Homework 9 Submission

Your Name Here

## Instructions

Download files at <https://github.com/DataScienceUWL/DS775>. The files for this HW are in Homeworks/Lesson09\_Download.

Complete the following problems and add your solutions to this word document. This HW works best if you just work directly on the R markdown file included in the download folder and knit it to produce your solutions. As in past weeks your submission should be a complete reference document. The tools this week are all in R.

**Python.** If you’d like to explore using Python to solve any of these pr oblems instead of OPL or Excel that’s fine (in fact I’d like it since I’m planning to include python in the next iteration of this course). Include your source code and output in this document and upload the corresponding .py or .ipynb file to the dropbox. If you decide to do the entire assignment in Python, feel free to submit a Jupyter notebook.

### Getting Help:

Post questions on Piazza. Always include the problem number in your subject line, e.g. “HW 1.3” so that it’s easy to search and find relevant posts. If your post would reveal a significant portion of a solution then make it a private post and tell us if it is OK to share it publically and we can judge whether or not to share it.

### What to hand in:

* Your completed .rmd file and the knitted .docx file.
* Please delete instructions and examples and include only your solutions and narration.

### The spirit of this HW

Read about the concepts behind these algorithms in the text (share any other resources you find on Piazza ), then tinker with the algorithms to try to find good answers to the problems below. We want you to get a feel for what it’s like to solve difficult optimization problems where it’s pretty much impossible to guarantee that you’ve got the best answer.

## Homework 9

## HW 9.1 - Textbook 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. Don’t knit with a for loop repeatedly as it will take a long time, rather use the for loop to find a good random number seed then use that seed to do one optimization in your final document. You may also wish to play with the population size (larger is better, but slower) and the number of iterations.

# 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))

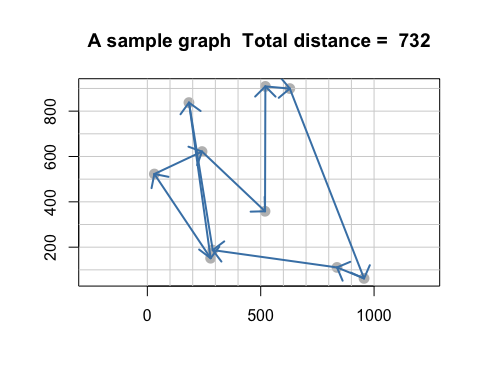
## Dem Rep Tot Win  
## 1 235 137 372 FALSE  
## 2 167 141 308 FALSE  
## 3 147 166 313 TRUE  
## 4 62 105 167 TRUE  
## 5 174 185 359 TRUE  
## 6 83 169 252 TRUE  
## 7 141 137 278 FALSE  
## 8 146 147 293 TRUE  
## 9 98 62 160 FALSE  
## 10 66 72 138 TRUE

# 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.)

## HW 9.2 - 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. The plotTour function below can be used to visualize your tours. Here is the tour plotting function and an example plot. Feel free to delete the example plot from your HW submission.

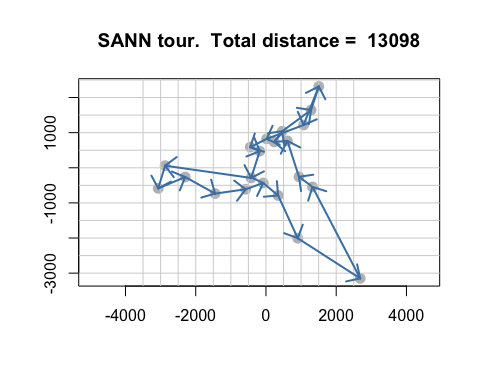


The fitness function below is is tspFitness(). Because GA() actually maximizes we take the reciprocal of the length as the objective so that finding a max corresponds to finding a short tour.

You may have to play with the optimization parameters a bit to find a good solution. popSize, pmutation, and pcrossover are good parameters to experiment with.

## HW 9.3 - TSP with Simulated Annealing

The example below shows how to apply simulated annealing to a problem of optimizing a tour between European cities. You can play with the parameters and random number seeds to improve the European tour (you don’t to need to submit this part). Next, modify the codeq to solve the 48 city TSP from Problem 2. Experiment with maxit, temp and try different random seeds to produce the shortest tour you can. Include a graph of the best tour you are able to find.



## HW 9.4 - Comparing Algorithms for a 30 dimensional Rastrigin function

The 30 dimensional Rastrigin function is considered very difficult to optimize and is a test case for many optimization algorithms. 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. The article “Continuous Global Optimzation in R,” by Katharine Mullen, gives an overview of optimizing continuous functions in R. You can find the article in the download packet.

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. Warning - some of the optimization routines find maxima instead of minima so you’ll need to maximize -f(x) to find the location of the minimum.

# 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: