METAHEURISTICS

# Load Libraries

library(GenSA)  
library(ggplot2)  
library(GA)

## Loading required package: foreach

## Loading required package: iterators

## Package 'GA' version 3.1.1  
## Type 'citation("GA")' for citing this R package in publications.

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

# Portfolio Optimization  
library("quantmod")

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

##   
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':  
##   
## first, last

## Loading required package: TTR

## Version 0.4-0 included new data defaults. See ?getSymbols.

tickers <- c("GE", "IBM", "GOOG", "AMZN", "AAPL")  
getSymbols(tickers, from = "2004-12-01", to = "2018-10-27")

## 'getSymbols' currently uses auto.assign=TRUE by default, but will  
## use auto.assign=FALSE in 0.5-0. You will still be able to use  
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")  
## and getOption("getSymbols.auto.assign") will still be checked for  
## alternate defaults.  
##   
## This message is shown once per session and may be disabled by setting   
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

##   
## WARNING: There have been significant changes to Yahoo Finance data.  
## Please see the Warning section of '?getSymbols.yahoo' for details.  
##   
## This message is shown once per session and may be disabled by setting  
## options("getSymbols.yahoo.warning"=FALSE).

## [1] "GE" "IBM" "GOOG" "AMZN" "AAPL"

P <- NULL  
for(ticker in tickers) {  
 tmp <- Cl(to.monthly(eval(parse(text = ticker))))  
 P <- cbind(P, tmp)  
}  
colnames(P) <- tickers  
R <- diff(log(P))  
R <- R[-1,]  
mu <- colMeans(R)  
sigma <- cov(R)  
library("PerformanceAnalytics")

##   
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':  
##   
## legend

pContribCVaR <- ES(weights = rep(0.2, 5), method = "gaussian", portfolio\_method = "component", mu = mu, sigma = sigma)$pct\_contrib\_ES  
obj <- function(w) {  
 fn.call <<- fn.call + 1  
 if (sum(w) == 0) { w <- w + 1e-2 }  
 w <- w / sum(w)  
 CVaR <- ES(weights = w, method = "gaussian", portfolio\_method = "component", mu = mu, sigma = sigma)  
 tmp1 <- CVaR$ES  
 tmp2 <- max(CVaR$pct\_contrib\_ES - 0.225, 0)  
 out <- tmp1 + 1e3 \* tmp2  
 return(out)  
}

# Using GenSA for Simulated Annealing  
set.seed(1234)  
fn.call <<- 0  
out.GenSA <- GenSA(fn = obj, lower = rep(0, 5), upper = rep(1, 5), control = list(smooth = FALSE, max.call = 3000))  
fn.call.GenSA <- fn.call  
out.GenSA$value

## [1] 0.1021234

out.GenSA$counts

## [1] 3000

cat("GenSA call functions", fn.call.GenSA, "times.\n")

## GenSA call functions 3000 times.

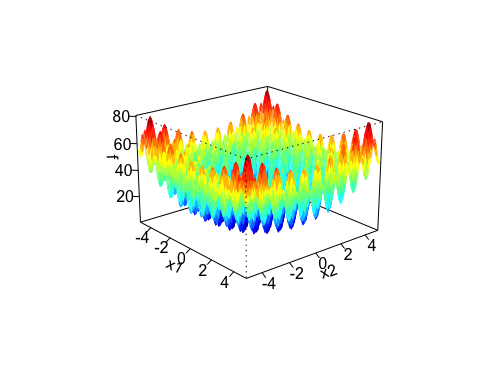
wstar.GenSA <- out.GenSA$par  
wstar.GenSA <- wstar.GenSA / sum(wstar.GenSA)  
rbind(tickers, round(100 \* wstar.GenSA, 2))

## [,1] [,2] [,3] [,4] [,5]   
## tickers "GE" "IBM" "GOOG" "AMZN" "AAPL"   
## "21.42" "29.17" "21.39" "8.91" "19.11"

100 \* (sum(wstar.GenSA \* mu) - mean(mu))

