Solving the Eltrut Problem with Hybrid Evolutionary Algorithms

Benjamin Bush
California State University, Los Angeles
5151 State University Drive
Los Angeles, California 90032
+1-(323) 343-3000
Nebson@gmail.com

ABSTRACT

The Eltrut problem is an optimization problem concerning the educational robot known as the LOGO turtle. In this study, I describe my formulation of the Eltrut problem and my attempts to solve it using evolutionary computing techniques. Motivation is given for the use of hybrid evolutionary algorithms, based in part on the consequences of the No Free Lunch theorem. The fitness landscapes that emerge while using hybrid evolutionary algorithms are compared to the more naïve simple algorithms that were initially used. Preliminary results are presented.

Categories and Subject Descriptors

I.2.8 **[Artificial Intelligence]**: Problem Solving, Control Methods, Search

General Terms

Theory, Algorithms

Keywords

Evolutionary algorithms, hybrid evolutionary algorithms, Eltrut, Turtle graphics, LOGO Turtle, No Free Lunch theorem.

1. INTRODUCTION

In the sections that follow, I describe my formulation of the Eltrut problem and my attempts to solve it using evolutionary computing techniques. In doing so, I have kept in mind the hard learned lessons gleaned from the No Free Lunch (NFL) theorem [4], which states that no search algorithm can perform better than any other search algorithm when performance is evaluated over the space of all possible optimization problems, so that for any given

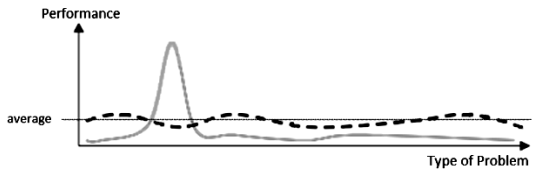

Figure 1. No Free Lunch

optimization algorithm, "gains in performance on one problem [are] offset by [loss in] performance on [other] problems." [2].

The NFL theorem illustrates that if knowledge of an optimization problem is not incorporated into a search algorithm, then there are no assurances that the algorithm will be effective [1]. When a researcher attempts to use an evolutionary algorithm to solve an optimization problem, he or she is working under the assumption that "points in the search space with high fitness are indicators of the location of points in the search space with higher fitness" [5]. It is also assumed that the solution representation is such that "nearby genotypes will have nearby phenotypes, and nearby phenotypes will have similar fitness values"[6].

When a search space does not satisfy these conditions (as is all too often the case), the researcher may attempt to reengineer the solution representation in a way that allows the desired properties to emerge. However, this is often a very difficult task. Enter hybrid evolutionary algorithms.

The term "hybrid algorithm" refers to an algorithm that is formed by combining 2 or more algorithms. For example, a problem can be split into 2 subproblems, the first of which is tackled by an evolutionary algorithm, after which a second algorithm takes over to finishes the job [7]. It is also common to embed non-evolutionary search algorithms, such as exhaustive local search, directly into the operators used in the evolutionary framework [7]. Hybrid approaches such as these grant the researcher greater power and flexibility to mold the search spaces he or she is working with, allowing the aforementioned desired properties to emerge. This is precisely the methodology used in this study.

2. THE LOGO TURTLE

The LOGO Turtle is a small educational 2-wheeled mobile robot. It has the ability to move forward, backward, or rotate about its center. In addition, the Turtle features a motorized pen which can be lowered so that the Turtle leaves a trail on the ground as it moves [3].

In practice, a real, physical robotic Turtle is rarely used. Instead, the Turtle is usually simulated on a computer, where it is usually depicted as a small isosceles triangle. The simulated Turtle draws line segments on the screen. To control the Turtle, users can write simple LOGO command sequences, such as:

```
forward 40, right 90, forward 40, right 90,
penUp, forward 20, penDown, forward 20
```

The preceding command sequence results in the monochrome bitmap image shown in Figure 2.

Figure 2. Turtle output.

On computers, monochrome bitmap images are stored as binary matrices, where "0" corresponds to a white pixel and "1" corresponds to a black pixel. The terms "bitmap" and "binary matrix" will be used interchangeably for the remainder of this paper. With this in mind, note that a turtle can be abstractly thought of as a function

$$t: C \rightarrow B$$

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

3. THE ELTRUT PROBLEM

Suppose one is given a goal bitmap that needs to be reproduced as accurately as possible using the LOGO Turtle, and that the reproduction process needs be done in a reasonable amount of time (using only k Turtle moves, for example). What LOGO command sequence must be given to the Turtle so that the Turtle will output a bitmap that is "close" to the goal bitmap? It is assumed that the goal bitmap contains a finite number of discrete curves, such as a line drawing or sketch.

