs and It's Applications*, San Diego, Morgan Kaufmann Publishers, Inc.
Ferrari, M., Ferrari, G. and Hempel, R., (2002) *Building Robots with LEGO Mindstorms*, Rockland, MA, Syngress Publishing.