HDS 5230 High Performance Computing Homework Week 8

Miao Cai

3/15/2019

Brutal force

Using the above code as a template, implement the following objective functions. Use both the brute force approach and the genetic algorithm to 'solve' the problem.

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\miaocai\Dropbox\@2018 SPRING HDS5230 High performance computing\hds5230Homework\HW8
## with 197 features
## It has 12 fields
## Integer64 fields read as strings:  OBJECTID
```

1. Change the objective function to minimize the median distance instead of the mean distance. Do you get a different optimal solution than when I used the mean?

```r
obj_func_min_mid <-
  function(solution,distMat) {
    setkey(distMat,OBJECTID)
    distMat[J(solution),
            list(min_dist=min(distancemiles)),
            by=pat_ID][,median(min_dist)]
  }

dist_temp_8 <- apply(
  possible_solutions, MARGIN = 1,
  obj_func_min_mid, distMat = pat_clinic_combinations)

temp_results_8 <- data.table(dist_temp_8, possible_solutions)
temp_results_8[min(dist_temp_8) == dist_temp_8]
```

```
##    dist_temp_8 V1 V2 V3 V4
## 1:     28.1379  3  5  1  8
```

2. Change the objective function to maximize the number of patients that live within 40 miles of a clinic. Do you get a different optimal solution than the median or mean?

```r
obj_func_max_40miles <-
  function(solution,distMat) {
    setkey(distMat,OBJECTID)
    distMat[J(solution),
            list(min_dist=min(distancemiles)),
            by=pat_ID][,sum(min_dist <= 40)]
  }

dist_temp_8 <- apply(
  possible_solutions, MARGIN = 1,
  obj_func_max_40miles, distMat = pat_clinic_combinations)
```

```r
temp_results_8 <- data.table(dist_temp_8, possible_solutions)
temp_results_8[min(dist_temp_8) == dist_temp_8]
```

```
##    dist_temp_8 V1 V2 V3 V4
## 1:        6569  6  4  2  7
```

3. In the prior 2 questions, we have not worried about the differences between urban and rural patients (they essentially had equal weights in our objective function). Now, I want you to make the rural patients 'worth more' in the objective function. You could simply exclude all the urban patients from the data and only consider the rural patients; But I still want you to consider the urban patients in the objective equation, I just want them to be worth 'less' than the rural patients. Use your answer from number two to construct a reasonable objective function for the following: 'Maximize the number of patients that live within 40 miles of a clinic. Rural patients should be worth 5 times as much as urban patients.' Does this give you a different answer than you got in number 2?

```r
obj_func_max_40miles_rural5_1 <-
  function(solution,distMat) {
    setkey(distMat,OBJECTID)
    distMat[J(solution),
            list(min_dist=min(distancemiles)),
            by=.(pat_ID, UR)][,sum(min_dist <= 40),UR][,sum(V1*c(5, 1))]
  }

dist_temp_8 <- apply(
  possible_solutions, MARGIN = 1,
  obj_func_max_40miles_rural5_1, distMat = pat_clinic_combinations)

temp_results_8 <- data.table(dist_temp_8, possible_solutions)
temp_results_8[min(dist_temp_8) == dist_temp_8]
```

