BioInspired Computing

Natural Computing Homework

Christopher Smith Zach Pierson

March 16, 2015

Contents

List of Figures

List of Tables

List of Algorithms

Document Preparation and Updates

Current Version [X.X.X]

Prepared By:
Team Member #1
Team Member #2
Team Member #3

Revision History

Date	***Author***	***Version***	***Comments***
2/2/15	*Team Member #1*	*1.0.0*	*Initial version*
3/4/15	*Team Member #3*	*1.1.0*	*Edited version*

1

Evolutionary Algorithms - Text Chapter 3

1.1 Problem 1 2

Implement the various hill-climbing procedures, the simulated annealing, and genetic algorithms to solve the problem exemplified in Section 3.3.3. Use a real valued representation scheme for the candidate solutions (variable x).

By comparing the performance of the algorithms, what can you conclude?

The performance of the simple hill-climbing algorithm depended on where it started. If the point started around 0.1 it would get the the maximum easily. However, if it started at a location where it was on another "hill" it would only find a local maximum instead of the global.

The iterative hill-climbing algorithm was able to find the global maximum since it had many points that it would compute to find local maximums depending on where each point started. It then took the highest one and considered that the global maximum. This approach had better performance overall compared to the standard but could also fall into the same position if one of the points that were initialized were not near the global maximum.

The stochastic hill-climbing

The simulated annealing algorithm

The genetic algorithm compared to the simple and iterative hill-climbing algorithms was a lot better where it almost always found the global maximum. There is still a small probability that it would find only a local maximum. Compared to the stochastic and simulated annealing algorithms it was on par with them since they both were able to find the global maximum every time they were run.

On a small search interval that these different algorithms ran on showed that they were about the same for this case. On larger intervals it would seem that the genetic algorithm could out perform the stochastic hill-climbing and simulated annealing algorithms because of the high randomness involved in the genetic algorithm compared to the other two.

For the simple hill-climbing, try different initial configurations as attempts at finding the global optimum. Was this algorithm Successful? Discuss the sensitivity of all the algorithms in relation to their input parameters.

- Simple hill-climbing and Iterative hill-climbing

  The simple hill-climbing and iterative hill-climbing algorithms use a basic approach of the hill climbing sudo code presented in the book. They are combined since the only difference between the two the iterative has more than 1 point but it is coded in a way to give the program 1 to how many points that someone would like to run the program against. The point(s) are initialized randomly across the search space and they follow whichever direction gives a larger y value after the points are evaluated and each direction is evaluated.

- Stochastic hill-climbing
- Genetic algorithm

  The genetic algorithm initialized the population randomly using a bit string representation. The population is then evaluated and the upper half is kept while the lower half the population is recreated using recombination of the upper half of the population. Then a random bit is flipped in the bit string representation. The new individual is evaluated and then the population is sorted after all the new individuals are evaluated. The best overall in the population by the ending generation is then displayed.

1.2 Problem 7

Determine, using genetic programming (GP), the computer program (S-expression)that produces exactly the outputs presented in Table 3.3 for each value of x. The following hypotheses are given:

- Use only functions with two arguments(binary trees)
- Largest depth allowed for each tree: 4
- Function set: F = { +, *}
- Terminal set: T = { 0, 1, 2, 3, 4, 5, x}.

x	Program output
-10	153
-9	120
-8	91
-7	66
-6	45
-5	28
-4	15
-3	6
-2	1
-1	0
0	3
1	10
2	21
3	36
4	55
5	78
6	105
7	136
8	171
9	210
10	253

The genetic programming algorithm is structured similarly to the genetic algorithm in the sense that it goes through a set number of generations, has a population of polynomials expressed as trees, evaluates the fitness of individuals, and does recombination and mutations on individuals. Instead of a bit string representation this algorithm uses a binary try that stores its value at each node in a character value. s are done by picking a random node in two trees and copying that node and its children as the left side of a new individual for the first tree and then the right side of the new individual for the second tree. Mutation is done by randomly selecting a node in a tree and changing it randomly to be within it's set. So the function

set just switches to the function the value is not. The terminal set will randomly be one of the terminals that the previous value was not.

