Project 1A

Tao Zhang
CS 472

February 11, 2014

1 Description of Algorithms

1.1 Hill Climbing (HC)

The hill climbing algorithm is basicly a local search algorithm to find the optimization solution. In my code, I have the main function which will generate a random solution to the function. Then start doing hillclimbing to get the best number.

```
int main()
{
  // generate a random solution "Rand"
   /* start doing hillclimbing algorithm
    * in the following function */
  best = hillclimbing(rand);
  print(best);
}
```

In the HC algorithm function, I will generate 60 neighbors for the previous solution, which plus or minus a interval of $0.01, 0.001 or...$ **for each** x_i**. As the function showed blow, it will keep running the loop until we get the best answer, and return to the main function.**

```
double hillclimbing()
{
  currentEval = evaluate(rand)
  bestEval = currentEval;
  loop do
    generate 60 neighbors
    for (all neighbors)
      evaluate(neighbors)
      compare and get the best Eval
```

```
    if(bestNeighborEval > currentEval)
      return currentEval

    currentEval = bestNeighborEval
    currentSolution = bestNeighbor
}
```

1.2 Simulated Annealing (SA)

The simulated annealing algorithm will do the global search to find the optimization solution. In my code, the main function is similar to hillclimbing, generate a random solution and put it into the annealing function to run a certain times to get the best solution.

In the SA algorithm, we have 100/i running times. Each time we get a randomNeighbor, compare with the best value. If it's better, replace it to best, otherwise keep looping or we have a probability to change our position (solution) in order to do a global search

```
double SA()
{
  s;      // current solution
  sbest;  // best solution so far
  snext;  // next solution

  e = evaluate(scurrent);  // current value
  ebest;  // best value
  enext;  // next value
  for T = 100 to 0 step i    // I tried i as 0.1, 0.001 ...0.000001)
    getRandomNeighbor;
    evaluate randomNeighbor;
    if(ebest > enext)
      ebest <- enext
    else if(P(e, enext, T) > random( 0 - 1))
      e <- enext


  return ebest;
}
```

2 Results

2.1 Hill Climbing for Sphere

HC Sphere	1	2	3
Interval	0.01	0.001	0.0001
Running Times	264840	3703380	41928240
RandomEval	250.592000	240.072000	308.152000
BestEval	0.000000	0.000000	0.000000
real time	0m0.094s	0m1.216s1	0m13.721s

As the data listed in the table above, I did get the correct best value for sphere function

2.2 Hill Climbing for Schwefel

HC Schwefel	1	2	3
Interval	0.01	0.01	0.01
Running Times	2771820	1625040	2129580
RandomEval	12377.579690	12104.257655	12523.885194
BestEval	9413.904662	10623.427370	10795.706182
real time	0m9.470s	0m5.622s	0m7.461s

The HC algorithm can't reach the best evaluation for Schwefel function as it is a local search.

2.3 Simulated Annealing for Sphere

SA Sphere	1	2	3	4
Interval of T	0.1	0.001	10^{-5}	10^{-7}
Running Times	1000	10^5	10^7	10^9
Change times due to P	958	94748	9469379	946966000
RandomEval	314.072000	236.6320005	278.632000	272.872000
BestEval	300.222689	178.742268	135.118743	89.081698
real time	0m0.006s	0m0.111s	0m10.986s	18m39.961s

My results doesn't have the correct best value. However, after analyzing the data, it could be found that as the running times keep going higher, the result is getting better and better. In short, I assume that I will get the correct best value if I have running times of 10^10 for this algorithm. But it already took me 18 mins to run 10^9 times, I don't want to waste hours running 10^{10} times.

2.4 Simulated Annealing for Schwefel

SA Schwefel	**1**	**2**	**3**
Interval of T	0.01	10^{-4}	10^{-6}
Running Times	10^4	10^6	10^8
Change times due to P	2941	280349	27002770
RandomEval	14665.118370	13208.873159	13004.752798
BestEval	14483.115589	13128.795451	12953.802056
real time	0m0.055s	0m10.986s	7m23.986s

The same as SA for Sphere, in limited running times, my coding still can't reach the best solution.

3 Conclusion

My hill climbing algorithm should succeed to finish the task, while the simulated annealing is not. After generating the data for SA, I believe there are some problems on the part of "generating a random neighbor" and also the part of "$P(e, enext, T) > random(0-1)$". Therefore, I may keep doing some research on how to generate a better neighbor and on the comparison of probability.