```
##    dist_temp_8 V1 V2 V3 V4
## 1:       22948  6  4  2  8
```

Implement a genetic algorithm

Using the above code as a template, implement the following objective functions. Use both the brute force approach and the genetic algorithm to 'solve' the problem.

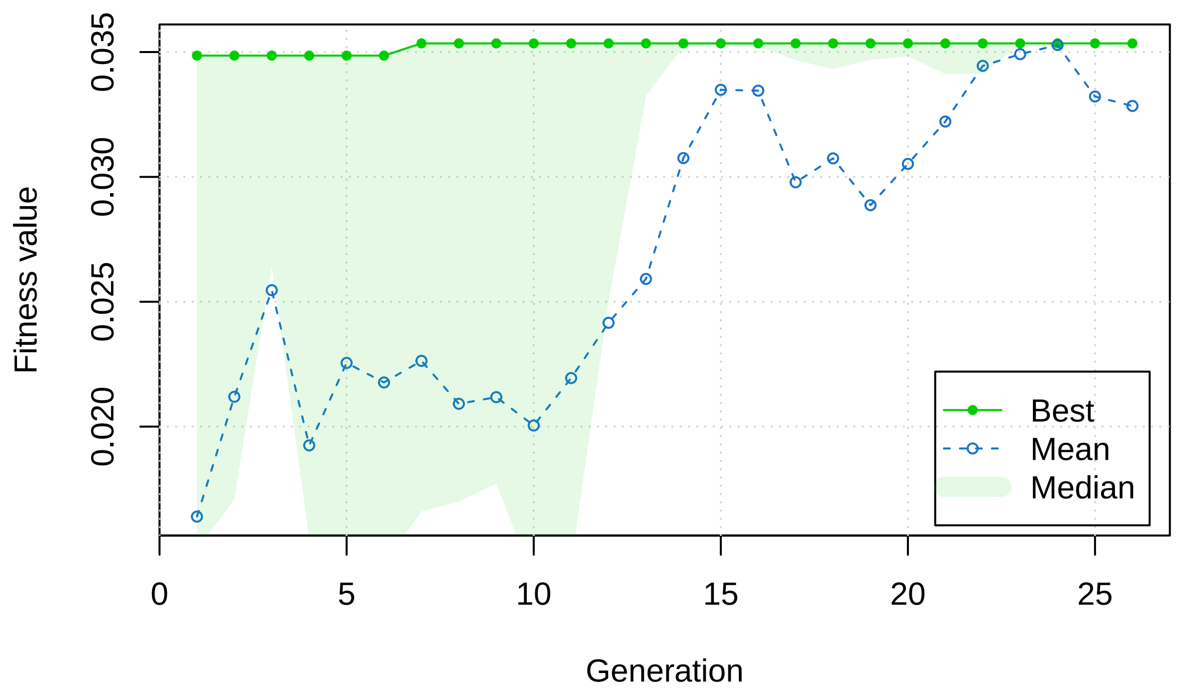
1. Change the objective function to minimize the median distance instead of the mean distance. Do you get a different optimal solution than when I used the mean?

```r
## Implement genetic algorithm
pacman::p_load(GA)

obj_func_min_mid2 <-
  function(solution,distMat,k) {
    setkey(distMat,OBJECTID)
    distMat[J(distMat[,unique(OBJECTID)[solution[1:k]]]), ## here is where I subset to only 4 sites
            list(min_dist=min(distancemiles)),
            by=pat_ID][,median(min_dist)]
  }
## make the inverse function so the maximum is the minimum...
Fitness_f <-
  function(solution,...) 1/obj_func_min_mid2(solution, ...)
```

```r
GA <- ga(type = "permutation", fitness = Fitness_f,
         distMat = pat_clinic_combinations, k=4,
         lower = 1, upper = 8, # these are the boundaries of the solution space (number of sites)
         popSize = 10, # total number of solutions per generation
         maxiter = 1000,  # max number of generations
         run = 20, # if we get the same best answer 20 times in a row, stop
         pmutation = 0.2) # this is the mutation rate per generation
## did it find the optimal solution?
summary(GA)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  10
## Number of generations =  1000
## Elitism               =  1
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 26
## Fitness function value = 0.03534746
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8
## [1,]  3  1  4  5  2  7  8  6
## [2,]  3  1  4  5  2  6  8  7
## [3,]  3  4  1  5  2  6  8  7
## [4,]  1  4  5  3  8  6  7  2
```

```r
plot(GA)
```



```
GA@solution # first 4 are the ones we choose
```

```
##      x1 x2 x3 x4 x5 x6 x7 x8
## [1,]  3  1  4  5  2  7  8  6
## [2,]  3  1  4  5  2  6  8  7
## [3,]  3  4  1  5  2  6  8  7
## [4,]  1  4  5  3  8  6  7  2
```