The results of this program give a consistent result of 0.952381 or 95 percent correct for the fitness of an individual. The equation ((((x + 5) + x) * x + 3) is one of those results. There are other results that are worse as well but that is because the trees representing the equations are randomly generated and potentially recombination and mutation would only allow it to slightly increase since they don't change the height past the deepest tree. Occasionally there is a result of a 100 percent correctness and the following two equations gave that result, (3 + (((x * 2) + (4 + 1)) * x)) and (3 + (x * ((x + 5) + (0 + x)))).

2

Artificial Neural Networks - Text Chapter 4

2.1 Problem 17

Apply the MLP network trained with the backprobagation learning algorithm to solve the character recognition task of Section 4.4.2

Determine a suitable network architecture and then test the resultant network sensitivity to noise in the test data. Test different noise levels, from 5% to 50%

2.1.1 Background

Artificial Neural networks are inspired by the neural networks that are at work within the human body. This problem gives a set of 8 different 12x10 grid of binary pixels. The input patterns should be feed in thought the network as 120 different input options.

2.1.2 Algorithm

As the input layer receives the neural network, weights will be applied and a sigmoid activation function will determine the next hidden layers output. This layer then feeds into the output layer which has a different set of weights.

After the input goes though the ANN, records of the output for the neurons will be recorded. This will be used for the backprobagation of error.

2.1.3 Results

The neural network took a long while to figure out. In fact a few bugs remain: the back propagation of the error didn't seem to work as the delta values where close if not already zero.

3

Swarms - Text Chapter 5

3.1 Problem 1

Write pseudo code for the Simple Ant Colony Optimization (S-ACO) algorithm considering pheromone evaporation, implement it continually, and apply it to solve the TSP. Discuss the results obtained.

Remove the pheromone evaporation term, apply the algorithm to the same problem, and discuss the results obtained

3.1.1 Background

Many biologists and psychologists have studied ants and their behaviors. Among other things, the research that shows how ants forage for food has produced the Ant Colony Optimization algorithm. The main element observed from ant colonies is **pheromone deposition**. This is a trail left by a foraging ant that attracts other ants. This is used in the algorithm to help exploit existing paths that other ants have seemed to take. The more the path is traveled the stronger the pheromone, and the stronger the the pheromone, the more likely an ant will follow the path.

3.1.2 Algorithm

While the pheromone deposition is the main way to exploit a food source or path, this can be problematic as the colony could be trapped in a sub-optimal path. There are two means for overcoming this obstacle: **probabilistic transition rule**, and the **pheromone decay rate**. The probabilistic transition rule gives the ant a probability of choosing a certain path, one that might lead to a more optimal solution. Secondly, the pheromone decay rate allows sub-optimal routes to be washed away by better paths.

3.1.3 Hypothesis

My hypothesis is that by removing the pheromone evaporation rate, the algorithm will be at the mercy of the initial population of edges traveled by ants.

3.1.4 Results

Too much time was spent trying to figure out the Artificial Neural Network that too little time was allocated for this problem.

3.2 Problem 8

Apply the PS algorithm described in Section 5.4.1 to the maximization problem of Example 3.3.3. Compare the relative performance of the PS algorithm with that obtained using a standard genetic algorithm.

The algorithm used for the particle swarm was a slightly modified version from the slides presented in class. Only the bounds of where the function was and the fitness function were changed so the swarm would stay within the bounds of the problem.

The particle swarm algorithm performance compared to standard genetic algorithm was overall better on the small search space. The swarm always got the global maximum like the genetic algorithm but it appeared to always get closer where the ga has a more random element to it and would get the global maximum with some error in the hundredths spot where the swarm's error appeared to be even smaller than that.

