ERABILIKO DITUGUN TEKNOLOGIAK/TRESNAK

Alde batetik, irakasgaian erabiltzen dugun paketea, metaheuR: https://github.com/b0rxa/metaheuR

Bestetik, Shiny: http://shiny.rstudio.com/

Aplikazioak grafikoak izango ditu. MetaheuR paketean ggplot2 erabiltzen da grafikoak egiteko. Apur bat berezia da, baina oso erabilgarria da. Atxikituta bidaliko dizut liburu bat ggplot2-ri buruz (ez irakurtzeko, erreferentzia gisa izateko). Bestela, onlineko laguntza oso ona da: http://docs.ggplot2.org/current/

Proiektua kudeatzeko normalean, publikoa ez bada, BitBucket erabiltzen dugu: https://bitbucket.org/

APLIKAZIOAREN EZAUGARRIAK

Komentatu dugun moduan, aplikazioa erabiltzean sekuentzia tipikoa hauxe izango da:

1. Zerrenda batetik, optimizazio problema mota bat aukeratu
2. Problema mota horreko instantzia bat kargatu (graph coloring problemarako, adibidez, grafo konkretu bat).
3. Zerrenda batetik optimizazio algoritmo bat aukeratu.
4. Optimizazio algoritmoaren parametroak/aukerak ezarri
5. Algoritmoa exekutatu eta exekuzioa bistaratu

Ideia bat egiteko, hementxe duzu hau era sinplean egiteko kodea:

```
library(metaheuR)

# Problema bat aukeratu, TSP-a adibidez
# Problemaren instantzia bat aukeratu eta problema sortu. Adibide honetan, fitxategi batetik
# kargatu beharrean ausaz sortuko dugu hirien arteko distantziak

dist.matrix <- matrix(runif(50*50), ncol=50)

# Problema sortu
problem <- tspProblem(cmatrix=dist.matrix)


# Orain algoritmo bat aukeratuko dugu, bilaketa lokal sinple bat.
# Parametroak ezartzeko, funtzioari pasatuko dizkiogun argumentuak zerrenda batean
# sartuko ditugu

args <- list()

# Lehenengoa, soluzioak ebaluatzeko TSP problemak duen ebaluazio funtzioa
args$evaluate <- problem$evaluate
```

```
# Gero, hasierako soluzioa. Kasu honetan, hirien ausazko permutazio bat (matrizea
10x10ekoa denez, 50 hiri ditugu)
args$initial.solution <- randomPermutation(50)

# Hurrengoa, inguruneko soluzioak nola lortzen diren. Hau da, soluzioen ingurunea. Swap
ingurunea erabiliko dugu
args$neighborhood <- swapNeighborhood(args$initial.solution)

# Inguruneko soluzioak nola aukeratuko ditugun soluzioak. Kasu honetan, inguruneko
guztietatik onena
args$selector <- greedySelector

# Amaitzeko, algoritmoa gelditzeko baldintzak. Kasu honetan, gehienez 5 segundu emango
dizkiogu
args$resources <- cResource(time=5)


# Orain algoritmoa exekutatzen dugu

res <- do.call(basicLocalSearch, args)

# Amaitzeko, emaitzak bistaratuko ditugu

plotProgress(res)
```