```
1/GA@fitnessValue
```

```
## [1] 28.29057
```

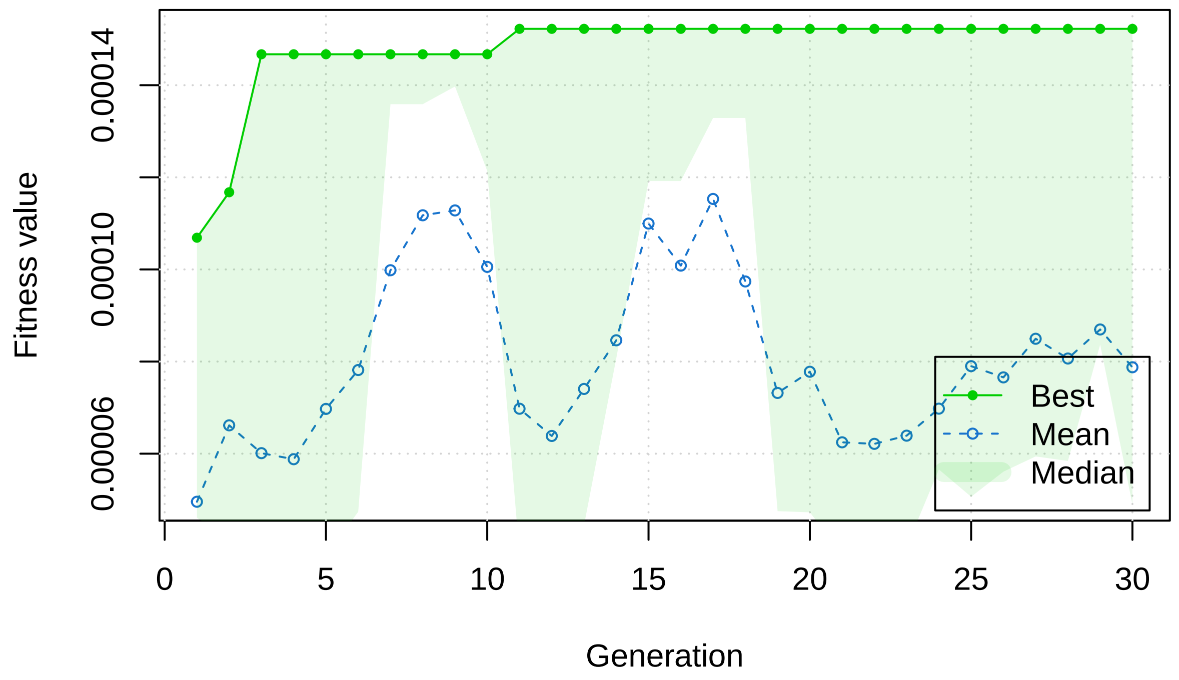
2. Change the objective function to maximize the number of patients that live within 40 miles of a clinic. Do you get a different optimal solution than the median or mean?

```
## Implement genetic algorithm
pacman::p_load(GA)

obj_func_max_40miles2 <-
  function(solution,distMat,k) {
    setkey(distMat,OBJECTID)
    distMat[J(distMat[,unique(OBJECTID)[solution[1:k]]]), ## here is where I subset to only 4 sites
            list(min_dist=min(distancemiles)),
            by=pat_ID][,sum(min_dist <= 40)]
  }
## make the inverse function so the maximum is the minimum...
Fitness_f <-
  function(solution,...) 1/obj_func_max_40miles2(solution, ...)
```

```r
GA <- ga(type = "permutation", fitness = Fitness_f,
         distMat = pat_clinic_combinations, k=4,
         lower = 1, upper = 8, # these are the boundaries of the solution space (number of sites)
         popSize = 10, # total number of solutions per generation
         maxiter = 1000,  # max number of generations
         run = 20, # if we get the same best answer 20 times in a row, stop
         pmutation = 0.2) # this is the mutation rate per generation
## did it find the optimal solution?
summary(GA)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  10
## Number of generations =  1000
## Elitism               =  1
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 30
## Fitness function value = 0.0001522302
## Solution =
##      x1 x2 x3 x4 x5 x6 x7 x8
## [1,]  4  2  7  6  1  5  3  8
```