More formally, Eltrut can be stated as follows: Given a goal bitmap $\boldsymbol{G}$, find a command sequence c such that the Hamming distance (see Figure 3) between $t(c)$ and $\boldsymbol{G}$ is minimized. We call this problem Eltrut (turtle spelled backwards) because it is essentially an attempt to find the inverse image of $\boldsymbol{G}$ under the Turtle function.

Bitmap **A**			Bitmap **B**		
1	**1**	0	1	**0**	0
1	**1**	0	1	**0**	0
1	**1**	0	1	**0**	0

Hamming(**A**,**B**) = 3

Figure 3. Hamming distance for bitmaps.

4. A NAÏVE ELTRUT SOLVER

A naïve approach to solving the Eltrut problem is to use an evolutionary algorithm in which each individual in the population is a k-tuple of LOGO commands $(m_1, m_2, \ldots, m_k)$. The individual's fitness is then found by feeding the commands to the Turtle and then comparing the resulting output to the goal bitmap using the Hamming distance.

In my experiments, however, I have found this approach to be problematic: the solution representation used here inherently leads to a rough fitness function, in which a small change in the individual's genotype leads to a large change in the individual's phenotype (and fitness). To see why, consider an individual who's phenotype is the bitmap with the long "zig zag" type curve illustrated in Figure 4 (left). As the Turtle executes the command sequence associated with this bitmap, it sweeps through each "zig" and "zag", one after another. But suppose now that at some early point in the command sequence, a mutation occurs which causes the Turtle to "zag" instead of "zig." Then all the commands which come after the mutation will now be in a completely different context, as the Turtle is not in the same position that it was in before. This causes a very large phenotypic change, as illustrated in Figure 4 (right).

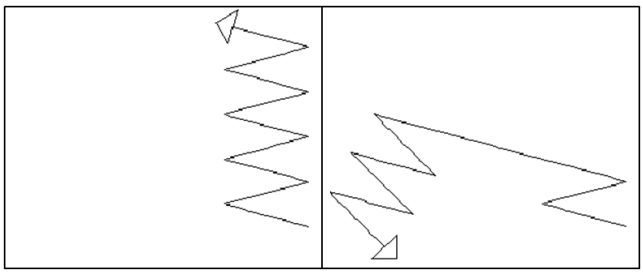

Figure 4. Instability present in the naïve approach.

5. HYBRID ALGORITHM FOR ELTRUT

The proposed algorithm divides the ELTRUT problem into two subproblems. The first subproblem consists of finding the set of k line segments which best approximates the given bitmap image. The second stage consists of finding a route the Turtle should use to traverse these line segments so as to minimize the total travelling distance required to complete the drawing (a generalization of the Travelling Salesman Problem). The set of k line segments and the tour that is used to traverse them together determine a LOGO command sequence, as required. The problem of approximating a bitmap image with a set of k line segments will from now on be referred to as the **Bitmap Approximation by Line Segments** problem, or **BALS**. The problem of finding the most efficient route in which the line segments should be traversed will from now on be referred to as the Traveling Turtle Problem, or TTP. Before discussing each of these subproblems individually, I must briefly discuss the issue of displaying a line segment on the computer screen.

5.1 Bresenham's Line Drawing Algorithm

To display a line segment on a computer screen, it is necessary to map the line segment to a set of pixels on a bitmap image in such a way that properties such as apparent continuity and uniform thickness are achieved. This is a non-trivial task.

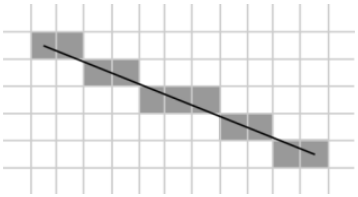

Figure 5. Bresenham's line drawing algorithm.

For our purposes, we will use the popular Bresenham's line drawing algorithm [1] depicted in Figure 5. Henceforth, if l is a line segment, then $Bresenham(l)$ will refer to the set of pixels the line segment is mapped to. Further, if $l_1, l_2, \dots, l_k$ are all line segments, then $Bresenham(l_1, l_2, \dots, l_k)$ will be defined as the union $\bigcup_{i=1}^{k} Bresenham(l_i)$.

5.2 Bitmap Approximation by Line Segments

The BALS problem can be formally stated as follows: given a goal bitmap $\boldsymbol{G}$, find a *k*-tuple of line segments $L = (l_1, l_2, \dots, l_k)$ such that the Hamming distance between $\boldsymbol{G}$ and the bitmap $\boldsymbol{B}$ is minimized, where $\boldsymbol{B}_{x,y} = 1$ if (x, y) is in $Bresenham(L)$, $\boldsymbol{B}_{x,y} = 0$ otherwise. Henceforth, we will call $\boldsymbol{B}$ the "associated bitmap" of the *k*-tuple of line segments $(l_1, l_2, \dots, l_k)$.