4

Immunocomputing - Text Chapter 6

4.1 Problem 1

Use a bone marrow algorithm to define genes for gene libraries to be used to generate the initial population of a genetic algorithm to solve the TSP problem presented in Chapter 3 and Chapter (Figure 6.24). Assume the following structure for the gene libraries:

Gene length $L_g = 4, number of libraries n = 8, and library length (number of genes in each library) L_l = 4. A sone gene from each gene library will be selected, the total chromosome length is L = L_g x n = 4x8 = 32, that corresponds to the implement this bone marrow model to define an initial population of chromosomes to be used in an evolutionary algorithm to solve th$

The bone marrow algorithm creates the gene libraries using stack encoding instead of representing each city with its number. Library one contains genes with the first element allowed to have a value from 0 to 31, second element 0 to 30, third 0-29, and fourth 0-28. This is the case for all genes in the first gene library. In the second it continues the stack encoding with values being 1 less then the previous element in the gene. so library 8 gene values would be 0-3, 0-2, 0-1, and 0 for the last element in the list. There is no need for a repair algorithm in this case since the libraries are created using stack encoding.

Since we didn't wright the project 1 in chapter 3 for the traveling salesman problem this algorithm will be compared to the ACO. The ACO drops ants randomly across the edges between the nodes and then start searching for a path. The bone marrow algorithm will create a path randomly for a individual using a gene from each library. Once the population is initialized the population is evolved using a standard genetic algorithm of keeping the best individuals and recombining based on the top individuals. ACO would scatter the ants again and have them search for another optimal path. The shortest distance found with the bone marrow algorithm was 61.544870 and the path was 28 21 15 17 25 31 29 22 18 2 10 6 1 14 9 8 5 0 4 3 11 12 7 13 16 20 19 26 30 23 27 24.

5

Latex

Big ole grab bag of latex sample code ....

5.1 Some LaTeX

See Figure 5.1. This is a floating figure environment. LaTeX will try to put it close to where it was typeset but will not allow the figure to be split if moving it can not happen. Figures, tables, algorithms and many other floating environments are automatically numbered and placed in the appropriate type of table of contents. You can move these and the numbers will update correctly.

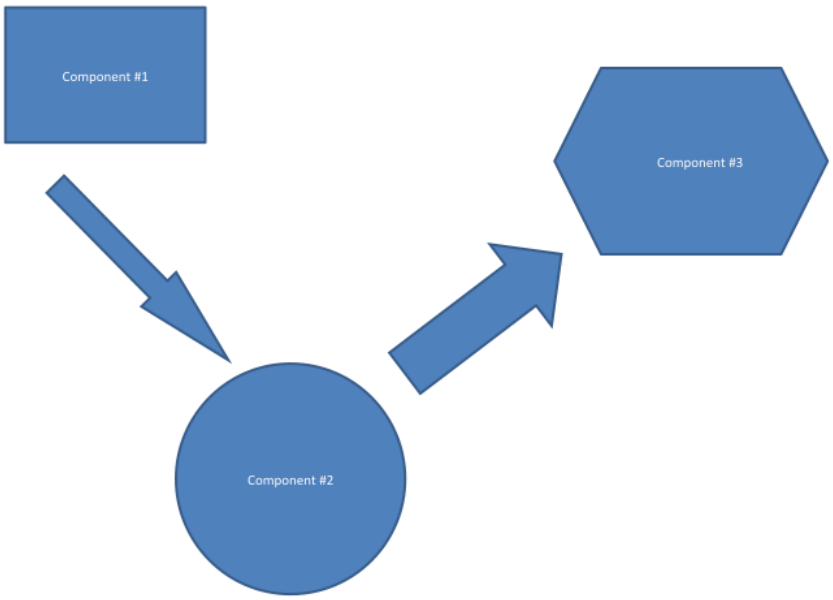


Figure 5.1: A sample figure .... System Diagram

See Table 5.1. This is a floating table environment. LaTeX will try to put it close to where it was typeset but will not allow the table to be split.

Sample bullet list environment:

- According to the all knowing wikipedia, C is an all purpose imperative programming language.

Table 5.1: A sample Table ... some numbers.

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

- Developed between 1969 and 1973 by Dennis Ritchie. [With help from Ken Thompson.]
- One of the most influential computer languages.

Sample numbered list:

1. Predictor: Small step in direction $\lambda \in \mathcal{N}(J_G(x))$:
2. Corrector: $y^{k+1} = y^k + (J_G(y^k))^{-1} G(y^k, \lambda)$

5.2 Section#1

Example Section.

5.2.1 Subsection #1

Example subsection.