```r
plot(GA)
```



```
GA@solution # first 4 are the ones we choose
```

```
##      x1 x2 x3 x4 x5 x6 x7 x8
## [1,]  4  2  7  6  1  5  3  8
```

```
1/GA@fitnessValue
```

```
## [1] 6569
```

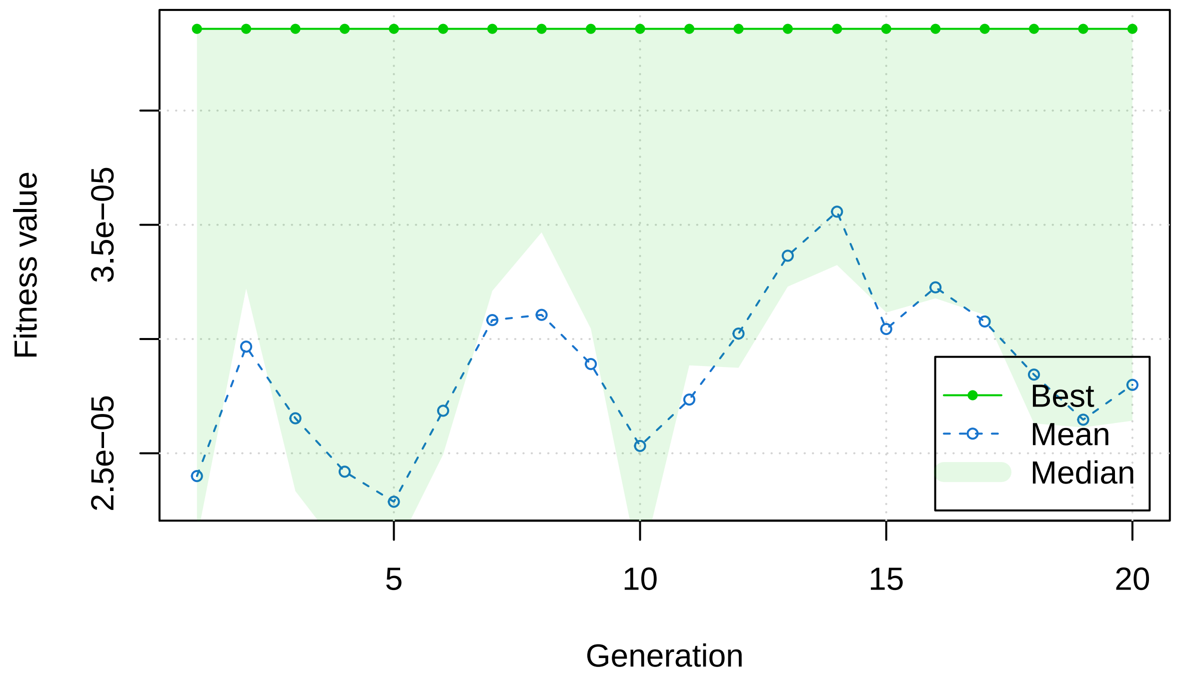
3. In the prior 2 questions, we have not worried about the differences between urban and rural patients (they essentially had equal weights in our objective function). Now, I want you to make the rural patients 'worth more' in the objective function. You could simply exclude all the urban patients from the data and only consider the rural patients; But I still want you to consider the urban patients in the objective equation, I just want them to be worth 'less' than the rural patients. Use your answer from number two to construct a reasonable objective function for the following: 'Maximize the number of patients that live within 40 miles of a clinic. Rural patients should be worth 5 times as much as urban patients.' Does this give you a different answer than you got in number 2?

```
## Implement genetic algorithm
pacman::p_load(GA)

obj_func_max_40miles_rural5_12 <-
  function(solution,distMat,k) {
    setkey(distMat,OBJECTID)
    distMat[J(distMat[,unique(OBJECTID)[solution[1:k]]]), ## here is where I subset to only 4 sites
            list(min_dist=min(distancemiles)),
            by=.(pat_ID, UR)][,sum(min_dist <= 40),UR][,sum(V1*c(5, 1))]
  }
```

```r
## make the inverse function so the maximum is the minimum...
Fitness_f <-
  function(solution,...) 1/obj_func_max_40miles_rural5_12(solution, ...)

GA <- ga(type = "permutation", fitness = Fitness_f,
         distMat = pat_clinic_combinations, k=4,
         lower = 1, upper = 8, # these are the boundaries of the solution space (number of sites)
         popSize = 10, # total number of solutions per generation
         maxiter = 1000,  # max number of generations
         run = 20, # if we get the same best answer 20 times in a row, stop
         pmutation = 0.2) # this is the mutation rate per generation
## did it find the optimal solution?
summary(GA)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  10
## Number of generations =  1000
## Elitism               =  1
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 20
## Fitness function value = 4.357678e-05
## Solution =
##      x1 x2 x3 x4 x5 x6 x7 x8
## [1,]  6  2  4  8  5  7  3  1
```

```r
plot(GA)
```



```
GA@solution # first 4 are the ones we choose
```

```
##      x1 x2 x3 x4 x5 x6 x7 x8
## [1,]  6  2  4  8  5  7  3  1
```

```
1/GA@fitnessValue
```

```
## [1] 22948
```