5.2.1 A Naïve Approach to Solving BALS

A naïve approach to solving the BALS problem is to use an evolutionary algorithm in which each individual in the population is a *k*-tuple of line segments $(l_1, l_2, \dots, l_k)$, where each line segment l_i is a 4-tuple of integers $(x_{i1}, y_{i1}, x_{i2}, y_{i2})$, representing the coordinates of the endpoints of l_i. Unfortunately, the evolutionary algorithms I ran using this method refused to converge to a good solution in a reasonable amount of time. I speculate that this poor performance is due to the relatively "flat" fitness landscape that results from the use of the Hamming distance: unless some of the line segments in $(l_1, l_2, \dots, l_k)$ are precisely aligned with one of the discrete curves found in the master bitmap, both in position and slope, the Hamming distance between the bitmap associated with $(l_1, l_2, \dots, l_k)$ and $\boldsymbol{G}$ will be very low. Thus the problem is quite "hit or miss", so that little information is available to the algorithm with which to direct the search toward more promising areas of the search space.

5.2.2 A Hybrid Algorithm for Solving BALS

The difficulties inherit in the naïve approach can be addressed by using an indirect solution representation and by incorporating local search methods that exploit problem-specific information. Before discussing the hybrid algorithm in full, we must first introduce the following equivalence relation on the set of all possible line segments: for any pair of line segments l_i and l_j, we write $l_i \equiv l_j$ iff they have the same midpoint and are of the same length. Put differently, two line segments are equivalent iff they are both diameters of the same circle. Thus each equivalence class of line segments is associated with a circle, which from now on will be called an "equivalence circle."

A typical individual in the population of the hybrid algorithm is a *k*-tuple of equivalence circles $(c_1, c_2, \dots, c_k)$, where each equivalence circle c_i is a triple of integers (x_i, y_i, r_i), where (x_i, y_i) are the coordinates of the center of c_i and r_i is the length of the radius of c_i. The typical individual is depicted in Figure 7 (top right).

To evaluate the fitness of the individual, it must first be mapped to a *k*-tuple of line segments $(l_1, l_2, \dots, l_k)$. This is accomplished in two steps, both of which employ exhaustive local search.

In the first step, each equivalence circle c_i is mapped to a translated equivalence circle $\widehat{c_i}$ whose center is the location on the master bitmap $\boldsymbol{G}$ closest to (x_i, y_i) which satisfies the condition $\boldsymbol{G}_{x,y} = 1$, as depicted in Figure 7 (bottom, left). This first step ensures that the midpoints of each of the line segments l_i will lie on one of the discrete curves found in $\boldsymbol{G}$.

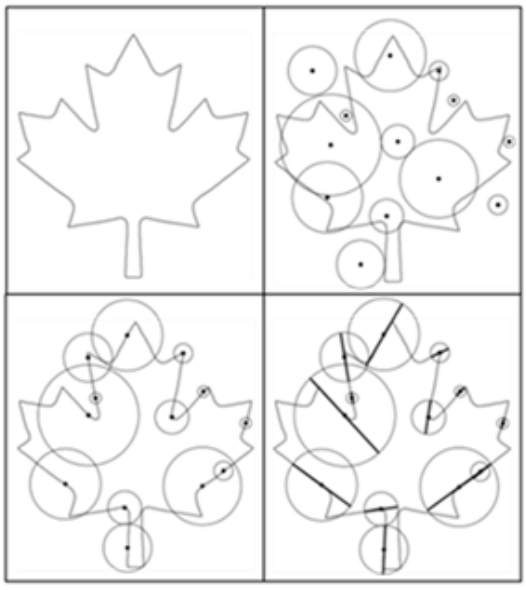

Figure 6. Mapping an individual to a k-tuple of line segments.

In the second step, the best representative line segment from each translated equivalence circle $\widehat{c_i}$ is determined by searching exhaustively through the finite number of diameters of $\widehat{c_i}$ (we only consider line segments with discrete endpoints), where the best line segment is the one which maximizes the number of elements in the set:

$$\{(x, y) : (x, y) \in Bresenham(l_i) \text{ and } \boldsymbol{G}_{x,y} = 1\}$$

This second step ensures that the slopes of each of the line segments l_i will be aligned with the local slope of the discrete curves in $\boldsymbol{G}$ on which their midpoints lie.