## [1] -0.2388974

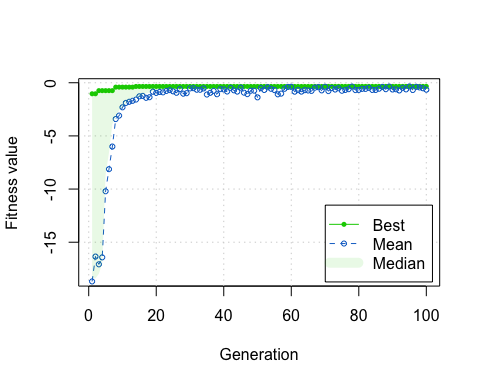
# Rastrigin function  
Rastrigin <- function(x1, x2)  
{  
 20 + x1^2 + x2^2 - 10\*(cos(2\*pi\*x1) + cos(2\*pi\*x2))  
}  
  
x1 <- x2 <- seq(-5.12, 5.12, by = 0.1)  
f <- outer(x1, x2, Rastrigin)  
persp3D(x1, x2, f, theta = 50, phi = 20, color.palette = jet.colors)



# Using GA for Genetic Algorithm  
gap <- ga(type = "real-valued",fitness=function(w) - Rastrigin(w[1],w[2]),lower=rep(0,5), upper=rep(1,5), popSize = 50, maxiter = 100)  
summary(gap)

## ── Genetic Algorithm ───────────────────   
##   
## GA settings:   
## Type = real-valued   
## Population size = 50   
## Number of generations = 100   
## Elitism = 2   
## Crossover probability = 0.8   
## Mutation probability = 0.1   
## Search domain =   
## x1 x2 x3 x4 x5  
## lower 0 0 0 0 0  
## upper 1 1 1 1 1  
##   
## GA results:   
## Iterations = 100   
## Fitness function value = -0.3441606   
## Solution =   
## x1 x2 x3 x4 x5  
## [1,] 0.005094966 0.04145404 0.5366386 0.7406573 0.5414703

plot(gap)



# Comparing GA and SA performance  
result <- matrix(c( 0.10,21.42,29.17,21.39,8.91,19.11,-0.042,0.012,0.008,0.469,0.439,0.371),nrow=6,ncol=2)  
colnames(result) <- c("Simulated Annealing","Genetic Algorithm")  
rownames(result) <- c("Fitness function value","x1","x2","x3","x4","x5")  
result

## Simulated Annealing Genetic Algorithm  
## Fitness function value 0.10 -0.042  
## x1 21.42 0.012  
## x2 29.17 0.008  
## x3 21.39 0.469  
## x4 8.91 0.439  
## x5 19.11 0.371

# Question 2 - Optimal parameters for regression   
X <- c(54,56,59,61,63,67,70,71,75,79,84,87)  
Y <- c(2.5,2.9,3.6,4.7,4.9,5.3,5.9,6.7,6.9,7.5,7.8,8.5)  
fun = function(x,y) { (sin(10\*x)\*cos(10\*y)+2)/sqrt(x^4+y^4+1) }   
obj = function(z)  
{  
 fn.call <<- fn.call + 1  
 SSYY=sum((Y-mean(Y))^2)  
 SSXY=sum((X-mean(X))\*(Y-mean(Y)))  
 SSX=sum((X-mean(X))^2)  
 b1= SSXY/SSX  
 b0=mean(Y)-b1\*mean(X)  
 Y\_Estimated=X\*b1+b0  
 Residuals= Y-Y\_Estimated  
 SSE=sum((Residuals -mean(Residuals))^2)  
 out <- SSE  
 return(out)  
}

# Using GenSA  
set.seed(1234)  
fn.call <<- 0  
out.GenSA <- GenSA(fn = obj, lower = rep(0, 2), upper = rep(5, 2), control = list(smooth = FALSE, max.call = 3000))  
fn.call.GenSA <- fn.call  
out.GenSA$value

