Simulated Annealing on Bumpy Surfaces

Assignment 2 CS472-S15 Due: Fri Feb 20, 2015 at 5pm PT

Points: 150

Reference Files (these should be live links):

- rand.cpp A simple fast random number generator
- rand.h
- func.cpp The fitness functions for this problem
- func.h

This assignment will get us familiar with the challenges of escaping local optima.

1 The Algorithm

Write a simple simulated annealing algorithm to try to find the global optimum of a function, assuming that we are maximizing the function.

We will try to maximize the function f_r found in the files `func.h` and `func.cpp`. The functions are defined over the 2D real space with $x, y \in [-100, 100]$. In this problem we will assume the genotype to simply be a pair of doubles.

The acceptance function will be:

```
bool accept(double newf, double f, double temp)
{
    return (newf>=f) || (randUnit()<exp((newf-f)/temp));
}
```

where `newf` is the fitness of the new trial point, `f` is the fitness of the current point we are at, and `temp` is the current annealing temperature.

For your mutation function you will try three different mutations:

1. Uniform:

   x is mutated by adding a sample from the uniform random number generator with in the range $[-0.1, 0.1]$ If the new x is outside the acceptable range for x then another random number is drawn (generally one must be careful with this approach that this isn't too frequent an event). y will be mutated by a second sample from the same generator.

2. Normal:

   x is mutated by adding a sample from the normally distributed random number generator with a mean of 0 and a standard deviation of 0.1 . If x is outside the acceptable range for x then another random number is drawn. y will be mutated by a second sample from the same generator in a similar way.

3. Cauchy:

   x is mutated by adding a sample from the Cauchy distributed random number generator with a mean of 0 and a scale 0.1 . If x is outside the acceptable range for x then another random number is drawn. y will be mutated by a second sample from the same generator in a similar way. The Cauchy distribution is a "pathological" distribution in that it samples out in the tail far enough with enough frequency that the integral for computing the variance is undefined. In short it is very "thick tailed" therefore **increasing exploration**.

The algorithm is similar to the one I described in class. You will write a program that will take 3 parameters on the command line. The parameters are:

1. The mutate type: 0 for uniform, 1 for normal, 2 for Cauchy.
2. A starting temperature.
3. A cooling constant $C \in [0, 1]$. If $C = 1$ then the starting temperature will be the temperature used throughout the run. If it is not 1 then every 200 evaluations it will test to see if the temperature needs to be changed[1] Everytime a new fitness is accepted with the accept function then a counter is incremented counting the number of accepted moves in each epoch (resetting the count at each epoch). If the ratio of this count to the number of evaluations in the epoch is greater than .9 then the temperature is lowered by $T_{k+1} = T_k * C$. If the ratio is less than .1 then the temperature is raised: $T_{k+1} = T_k/C$. This helps find a good starting temperature if $C \neq 1$.

Your program should run a test for 100000 evaluations and then report the same numbers you reported for the last assignment:

1. The number of fitness evaluations when it found the best fitness it found in the number of evaluations allowed.
2. The total number of accepted moves made.
3. The x and y that gave that fitness.
4. The best fitness found.

You should run 100 trials of the experiment for a given invocation.

Here is a sketch of the algorithm with details left for you to figure out:

```
for (int trial=1; trial<=numTrials; trial++) {

    bestX = X = random();
    bestf = f = fitness(X);

    while (tries<tryLimit) {

        epochImproves = 0;
        for (int i=0; i<epoch; i++) {
            newX = mutate(X);
            newf = fitness(newX);

            if (accept(newf, f, temp)) {
                epochImproves++;

                X = newX;
                f = newf;

                if (f>bestf) {
                    bestX = X;
                    bestf = f;
                }
```

[1]In fact you can use $C = 1$ as the multiplier and it will do the trick with no special test.

```
            }
        }

        rate = epochImproves/numEvalsInEpoch;
        if (cooling<1) {
            if (rate>.9) temp *= cooling; // make it cooler
            if (rate<.1) temp /= cooling; // make it hotter
        }
    }
    report(results);
}
```

The program will be run with a test script that looks something like what follows. Beware: this script destroys all files with two letter names beginning with z.

```
#!/bin/sh
rm z?
sa 0 100 1   > z1 &
sa 1 100 1   > z2 &
sa 2 100 1   > z3 &

sa 0 100 .99 > z4 &
sa 1 100 .99 > z5 &
sa 2 100 .99 > z6 &

sa 0 .05 1   > za &
sa 1 .05 1   > zb &
sa 2 .05 1   > zc &

sa 0 .05 .99 > zd &
sa 1 .05 .99 > ze &
sa 2 .05 .99 > zf &

wait
catfiles z?
```

2 The Experiments

Write a simple half page report named "yourname.pdf" describing what you observe. Turn in your code and the report in a tar. The tar should include a makefile that will make the program `sa`.

3 Grading this Assignment

I will grade this based first be seeing that your code compiles, runs and returns a sensible answer. I will read the report to make sure it is clear and that you understand what happened when you ran the experiments. Your report needs to be concise and to the point. Supply data to support your observations. You can run your code on the test system then write your report and resubmit them together.

4 Submission

Homework will be submitted as an uncompressed tar file to the homework submission page linked from the main class page. You can submit as many times as you like. The LAST file you submit BEFORE the deadline will be the one graded. For all submissions you will receive email giving you some automated feedback on the unpacking and compiling and running of code and possibly some other things that can be autotested. I will read the results of the runs and the reports you submit.

Have fun.