And so the burden of finding line segments which precisely align in position and slope with the discrete curves found in $\boldsymbol{G}$ has been lifted by local search techniques, leaving the globally oriented evolutionary framework to answer the more globally oriented question: "in what vicinity should each of the *k* line segments be

placed, and what should the length of each line segment be?" From this point of view, the fitness function is quite smooth, since a small change in an individual's genotype yields a correspondingly small change in the individual's fitness, as depicted in the top two sections of Figure 8. This is a significant improvement over the predicament vexing the naïve approach, in which even a small change in the position of a line segment's endpoints can cause the line segment to become almost completely separated from the discrete curve to which it was previously aligned, as depicted in the bottom two sections of Figure 8.

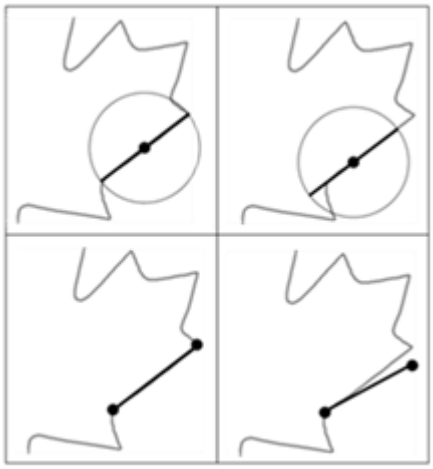

Figure 7. Comparison of hybrid and naïve algorithms

5.3 The Travelling Turtle Problem (TTP)

TTP can be stated formally as follows: given a *k*-tuple of line segments $(l_1, l_2, \ldots, l_k)$, find the tour of minimal length which traverses all of the segments. One can easily verify that TTP is a generalization of the Traveling Salesman Problem (TSP) by noting that if each line segment l_i is of length 0, then each line segment is in fact nothing more than a point and we are left with an instance of TSP. Thus TSP reduces to TTP so that TTP is NP complete (note that this makes Eltrut NP complete as well).

TTL has been previously described and tackled by Xu, Lin and Yang in [8], where they refer to the problem as "segment TSP" and develop a non-evolutionary polynomial time approximation scheme to produce near optimal tours in polynomial time. It would be a simple task to couple their approximation scheme with the BALS solver described in the previous section to form a hybrid algorithm capable of solving the ELTRUT problem. However, the evolutionary algorithm community has had considerable success with TSP and has produced a large body of work on this topic. This leads me to believe that it may be more worthwhile to modify an evolutionary algorithm designed to solve TSP, thereby enabling it to solve TTL. An extensive review of the literature on the topic of using genetic algorithms to solve TSP is resented in [9]. As this project is still a work in progress, I will report on my efforts to solve TTP in the near future.

6. PRELIMINARY RESULTS

As a preliminary test of the effectiveness of the hybrid algorithm described in section 5.2.2, I attempted to approximate the mouse-drawn bitmap shown in Figure 8 (left) with 5 line segments. The evolutionary algorithm had a population of 800 individuals, roulette wheel selection, mutation, but no crossover. Running the algorithm for 1000 generations yielded the suboptimal, yet very encouraging, result shown in Figure 8 (right).

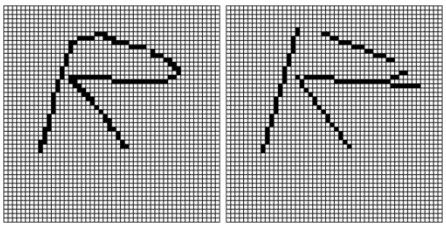

Figure 8. Preliminary results.

7. ACKNOWLEDGMENTS

Thank you Dr. Todd Ebert, Dr. Russ Abott, Dr. David Fogel, and Chuck Esterbrook for providing me with insightful discussion on the topic of evolutionary algorithms.

8. REFERENCES

[1] Bresenham, J. E. Algorithm for computer control of a digital plotter. *Seminal Graphics: Poineering Efforts that Shaped the Field* ACM, New York, NY, 1-6, 1998.

[2] Fogel, D. B. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, New York, IEEE Press, 2000.

[3] Ferrari, M., Ferrari, G. and Hempel, R., *Building Robots with LEGO Mindstorms*, Rockland, MA, Syngress Publishing, 2002.

[4] Wolpert, D.H., and Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82, 1997.

[5] Sharpe, O. Beyond NFL: A few tentative steps, In J. Koza, editor, *Proceedings of the Third Annual Genetic Programming Conference*, 1998.

[6] Sharpe, O. Continuing Beyond NFL: Dissecting Real World Problems. Proceedings of the Genetic and Evolutionary Computation Conference, 1:595-602, Orlando, Florida, USA, 13-17, 1999.

[7] Talbi, E.-G., A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, 8: 541-564, 2002.

[8] XU, J., Zhiyong, L., and Yang, Y. Traveling Salesman Problem of Segments. *International Journal of Computational Geometry & Applications*, Vol. 14 , Nos. 1& 2 19-40, 2004.

[9] Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., and Dizdarevic, S. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*, 13: 129-170, 1999.