Title	Randomized Optimization
Due	Mar 15, 2015 11:55 pm
Status	Not Started
Grade Scale	Points (max 100.0)

Instructions

Numbers

The assignment is worth 10% of your final grade.

Why?

The purpose of this project is to explore random search. As always, it is important to realize that understanding an algorithm or technique requires more than reading about that algorithm or even implementing it. One should actually have experience seeing how it behaves under a variety of circumstances.

As such, you will be asked to implement or steal several randomized search algorithms. In addition, you will be asked to exercise your creativity in coming up with problems that exercise the strengths of each.

As always, you may program in any language that you wish (we do prefer java, matlab, Lisp or C++; let us know beforehand if you're going to use something else). In any case *it is your responsibility* to make sure that the code runs on the standard CoC linux boxes. Re-read that last sentence.

Read everything below carefully!

The Problems Given to You

You must implement four local random search algorithms. They are:

1. randomized hill climbing
2. simulated annealing
3. a genetic algorithm
4. MIMIC

You will then use the first three algorithms to find good weights for a neural network. In particular, you will use them instead of backprop for the neural network you used in assignment #1 on at least one of the problems you created for assignment #1. Notice that weights in a neural network are continuous and real-valued instead of discrete so you might want to think a little bit about what it means to apply these sorts of algorithms in such a domain.

The Problems You Give Us

In addition to finding weights for a neural network, you must create three optimization problem domains on your own. For the purpose of this assignment an "optimization problem" is just a *fitness function* one is trying to *maximize* (as opposed to a cost function one is trying to minimize). This doesn't make things easier or harder, but picking one over the other makes things easier for us to grade.

Please note that *the problems you create should be over discrete-valued parameter spaces. Bit strings are preferable*.

The first problem should highlight advantages of your genetic algorithm, the second of simulated annealing, and the third of MIMIC. Be creative and thoughtful. It is not required that the problems be complicated or painful. They can be simple. For example, the 4-peaks and k-color problems are rather straightforward, but illustrate relative strengths rather neatly.

What to Turn In

You must submit via T-Square a tar or zip file named *yourgtaccount*.{zip,tar,tar.gz} that contains a single folder or directory named*yourgtaccount*. That directory in turn contains:

1. a file named *README.txt*containing instructions for running your code
2. your code
3. a file named *yourgtaccount-analysis.pdf* containing your writeup
4. any supporting files you need

The file yourgtaccount-*analysis*.pdf should contain:

- the results you obtained running the algorithms on the networks: why did you get the results you did? what sort of changes might you make to each of those algorithms to improve performance? Feel free to include any supporting graphs or tables. And by "feel free to", of course, I mean "do".

- a description of your optimization problems, and why you feel that they are interesting and exercise the strengths and weaknesses of each approach. Think hard about this.

- analyses of your results. Why did you get the results you did? Compare and contrast the different algorithms. What sort of changes might you make to each of those algorithms to improve performance? How fast were they in terms of wall clock time? Iterations? Which algorithm performed best? How do you define best? Be creative and think of as many questions you can, and as many answers as you can. You know the drill.**Note: Analysis writeup is limited to 10 pages.**

Grading Criteria

At this point you are not surprised to read that you are being graded on your analysis more than anything else. I will refer you to this section from assignment #1 for a more detailed explanation. On the other hand, I will also point out that implementing three of these algorithms is very easy (almost not worth stealing the code, but feel free to do so anyway) but at least one of them requires some time (luckily, there are now versions of this algorithm out there to steal). You should start now.