Subsubsection #1

Because I can. [But I did not assign a color to the font.]

$$\frac{\partial u}{\partial t} = k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{5.1}$$

5.2.2 Code Details

Here is an example code listing:

```
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
```

```
}
```

This code listing is not floating or automatically numbered. If you want auto-numbering, but it in the algorithm environment (not algorithmic however) shown above.

Sample algorithm: Algorithm 1. This algorithm environment is automatically placed - meaning it floats. You don't have to worry about placement or numbering.

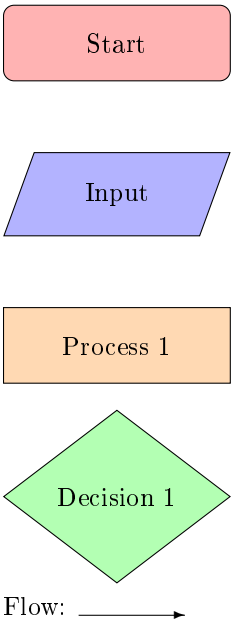
Algorithm 1 Calculate $y = x^n$

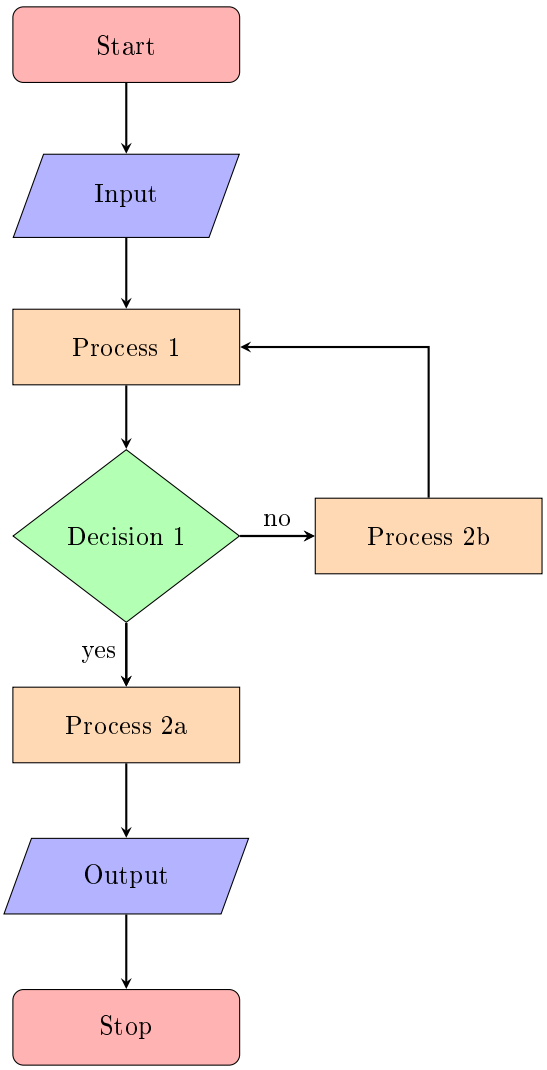
Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \Leftarrow 1$
  if $n < 0$ **then**
    $X \Leftarrow 1/x$
    $N \Leftarrow -n$
  else
    $X \Leftarrow x$
    $N \Leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \Leftarrow X \times X$
      $N \Leftarrow N/2$
    else {N is odd}
      $y \Leftarrow y \times X$
      $N \Leftarrow N - 1$
    end if
  end while

Citations look like [?, ?, ?] and [?, ?, ?]. These are done automatically. Just fill in the database `designrefs.bib` using the same field structure as the other entries. Then pdflatex the document, bibtex the document and pdflatex twice again. The first pdflatex creates requests for bibliography entries. The bibtex extracts and formats the requested entries. The next pdflatex puts them in order and assigns labels. The final pdflatex replaces references in the text with the assigned labels. The bibliography is automatically constructed.

5.3 Section #2

An example of a minipage environment (gets side by side content - like two column mode). Also there is an example of a flow chart using tikz. Flowcharts





More in the sample document at the end.

A

Supporting Materials

This document will contain several appendices used as a way to separate out major component details, logic details, or tables of information. Use of this structure will help keep the document clean, readable, and organized.

B

Code

Insert code here. You can use the listing environment or use doxygen.

