Week 9 Homework

Jeff Baggett

10/30/2016

We'll define several functions for you and then you'll use the optimization tools introduced in the presentation to try to find (approximately) optimal solutions to the problems detailed below. You should include your solutions right in this document and "knit" it into a Word document (just like you did for DS705 assignments). When you're done, submit both the Word document and this .rmd file to the D2L dropbox.

## Problem 1 - Redistricting with a Genetic Algorithm

Solve problem 13.10-6. Install the 'gramEvol' package to get access to a genetic algorithm that uses integer encoding called GeneticAlg.int(). Use that algorithm to solve this problem. You'll have to read the documentation to figure out how to use the algorithm.

In the block below add your code to use the genetic algorithm. Either experiment with different random number seeds or use a for loop to conduct the optimization many times to find the best solution you can.

# you can delete the lines below, they are just there to show you how the functions work  
x = c(1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8); # a sample assignment  
(demrepFit(x)) # extra parentheses force R to print the result

## [1] 37

# after you've found a solution, you can inspect the results using  
(distAssign(x))

## Dem Rep Tot Win  
## 1 235 137 372 FALSE  
## 2 167 141 308 FALSE  
## 3 147 166 313 TRUE  
## 4 62 105 167 TRUE  
## 5 174 185 359 TRUE  
## 6 83 169 252 TRUE  
## 7 141 137 278 FALSE  
## 8 146 147 293 TRUE  
## 9 98 62 160 FALSE  
## 10 66 72 138 TRUE

# for this x Republicans win 6 districts, but the constraints are violated

Make sure you print out your best solution. How many districts do Republicans win with your solution? (I don't think Republicans can win all 10 in this example, but they can get close.)

## Problem 2 - TSP with a Genetic Algorithm

Use the ga() function with permutation encoding from the 'GA' package to approximate a solution to this 48 city TSP problem. Try different random number seeds and report the best result you can find. Copy the code from saTSP.R to make a graph of the tour. Your fitness function is tspFitness().

## [1] 2.980537e-05

## Problem 3 - TSP with Simulated Annealing

Modify the code in saTSP.R and include it below to approximate an optimal tour for the 48 city TSP problem in Problem 2. Include a graph of the best tour you are able to find.

How do the results compare to those achieved using a genetic algorithm? Is one algorithm significantly more efficient than the other? Compare the total tour distances produced by the two algorithms, does one algorithm consistently produce better results?

## Problem 4 - Comparing Algorithms for a 30 dimensional Rastrigin function

The 30 dimensional Rastrigin function is considered very difficult to optimize. We know that the global minimum value of 0 occurs at the origin. For this problem you should compare the performance of Naive Multistart, the Genetic Algorithm plus local search, and the Simulated Annealing algorithm GenSA() from the 'GenSA' package. If you can get it to work, then also try the mlsl() function in the 'nloptr' package as it should work considerably better than Naive Multistart.

This is a somewhat open problem, but at the very least you should try each algorithm multiple times (possibly in for loop) and report on which algorithms are most efficient (fewest function calls) and which are most reliable (able to consistently identify the global minimum). Experiment with the algorithm parameters (population size, number of iterations of local search, etc.) You'll likely have to increase population sizes and the maximum number of iterations to successfully solve the 30 dimensional problem. Look at the source code in the presentation .Rmd file included in the download packet for guidance in setting up your algorithms.

# your code goes in this block  
dimension = 30;  
lower = rep(-5.12,dimension); upper = rep(5.12,dimension);   
x0 = runif(dimension,-5.12,5.12)

Discuss your algorithm comparison here: