

CSC 320: Programming Assignment 6  
Stochastic Search  
Due Monday October 26, 23:59

## Introduction

The purpose of this assignment is to introduce you to some simple stochastic search techniques. The programming is relatively simple but, unlike earlier assignments, you will need to collect data. I have made this last aspect relatively simple, but be sure to take this into consideration when planning your time.

## Preparation:

Download the tarballed file for the assignment from the website, **ps6.tgz**, and unpack it to find the following files:

**genotype.py** - an abstract class for genotypes  
**bitgenotype.py** - an implementation of a bit-based genotype  
**stochasticsearch.py** - an abstract class for all stochastic search methods  
**randmutationhillclimber.py** - an implementation of a 1+1 random mutation hill climber

there are also some files that you don't need to touch:

**plotsearchdata.py** - you shouldn't need to touch anything here  
**Gnuplot/** (a directory) - this contains the python wrapper for gnuplot

## Part 1: Getting your feet wet

- A) Convert randmutationhillclimber to a 1+N hillclimber. This should require only a single change to a single line of code in randmutationhillclimber.
- B) update the code to avoid duplicates in the population. This will require changes to makeNewPop() and createPop(). **Note** - you do not have to avoid repeated states between generations (why not?)
- C) Gather data and produce graphs for the following experiments:
  - performance of your RMHC using the sumList() fitness function across a range Ns
  - performance of your RMHC using the boolToFloat() fitness function across a range of Ns

## Part 2: A Simple Genetic Algorithm

Write a new class GeneticAlgorithm, a subclass of StochasticSearch, which implements a Genetic Algorithm. Specifically:

- A) Write new functions for the genotype class which implement per-locus **mutation** (i.e. each locus changes value with a probability p) and **two-parent single-point crossover**.
- B) makeNewPop() will need to keep the N best parents, and then create new individuals through mutation and crossover. You may use either fitness proportional selection or uniform selection, but justify your choice.
- C) Gather data and produce graphs for the following experiments:

- performance of your GA using the sumList() fitness function across a range of population sizes
- performance of your GA using the boolToFloat() fitness function across a range of population sizes

### Part 3: Function Optimization:

A) Write a new fitness function:

-- calculates fitness as  $= (4*\cos(9x)) + (x-5)^2 + 5$

-- where your 32-bit string has first been converted into a floating point number between 0 and 10

B) Using your GA from Part 2, evolve an optimal (minimal) value to this solution

C) Gather data and produce graphs for a set of experiments from (B) above, over a range of parameters.

### How to plot data:

I have included utilities to make producing graphs of your data easier. These utilities require a \*nix machine with gnuplot installed. This should encompass all of the CS Lab machines.

- if you pass a filename into runSearch() - as in, runsearch("test.txt") it will produce a data text file for you. Subsequent calls to runsearch with the same filename will append data to that file.

- if you are on a computer with gnuplot installed, then you can call plotsearchdata.py (after modifying the last lines) to help produce pretty graphs like this:

```
processDataFile(filename)
```

which will create a file (inelegantly named) tempdata.txt

- if you have multiple data files that you want to compare, then you can call the following:

```
compareDataFiles([list of data files],[list of titles])
```

which will create (inelegantly named) files called tempdata0.txt, tempdata1.txt, tempdata2.txt etc...

Now, to plot you'll have to use **gnuplot**:

### to plot a single course of data:

open gnuplot (type gnuplot at the prompt)

```
gnuplot> set xlabel 'generation'
```

```
gnuplot> set ylabel 'fitness'
```

```
gnuplot> plot 'tempdata.txt' with errorbars title 'mytitle'
```

### to plot and compare multiple data files:

```
gnuplot> plot 'tempdata0.txt' with errorbars title 'value1', 'tempdata1.txt'with errorbars title 'value2'
```

(and more files with the same pattern)

**to save a postscript:**

```
gnuplot> set term postscript enhanced color  
gnuplot> set output 'myoutputfile.ps'  
gnuplot> replot
```

and once you are done:

```
gnuplot> set term x11
```