LaTeX Example

LaTeX sample file:

B.1 Introduction

This is a sample input file. Comparing it with the output it generates can show you how to produce a simple document of your own.

B.2 Ordinary Text

The ends of words and sentences are marked by spaces. It doesn't matter how many spaces you type; one is as good as 100. The end of a line counts as a space.

One or more blank lines denote the end of a paragraph.

Since any number of consecutive spaces are treated like a single one, the formatting of the input file makes no difference to TeX, but it makes a difference to you. When you use LaTeX, making your input file as easy to read as possible will be a great help as you write your document and when you change it. This sample file shows how you can add comments to your own input file.

Because printing is different from typewriting, there are a number of things that you have to do differently when preparing an input file than if you were just typing the document directly. Quotation marks like "this" have to be handled specially, as do quotes within quotes: " 'this' is what I just wrote, not 'that' ".

Dashes come in three sizes: an intra-word dash, a medium dash for number ranges like 1–2, and a punctuation dash—like this.

A sentence-ending space should be larger than the space between words within a sentence. You sometimes have to type special commands in conjunction with punctuation characters to get this right, as in the following sentence. Gnats, gnus, etc. all begin with G. You should check the spaces after periods when reading your output to make sure you haven't forgotten any special cases. Generating an ellipsis ... with the right spacing around the periods requires a special command.

TeX interprets some common characters as commands, so you must type special commands to generate them. These characters include the following: $ & % # { and }.

In printing, text is emphasized by using an *italic* type style.

A long segment of text can also be emphasized in this way. Text within such a segment given additional emphasis with Roman *type. Italic type loses its ability to emphasize and become simply distracting when used excessively.*

It is sometimes necessary to prevent TeX from breaking a line where it might otherwise do so. This may be at a space, as between the "Mr." and "Jones" in "Mr. Jones", or within a word—especially when the word is a symbol like *itemnum* that makes little sense when hyphenated across lines.

Footnotes[1] pose no problem.

TeX is good at typesetting mathematical formulas like $x - 3y = 7$ or $a_1 > x^{2n}/y^{2n} > x'$. Remember that a letter like x is a formula when it denotes a mathematical symbol, and should be treated as one.

[1] This is an example of a footnote.

B.3 Displayed Text

Text is displayed by indenting it from the left margin. Quotations are commonly displayed. There are short quotations

> This is a short a quotation. It consists of a single paragraph of text. There is no paragraph indentation.

and longer ones.

> This is a longer quotation. It consists of two paragraphs of text. The beginning of each paragraph is indicated by an extra indentation.
>
> This is the second paragraph of the quotation. It is just as dull as the first paragraph.

Another frequently-displayed structure is a list. The following is an example of an *itemized* list.

- This is the first item of an itemized list. Each item in the list is marked with a "tick". The document style determines what kind of tick mark is used.
- This is the second item of the list. It contains another list nested inside it. The inner list is an *enumerated* list.
  1. This is the first item of an enumerated list that is nested within the itemized list.
  2. This is the second item of the inner list. LaTeX allows you to nest lists deeper than you really should.

  This is the rest of the second item of the outer list. It is no more interesting than any other part of the item.
- This is the third item of the list.

You can even display poetry.

> There is an environment for verse
> Whose features some poets will curse.
>
> For instead of making
> Them do *all* line breaking,
> It allows them to put too many words on a line when they'd rather be forced to be terse.

Mathematical formulas may also be displayed. A displayed formula is one-line long; multi-line formulas require special formatting instructions.

$$x' + y^2 = z_i^2$$

Don't start a paragraph with a displayed equation, nor make one a paragraph by itself.

B.4 Build process

To build LaTeX documents you need the latex program. It is free and available on all operating systems. Download and install. Many of us use the TexLive distribution and are very happy with it. You can use a editor and command line or use an IDE. To build this document via command line:

```
alta>  pdflatex SystemTemplate
```

If you change the bib entries, then you need to update the bib files:

```
alta>  pdflatex SystemTemplate
alta>  bibtex SystemTemplate
alta>  pdflatex SystemTemplate
alta>  pdflatex SystemTemplate
```

The template files provided also contain a Makefile, which will make things much easier.

Acknowledgment

Thanks to Leslie Lamport.