## [1] 1.733883

out.GenSA$counts

## [1] 3000

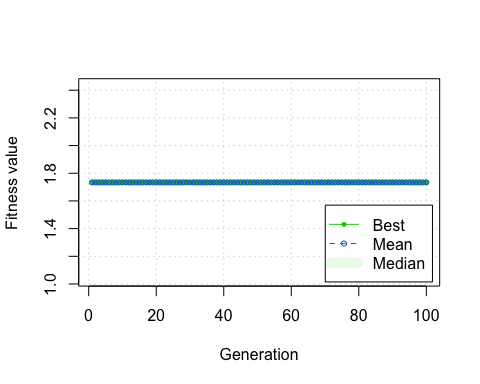
cat("GenSA call functions", fn.call.GenSA, "times.\n")

## GenSA call functions 3000 times.

# Using GA  
gap <- ga(type = "real-valued", fitness=obj, lower=rep(0,2), upper=rep(5,2), popSize = 50, maxiter = 100)  
summary(gap)

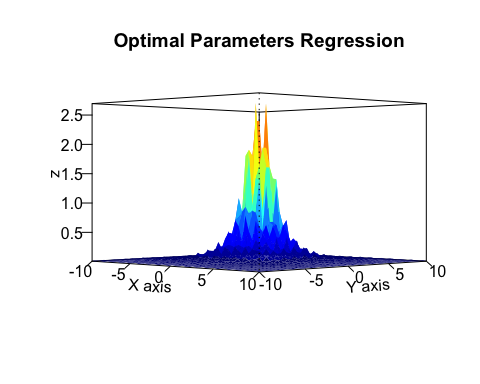
## ── Genetic Algorithm ───────────────────   
##   
## GA settings:   
## Type = real-valued   
## Population size = 50   
## Number of generations = 100   
## Elitism = 2   
## Crossover probability = 0.8   
## Mutation probability = 0.1   
## Search domain =   
## x1 x2  
## lower 0 0  
## upper 5 5  
##   
## GA results:   
## Iterations = 100   
## Fitness function value = 1.733883   
## Solutions =   
## x1 x2  
## [1,] 1.6018605 3.844055  
## [2,] 2.1611545 3.586050  
## [3,] 1.5885539 3.820426  
## [4,] 1.8108214 3.764008  
## [5,] 1.6475185 3.321844  
## [6,] 0.6831986 3.681029  
## [7,] 1.3628312 3.699395  
## [8,] 1.5601833 3.768104  
## [9,] 1.9335305 3.692252  
## [10,] 1.4089009 2.529521  
## ...   
## [48,] 1.6479827 3.800979  
## [49,] 0.8062318 3.926929

plot(gap)

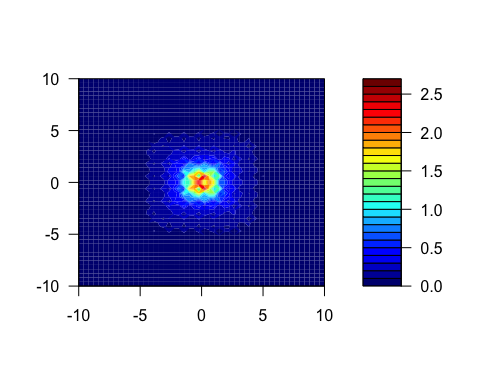


# Plot for Regression

x=seq(-10,10,length=50)  
y=seq(-10,10,length=50)   
z=outer(x,y,fun)   
 persp3D(x, y, z, phi = 0, theta = 45,  
 xlab = "X axis", ylab = "Y axis",  
 main = "Optimal Parameters Regression",  
 color.palette = jet.colors  
)



filled.contour(x, y, z, color.palette = jet.colors)



# Optimal Regression line  
plot(X,Y,xlim=c(0, 100),xlab="Passengers", ylab="Cost in ($1000)", col="red")  
  
abline(lsfit(X, Y),col = "green")

